ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

ΜΟΝΤΕΛΟΚΕΝΤΡΙΚΗ ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΔΙΑΧΕΙΡΙΣΗΣ ΥΠΗΡΕΣΙΩΝ ΔΙΑΔΙΚΤΥΟΥ ΜΕ ΕΠΙΓΝΩΣΗ ΤΟΥ ΠΛΑΙΣΙΟΥ ΧΡΗΣΗΣ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

της

Γεωργίας Μ. Καπιτσάκη

Διπλωματούχου Ηλεκτρολόγου Μηχανικού & Μηχανικού Υπολογιστών Ε.Μ.Π.

Αθήνα, Ιούνιος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

ΜΟΝΤΕΛΟΚΕΝΤΡΙΚΗ ΑΝΑΠΤΥΞΗ ΚΑΙ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΔΙΑΧΕΙΡΙΣΗΣ ΥΠΗΡΕΣΙΩΝ ΔΙΑΔΙΚΤΥΟΥ ΜΕ ΕΠΙΓΝΩΣΗ ΤΟΥ ΠΛΑΙΣΙΟΥ ΧΡΗΣΗΣ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Γεωργία Μ. Καπιτσάκη

Συμβουλευτική Επιτροπή : Ιάκωβος, Στ. Βενιέρης, Καθηγητής ΕΜΠ.

Δήμητρα-Θ. Ι. Κακλαμάνη, Αν. Καθηγήτρια ΕΜΠ.

Μιχαήλ Θεολόγου, Καθηγητής ΕΜΠ.

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την 4η Ιουνίου 2009.

.....	.....	.....
Ιάκωβος Στ. Βενιέρης Καθηγητής Ε.Μ.Π.	Δήμητρα-Θ. Ι. Κακλαμάνη Καθηγήτρια Ε.Μ.Π.	Μιχαήλ Θεολόγου Καθηγητής Ε.Μ.Π
.....	.....	.....
Γιώργος Στασινόπουλος Καθηγητής Ε.Μ.Π	Νικόλαος Ουζούνογλου Καθηγητής Ε.Μ.Π	Κώστας Κοντογιάννης Αναπληρωτής Καθηγητής Ε.Μ.Π.

.....
Χρήστος Κακλαμάνης
Καθηγητής
Πανεπιστημίου Πατρών

Αθήνα, Ιούνιος 2009

.....

Γεωργία Μ. Καπιτσάκη

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

Περίληψη

Η πληροφορία πλαισίου χρήσης (context) αποτελεί σημαντική παράμετρο στην ανάπτυξη και παροχή υπηρεσιών στην προσπάθεια προσφοράς προσωποποιημένων υπηρεσιών στους τελικούς χρήστες, οι οποίες λαμβάνουν υπόψη τόσο τις προτιμήσεις των χρηστών, όσο και διάφορα άλλα στοιχεία που σχετίζονται με την τρέχουσα ενασχόληση τους και τα χαρακτηριστικά του περιβάλλοντος στο οποίο βρίσκονται, όπως είναι η τοποθεσία, οι καιρικές συνθήκες, κτλ. Το πρόβλημα της διαχείρισης του πλαισίου χρήσης, καθώς και η ανάπτυξη εφαρμογών και υπηρεσιών με επίγνωση του πλαισίου χρήσης (context-awareness), έχουν γίνει ιδιαίτερα σημαντικές και ενδιαφέρουσες ερευνητικές περιοχές τα τελευταία χρόνια. Οι υπηρεσίες διαδικτύου (web services) ως η επικρατέστερη τεχνολογία για την υλοποίηση επιχειρησιακών διεργασιών και την ενσωμάτωση εφαρμογών χρησιμοποιούνται συχνά σε αυτά τα περιβάλλοντα οδηγώντας σε υπηρεσίες διαδικτύου με επίγνωση του πλαισίου χρήσης.

Η παρούσα διδακτορική διατριβή ασχολείται με τη διαχείριση της πληροφορίας πλαισίου χρήσης σε υπηρεσίες διαδικτύου και εφαρμογές που αποτελούνται από υπηρεσίες διαδικτύου. Το κύριο μέλημα της προτεινόμενης λύσης είναι ο διαχωρισμός της διαχείρισης του πλαισίου χρήσης από τη λειτουργία της εφαρμογής τόσο κατά την εκτέλεση της, όσο και κατά τη διαδικασία της ανάπτυξης. Επιπλέον, βασική παράμετρος αποτελεί η δυνατότητα επαναχρησιμοποίησης υπηρεσιών διαδικτύου που λειτουργούν ως συστατικά για την ανάπτυξη μιας σύνθετης εφαρμογής.

Για την εξασφάλιση των ανωτέρω, η διατριβή περιλαμβάνει μια αρχιτεκτονική διαχείρισης της πληροφορίας πλαισίου χρήσης για υπηρεσίες διαδικτύου που υλοποιείται μέσω κατάλληλης “σύλληψης” των μηνυμάτων αιτήσεων και απαντήσεων των υπηρεσιών. Επιπροσθέτως, προτείνεται μια μοντελοκεντρική μεθοδολογία για την ανάπτυξη εφαρμογών διαδικτύου που απαρτίζονται από υπηρεσίες με επίγνωση του πλαισίου χρήσης, καθώς και μια διαδικασία εύρεσης υπηρεσιών που παρέχουν πρόσβαση σε πληροφορίες πλαισίου χρήσης μέσω της αντιστοίχισής τους με κατάλληλες υπηρεσίες διαδικτύου. Μέσω της μεθοδολογίας δίνεται μια ολοκληρωμένη λύση στο πρόβλημα της ανάπτυξης εφαρμογών που αποτελούνται από υπηρεσίες διαδικτύου με επίγνωση του πλαισίου χρήσης, καθώς η πληροφορία διαχειρίζεται σε όλα τα στάδια ανάπτυξης. Το προτεινόμενο σύστημα, η μεθοδολογία και οι επιμέρους διαδικασίες περιγράφονται λεπτομερώς σε σχέση με την τρέχουσα βιβλιογραφία και επιπλέον τεκμηριώνονται με παραδείγματα της λειτουργίας τους, ενώ αξιολογούνται βάσει αναγνωρισμένων κριτηρίων.

Λέξεις κλειδιά: υπηρεσία διαδικτύου, πλαίσιο χρήσης, επίγνωση πλαισίου χρήσης, εφαρμογή διαδικτύου, μοντελοκεντρική ανάπτυξη, UML μοντελοποίηση, UML μεταμοντέλο, πηγή πλαισίου χρήσης.

Abstract

Context information constitutes an important parameter in the development and provision of services towards the need for personalizing the user experience during service provision. Context may refer to user preferences and other information related with the current user activity and the characteristics of the execution environment, such as current location, weather conditions, etc. The issue of context management, as well as the development of context-aware applications, have become interesting and challenging research fields in the latest years. Web services as the most popular technology for the implementation of business processes and the application integration are usually exploited in this field leading to the notion of context-aware web services.

The main subject of the current doctoral thesis lies in the management of context information for web services and web applications consisting of web services. The primary goal of the proposed solution is the decoupling of context management from the main business logic both in the phase of the application provision and in the preceding development stages. Moreover, the solution concentrates on the provision of appropriate constructs that allow the reuse of web services that can act as main building blocks towards the creation of new, composite web applications that integrate the functionality of different components.

In order to guarantee the above, the thesis proposes a context adaptation mechanism for web services implemented through a modularized management architecture that is based on request and response message interception. The architecture is accompanied by a model-driven methodogy for the development of composite web applications consisting of context-aware web services. A matching approach for the identification of web services that provide access to context information and that can be efficiently combined with appropriate business web services offering specific functionality is also proposed. The methodology provides a complete solution towards the development of applications consisting of context-aware web services, where context information is appropriately handled through all development stages. The system, the methodology and the relevant procedures proposed are described in detail in respect to the current research literature, documented through different use cases for their validation and evaluated through appropriate assessment metrics.

Keywords: web service, context, context-awareness, web application, model-driven engineering, Unified Modeling Language, UML metamodel, context source.

Πρόλογος

Καθώς φαίνεται, ήρθε η πολυπόθητη στιγμή, όπου το περιπετειώδες ταξίδι της εκπόνησης της διδακτορικής διατριβής έφτασε στο πέρας του. Με τον παρόν πρόλογο θα ήθελα να εκφράσω τη μεγάλη μου χαρά, αλλά και να ευχαριστήσω όσους με βοήθησαν και μου συμπαραστάθηκαν σε αυτό το έργο.

Αρχικά θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή της διατριβής μου, τον κ. Ιάκωβο Βενιέρη, για την ευκαιρία που μου έδωσε και την καθοδήγησή του κατά τις ερευνητικές μου δραστηριότητες προς την περάτωση της διατριβής. Κατάφερνε πάντα να βρίσκει λύσεις στα προβλήματα και να μου δίνει κίνητρο και παρότρυνση να συνεχίσω μπροστά. Επίσης, την καθηγήτρια κ. Δήμητρα-Θεοδώρα Κακλαμάνη για την ενίσχυσή της καθ' όλη τη διάρκεια εκπόνησης της διατριβής, αλλά και το τρίτο μέλος της τριμελούς επιτροπής, τον καθηγητή κ. Μιχάλη Θεολόγου, για τη συνεχή παρακίνησή του για την περάτωση της διατριβής και την πάντα φιλική του διάθεση. Ιδιαίτερες ευχαριστίες αρμόζουν και στο Εθνικό Μετσόβιο Πολυτεχνείο και τη Σχολή των Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, αλλά και το Ίδρυμα Κρατικών Υποτροφιών για την ενίσχυση της προσωπικής μου προσπάθειας.

Θα ήθελα ακόμα να ευχαριστήσω όλα τα μέλη της εφταμελούς επιτροπής εξέτασης της διατριβής μου: τους καθηγητές Γιώργο Στασινόπουλο, Νικόλαο Ουζούνογλου, Κώστα Κοντογιάννη και Χρήστο Κακλαμάνη, για την υποστήριξή τους.

Από τους συναδέλφους και συνεργάτες ιδιαίτερη αναφορά αρμόζει πρώτα στους Δημήτρη Κατέρο και Γιώργο Πρεζεράκο, οι οποίοι προσέφεραν σημαντική βοήθεια με τις ιδέες τους και τις διεξόδους που υποδείκνυαν καθ' όλη τη διάρκεια εκπόνησης της διατριβής και με τους οποίους συνεργάστηκα για την εκπόνηση ερευνητικού έργου και τη συγγραφή διαφόρων ερευνητικών εργασιών για διεθνή περιοδικά και συνέδρια. Επίσης, στους αγαπητούς Γιώργο Λιουδάκη και Νίκο Τσελίκα ένα θερμό ευχαριστώ για τη συνεργασία τους σε ερευνητικές αναζητήσεις και δημοσιεύσεις, αλλά και τη βοήθεια που προσέφεραν σε διάφορα επίπεδα και τις πολύτιμες συμβουλές τους. Να ευχαριστήσω ακόμα το Γιάννη Φουκαράκη και το Χρήστο Παππά για την ωραία μας συνεργασία και τα ευχάριστα ταξίδια.

Στον υπόλοιπο κόσμο του εργαστηρίου και τους συναδέλφους και συμπορευόμενους, αξίζουν πολλά ευχαριστώ, καθώς η καθημερινή μας επαφή έκανε το κλίμα ιδιαίτερα ευχάριστο και βοηθούσε στο να ξεχάσουμε για λίγο όλοι μας τις διάφορες υποχρεώσεις. Να αναφέρω τον πολυμήχανο Χρήστο Κατσιγιάννη, το χαμογελαστό Βαγγέλη

Κοσμάτο και την πάντα φιλική Σοφία Καπελλάκη, αλλά και τους Νίκο Δέλλα, Λευτέρη Κουτσολουκά, Χρύσα Παπαγιάννη και τους νεότερους, Άννα Αντωνακοπούλου και Φώτη Γώγουλο για την ευχάριστη νότα στο κλίμα του εργαστηρίου.

Μεταξύ των φίλων μου, τους οποίους ωστόσο δε μπορώ να απαριθμήσω εξαντλητικά σε αυτό το χώρο, θα ήθελα να αναφέρω την Ευαγγελία Αντωνιάδου για τη στήριξη της σε όλους τους τομείς, τη Λίλα Δημάκη για τη φιλία μας που μετράει πολλά χρόνια και τη Διονυσία Πετράκη για την κατανόησή της και τις ξένοιαστες στιγμές μας. Δε μπορώ να μετρήσω το μέγεθος της σημασίας που έχετε για μένα. Ξέρετε πως έχετε όλη μου την αγάπη και εκτίμηση.

Τέλος, το πιο μεγάλο ευχαριστώ μαζί με την απεριόριστη αγάπη μου πηγαίνει στην οικογενειά μου, τους γονείς μου Κυριακή και Μηνά και τα αδέρφια μου Μιράντα και Αλέξανδρο, που μου παρείχαν τα απαραίτητα εφόδια για τη διαρκή πορεία μου. Με τη συνεχή υποστήριξή τους καθ' όλη τη διάρκεια των σπουδών μου και τη διαρκή ενθάρρυνσή τους μου έδιναν πάντα δύναμη να συνεχίσω μπροστά χωρίς ποτέ να κοιτάω πίσω και να εξελιχθώ σε έναν ολοκληρωμένο άνθρωπο.

Γεωργία Μ. Καπιτσάκη

Αθήνα, Ιούνιος 2009

"Ο άνθρωπος δεν επιθυμεί κάτι
επειδή είναι πραγματικό.
Αντιθέτως, το θεωρεί πραγματικό
επειδή το επιθυμεί."
– Μπαρούχ Σπινόζα

Περιεχόμενα

Λίστα Εικόνων

Λίστα Πινάκων

1ο Κεφάλαιο

Εισαγωγή

Στο παρόν Κεφάλαιο γίνεται μια εισαγωγή στη διδακτορική διατριβή μέσω της περιγραφής του ερευνητικού προβλήματος που μελετάται και της σύντομης περιγραφής της προτεινόμενης προσέγγισης, ενώ αναφέρεται και η διάρθρωση των κεφαλαίων της διατριβής. Για να μπορέσει να γίνει περισσότερο κατανοητή η ερευνητική περιοχή που συνάδει με τη διατριβή και να γίνει καλύτερα αντιληπτό το προς αντιμετώπιση πρόβλημα δίνονται ακόμα βασικοί ορισμοί, όπως έχουν προταθεί κατά καιρούς στη βιβλιογραφία και όπως συνηθίζεται να χρησιμοποιούνται σήμερα σε αντίστοιχες μελέτες και εργασίες.

1.1 Κίνητρο και Ερευνητική Προσέγγιση

Τα τελευταία χρόνια παρατηρείται μια αυξανόμενη τάση προς την ανάπτυξη συστημάτων και υπηρεσιών που σχετίζονται με το κινητό διαδίκτυο (mobile Internet ή mobile computing). Πράγματι, η φύση της εργασίας πολλών ατόμων στο σύγχρονο κόσμο περιλαμβάνει συχνές μετακινήσεις σε διάφορα μέρη του κόσμου. Ακόμα, συσκευές που χρησιμοποιούνται καθημερινά σε όλο τον κόσμο από την πλειοψήφία του πληθυσμού – τουλάχιστον των ανεπτυγμένων χωρών – (π.χ. κινητά τηλέφωνα, Προσωπικοί Ψηφιακοί Βοηθοί ή Personal Digital Assistants / PDAs, φορητοί υπολογιστές, κτλ.) έχουν εισάγει τις έννοιες της κινητής επικοινωνίας, πληροφόρησης, αλλά και ψυχαγωγίας στο καθημερινό λεξιλόγιο. Ένα σημαντικό παραδείγμα τέτοιων υπηρεσιών αποτελεί το i-mode για την κινητή τηλεφωνία της ιαπωνικής εταιρίας NTT DoCoMo [26] που ξεκίνησε από την Ιαπωνία, αλλά που έχει πλέον διαδοθεί και στον υπόλοιπο κόσμο. Το παραπάνω φαινόμενο καθιστά την παροχή υπηρεσιών που να μπορούν να κεντρίσουν το ενδιαφέρον, αλλά και να κερδίσουν την εμπιστοσύνη των χρηστών ιδιαίτερα ενδιαφέρουσα ενασχόληση, αλλά και μια σημαντική πρόκληση τόσο για τους ερευνητές όσο και για τους παρόχους υπηρεσιών. Η

εξέλιξη των κινητών συστημάτων αναμένεται να συνεχιστεί και στο μέλλον οδηγώντας σε περιβάλλοντα με μεγάλο αριθμό συσκευών ενσωματωμένων ακόμα στο ίδιο το περιβάλλον των χρηστών.

Στο τρέχον περιβάλλον ένα από τα πρώτα μελήματα των μηχανικών υπηρεσιών και λογισμικού και των παρόχων είναι να προσφέρουν στους χρήστες προσωποποιημένες υπηρεσίες και εφαρμογές που λαμβάνουν υπόψη τόσο τις προτιμήσεις των χρηστών, όσο και διάφορα άλλα στοιχεία που σχετίζονται με την τρέχουσα ενασχόληση τους και τα χαρακτηριστικά του περιβάλλοντος στο οποίο βρίσκονται, όπως είναι η τοποθεσία, οι καιρικές συνθήκες, κτλ. Υπό αυτό το πλαίσιο ανακύπτει η ανάγκη διαχείρισης αυτής της πληροφορίας – που αναφέρεται ως πλαίσιο χρήσης (context) – και της σωστής χρήσης της στην ανάπτυξη και παροχή υπηρεσιών στους χρήστες. Το πλαίσιο χρήσης αποτελεί ένα σημαντικό κομμάτι του κινητού διαδικτύου, καθώς οι συσκευές μπορούν να κάνουν ουσιαστική χρήση αυτής της πληροφορίας για να παρέχουν νέες υπηρεσίες (π.χ. βασισμένες στην τοποθεσία του χρήστη) ή ακόμα για να εμπλουτίσουν τις υπάρχουσες υπηρεσίες. Επιπλέον, τα τελευταία χρόνια παρατηρείται μια τάση μεταφοράς των τεχνολογιών του παραδοσιακού διαδικτύου (internet) στο κινητό διαδίκτυο. Οι υπηρεσίες διαδικτύου (Web Services / WSs) ως η επικρατέστερη τεχνολογία για την υλοποίηση επιχειρησιακών διεργασιών και την ενσωμάτωση εφαρμογών είναι ευρέως διαδεδομένες και σε αυτά τα περιβάλλοντα κυρίως χάριν της συμβατότητας τους με διάφορα συστήματα και υλοποιήσεις.

Καθώς αυξάνεται το πλήθος των διαθέσιμων υπηρεσιών διαδικτύου, ενισχύεται η τάση ανάπτυξης εφαρμογών που ενσωματώνουν τις λειτουργίες των υπάρχουσων υπηρεσιών. Υπό αυτό το πρίσμα οι υπηρεσίες διαδικτύου μπορούν να θεωρηθούν αυτόνομα δομικά συσταστικά για τη σύνθεση και δημιουργία πλουσιότερων εφαρμογών. Η πλειοψηφία των ερευνητικών προσπαθειών στο πεδίο της σύνθεσης υπηρεσιών επικεντρώνεται στις αλληλεπιδράσεις μεταξύ εφαρμογών [21]. Ωστόσο, το τελευταίο διάστημα παρατηρείται ερευνητική δραστηριότητα που σχετίζεται με τη δημιουργία εφαρμογών διαδικτύου που κατασκευάζονται γύρω από τις ανάγκες του τελικού χρήστη. Ένα παράδειγμα αποτελούν τα mashup που προκύπτουν από ένα συνδυασμό δεδομένων από διαφορετικές πηγές και συχνά χρησιμοποιούν υπηρεσίες διαδικτύου ως πηγές δεδομένων.

Έχοντας ως αφετηρία τα παραπάνω το ερευνητικό ζήτημα που μελετάται στην παρούσα διατριβή είναι η διαχείριση της πληροφορίας πλαισίου χρήσης για την περίπτωση δικτυακών εφαρμογών που αποτελούνται από υπηρεσίες διαδικτύου. Κάτι τέτοιο είναι

απαραίτητο για να γίνει εφικτή η χρήση της πληροφορίας από τις ίδιες τις υπηρεσίες. Η προσαρμογή των υπηρεσιών διαδικτύου στο πλαίσιο χρήσης επιχειρείται μέσω μιας κατανεμημένης αρχιτεκτονικής που διαχειρίζεται τις αιτήσεις που στέλνονται προς την υπηρεσία από τον χρήστη και τις απαντήσεις που δίνονται έπειτα από την υπηρεσία ως αποτέλεσμα της λειτουργίας της. Σε κάθε περίπτωση η διαχείριση αφορά εφαρμογές που αποτελούνται απο υπηρεσίες διαδικτύου. Εφαρμογές αυτού του τύπου είναι ευρέως διαδεδομένες και μπορούν να χρησιμοποιηθούν σε διάφορα περιβάλλοντα. Η αρχιτεκτονική προσαρμογής επικεντρώνεται στη χρήση των υπηρεσιών διαδικτύου για εφαρμογές διαδικτύου (web applications), για εφαρμογές δηλ. που μπορούν να προσπελαστούν μέσω ενός “αδύνατου” ή slim πελάτη, όπως αναφέρεται συχνά ο φυλλομετρητής ιστού (browser). Επιλέχθηκε αυτή ή προσέγγιση για να εξασφαλιστεί η ανεξαρτησία της εκάστοτε εφαρμογής από το λογισμικό στο τερματικό του χρήστη, αφού ένας slim πελάτης μπορεί να υποστηρίξει μια πληθώρα εφαρμογών χωρίς να απαιτείται κάποια ιδιαίτερη διαμόρφωση για την προς εκτέλεση εφαρμογή.

Πολλές φορές η προσπάθεια διαχείρισης της πληροφορίας πλαισίου χρήσης γίνεται από τις ίδιες τις υπηρεσίες ως κομμάτι της λειτουργίας τους. Κάτι τέτοιο όμως δεν είναι πάντα επιθυμητό, καθώς είναι βασικό να μπορεί μια υπηρεσία να λειτουργεί απρόσκοπτα είτε με είτε χωρίς την πληροφορία πλαισίου χρήσης που σχετίζεται με την ίδια, τον τελικό χρήστη και το περιβάλλον του. Είναι επομένως προτιμότερο η συγκεκριμένη διαχείριση να διατηρείται ανεξάρτητη από την κυρίως λειτουργία της εφαρμογής. Κάτι τέτοιο προσδίδει μεγαλύτερη ευελιξία στη χρήση και τις δυνατότητες της υπηρεσίας. Επιπροσθέτως, αυτή η προσέγγιση καθιστά την ανάπτυξη της υπηρεσίας πιο γρήγορη και εύκολη για τον μηχανικό λογισμικού, αφού η διαχείριση του πλαισίου χρήσης είναι ανεξάρτητη και δεν επηρεάζει ούτε σχετίζεται με τη βασική της λειτουργία.

Ακόμα αντιμετωπίζεται το πρόβλημα της ανάπτυξης εφαρμογών προσαρμοσμένων στο πλαίσιο χρήσης, ενώ ενισχύεται η επαναχρησιμοποίηση υπηρεσιών ως δομικών στοιχείων της εφαρμογής για την ολοκλήρωση διαφόρων λειτουργιών. Συγκεκριμένα, η διατριβή πραγματεύεται τη μοντελοκεντρική ανάπτυξη εφαρμογών διαδικτύου προσαρμοσμένων στο πλαίσιο χρήσης. Μια δομημένη μεθοδολογία επαρκεί για να προκύψει μια λειτουργική εφαρμογή διαδικτύου, στην οποία η πληροφορία πλαισίου χρήσης διαχειρίζεται σε όλα τα στάδια ανάπτυξης διατηρώντας συγχρόνως αυτή τη διαχείριση ανεξάρτητη από τη βασική λειτουργία της εφαρμογής. Επιπλέον, κατά την ανάπτυξη εφαρμογών υπάρχοντα δομικά συστατικά που προσφέρουν συγκεκριμένες λειτουργίες μπορούν να διατεθούν ως υπηρεσίες διαδικτύου και να

επαναχρησιμοποιηθούν. Με αυτόν τον τρόπο ενισχύεται η επαναχρησιμοποίηση κώδικα και η διαδικασία ανάπτυξης βασίζεται σε τμήματα λογισμικού που έχουν ήδη εφαρμοσθεί επιτυχώς. Αυτό είναι εφικτό τόσο για τις υπηρεσίες διαδικτύου που προσφέρουν συγκεκριμένη λειτουργικότητα, όσο και για τις υπηρεσίες που παρέχουν έμμεση πρόσβαση σε πληροφορίες πλαισίου χρήσης ενσωματώνοντας τη λειτουργία πηγών πλαισίου χρήσης. Για το λόγο αυτό στα πλαίσια της διατριβής προτείνεται ακόμα μια διαδικασία εύρεσης πιθανών πηγών πλαισίου χρήσης, προκειμένου να επαναχρησιμοποιηθούν στην ανάπτυξη νέων εφαρμογών.

1.2 Διάρθρωση Διατριβής

Η παρουσίαση της διατριβής εξελίσσεται σε συνολικά εννέα Κεφάλαια. Πέρα από το εισαγωγικό, το περιεχόμενο των υπόλοιπων κεφαλαίων διανέμεται όπως περιγράφεται ακολούθως.

Στο δεύτερο Κεφάλαιο γίνεται μια επισκόπηση των πιο διαδεδομένων τεχνολογιών διαδικτύου. Παρουσιάζονται αναλυτικά οι υπηρεσίες διαδικτύου και τα βασικά πρωτόκολλα που χρησιμοποιούνται για την περιγραφή και ανακάλυψη, καθώς και τη μεταφορά των δεδομένων των υπηρεσιών διαδικτύου. Ακολούθως, παρουσιάζονται τα πιο διαδεδομένα πλαίσια εργασίας για υπηρεσίες διαδικτύου, τα οποία διευκολύνουν τους προγραμματιστές υπηρεσιών διαδικτύου υλοποιώντας τα βασικά πρωτόκολλα των υπηρεσιών. Το Κεφάλαιο κλείνει με την περιγραφή του αρχιτεκτονικού προτύπου Μοντέλου-Όψης-Ελεγκτή και τα πιο γνωστά πλαίσια που ακολουθούν τις αρχιτεκτονικές του αρχές.

Στο τρίτο Κεφάλαιο πραγματοποιείται μία αναδρομή στις ερευνητικές προσπάθειες που έχουν προταθεί μέχρι σήμερα αναφορικά με την προσαρμογή εφαρμογών και υπηρεσιών στο πλαίσιο χρήσης. Οι λύσεις που προτείνονται κατηγοριοποιούνται σε τρεις κατηγορίες: τις λύσεις μεσισμικού και ειδικές πλατφόρμες υπηρεσιών, όπου προτείνονται ειδικά συστήματα για την παροχή υπηρεσιών με επίγνωση του πλαισίου χρήσης, τις λύσεις σε επίπεδο πηγαίου κώδικα ή επεκτάσεις γλωσσών προγραμματισμού, όπου η διαχείριση της πληροφορίας πλαισίου χρήσης γίνεται απευθείας σε επίπεδο κώδικα εμπλουτίζοντας τη λογική της εφαρμογής με κομμάτια κώδικα υπεύθυνα για το χειρισμό του πλαισίου χρήσης, και τις προσεγγίσεις μέσω “σύλληψης” μηνυμάτων, όπου προσαρμογή επιτυγχάνεται συλλέγοντας τα εισερχόμενα και εξερχόμενα μηνύματα και τροποποιώντας τα κατάλληλα χωρίς να μεταβληθεί η κύρια εφαρμογή και η λειτουργία της.

Στο τέταρτο Κεφάλαιο περιγράφεται η αρχιτεκτονική που αναπτύχθηκε στα πλαίσια της διατριβής και που προσφέρει ένα αποδοτικό χειρισμό της πληροφορίας πλαισίου χρήσης για τις υπηρεσίες διαδικτύου έχοντας ως αποτέλεσμα την προσαρμογή των υπηρεσιών. Στο Κεφάλαιο περιγράφονται αρχικά τα επίπεδα προσαρμογής των μηνυμάτων των υπηρεσιών στο πλαίσιο χρήσης, ενώ ακολουθεί η περιγραφή της γενικής λειτουργίας της αρχιτεκτονικής. Εν συνεχεία, πραγματοποιείται μια ανάλυση των στοιχείων που απαρτίζουν την αρχιτεκτονική (διαχειριστής μηνυμάτων, προσαρμογείς στο πλαίσιο χρήσης, πηγές πληροφορίας) και, τέλος, περιγράφονται οι λεπτομέρειες υλοποίησης του μηχανισμού προσαρμογής.

Στο πέμπτο Κεφάλαιο της διατριβής πραγματοποιείται μια επισκόπηση των λύσεων για τη μοντελοκεντρική ανάπτυξη υπηρεσιών. Προσεγγίζεται το θέμα της μοντελοποίησης της πληροφορίας πλαισίου χρήσης και περιγράφονται οι προτεινόμενες λύσεις της διεθνούς βιβλιογραφίας. Ακολούθως, γίνεται μια επισκόπηση προτεινόμενων προσεγγίσεων και τεχνικών που σχετίζονται με τη μοντελοκεντρική ανάπτυξη υπηρεσιών διαδικτύου και, τέλος, παρουσιάζονται κάποια γενικά εργαλεία μοντελοκεντρικής ανάπτυξης λογισμικού.

Το έκτο Κεφάλαιο πραγματεύεται τη μοντελοκεντρική ανάπτυξη εφαρμογών με επίγνωση του πλαισίου χρήσης, οι οποίες αποτελούνται από υπηρεσίες διαδικτύου, όπως προτείνεται στα πλαίσια της διατριβής. Αρχικά, δίνεται μια γενική περιγραφή της μεθοδολογίας και των βημάτων που την απαρτίζουν και εν συνεχεία περιγράφονται τα τρία προφίλ που έχουν οριστεί για τη σχεδίαση της εφαρμογής: το προφίλ υπηρεσιών διαδικτύου, το μεταμοντέλο πληροφορίας πλαισίου χρήσης και το προφίλ παρουσίασης. Ακολουθεί η αναλυτική περιγραφή των βημάτων της μεθοδολογίας από τη μοντελοποίηση της εφαρμογής και των εξαρτήσεων με το πλαίσιο χρήσης μέχρι το μετασχηματισμό του μοντέλου σε εκτελέσιμο κώδικα.

Στο έβδομο Κεφάλαιο δίνονται κάποια παραδείγματα χρήσης της προτεινόμενης μεθοδολογίας και του μηχανισμού προσαρμογής μέσω μιας απλής γενικής εφαρμογής, ενώ αναλυτικά περιγράφεται η ανάπτυξη ενός πιο πολύπλοκου σεναρίου. Πραγματοποιείται ακόμα η αξιολόγηση της λύσης λαμβάνοντας υπόψη διάφορα ποιοτικά κριτήρια και μετρικές αποτίμησης, όπως προκύπτουν από τη μηχανική λογισμικού γενικότερα και από τις αρχές της μοντελοκεντρικής ανάπτυξης και τα αντίστοιχα εργαλεία ειδικότερα. Επιπλέον δίνονται και κάποια ειδικά κριτήρια που σχετίζονται με την ανάπτυξη, αλλά και παροχή υπηρεσιών με επίγνωση του πλαισίου χρήσης.

Στο όγδοο Κεφάλαιο παρουσιάζεται η λύση που έχει αναπτυχθεί για την εύρεση πιθανών υπηρεσιών-πηγών πλαισίου χρήσης για συνδυασμό με υπηρεσίες εφαρμογών στα πλαίσια της ανάπτυξης εφαρμογών με επίγνωση του πλαισίου χρήσης. Η διαδικασία εύρεσης βασίζεται σε κατάλληλες περιγραφές υπηρεσιών διαδικτύου που εισάγονται σε μια "αποθήκη" υπηρεσιών. Η προτεινόμενη προσέγγιση αξιολογείται μέσω μετρικών που προέρχονται από το πεδίο της ανάκτησης πληροφοριών.

Τέλος, το ένατο Κεφάλαιο αποτελεί τον επίλογο της διατριβής, όπου συνοψίζεται η προτεινόμενη λύση και παρουσιάζονται τα βασικά συμπεράσματα. Επιπλέον, δίνονται κατευθύνσεις μελλοντικής έρευνας και επέκτασης έχοντας ως βάση τη λύση που προτείνεται στη διατριβή.

1.3 Βασικές Έννοιες

Στην παρούσα ενότητα δίνονται κάποιες βασικές έννοιες για να γίνει πιο κατανοητό το ερευνητικό πλαίσιο στο οποίο κινείται η διατριβή και το προς επίλυση πρόβλημα που μελετάται.

1.3.1 Υπηρεσίες Διαδικτύου

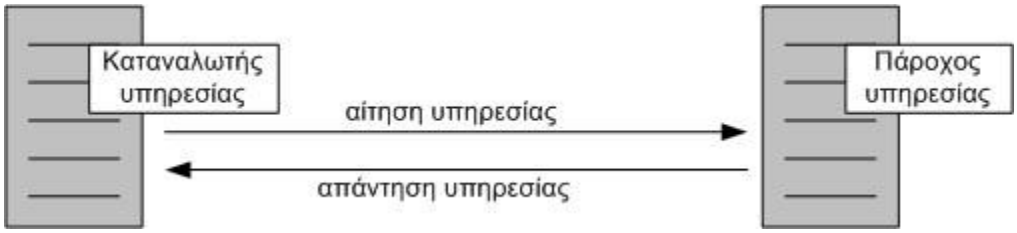
Η Προσανατολισμένη σε Υπηρεσίες Αρχιτεκτονική ή Service-oriented Architecture (SOA) [17] αποτελεί μια μεθοδολογία για την ανάπτυξη και την ενσωμάτωση συστημάτων, όπου οι βασικές λειτουργίες ομαδοποιούνται σε επιχειρησιακές διεργασίες οδηγώντας σε διαλειτουργικές υπηρεσίες, υπηρεσίες που μπορούν δηλ. να αλληλεπιδράσουν μεταξύ τους επιτυχώς. Ο βασικός στόχος της SOA είναι η "χαλαρή" συσχέτιση των υπηρεσιών με λειτουργικά συστήματα, γλώσσες προγραμματισμού και άλλες τεχνολογίες που αποτελούν την υποδομή βάσης των εφαρμογών. Η SOA χωρίζει τις λειτουργίες σε διακριτά τμήματα ή υπηρεσίες που καθίστανται προσβάσιμα σε ένα δίκτυο, ώστε να μπορούν να συνδυαστούν με άλλα και να επαναχρησιμοποιηθούν στην παραγωγή επιχειρησιακών εφαρμογών. Αυτές οι υπηρεσίες επικοινωνούν μεταξύ τους μέσω της ανταλλαγής δεδομένων ή μέσω του συντονισμού δραστηριοτήτων μεταξύ δύο ή παραπάνω υπηρεσιών. Συχνά θεωρείται ότι οι αρχές της SOA έχουν δομηθεί και αποτελούν εξέλιξη των αρχών των κατανεμημένων συστημάτων (distributed computing) και του αρθρωτού προγραμματισμού.

Σύμφωνα με τα παραπάνω ένας ορισμός που μπορεί να δοθεί για την Service-Oriented Architecture είναι ο εξής:

"Η SOA αναφέρεται σε μι*α ομάδα υπηρεσιών που μπορούν να επικοινωνούν μεταξύ τους. Η διαδικασία της επικοινωνίας περιλαμβάνει είτε απλή ανταλλαγή δεδομένων ή μπορεί και να*

περιλαμβάνει δύο ή περισσότερες υπηρεσίες που συντονίζουν μια δραστηριότητα. Για αυτή τη λειτουργία είναι απαραίτητη η ύπαρξη κάποιου τρόπου διασύνδεσης των υπηρεσιών."

Στην επόμενη εικόνα (Εικόνα 1) παρουσιάζεται μια βασική προσανατολισμένη σε Υπηρεσίες Αρχιτεκτονική. Φαίνεται ένας καταναλωτής ή χρήστης ή πελάτης της υπηρεσίας, ο οποίος αποστέλλει στον πάροχο της υπηρεσίας ένα μήνυμα αίτησης υπηρεσίας. Ο πάροχος επιστρέφει με τη σειρά του ένα μήνυμα απάντησης στον καταναλωτή. Οι συνδέσεις αιτήσεων και απαντήσεων ορίζονται με τέτοιο τρόπο, ώστε τα μηνύματα να είναι κατανοητά τόσο από τον καταναλωτή της υπηρεσίας, όσο και από τον πάροχο. Ο πάροχος μπορεί να λειτουργήσει και αυτός ως καταναλωτής κάποιας υπηρεσίας.



Εικόνα 1. Service-oriented architecture.

Η SOA αποτελεί μια αφαιρετική αρχιτεκτονική έννοια, ενώ οι υπηρεσίες διαδικτύου ή Web Services [48] αποτελούν μια συγκεκριμένη υλοποίηση της [25]. Είναι μια προσέγγιση για την κατασκευή συστημάτων λογισμικού που βασίζεται στη χρήση ελαφρώς συσχετισμένων δομικών συστατικών (components) που έχουν περιγραφθεί με ενιαίο τρόπο και τα οποία μπορούν να ανακαλυφθούν και να συνθεθούν ή να συνδυαστούν με άλλα. Σύμφωνα με το World Wide Web Consortium (W3C) [50] μια υπηρεσία διαδικτύου μπορεί να οριστεί ως:

"Ένα σύστημα λογισμικού σχεδιασμένο για να υποστηρίζει διαλειτουργικότητα στην αλληλεπίδραση μηχανής με μηχανή πάνω από κάποιο δίκτυο. Έχει μια διεπαφή που περιγράφεται σε μια μορφή που μπορεί να την επεξεργαστεί μια μηχανή (που συγκεκριμένα ονομάζεται WSDL). Άλλα συστήματα αλληλεπιδρούν με την υπηρεσία διαδικτύου με έναν τρόπο προκαθορισμένο από την περιγραφή της κάνοντας χρήση SOAP μηνυμάτων που μεταφέρονται μέσω του πρωτοκόλλου HTTP [46] *σε σειριοποίηση Extensible Markup Language / XML* [18] *σε συνδυασμό με άλλα σχετικά με το διαδίκτυο πρότυπα."*

Αν και η τεχνολογία των υπηρεσιών διαδικτύου δεν είναι η μόνη προσέγγιση για την υλοποίηση της SOA, είναι η επικρατέστερη και έχει υιοθετηθεί ευρέως από τη βιομηχανία. Με τις υπηρεσίες διαδικτύου η βιομηχανία αντιμετωπίζει και πάλι τη βασική πρόκληση που έχει προκύψει από τα κατανεμημένα συστήματα: την παροχή ενός ενιαίου τρόπου για την περιγραφή δομικών συστατικών ή υπηρεσιών μέσα σε ένα δίκτυο, τον εντοπισμό τους και

την προσπέλασή τους. Η διαφορά μεταξύ των υπηρεσιών διαδικτύου και άλλων προσεγγίσεων (π.χ. τεχνολογίες κατανεμημένων αντικειμένων, όπως οι Common Object Request Broker Architecture / CORBA [11] και Microsoft Distributed Component Object Model / DCOM [16]) έγκειται στην έννοια της "χαλαρής" συσχέτισης της αρχιτεκτονικής τους. Αντί να δημιουργούνται εφαρμογές που οδηγούν σε άρρηκτα ενσωματωμένες συλλογές αντικειμένων ή δομικών συστατικών, η όλη προσέγγιση είναι πολύ πιο δυναμική και εύκολα προσαρμόσιμη σε αλλαγές.

Η τεχνολογία των υπηρεσιών διαδικτυού μπορεί να χρησιμοποιηθεί με διάφορους τρόπους. Οι υπηρεσίες διαδικτυού μπορούν να χρησιμοποιηθούν σε σταθερές και κινητές συσκευές για την πρόσβαση σε εφαρμογές διαδικτύου, όπως συστήματα κρατήσεων και παρακολούθησης παραγγελιών. Μπορούν ακόμα να χρησιμοποιηθούν στην ενσωμάτωση διεπιχειρησιακών εφαρμογών (B2B) [7] και τη σύνδεση εφαρμογών που διευθύνονται από διάφορους οργανισμούς στην ίδια εφοδιαστική αλυσίδα. Επιπλέον, οι υπηρεσίες διαδικτυού μπορούν να βοηθήσουν στην επίλυση του γενικότερου προβλήματος της ενσωμάτωσης επιχειρησιακών εφαρμογών (Enterprise Application Integration / EAI) συνδέοντας πολλαπλές εφαρμογές από έναν οργανισμό σε άλλους. Σε όλες αυτές τις περιπτώσεις οι τεχνολογίες των υπηρεσιών διαδικτύου παρέχουν τα πρότυπα για τη διασύνδεση των ποικίλων κομματιών λογισμικού. Στην παρούσα διατριβή η χρήση των υπηρεσιών διαδικτύου στην οποία εστιάζουμε σε σχέση με το πλαίσιο χρήσης είναι κυρίως η πρώτη από τις προαναφερόμενες.

Διάφορα βιβλία και άρθρα για τη χρήση των υπηρεσιών διαδικτύου που απευθύνονται σε προγραμματιστές, ερευνητές και άλλους χρήστες έχουν εκδοθεί (π.χ. [44], [2]), ενώ λεπτομέρειες για τα βασικά πρωτόκολλα που χρησιμοποιούνται από τις υπηρεσίες διαδικτύου (SOAP, WSDL και UDDI) δίνονται στο επόμενο Κεφάλαιο.

1.3.2 Ορισμός Πλαισίου Χρήσης

Το πλαίσιο χρήσης μιας εφαρμογής ή contextual information ή context ,όπως συχνά αναφέρεται, σχετίζεται με διάφορα χαρακτηριστικά που αφορούν το χρήστη, τη συσκευή που χρησιμοποιεί, την ίδια την υπηρεσία ή εφαρμογή που εκτελείται και το περιβάλλον εκτέλεσης. Στη βιβλιογραφία μπορεί κανείς να συναντήσει διάφορους ορισμούς που έχουν δοθεί κατά καιρούς (π.χ. [36]). Ωστόσο, είναι δύσκολο να βρεθεί κάποιος αρκετά γενικός ορισμός που να καλύπτει όλες τις πιθανές περιπτώσεις αναφοράς στο πλαίσιο χρήσης που υπάρχουν στις διάφορες ερευνητικές προσπάθειες. Έτσι, εδώ αναφερόμαστε στο πλαίσιο χρήσης που σχετίζεται με τις υπηρεσίες και τις εφαρμογές του κινητού διαδικτύου.

Παρατηρείται πως οι ορισμοί που σχετίζονται με το κινητό διαδίκτυο έχουν εξελιχθεί στη διάρκεια του χρόνου και κάποιοι συγκεκριμένοι με τις παραλλαγές τους έχουν επικρατήσει περισσότερο στην ερευνητική κοινότητα.

Αρχικά, το πλαίσιο χρήσης ως έννοια προέκυψε από τις υπηρεσίες που προσαρμόζονται στη θέση του χρήστη, αναφερόμενες ως location-based υπηρεσίες. Στη συνέχεια, ο όρος εμπλουτίστηκε με άλλα χαρακτηριστικά στα οποία μπορεί να προσαρμοστεί η υπηρεσία. Οι πρώτοι ορισμοί που δίνονταν αναφέρονταν είτε σε απαρίθμηση παραδειγμάτων του πλαισίου χρήσης είτε σε χρήση συνώνυμων λέξεων, όπως για παράδειγμα στο [38], όπου το πλαίσιο χρήσης χωρίζεται στις εξής κατηγορίες:

- *Υπολογιστικό πλαίσιο χρήσης*, όπως η συνδεσιμότητα του δικτύου, το κόστος επικοινωνίας, το έυρος ζώνης και οι διάφορες κοντινές συσκευές (π.χ. εκτυπωτές, οθόνες και σταθμοί εργασίας).
- *Πλαίσιο χρήσης του χρήστη*, όπως πληροφορίες για το χρήστη (προφίλ χρήστη), θέση του χρήστη, άτομα κοντά στο χρήστη, μέχρι και η κοινωνική του κατάσταση.
- *Φυσικό πλαίσιο χρήσης*, όπως ο φωτισμός, τα επίπεδα θορύβου, οι συνθήκες κίνησης και η θερμοκρασία.

Από τους πρώτους πιο γενικούς όρους που προέκυψαν είναι αυτός που δόθηκε στο [39], όπου το πλαίσιο χρήσης αναφέρεται ως έννοια που σχετίζεται με:
"*Τη θέση χρήσης, το σύνολο των κοντινών ανθρώπων και αντικειμένων, καθώς και τις αλλαγές σε αυτά τα αντικείμενα*."

Από τότε έχει προκύψει μια πληθώρα άλλων ορισμών, άλλοι πιο ειδικοί με εφαρμογή σε συγκεκριμένες εφαρμογές, όπως το e-note [5] όπου το πλαίσιο χρήσης είναι: "*Συνδυασμός στοιχείων του περιβάλλοντος για τα οποία γνωρίζει ο υπολογιστής του χρήστη"* και άλλοι με μια προσπάθεια γενίκευσης, όπως ο όρος [40]: "*Πλαίσιο χρήσης είναι αυτό που περιβάλλει και δίνει σημασία σε κάτι άλλο*", ενώ ένας παρόμοιος ορισμός με τον δεύτερο ορισμό του Schilit δίνεται στο [6] εμπλουτίζοντας τον με στοιχεία όπως η ώρα, η εποχή, η θερμοκρασία, κτλ.

Συνολικά το context αποτελείται από τη *φυσική, κοινωνική, συναισθηματική και υπολογιστική κατάσταση του χρήστη* [15]. Συχνά ως συνώνυμο του πλαισίου χρήσης χρησιμοποιείται και "*η πληροφορία περιβάλλοντος" ή* "τα στοιχεία του *περιβάλλοντος"*. Στην πλειοψηφία των ορισμών εμφανίζεται και η συσχέτιση με την εφαρμογή και τη λειτουργία της. Κάποια τέτοια παραδείγματα ορισμών που έχουν δοθεί την ίδια χρονική περίοδο και αναφέρονται στο πλαίσιο χρήσης της εφαρμογής είναι τα εξής:

"*Η πληροφορία περιβάλλοντος ή το πλαίσιο χρήσης περιλαμβάνει πληροφορίες που είναι κομμάτι του περιβάλλοντος λειτουργίας μιας εφαρμογής και που μπορούν να ανιχνευθούν από την εφαρμογή. Αυτό τυπικά περιλαμβάνει τη θέση, την ταυτότητα, τη δραστηριότητα και την κατάσταση ανθρώπων, ομάδων και αντικειμένων. Το πλαίσιο χρήσης μπορεί ακόμα να σχετίζεται με τοποθεσίες ή με το υπολογιστικό περιβάλλον. (...) Τέλος, μια εφαρμογή μπορεί να χρησιμοποιήσει το περιβάλλον λογισμικού και υλικού για να ανιχνεύσει για παράδειγμα τις δυνατότητες των κοντινών σε αυτή πόρων.*" [37]

"*Πλαίσιο χρήσης είναι το σύνολο των καταστάσεων και των χαρακτηριστικών του περιβάλλοντος που καθορίζει τη συμπεριφορά μιας εφαρμογής ή που μέσα στο οποίο συμβαίνουν τα γεγονότα μιας εφαρμογής και που ενδιαφέρει το χρήστη*." [9]

"Τ*ο πλαίσιο χρήσης αποτελεί έναν μηχανισμό που βοηθά τις εφαρμογές να εκτελέσουν ενέργειες για λογαριασμό του χρήστη με αυτόνομο και εύκαμπτο τρόπο. Έτσι, αποτελείται από το σύνολο των συνθηκών που περιβάλλουν την ενέργεια και που είναι πιθανόν σχετικές με την ολοκλήρωσή της.*" [20]

Διάφορους όρους για το πλαίσιο χρήσης μπορεί να συναντήσει κανείς και στο διαδίκτυο, όπως στο λεξικό Merriam-Webster's Collegiate, όπου αναφέρεται ως "*Οι αλληλοσυσχετισμένες συνθήκες στις οποίες κάτι υπάρχει ή συμβαίνει*" [24] και στο λεξικό του [8]: "*Οι συνθήκες σχετικές με κάτι που βρίσκεται υπό θεώρηση*".

Σημαντικό για το πλαίσιο χρήσης είναι και ο συνδυασμός των τιμών των πληροφοριών που περιλαμβάνει. Κάτι τέτοιο μπορεί να οδηγήσει σε καλύτερη κατανόηση της τρέχουσας κατάστασης της εφαρμογής. Έτσι, βασικά στοιχεία του πλαισίου χρήσης, όπως είναι η θέση, η οντότητα, η δραστηριότητα και ο χρόνος μπορούν να λειτουργήσουν ως δείκτες για άλλες πηγές πλαισίου χρήσης. Για παράδειγμα, γνωρίζοντας την τρέχουσα θέση και την ώρα μαζί με το ημερολόγιο του χρήστη μπορεί κανείς να εξάγει παραπάνω συμπεράσματα για το πλαίσιο χρήσης σχετικά π.χ. με προγραμματισμένες συναντήσεις του χρήστη, αναμονή στο αεροδρόμιο, κτλ.

Σε άλλες περιπτώσεις (όπως και στο [38]) οι ερευνητές επιχειρούν να κατηγοριοποιήσουν το πλαίσιο χρήσης που μπορεί για παράδειγμα να διακριθεί σε *ενεργό*, το οποίο επηρεάζει τη συμπεριφορά μιας εφαρμογής, και σε *παθητικό*, το οποίο είναι μεν σχετικό με την εφαρμογή, αλλά όχι κρίσιμο για αυτή [9].

Ωστόσο, ο πιο συχνά αναφερόμενος ορισμός έχει δοθεί από τους Dey και Abowd [14] [13], οι οποίοι περιγράφουν το πλαίσιο χρήσης ως:

"Πλαίσιο χρήσης είναι οποιαδήποτε πληροφορία μπορεί να χρησιμοποιηθεί για να χαρακτηρίσει την κατάσταση μιας οντότητας. Μια οντότητα μπορεί να είναι ένα άτομο, ένα μέρος ή ένα αντικείμενο που θεωρείται σχετικό με την αλληλεπίδραση μεταξύ ενός χρήστη και μιας εφαρμογής, συμπεριλαμβανομένων του χρήστη και της εφαρμογής."

Στη διατριβή έχει ληφθεί υπόψη κυρίως ο τελευταίος ορισμός, ο οποίος είναι και αυτός που μνημονεύεται από την πλειοψηφία των σχετικών με το πλαίσιο χρήσης επιστημονικών άρθρων από το 2001 και μετά. Ωστόσο, ένας ορισμός που μπορεί να προκύψει συμπερασματικά από τα παραπάνω και εμπεριέχει και την έννοια του πλαισίου χρήσης που χρησιμοποιείται στο σύστημα διαχείρισης του πλαισίου χρήσης της διατριβής είναι ο εξής:
"Το πλαίσιο χρήσης αναφέρεται σε οποιαδήποτε πληροφορία σχετίζεται με τον χρήστη μιας υπηρεσίας, το περιβάλλον, στο οποίο κινείται, και τις δραστηριότητες, στις οποίες εμπλέκεται. Σχετίζεται με οποιοδήποτε χαρακτηριστικό μπορεί να χρησιμοποιηθεί για να κάνει τη χρήση της εφαρμογής πιο προσωποποιημένη και σχετική με το περιβάλλον χρήσης ή εκτέλεσης."

Έτσι, το πλαίσιο χρήσης συνδέεται κυρίως με το χρήστη και κάθετι που τον αφορά και είναι επιπλέον σχετικό με την εκτέλεση της εφαρμογής.

1.3.3 Ορισμός Επίγνωσης Πλαισίου Χρήσης

Οι τεχνικές που καθιστούν δυνατή τη χρήση της σχετικής με το πλαίσιο χρήσης πληροφορίας αναφέρονται ως "τεχνικές διαχείρισης του πλαισίου χρήσης". Ο όρος επίγνωση πλαισίου χρήσης ή context-awareness χρησιμοποιείται για αναφορά σε υπηρεσίες, εφαρμογές και συστήματα που λαμβάνουν υπόψη το πλαίσιο χρήσης και προσαρμόζουν τη λειτουργία τους ανάλογα. Διάφοροι ορισμοί έχουν δοθεί κατά καιρούς για την "επίγνωση του πλαισίου χρήσης" και για τις "εφαρμογές με επίγνωση του πλαισίου χρήσης". Και πάλι παρατηρείται πως πολλοί από αυτούς τους ορισμούς είναι συνδεδεμένοι με συγκεκριμένες εφαρμογές και συστήματα (π.χ. [5]).

Σε κάποιους ορισμούς που έχουν δοθεί έχει ειπωθεί πως "*επίγνωση του πλαισίου χρήσης είναι η ικανότητα της εφαρμογής του κινητού χρήστη να ανακαλύπτεί και να αντιδρά στις αλλαγές του περιβάλλοντος, στο οποίο βρίσκεται.*" [39], ενώ για να έχει μια υπηρεσία επίγνωση του πλαισίου χρήσης "*πρέπει να έχει επίγνωση του τρέχουντος πλαισίου χρήσης του χρήστη και να προσαρμόζεται μόνη της στις αλλαγές του*" [35] ή *"να*

παρακολουθεί τις αλλαγές στο περιβάλλον και να τροποποιείται βάσει προκαθορισμένων ή καθορισμένων από το χρήστη κανόνων" [19].

Οι παραπάνω ορισμοί αναφέρονται περισσότερο στην ικανότητα παρακολούθησης της κατάστασης του πλαισίου χρήσης και αυτοπροσαρμογής μιας εφαρμογής σε αυτή. Αυτό σημαίνει πως οι εφαρμογές θα πρέπει να παρακολουθούν τις αλλαγές που συμβαίνουν και να αλλάζουν τη συμπεριφορά τους ανάλογα, να είναι δηλαδή προσαρμόσιμες ή *αυτόματα αντιδραστικές και ευαίσθητες* στο πλαίσιο χρήσης [12]. Αυτό όμως δε συμβαίνει πάντα και μπορούμε να πούμε πως οι συγκεκριμένοι ορισμοί αναφέρονται σε συγκεκριμένου τύπου συστήματα. Στις περισσότερες περιπτώσεις μας ενδιαφέρει να μπορούν οι εφαρμογές να προσαρμόζουν τη λειτουργία τους στο πλαίσιο χρήσης όταν εκτελούνται χωρίς να απαιτείται η συνεχής παρακολούθησή του από την ίδια την εφαρμογή.

Πράγματι κάποιοι άλλοι ορισμοί αναφέρονται στην προσαρμογή της υπηρεσίας και όχι σε αλλαγές του περιβάλλοντος. Κάποια παραδείγματα είναι τα παρακάτω:

"*Το λογισμικό με επίγνωση του πλαισίου χρήσης προσαρμόζεται στη θέση χρήσης, τις ομάδες των κοντινών ατόμων, τους εξυπηρετητές και τις προσβάσιμες συσκευές, καθώς και σε αλλαγές αυτών των αντικειμένων."* [38]

Μια εφαρμογή ή ένας υπολογιστής με επίγνωση του πλαισίου χρήσης "*αλλάζει τη συμπεριφορά του ανάλογα με το πώς και πού χρησιμοποιείται. Πρέπει να είναι σε θέση να καθορίσει την κατάσταση του περιβάλλοντός του.*" [47]

"*Επίγνωση του πλαισίου χρήσης είναι η ικανότητα ενός προγράμματος ή μιας συσκευής να διαισθάνεται διάφορες καταστάσεις στο περιβάλλον του και στον εαυτό του.*" [34]

"*Εφαρμογές με επίγνωση του πλαισίου χρήσης αισθάνονται το πλαίσιο χρήσης και τροποποιούν τη συμπεριφορά τους ανάλογα χωρίς τη ρητή μεσολάβηση του χρήστη.*" [37]

Ο πιο γενικός, αλλά μάλλον όχι τόσο επεξηγηματικός ορισμός έχει δοθεί από τους Dey και Abowd στο [14]: "*Η χρήση του πλαισίου χρήσης για την παροχή σχετικών πληροφοριών και υπηρεσιών στο χρήστη, όπου η καταλληλότητα εξαρτάται από τη δραστηριότητα του χρήστη.*"

Όσον αφορά τις εφαρμογές με επίγνωση του πλαισίου χρήσης στο [13] επιχειρείται να συνδυαστούν οι ιδέες προηγούμενων κατηγοριοποιήσεων και προσπαθειών σε μια γενικευμένη έννοια που ικανοποιεί όλες τις πιθανές εφαρμογές με επίγνωση του πλαισίου χρήσης. Συγκεκριμένα, γίνεται αναφορά σε τρεις κατηγορίες ή χαρακτηριστικά που μπορεί να υποστηρίζει μια εφαρμογή με επίγνωση του πλαισίου χρήσης:

- Παρουσίαση των πληροφοριών και των υπηρεσιών σε ένα χρήστη
- Αυτόματη εκτέλεση μιας εφαρμογής για ένα χρήστη
- Τοποθέτηση ετικετών (tagging) στην πληροφορία πλαισίου χρήσης για μετέπειτα ανάκτηση

Στη διατριβή η έννοια που χρησιμοποιείται για τις εφαρμογές με επίγνωση του πλαισίου χρήσης είναι η εξής:

"*Εφαρμογές με επίγνωση του πλαισίου χρήσης είναι οι εφαρμογές ή ακόμα οι υπηρεσίες και τα συστήματα που λαμβάνουν υπόψη το πλαίσιο χρήσης κατά την εκτέλεση τους και προσαρμόζουν τη λειτουργία τους σε αυτή την πληροφορία προκαταβολικά.*"

Διάφορες εφαρμογές προσαρμοσμένες στο πλαίσιο χρήσης έχουν αναπτυχθεί και δοκιμαστεί από τους τελικές χρήστες σε πειραματικά, αλλά και πραγματικά περιβάλλοντα, και συνήθως σχετίζονται με συγκεκριμένες περιπτώσεις εφαρμογής [35].

1.3.4 Μοντελοκεντρική Ανάπτυξη

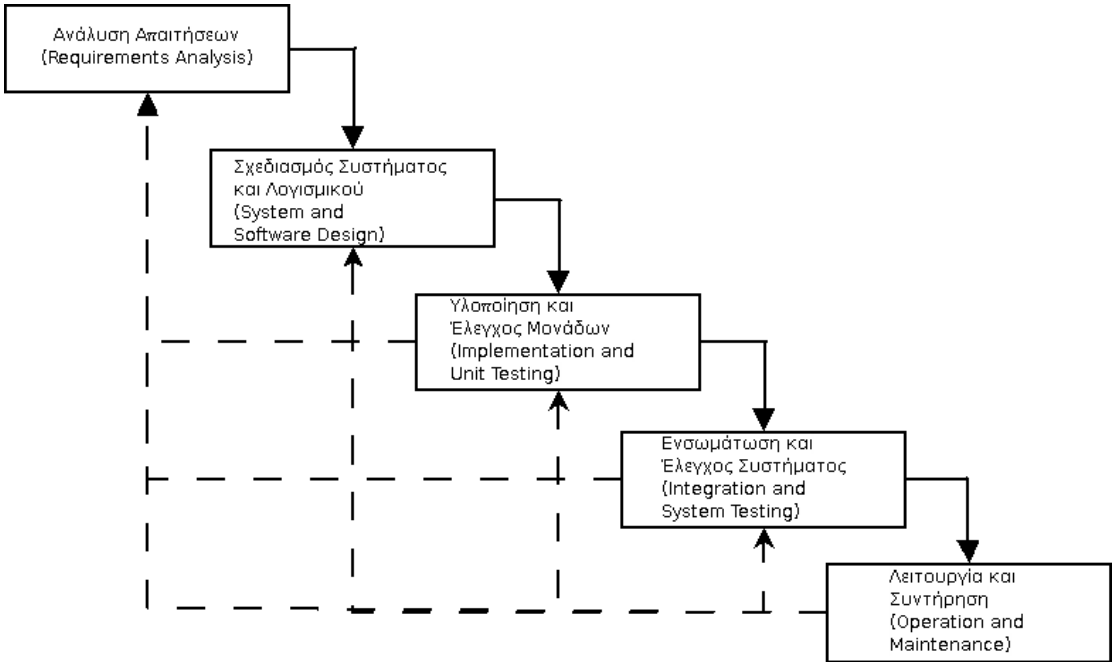
1.3.4.1 Ανάπτυξη λογισμικού

Η ανάπτυξη λογισμικού και εφαρμογών καθίσταται όλο και πιο πολύπλοκη ως διαδικασία λόγω της πολυπλοκότητας των προς ανάπτυξη συστημάτων που απαιτούνται στις μέρες μας. Παρατηρείται πως κατά καιρούς διάφοροι μέθοδοι ανάπτυξης έχουν προταθεί από την Τεχνολογία Λογισμικού ή Μηχανική Λογισμικού (Software Engineering) για να βοηθήσουν στην ανάπτυξη και να καθορίσουν συγκεκριμένα βήματα που μπορούν να διευκολύνουν το έργο του μηχανικού λογισμικού. Το πιο γνωστό από αυτά τα μοντέλα είναι το μοντέλο του καταρράκτη (waterfall model) που εμφανίστηκε περί το 1970 σε ένα άρθρο του Winston W. Royce [49] (Εικόνα 2).

Έκτοτε διάφορες άλλες μεθοδολογίες έχουν προταθεί, όπως οι agile μεθοδολογίες [1] [10] που χαρακτηρίζονται από μικρούς και διακριτούς κύκλους ανάπτυξης με σαφώς καθορισμένα αποτελέσματα και διαρκή έλεγχο του λογισμικού. Αν και οι agile μεθοδολογίες έχουν υιοθετηθεί ευρέως, η βασικότερη μεθοδολογία αναφοράς παραμένει το μοντέλο του καταρράκτη με τα πλεονεκτήματα και μειονεκτήματα που παρουσιάζει.

Ειδικά τα τελευταία χρόνια παρατηρείται μια εκρηκτική ανάπτυξη της βιομηχανίας παραγωγής λογισμικού και ανάπτυξης εφαρμογών παγκοσμίως. Η ανάπτυξη αυτής της βιομηχανίας έχει βάλει στο επίκεντρο του ενδιαφέροντος το ζήτημα της διαλειτουργικότητας και της ανεξαρτησίας από συγκεκριμένες υλοποιήσεις. Μάλιστα, από παλαιότερα γίνονταν προσπάθειες αφαιρετικότητας στις γλώσσες προγραμματισμού και

στις διαδικασίες υλοποίησης για να διευκολυνθεί η διαδικασία ανάπτυξης. Η παλαιότερη ίσως από αυτές τις προσπάθειες έγινε με τη γλώσσα μηχανής (assembly), όπου ο προγραμματιστής μπορούσε να γράψει προγράμματα στη γλώσσα μηχανής και η μηχανή να ακολουθήσει τις εκάστοτε οδηγίες. Το επόμενο μεγάλο βήμα ήταν οι γλώσσες προγραμματισμού, που επέτρεψαν την ανάπτυξη προγραμμάτων σε μια γλώσσα ευκολότερα κατανοητή από τον άνθρωπο. Καθορίστηκαν πρότυπα για διάφορες γλώσσες με αποτέλεσμα να είναι δυνατή η χρησιμοποίησή τους σε πολλές διαφορετικές πλατφόρμες, ενώ η ανάπτυξη, κατανόηση και συντήρηση των προγραμμάτων έγιναν πιο εύκολες διαδικασίες.



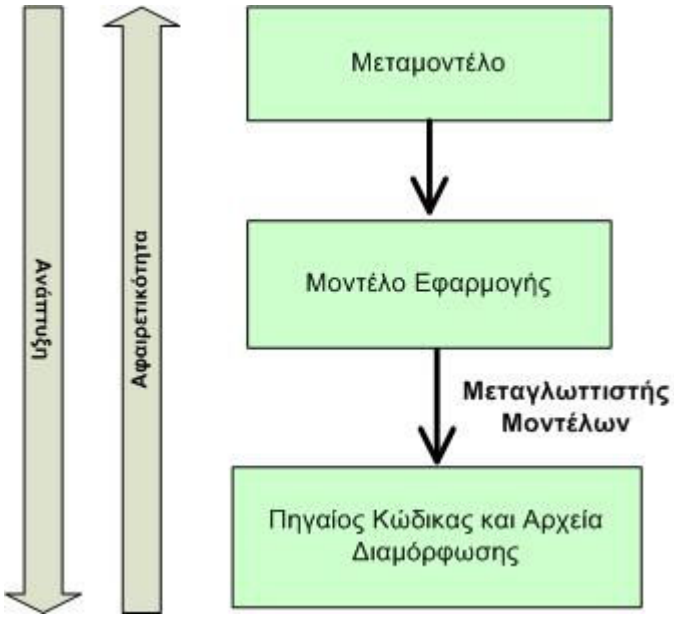
Εικόνα 2. Το μοντέλο του καταρράκτη.

Αυτό που προσπαθούν τώρα να προσφέρουν οι νέες γλώσσες προγραμματισμού σε σχέση με τις παλαιότερες είναι ένα ακόμα υψηλότερο επίπεδο αφαιρετικότητας, καθώς στις μέρες μας υπάρχει έντονη η ανάγκη της διαλειτουργικότητας των συστημάτων, αλλά και της μετατόπισης σε τρόπους προγραμματισμού που ενστερνίζονται όλο και περισσότερο αφαιρετικές μεθοδολογίες.

1.3.4.2 Μοντελοκεντρική ανάπτυξη

Για να μπορεί μια εφαρμογή να αναπτυχθεί με όσο το δυνατόν λιγότερα βήματα επαναχρησιμοποιώντας υπάρχουσες υλοποιήσεις η προσοχή έχει στραφεί προς τη Μοντελοκεντρική Ανάπτυξη (Εικόνα 3). Με τη μοντελοκεντρική ανάπτυξη η αρχική ανάπτυξη του μοντέλου της εφαρμογής παραμένει επαρκώς ανεξάρτητη από την

πλατφόρμα λογισμικού που θα χρησιμοποιηθεί. Από τα παραπάνω έχει προκύψει και η Μοντελοκεντρική Μηχανική (Model-Driven Engineering / MDE) [41], η συστηματική δηλ. χρήση μοντέλων ως τεχνολογικά αντικείμενα πρώτης τάξης για ολόκληρο τον κύκλο ζωής ενός τεχνικού έργου. Η Μοντελοκεντρική Μηχανική βασίζεται στις παραδοχές πως κάθε σύστημα μπορεί να αναπαρασταθεί από μοντέλα και πως κάθε μοντέλο υπακούει συντακτικά σε κάποιο μεταμοντέλο [3].

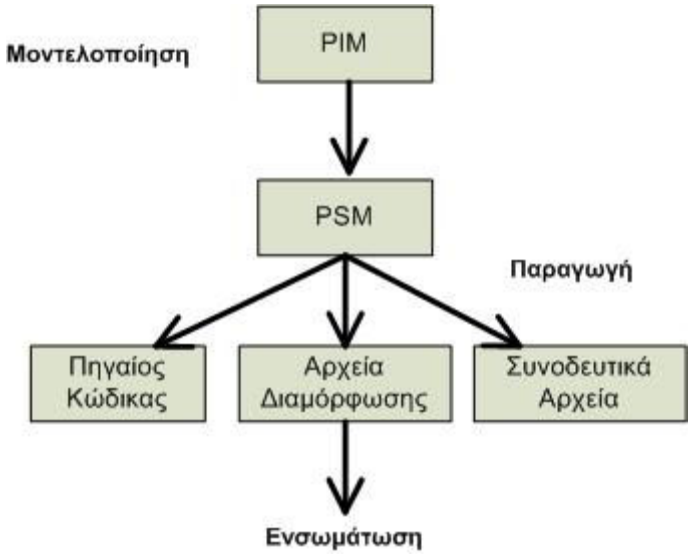


Εικόνα 3. Αφαιρετικότητα στη μοντελοκεντρική ανάπτυξη.

Ένας από τους κύριους στόχους της μοντελοκεντρικής ανάπτυξης λογισμικού είναι ο διαχωρισμός της σχεδίασης από την αρχιτεκτονική. Καθώς οι αρχές και οι τεχνολογίες που χρησιμοποιούνται για τη σχεδίαση των εφαρμογών και οι τεχνολογίες που χρησιμοποιούνται για την υλοποίηση των αρχιτεκτονικών μεταβάλλονται ανεξάρτητα, η χαλαρή τους σύνδεση επιτρέπει την επιλογή της καλύτερης λύσης σε κάθε περιοχή κατά την ανάπτυξη συστημάτων.

Η μοντελοκεντρική ανάπτυξη σχετίζεται συχνά με τη Μοντελοκεντρική Αρχιτεκτονική (Model-driven Architecture / MDA) [29] [43] που προτάθηκε από την Ομάδα Διαχείρισης Αντικειμένων (Object Management Group / OMG) [27] το 2001 και αποτελεί τη σημαντικότερη ίσως μοντελοκεντρικη προσέγγιση στο πεδίο της τεχνολογίας λογισμικού. Η μοντελοκεντρική αρχιτεκτονική παρέχει ένα σύνολο κατευθυντήριων γραμμών για την κατασκευή προδιαγραφών που εκφράζονται ως μοντέλα και υποστηρίζει τη μοντελοκεντρική ανάπτυξη συστημάτων λογισμικού. Η OMG με τη μοντελοκεντρική αρχιτεκτονική εστιάζει στην ευθεία μηχανική (forward engineering), δηλ. στην παραγωγή κώδικα από αφαιρετικές προδιαγραφές που καθορίζονται από ανθρώπους.

Σύμφωνα με τη μοντελοκεντρική αρχικτεκτονική [42] η ανάπτυξη της εφαρμογής αρχίζει από τη δημιουργία του ανεξάρτητου από την πλατφόρμα μοντέλου (Platform Independent Model / PIM) (Εικόνα 4) που εκφράζεται συνήθως στη γλώσσα Unified Modeling Language (UML) [32]. Το PIM εκφράζει μόνο τη λειτουργία και τη συμπεριφορά της εφαρμογής και περιλαμβάνει και κάποια στοιχεία τεχνολογίας, αλλά λείπουν λεπτομέρειες σχετικές με την πλατφόρμα υλοποίησης. Στη συνέχεια δημιουργείται ή παράγεται το ειδικό για την πλατφόρμα μοντέλο (Platform Specific Model / PSM). Το PSM ορίζει τις δομές που θα χρησιμοποιηθούν στην πλατφόρμα που θα πραγματοποιηθεί η υλοποίηση. Ένα PIM μπορεί να μετατραπεί σε παραπάνω από ένα PSM, αν χρησιμοποιούνται παραπάνω από μία τεχνολογικές πλατφόρμες για την εφαρμογή.



Εικόνα 4. Βήματα ανάπτυξης εφαρμογών στη μοντελοκεντρική αρχιτεκτονική.

Η μοντελοκεντρική ανάπτυξη οραματίζεται ότι το μοντέλο που υλοποιεί τις λειτουργικές απαιτήσεις της εφαρμογής θα παραμείνει αναλλοίωτο και ανεξάρτητο από αλλαγές στις τεχνολογίες υλοποίησης. Στοχεύει στον ορισμό γλωσσών μοντελοποίησης για συγκεκριμένες περιοχές ενδιαφέροντος και υποστηρίζει μετασχηματισμούς μεταξύ μετα-μοντέλων και την ημι-αυτόματη ή πλήρως αυτόματη παραγωγή εκτελέσιμου κώδικα από το μοντέλο του συστήματος. Η διαδικασία ανάπτυξης είναι μοντελοκεντρική υπό την έννοια ότι εστιάζει στο μοντέλο της εφαρμογής που κατευθύνει την παραγωγή του κώδικά.

Στην περίπτωση της μοντελοκεντρικής αρχιτεκτονικής ιδιαίτερη σημασία έχει η αρχή της μετατροπής μοντέλων (model transformation). Μια συγκεκριμένη πρότυπη γλώσσα για το μετασχηματισμό μοντέλων έχει οριστεί από την OMG με το όνομα Ερώτημα/Όψη/Μετασχηματισμός (Query/View/Transformation / QVT) [31]. Η μοντελοκεντρική αρχιτεκτονική σχετίζεται με διάφορα άλλα πρότυπα, όπως τα: Meta-

Object Facility (MOF) [30], XML Metadata Interchange (XMI) [33] και Enterprise Distributed Object Computing (EDOC) [28]. Ο όρος “αρχιτεκτονική” μάλιστα δεν αναφέρεται τόσο στην αρχιτεκτονική του συστήματος που μοντελοποιείται, αλλά περισσότερο στην αρχιτεκτονική των διαφόρων προτύπων και μοντέλων που αποτελούν την τεχνολογική βάση για την MDA.

Τα βασικότερα πλεονεκτήματα που προσδίδει η μοντελοκεντρική ανάπτυξη στην ανάπτυξη εφαρμογών είναι τα εξής:

- Αφαιρετική διαδικασία ανάπτυξης
- Ευελιξία στις δυνατότητες επαναπρογραμματισμού
- Ανεξαρτησία των αρχικών σταδίων ανάπτυξης από συγκεκριμένες υλοποιήσεις
- Επιτάχυνση της ανάπτυξης μέσω της παραγωγής κώδικα
- Μη απαίτηση για πλήρη γνώση της τεχνολογίας υλοποίησης
- Πιο εύκολη αντικατάσταση της τεχνολογίας υλοποίησης

Ωστόσο, για να μπορεί να υιοθετηθεί η μοντελοκεντρική ανάπτυξη απαιτείται η γνώση των βασικών της αρχών και εργαλείων. Έχουν αναπτυχθεί διάφορα εργαλεία και πλατφόρμες που διευκολύνουν την ανάπτυξη συστημάτων με τις αρχές της μοντελοκεντρικής αρχιτεκτονικής, ενώ πολλά βιβλία παρουσιάζουν εκτενώς τις αρχές της [22] [23] [45]. Η μοντελοκεντρική ανάπτυξη έχει εφαρμογή σε πολλές περιπτώσεις συστημάτων και πολλοί ερευνητές την έχουν ενσωματώσει στις προσπάθειες τους σε διάφορα ερευνητικά πεδία (π.χ. MD Security [4]).

Αν και μια πλήρης υλοποίηση της προδιαγραφής της μοντελοκεντρική αρχιτεκτονικής περιλαμβάνει και τα τρία στάδια, για τεχνολογίες ή προδιαγραφές, για τις οποίες μπορεί να προκύψει ένα κοινό μεταμοντέλο, παρατηρείται συχνά η χρησιμοποίηση του μεταμοντέλου ως αρχικό στάδιο υλοποίησης ακολουθώντας τις γενικές αρχές της μοντελοκεντρικής μηχανικής. Για την περίπτωση των υπηρεσιών διαδικτύου μπορεί να χρησιμοποιηθεί ένα κοινό μεταμοντέλο για να διευκολύνει την υλοποίηση εφαρμογών που βασίζονται σε υπηρεσίες διαδικτύου. Το μοντέλο της εφαρμογής σε αυτό το στάδιο παραμένει ανεξάρτητο από τη συγκεκριμένη αρχιτεκτονική υλοποίησης υπηρεσιών διαδικτύου. Έτσι, μπορούν να δοθούν διάφορες περιπτώσεις μετατροπής για την παραγωγή κώδικα από το μοντέλο, με κάθε περίπτωση να σχετίζεται με διαφορετική πλατφόρμα υλοποίησης των υπηρεσιών. Αυτή η προσέγγιση έχει ακολουθηθεί και στη διατριβή.

1.4 Βιβλιογραφία

[1] Agile Alliance, http://www.agilealliance.com/.

[2] Barry, D. K., “Web Services and Service-Oriented Architecture: The Savvy Manager's Guide”, Morgan Kaufmann Publishers, 2003.

[3] Bezivin, J., Barbero, M., Jouault, F., “On the applicability scope of model driven engineering”, Proc. Fourth International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES '07), 2007, pp. 3-7.

[4] Breu, R., Hafner, M., Weber, B., Novak, A., “Model Driven Security for Inter-organizational Workflows in e-Government”, E-Government: Towards Electronic Democracy, ISBN 978-3-540-25016-6, 2005, pp. 122-133.

[5] Brown, P. J., “The stick-e document: a framework for creating context-aware applications”, Electronic Publishing, vol. 8(2 & 3), 1995, pp. 259-272.

[6] Brown, P. J., Bovey, J. D. Chen, X. “Context-Aware Applications: From the Laboratory to the Marketplace”. IEEE Personal Communications, vol. 4(5), 1997, pp. 58-64.

[7] Business-to-business Electronic Commerce (B2B), Wikipedia, http://en.wikipedia.org/wiki/Business-to-business_electronic_commerce.

[8] Cern Engineering Data Management Service, cedar.web.cern.ch/CEDAR/glossary.html.

[9] Chen, G., Kotz, D., “A Survey of Context-Aware Mobile Computing Research”, Technical Report TR2000-381, Dartmouth College, 2000.

[10] Cohen, D., Lindvall, M., Costa, P., “An introduction to agile methods. In Advances in Computers”, New York: Elsevier Science, 2004, pp. 1-66.

[11] Common Object Request Broker Architecture (OMG CORBA), http://www.corba.org/.

[12] Cooperstock,J. R., Tanikoshi, K., Beirne, G., Narine, T., Buxton, W. A. S., “Evolution of a reactive environment”, Proc. SIGCHI conference on Human factors in computing systems , 1995, pp. 170-177.

[13] Dey, K., “Understanding and Using Context”, Personal and Ubiquitous Computing archive, vol. 5(1), February 2001, pp. 4-7.

[14] Dey, K., Abowd, G. D., “Towards a Better Understanding of Context and Context-Awareness”, Proc. Workshop on The What, Who, Where, When, and How of Context-

Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI'00), The Hague, The Netherlands, 2000.

[15] Dey, A. K., Abowd, G.D., Wood, A., "CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services", Knowledge-Based Systems, vol. 11, 1999, pp. 3-13.

[16] Distributed Component Object Model (Microsoft DCOM), http://msdn.microsoft.com /en-us/ library/ms809340.aspx.

[17] Erl, T., "Service-Oriented Architecture (SOA): Concepts, Technology, and Design", Prentice Hall Service-Oriented Computing Series, July 2005, ISBN 0-13-185858-0.

[18] Extensible Markup Language (XML), http://www.w3.org/XML/.

[19] Fickas, S., Kortuem, G.; Segall, Z., "Software organization for dynamic and adaptable wearable systems", Proc. First International Symposium on Wearable Computers, 1997, pp. 56-63.

[20] Henricksen, K., "A Framework for Context-Aware Pervasive Computing Applications", PhD thesis, University of Queensland, September 2003.

[21] Kapitsaki, G., Kateros, D. A., Foukarakis, I. E., Prezerakos, G. N., Kaklamani, D. I., Venieris, I. S., "Service Composition: State of the art and future challenges", Proc. 16th IST Mobile and Wireless Communications Summit, Budapest, Hungary, 2007, pp. 1-5.

[22] Kleppe, A., Warmer, J., Bast, W., Warmer, J. B., Watson, A., "MDA Explained: The Model Driven Architecture: Practice and Promise", Addison Wesley, 2003, ISBN: 0-321-19442-X.

[23] Mellor, S. J., Kendall, S., Uhl, A., Weise, D., "MDA Distilled", Addison Wesley Longman Publishing Co., Inc., 2004.

[24] Merriam-Webster's Collegiate Dictionary, Context, http://www.merriam-webster.com/dictionary/context.

[25] Newcomer, E., Lomow, G., "Understanding SOA with Web Services", Addison Wesley. 2005, ISBN 0-321-18086-0.

[26] NTT Docomo i-mode, http://www.nttdocomo.com/services/imode/index.html.

[27] Object Management Group (OMG), http://www.omg.org/.

[28] OMG, “Enterprise distributed Object Computing (EDOC)”, http://www.omg.org/technology/documents/formal/edoc.htm.

[29] OMG, “MDA Guide version 1.0.1.”, 2003, http://www.omg.org/docs/omg/03-06-01.pdf.

[30] OMG, “Meta-Object Facility (MOF) version 1.4”, 2003, http://www.omg.org/docs/formal/02-04-03.pdf.

[31] OMG, “Meta Object Facility (MOF) version 2.0 Query/View/Transformation Specification”, v1.0, April 2008.

[32] OMG, “Unified Modeling Language (UML), Infrastructure, version 2.1.2”, November 2007, http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF.

[33] OMG, “XMI Mapping Specification version 2.1.1”, 2001, http://www.omg.org/docs/formal/07-12-01.pdf.

[34] Pascoe, J., “Adding Generic Contextual Capabilities to Wearable Computers”, Proc. 2nd International Symposium on Wearable Computers, 1998, pp. 92-99.

[35] Pauty, J., Preuveneers, D., Rigole, P., Berbers, Y., “Research Challenges in Mobile and Context-Aware Service Development”, Proc. Future Research Challenges in Software and Services (FRCSS’06), 2006.

[36] Rodden, T., Cheverst, K., Davies, K. Dix, A., “Exploiting Context in HCI Design for Mobile Systems”, Proc. Workshop on Human Computer Interaction with Mobile Devices, 1998.

[37] Salber, D., Dey, A. K., Abowd, G. D., “The Context Toolkit: Aiding the Development of Context-Enabled Applications”, Proc. SIGCHI conference on Human factors in computing systems (CHI’99), 1999, pp. 434-441.

[38] Schilit, B., Adams, N., Want, R., “Context-aware computing applications”, Proc. IEEE Workshop on Mobile Computing Systems and Applications, 1994, pp. 85-90.

[39] Schilit, B. N., Theimer, M. M. “Disseminating active map information to mobile hosts”, IEEE Network, vol. 8(5), September/October 1994, pp. 22-32.

[40] Schmidt, A., Beigl, M., Gellersen, H.-W, “There is more to context than location”, Computers and Graphics, vol. 23(6), 1999, pp. 893-901.

[41] Schmidt, D. C.,”Model-Driven Engineering”, IEEE Computer, vol. 39(2), Cover Feature, 2006, pp. 25-31.

[42] Siegel, J., OMG Staff Strategy Group, “Developing in OMG’s Model-Driven Architecture”, Object Management Group White Paper, Revision 2.6, November 2001.

[43] Soley, R., OMG Staff Strategy Group, “Model Driven Architecture”, Object Management Group White Paper, Draft 3.2, November 2000.

[44] Topley, K., “Java Web Services in Nutshell”, O'Reilly, 2003.

[45] Trompeter, J., Pietrek, G., Flores Beltran, J. C., Holyer, B., Mork. S. A., Niehues, B., Thoms, K., “Modellgetriebene Softwareentwicklung, MDA und MDSD in der Praxis”, Entwickler Press, 2007, ISBN: 978-3-939084-11-2.

[46] W3C, ”Hypertext Transfer Protocol (HTTP)”, http://www.w3.org/Protocols/.

[47] Ward, A., Jones, A., Hopper, A., “A new location technique for the active office”, IEEE Personal Communications, vol. 4(5), October 1997, pp. 42-47.

[48] Web Services Architecture, W3C, http://www.w3c.org/TR/ws-arch/.

[49] Winston, R. W., "Managing the Development of Large Software Systems", Proc. IEEE WESCON, 1970, pp. 1-9.

[50] World Wide Web Consortium (W3C), http://www.w3.org/.

2ο Κεφάλαιο

Επισκόπηση Τεχνολογιών Διαδικτύου

2.1 Υπηρεσίες Διαδικτύου

Για την επικοινωνία, περιγραφή και ανακάλυψη υπηρεσιών διαδικτύου έχει προταθεί ένα κοινό σύνολο προτύπων που χρησιμοποιούνται από όλα τα εργαλεία υπηρεσιών διαδικτύου. Τα βασικά πρωτόκολλα που χρησιμοποιούνται είναι τα: Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP) και Universal Description, Discovery, and Integration (UDDI), ενώ διάφορα πρωτόκολλα επέκτασης είναι επίσης διαθέσιμα (π.χ. WS-Security [27]). Η γλώσσα που χρησιμοποιείται από όλα τα πρωτόκολλα είναι η XML. Η XML παρέχει την περιγραφή, αποθήκευση και μορφοποίηση για τη μετάδοση των δεδομένων των υπηρεσιών διαδικτύου. Τα παραπάνω βασικά πρωτόκολλα περιγράφονται στις παρακάτω ενότητες.

2.1.1 Web Services Description Language

Σκοπός της γλώσσας Περιγραφής Υπηρεσιών Διαδικτύου (Web Services Description Language / WSDL) [36] είναι η δημιουργία, περιγραφή και δημοσίευση των πρωτοκόλλων μιας υπηρεσίας διαδικτύου με έναν τυποποιημένο τρόπο. Τα στοιχεία της WSDL περιλαμβάνουν μια περιγραφή των δεδομένων – χρησιμοποιώντας ένα η παραπάνω XML σχήματα (XML schemas) – που θα σταλούν από την υπηρεσία διαδικτύου, έτσι ώστε ο αποστολέας και ο παραλήπτης να μπορούν να καταλάβουν τα δεδομένα που ανταλλάσσονται. Τα στοιχεία της WSDL περιλαμβάνουν ακόμα περιγραφές των μεθόδων που θα εφαρμοσθούν στα δεδομένα, ώστε ο παραλήπτης του μηνύματος να ξέρει πώς να το επεξεργαστεί, και ένα πρωτόκολλο ή τρόπο αποστολής, ώστε ο αποστολέας να ξέρει πώς

να το στείλει. Τυπικά, η WSDL χρησιμοποιείται μαζί με το πρωτόκολλο SOAP. Η WSDL αναπτύχθηκε από τις εταιρίες Microsoft, Ariba και IBM, ενώ η έκδοση 1.1 της προδιαγραφής της έγινε δεκτή από το W3C.

Τα δύο μέρη που συμμετέχουν σε μια επικοινωνία ή αλληλεπίδρ

αση με υπηρεσίες διαδικτύου πρέπει να έχουν πρόσβαση στο ίδιο WSDL για να είναι σε θέση να κατανοήθουν εκατέρωθεν. Με άλλα λόγια, τόσο ο αποστολέας όσο και ο παραλήπτης ενός μηνύματος πρέπει να έχουν πρόσβαση στο ίδιο XML σχήμα. Ο αποστολέας πρέπει να γνωρίζει πώς να μορφοποιήσει το προς αποστολή μήνυμα σωστά και ο παραλήπτης πώς να κατανοήσει σωστά το περιεχόμενο του μηνύματος. Από τη στιγμή που έχουν και τα ίδια μέρη το ίδιο WSDL αρχείο, οι υλοποιήσεις των υπηρεσιών διαδικτύου μπορούν να έχουν οποιαδήποτε μορφή. Αυτό είναι και το πλεονέκτημα της WSDL που επιτρέπει την κωδικοποίηση και αποκωδικοποίηση μηνυμάτων από και προς οποιαδήποτε εφαρμογή, όπως CORBA, Enterprise JavaBeans (EJB), συστήματα Enterprise Resource Planning (ERP) [25], κτλ.

Οι WSDL περιγραφές χωρίζονται σε βασικά, διαχωρίσιμα στοιχεία (Εικόνα 5):

- ***Τύποι δεδομένων (Data Types):*** οι τύποι δεδομένων (συνήθως σε μορφή XML σχημάτων) που θα χρησιμοποιηθούν στα μηνύματα
- ***Μήνυμα (Message):*** αφαιρετικός ορισμός των δεδομένων σε μορφή μηνυμάτων για να χρησιμοποιηθούν στην κλήση κάποιας μεθόδου.
- ***Μέθοδος (Operation):*** αφαιρετικός ορισμός των μεθόδων για ένα μήνυμα.
- ***Τερματικά σημεία (Port Types):*** περιγραφή διεπαφής ή αφαιρετικό σύνολο μεθόδων που αντιστοιχίζονται σε μία ή περισσότερες υλοποιήσεις.
- ***Πρόσδεση (Binding):*** τα πρωτόκολλα και οι τύποι δεδομένων για τις μεθόδους και τα μηνύματα που ορίζονται για ένα συγκεκριμένο τύπο τερματικών σημείων.
- ***Θύρα (Port):*** συνδυασμός μιας πρόσδεσης και μιας δικτυακής διεύθυνσης που παρέχει τη διεύθυνση επικοινωνίας για την υπηρεσία.
- ***Υπηρεσία (Service):*** τμήμα που περικλείει τους ορισμούς της υπηρεσίας στο WSDL αρχείο.

Συνηθίζεται τα τμήματα του WSDL αρχείου να παράγονται από διάφορα εργαλεία που μετατρέπουν τα μεταδεδομένα της εφαρμογής από τον κώδικά της σε πληροφορία σε μορφή XML σχήματος, από την οποία προκύπτει τελικά το WSDL αρχείο.

Από τις παραπάνω πληροφορίες προκύπτει το πώς δουλεύει η υπηρεσία, τι αποτέλεσμα έχει η λειτουργία της, ποιες δομές δεδομένων περιμένει να λάβει και ποιες

αποστέλλει. Βέβαια χρειάζεται ο συνδυασμός όλων των παραπάνω πληροφοριών για να γίνουν πλήρως κατανοητές οι δυνατότητες της υπηρεσίας.

Εικόνα 5. Συντακτική δομή WSDL αρχείων.

2.1.2 Simple Object Access Protocol

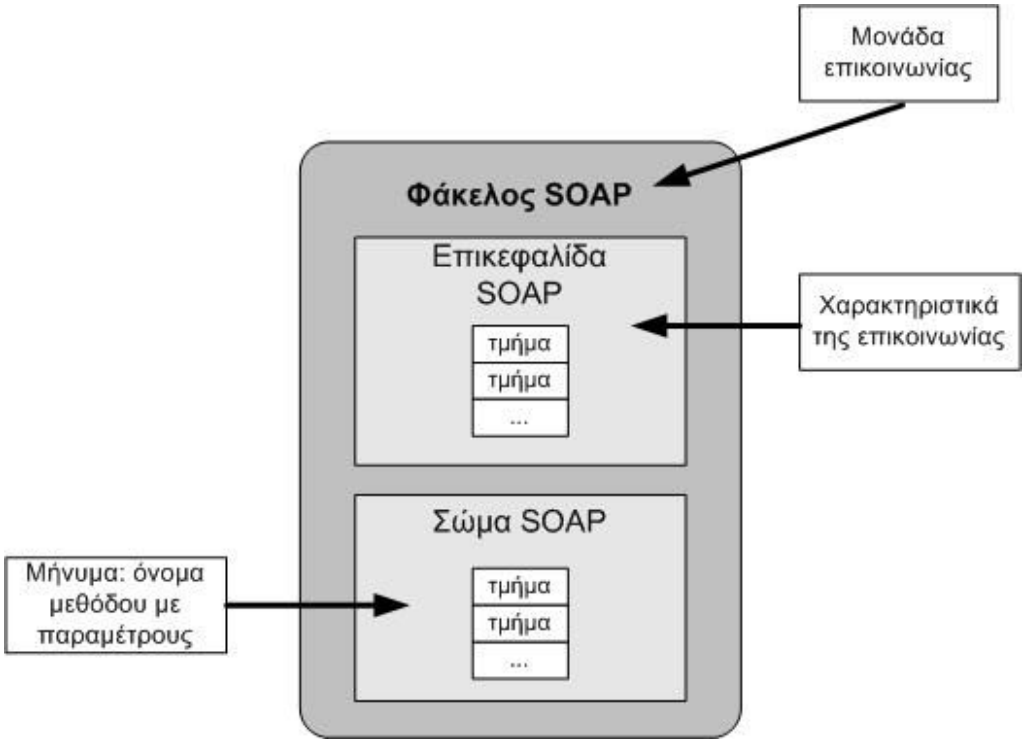
Το Απλό Πρωτόκολλο Πρόσβασης Αντικειμένων (Simple Object Access Protocol / SOAP) [35] αποτελεί ίσως την πιο σημαντική τεχνολογία των υπηρεσιών διαδικτύου. Ο ρόλος του είναι να μεταφέρει τα δεδομένα των υπηρεσιών διαδικτύου από ένα σημείο του δικτύου σε ένα άλλο. Η πρώτη έκδοση της προδιαγραφής του αναπτύχθηκε και δημοσιεύτηκε στα τέλη της δεκαετίας του 1990. Το SOAP αποτελεί μια επέκταση της προδιαγραφής της XML Διαδικασίας Απομακρυσμένης Κλήσης (XML-Remote Procedure Call / XML-RPC) [40] που επίσης συνηθίζεται να χρησιμοποιείται για επικοινωνία στις υπηρεσίες διαδικτύου.

Το SOAP επιτρέπει στον αποστολέα και τον παραλήπτη των XML αρχείων να υποστηρίζουν ένα κοινό πρωτόκολλο μεταφοράς δεδομένων για αποδοτική δικτυακή επικοινωνία. Σε σχέση με το διαδίκτυο, το SOAP μπορεί να θεωρηθεί ως ένα είδος επέκτασης του πρωτοκόλλου HTTP. Αντί να χρησιμοποιείται το HTTP για την αίτηση για μια HTML σελίδα σε ένα φυλλομετρητή ιστού, το πρωτόκολλο SOAP στέλνει ένα XML μήνυμα

μέσω μιας HTTP αίτησης (request) και λαμβάνει μια απάντηση – αν υπάρξει – μέσω μιας HTTP απάντησης (reply). Για να γίνει σωστά ο χειρισμός του XML μηνύματος, χρειάζεται ένας επεξεργαστής ή χειριστής SOAP μηνυμάτων. Πιο συγκεκριμένα, ο HTTP παραλήπτης του SOAP μηνύματος πρέπει να είναι σε θέση να ελέγξει την αξιοπιστία και να καταλάβει την XML μορφή που ορίζεται από την προδιαγραφή του SOAP. Η βασισμένη στο SOAP επικοινωνία μοντελοποιείται θεωρώντας ότι λαμβάνει χώρα μεταξύ SOAP κόμβων, που μπορεί να είναι αποστολείς SOAP μηνυμάτων ή παραλήπτες ή και τα δύο. Ένας SOAP κόμβος υποστηρίζει έναν ή παραπάνω χειριστές SOAP μηνυμάτων και είναι υπεύθυνος για το χειρισμό των διαφόρων τμημάτων του μηνύματος.

Τα τρία βασικά τμήματα ενός SOAP μηνύματος είναι ο φάκελος, η επικεφαλίδα και το σώμα (Εικόνα 6). Ο φάκελος είναι υποχρεωτικός και καθορίζει την αρχή και το τέλος του SOAP μηνύματος. Η επικεφαλίδα είναι προαιρετική και μπορεί να περιλαμβάνει ένα ή περισσότερα τμήματα επικεφαλίδας που μεταφέρουν για παράδειγμα τα χαρακτηριστικά του μηνύματος ή περιγράφουν την ποιότητα υπηρεσίας για το μήνυμα. Οι επικεφαλίδες έχουν σκοπό να μεταφέρουν πληροφορίες που σχετίζονται με το μήνυμα, όπως αναγνωριστικά συναλλαγών, χαρακτηριστικά ασφάλειας ή μηχανισμούς συσχέτισης μηνυμάτων. Το σώμα του μηνύματος είναι υποχρεωτικό και περιλαμβάνει ένα ή παραπάνω τμήματα που αποτελούν το ίδιο το μήνυμα. Οι κανόνες κωδικοποίησης επιτρέπουν την κωδικοποίηση σύνθετων δομών δεδομένων με τέτοιο τρόπο, ώστε να είναι δυνατή η μεταφορά τους με SOAP μηνύματα. Το SOAP υποστηρίζει συγκεκριμένους τύπους δεδομένων (π.χ. ακέραιους), ενώ μπορούν ακόμα να περιγραφθούν σύνθετοι τύποι, όπως λίστες ή δομές που αναπαριστούν π.χ. μια εγγραφή σε μια βάση δεδομένων.

Το SOAP μεταφέρει ένα XML αρχείο κατα μήκος του διαδικτύου και – πιθανόν – κατά μήκος και άλλων τύπων δικτύων για να φτάσει σε μια υλοποίηση υπηρεσίας διαδικτύου. Ένα SOAP μήνυμα είναι ουσιαστικά ένα XML αρχείο που έχει συγκεκριμένη μορφή. Μια συμβατή με το SOAP υλοποίηση γνωρίζει πώς να ερμηνεύει ένα τέτοιο αρχείο και πώς να αντιστοιχεί τα δεδομένα που περιέχει στην υποκείμενη υλοποίηση λογισμικού της υπηρεσίας. Ωστόσο, το SOAP δεν ορίζει την ίδια την υπηρεσία, αλλά ορίζει αρκετά στοιχεία για το μήνυμα, ώστε να μπορεί να το αναγνωρίσει ο επεξεργαστής SOAP μηνυμάτων. Συνήθως, όταν το SOAP μήνυμα ληφθεί από κάποιο άκρο περνάει μέσα από ένα σύνολο εργαλείων του επεξεργαστή μηνυμάτων που το αναλύουν για να κατανοήσουν το περιεχόμενό του ή ακόμα και να το τροποποιήσουν αν χρειαστεί.



Εικόνα 6. Δομή SOAP μηνύματος.

2.1.3 Universal Description, Discovery, and Integration

Αφού δημιουργηθεί μια υπηρεσία διαδικτύου, χρειάζεται να υπάρχει κάποιος τρόπος για να ανακαλυφθεί. Αυτός είναι ο σκοπός της εγγραφής Καθολικής Περιγραφής, Ανακάλυψης και Ενσωμάτωσης (Universal Description, Discovery, and Integration / UDDI) [34] που δημιουργήθηκε για να υλοποιηθεί ένας κατάλογος υπηρεσιών διαδικτύου. Η UDDI αποτελεί μια προδιαγραφή για κατανεμημένα μητρώα (registry) πληροφοριών υπηρεσιών διαδικτύου. Ο UDDI κατάλογος με εγγραφές δέχεται πληροφορίες που περιγράφουν μια επιχείρηση, συμπεριλαμβανομένων των υπηρεσιών διαδικτύου που προσφέρει, και επιτρέπει στους ενδιαφερόμενους να εκτελέσουν διαδικτυακές αναζητήσεις και να λάβουν πληροφορίες για τις υπηρεσίες.

Η UDDI ξεκίνησε ως μια συνεργασία μεταξύ των εταιριών Microsoft, IBM και Ariba για την προώθηση της υιοθέτησης των προτύπων των υπηρεσιών διαδικτύου. Οι εταιρίες ίδρυσαν το *UDDI.org* και προσκάλεσαν και άλλες εταιρίες να συμμετάσχουν. Οι εταιρίες που προσκλήθηκαν ως ιδρυτές έθεσαν τους βασικούς κανόνες, όρισαν τις αρχικές προδιαγραφές και αποφάσισαν για την τελική διάθεση της δουλειάς τους. Αν και η UDDI μοιάζει με το Χρυσό Οδηγό (ή Yellow Pages) για υπηρεσίες διαδικτύου, επιτρέπει ακόμα στους μηχανικούς λογισμικού να αλληλεπιδρούν με τη UDDI τόσο κατά τον σχεδιασμό όσο

και κατά την εκτέλεση. Οι UDDI πόροι μπορούν να θεωρηθούν κομμάτι της αρχιτεκτονικής και της υποδομής των υπηρεσιών διαδικτύου.

Η UDDI έχει δύο βασικά μέρη: εγγραφή (registration) και ανακάλυψη (discovery). Το κομμάτι της εγγραφής σημαίνει ότι οι επιχειρήσεις μπορούν να στείλουν πληροφορίες στη UDDI που οι άλλες επιχειρήσεις μπορούν έπειτα να ψάξουν και να ανακαλύψουν, κάτι το οποίο αποτελεί το δεύτερο κομμάτι της ανακάλυψης.

Εννοιολογικά, οι πληροφορίες που παρέχονται σε μια επιχειρησιακή εγγραφή UDDI αποτελούνται από τρία συστατικά:

- ***Ασπρες σελίδες (White Pages):*** περιλαμβάνουν το όνομα, τη διεύθυνση και άλλες πληροφορίες για την επιχείρηση.
- ***Κίτρινες σελίδες (Yellow Pages):*** περιλαμβάνουν το είδος της επιχείρησης, τη θέση και τα προϊόντα της, συμπεριλαμβανομένων διαφόρων τυποποιημένων ταξινομιών για τη γεωγραφική της θέση, το είδος της βιομηχανίας, κτλ.
- ***Πράσινες σελίδες (Green Pages):*** περιλαμβάνουν τεχνικές πληροφορίες για τις υπηρεσίες της επιχείρησης, όπως τις προδιαγραφές, τους ορισμούς των επιχειρησιακών διεργασιών, κτλ. Εδώ ενδέχεται να υπάρχει και μια αναφορά στο WSDL αρχείο. Οι πράσινες σελίδες περιλαμβάνουν με άλλα λόγια τα χαρακτηριστικά της υπηρεσίας, συμπεριλαμβανομένου ενός μοναδικού αναγνωριστικού για την υπηρεσία.

Η UDDI δομή δεδομένων εκφράζεται με πολύπλοκους τύπους σε XML σχήματα. Αυτά τα σχήματα επιτρέπουν την επεκτασιμότητα των δεδομένων που αποθηκεύονται για μια συγκεκριμένη επιχείρηση ή οντότητα. Οι πληροφορίες ταξινόμησης είναι χρήσιμες για την αναζήτηση και την ανάκτηση συγκεκριμένων λεπτομερειών για τις επιχειρήσεις, ενώ η κάθε επιχείρηση μπορεί να προσθέσει οποιοδήποτε αριθμό ταξινομήσεων στις εγγραφές της για να βοηθήσει την αναζήτηση.

2.2 Πλαίσια Διαχείρισης Υπηρεσιών Διαδικτύου

Διάφορα πλαίσια εργασίας για υπηρεσίες διαδικτύου (web service frameworks) είναι διαθέσιμα. Ο ρόλος τους είναι να διευκολύνουν τους μηχανικούς υπηρεσιών διαδικτύου υλοποιώντας τα βασικά πρωτόκολλα των υπηρεσιών. Στον Πίνακα 1 παρουσιάζεται μια λίστα των πιο γνωστών framework που υπάρχουν, ενώ στις υποενότητες δίνονται πιο αναλυτικές περιγραφές για δύο από τα επικρατέστερα (Apache Axis2 και Apache CXF).

Όνομα Framework	Πλατφόρμα	Πρωτόκολλα που υποστηρίζει
Apache Axis [2]	Java/C++	SOAP, WSDL
Apache Axis2 [3]	Java/C	SOAP, MTOM, WSDL 2.0, WSDL
Apache CXF [5]	Java	SOAP1.1, SOAP1.2, MTOM, WSDL 2.0, WSDL
Csoap [13]	C	SOAP
Hessian [18]	Java, Ruby, Python, Erlang, PHP κτλ.	Hessian
Java Web Services Development Pack [32]	Java	SOAP, WSDL
NuSOAP [26]	PHP	SOAP, WSDL
Web Services Invocation Framework [8]	Java	SOAP, WSDL
Windows Communication Foundation [39]	.Net	SOAP, WSDL
XML Interface for Network Services [44]	Java	SOAP, XML-RPC
gSoap [17]	C/C++	SOAP, XML-RPC, WSDL
WSO2 Web Services Framework for C (WSO2 WSF/C) [41]	C	SOAP, WSDL, TLS

Πίνακας 1. Frameworks για υπηρεσίες διαδικτύου.

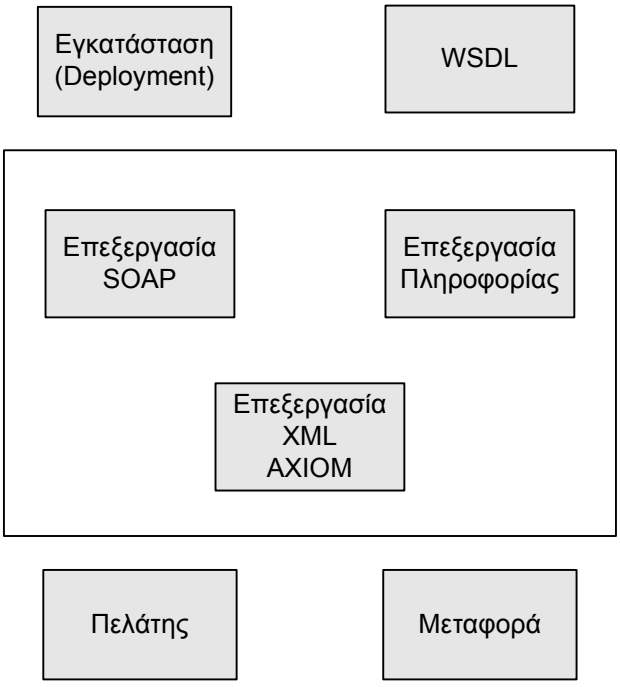
2.2.1 Apache Axis2

Το Apache Axis2 αποτελεί μια υλοποίηση της πλευράς του πελάτη, αλλά και του εξυπηρετητή της επικοινωνίας υπηρεσιών διαδικτύου, και υπάρχει διαθέσιμο σε διαφορετικές εκδόσεις στις γλώσσες προγραμματισμού Java και C. Υλοποιήθηκε με σκοπό να βελτιώσει το προηγούμενο Apache Axis. Έτσι, παρέχει μια αρθρωτή αρχιτεκτονική που

επιτρέπει την προσθήκη νέων λειτουργιών για την υποστήριξη των νέων προδιαγραφών που προκύπτουν για τις υπηρεσίες διαδικτύου. Μέσω του Axis2 είναι εφικτές μεταξύ άλλων οι εξής ενέργειες:

- Αποστολή SOAP μηνυμάτων
- Λήψη και επεξεργασία SOAP μηνυμάτων
- Δημιουργία μιας υπηρεσίας διαδικτύου από μια απλή Java κλάση (μέσω του εργαλείου Java2WSDL)
- Δημιουργία κλάσεων υλοποίησης για τον πελάτη και τον εξυπηρετήτή χρησιμοποιώντας το WSDL αρχείο (μέσω του εργαλείου WSDL2Code)
- Ανάκτηση του WSDL αρχείου μιας υπηρεσίας
- Αποστολή και λήψη SOAP μηνυμάτων με συνημμένα
- Δημιουργία και χρησιμοποίηση υπηρεσιών διαδικτύου βασισμένων στο πρωτόκολλο Representational State Transfer (REST) [28]

Η αρθρωτή αρχιτεκτονική του Apache Axis2 φαίνεται στην Εικόνα 7.



Εικόνα 7. Αρχιτεκτονική του Apache Axis 2.

Τα μηνύματα που λαμβάνονται από το Axis2 διατρέχουν ένα σύνολο χειριστών (handlers) που τα επεξεργάζονται καταλλήλως. Κάθε χειριστής λειτουργεί σε ξεχωριστή φάση (phase) πάνω στο μήνυμα, ενώ ειδικοί χειριστές μπορούν να εισαχθούν και από τους χρήστες υπό τη μορφή υπομονάδων (modules) και να συσχετιστούν με κάποια

συγκεκριμένη φάση. Αυτή τη δυνατότητα του Axis2 έχουμε εκμεταλλευτεί και στα πλαίσια της διατριβής.

Στα πλεονεκτήματα του Axis2 συγκαταλέγεται ακόμα η υποστήριξη διαφόρων framework αντιστοίχισης, όπως τα Apache XML Beans [43]. Αυτή η δυνατότητα είναι χρήσιμη για υπηρεσίες με πολύπλοκο σχήμα ορισμού, καθώς τα XMLBeans αποτελούν μια τεχνολογία για την πρόσβαση XML αρχείων μέσω της σύνδεσης των στοιχείων τους σε τύπους της γλώσσας προγραμματισμού Java.

2.2.2 Apache CXF

Το framework Apache CXF (CeltiXFire) έχει προκύψει ως συνδυασμός και έχει αντικαταστήσει τα Xfire [42] και Celtix. Υποστηρίζει διάφορα πρωτόκολλα, όπως τα SOAP, XML/HTTP, RESTful HTTP και CORBA, ενώ μπορεί να χρησιμοποιηθεί πάνω από διάφορα πρωτόκολλα του επιπέδου μεταφοράς, όπως τα HTTP και Java Message Service (JMS).

Η γενική αρχιτεκτονική του CXF αποτελείται από τα παρακάτω μέρη:

- ***Δίαυλος (Bus):*** αποτελεί τη ραχοκοκαλιά της αρχιτεκτονικής του CXF.
- ***Μηνύματα και χειριστές (Messaging & Interceptors):*** παρέχουν το στρώμα μηνυμάτων πάνω στο οποίο χτίζονται οι περισσότερες λειτουργίες.
- ***Διεπαφές χρηστών-συστήματος (Front ends):*** παρέχουν ένα προγραμματιστικό μοντέλο για τη δημιουργία υπηρεσιών (π.χ. JAX-WS [21]).
- ***Υπηρεσίες (Services):*** παρέχουν ένα μοντέλο υπηρεσιών για την περιγραφή της υπηρεσίας που μοιάζει με το μοντέλο της WSDL.
- ***Προσδέσεις (Bindings):*** παρέχουν τη λειτουργία μετάφρασης του πρωτοκόλλου (π.χ. SOAP, REST).
- ***Μεταφορές (Transports):*** προορισμοί και κανάλια που δημιουργούν την αφαιρετικότητα μεταφοράς που χρησιμοποιεί το CXF για να επιτύχει την ουδετερότητα μεταφοράς.

Το CXF παρέχει ακόμα υποστήριξη για διάφορα εργαλεία μετατροπής μεταξύ κλάσεων Java beans, υπηρεσιών διαδικτύου και WSDL αρχείων. Υποστηρίζει την ενσωμάτωση με εργαλεία ανάπτυξης εφαρμογών (build tools), όπως είναι το Apache Maven [6] και το Apache Ant [1], καθώς και με το Spring [29]. Οι προσδέσεις που προσφέρει περιλαμβάνουν και ένα κομμάτι χωρίς XML μορφοποίηση, όπου μπορεί να γίνει χρήση της μορφοποίησης JavaScript Object Notation (JSON) [22] και της CORBA.

Το CXF, όπως και το Apache Axis2, υποστηρίζει χειριστές μηνυμάτων. Όταν καλείται μια υπηρεσία δημιουργείται και ενεργοποιείται μια σειρά χειριστών. Κάθε χειριστής μπορεί

να διαβάσει ή να τροποποιήσει το μήνυμα, να επεξεργαστεί την επικεφαλίδα του, να ελέγξει την αξιοπιστία του, κτλ. Οι χειριστές χρησιμοποιούνται τόσο στους CXF πελάτες όσο και στους εξυπηρετήτες.

2.3 Πλαίσια Μοντέλου-Όψης-Ελεγκτή

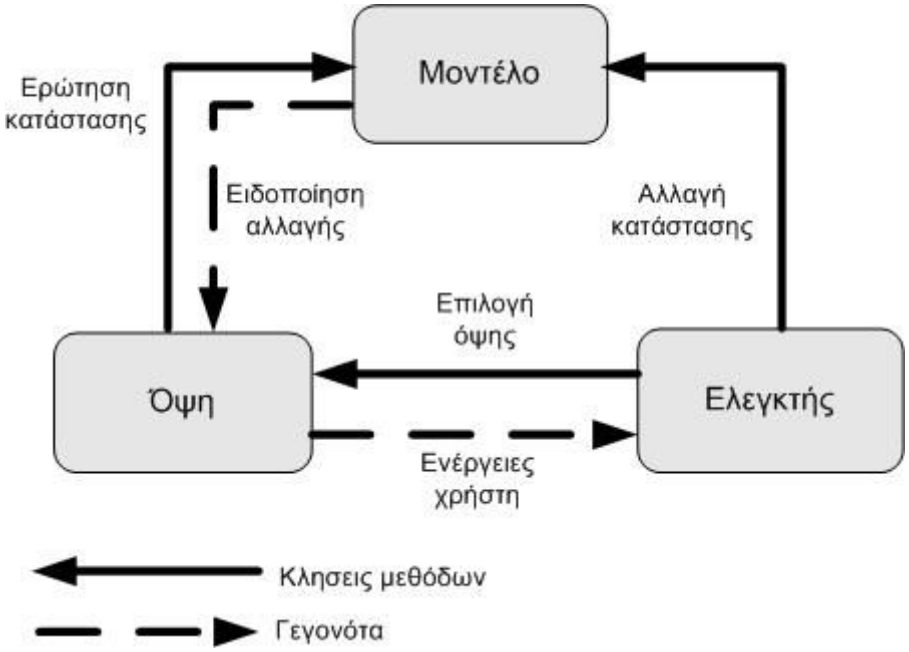
Ο όρος Μοντέλο-Όψη-Ελεγκτής (Model-View-Controller / MVC) [19] χρησιμοποιείται για το αρχιτεκτονικό πρότυπο (pattern) [10] που χρησιμοποιείται στη μηχανική λογισμικού με σκοπό να απομονώσει τη λογική της εφαρμογής από το κομμάτι της εφαρμογής που σχετίζεται με τη διεπαφή του χρήστη. Με αυτόν τον τρόπο καθίσταται πιο εύκολη η μετατροπή της εμφάνισης της εφαρμογής ή των υποκείμενων κανόνων χωρίς να επηρεάζεται η συνολική εφαρμογή.

Το MVC αποτελείται από τα εξής συστατικά:

- ***Μοντέλο (Model):*** το μοντέλο αναπαριστά τα δεδομένα της εφαρμογής και τους κανόνες που χρησιμοποιούνται για το χειρισμό των δεδομένων. Αποτελεί τη συγκεκριμένη αναπαράσταση της πληροφορίας την οποία χρησιμοποιεί η εφαρμογή. Πολλές εφαρμογές χρησιμοποιούν κάποιο μηχανισμό αποθήκευσης (π.χ. μια βάση δεδομένων), αλλά το MVC δεν αναφέρεται συγκεκριμένα στο επίπεδο πρόσβασης δεδομένων, γιατί θεωρεί ότι αυτό υπόκειται του μοντέλου. Ο ρόλος του μοντέλου είναι να χειρίζεται την πληροφορία και να ενημερώνει για κάποια αλλαγή σε αυτή.
- ***Όψη (View):*** η όψη αντιστοιχεί στα διάφορα στοιχεία της διεπαφής του χρήστη (π.χ. κείμενο, λίστες επιλογής, κτλ.). Πολλαπλές όψεις μπορούν να συνυπάρχουν σε ένα μοντέλο για διαφορετικούς λόγους. Η όψη είναι υπεύθυνη για την αντιστοίχιση των γραφικών συστατικών σε μια συσκευή. Συνδέεται στο μοντέλο και απεικονίζει τα περιεχόμενά του σε κάποια οθόνη. Όταν το μοντέλο αλλάζει, το τμήμα της διεπαφής που επηρεάζεται ανανεώνεται για να εκφράσει την αλλαγή. Στο ίδιο μοντέλο μπορούν να υπάρχουν παραπάνω από μια όψεις, με καθεμία να απεικονίζει τα περιεχόμενα του μοντέλου σε διαφορετική οθόνη. Επιπλέον, μια όψη μπορεί να περιλαμβάνει διάφορες υπο-όψεις που να περικλείουν και αυτές με τη σειρά τους πολλές υπο-όψεις, κοκ.
- ***Ελεγκτής (Controller):*** ο ελεγκτής είναι υπεύθυνος για τις λεπτομέρειες που σχετίζονται με τη μετάδοση των ενεργειών του χρήστη (π.χ. κίνηση του ποντικιού) στο μοντέλο. Ο ελεγκτής επεξεργάζεται και απαντά σε αυτές τις ενέργειες, ενώ

μπορεί να ζητήσει αλλαγές στο μοντέλο μετά από την ανίχνευση κάποιας ενέργειας. Μέσω του ελεγκτή ο χρήστης αλληλεπιδρά με την εφαρμογή.

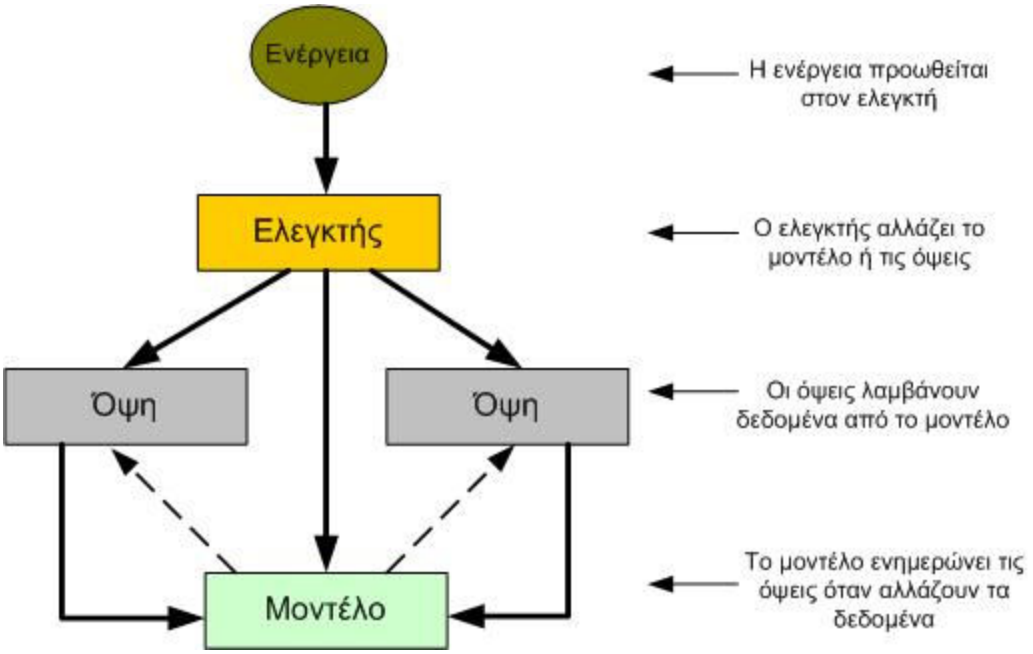
Το μοντέλο, η όψη και ο ελεγκτής είναι άρρηκτα συνδεδεμένα και σε συνεχή επαφή. Η αλληλεπίδρασή τους φαίνεται στην Εικόνα 8.



Εικόνα 8. Το πρότυπο μοντέλου-όψης-ελεγκτή.

Τα γεγονότα που συμβαίνουν γίνονται αιτία για να αλλάξει ο ελεγκτής ένα μοντέλο ή μία όψη ή και τα δύο. Κάθε φορά που ο ελεγκτής αλλάζει τα δεδομένα ή τα χαρακτηριστικά του μοντέλου, όλες οι εξαρτώμενες όψεις ενημερώνονται αυτόματα. Ομοίως, κάθε φορά που ο ελεγκτής αλλάζει μια όψη, εμφανίζοντας π.χ. περιοχές που δεν ήταν προηγουμένως ορατές, το μοντέλο χρησιμοποιεί τα δεδομένα από το υποκείμενο μοντέλο για να ενημερωθεί (Εικόνα 9).

Διάφορα framework που ακολουθούν τις αρχιτεκτονικές αρχές του προτύπου μοντέλου-όψης-ελεγκτή έχουν αναπτυχθεί σε διάφορες γλώσσες προγραμματισμού. Κάποια από αυτά παρουσιάζονται στον Πίνακα 2, ενώ στις υποενότητες περιγράφονται τα επικρατέστερα framework στη γλώσσα προγραμματισμού Java (Apache Struts και Spring MVC Framework).



Εικόνα 9. Γεγονότα στο πρότυπο μοντέλου-όψης-ελεγκτή.

MVC Frameworks	Γλώσσα Υλοποίησης
ASP.NET MVC Framework [9]	.NET
Maverick.NET [24]	.NET
Apache Cocoon [4]	Java
Google Web Toolkit (GWT) [16]	Java
Spring MVC Framework [31]	Java
Apache Struts [7]	Java
Tapestry [33]	Java
WebObjects [38]	Java
JavaServer Faces [20]	Java
Wavemaker [37]	Javascript
Catalyst [12]	Perl
Fuse Framework [15]	PHP

Django [14]	Python
Camping [11]	Ruby

Πίνακας 2. Frameworks μοντέλου-όψης-ελεγκτή.

2.3.1 Apache Struts

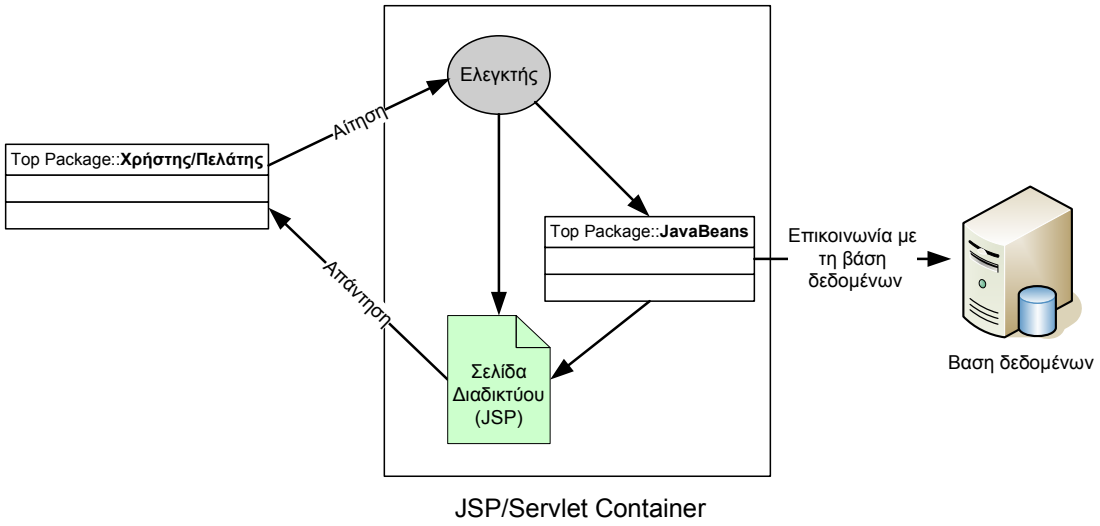
Το Apache Struts αποτελεί ένα framework ανοικτού κώδικα (open source) για την ανάπτυξη εφαρμογών διαδικτύου. Χρησιμοποιεί και επεκτείνει το σύνολο βιβλιοθηκών Java Servlet για να ενισχύσει την υιοθέτηση αρχιτεκτονικών μοντέλου-όψης-ελεγκτή. Στόχος του Struts είναι να διαχωρίσει το μοντέλο της εφαρμογής από την όψη, η οποία σχετίζεται με HTML σελίδες που παρουσιάζονται στο χρήστη, και από τον ελεγκτή, όπως ακριβώς συμβαίνει στο MVC. Το Struts παρέχει τον ελεγκτή (`ActionServlet`) και διευκολύνει την εγγραφή προτύπων για την όψη ή την παρουσίαση της εφαρμογής, που συνήθως χρησιμοποιεί την τεχνολογία σελίδων JavaServer Pages (JSPs). Ο προγραμματιστής της εφαρμογής διαδικτύου είναι υπεύθυνος για την ανάπτυξη του κώδικα του μοντέλου και για τη δημιουργία ενός κεντρικού αρχείου διαμόρφωσης (*struts-config.xml*), που ενώνει το μοντέλο, την όψη και τον ελεγκτή.

Αιτήσεις στέλνονται από το χρήστη στον ελεγκτή υπό τη μορφή “ενεργειών” (Actions) που ορίζονται στο αρχείο διαμόρφωσης. Όταν ο ελεγκτής λάβει μια τέτοια αίτηση, καλεί την αντίστοιχη `Action` κλάση που αλληλεπιδρά με τον κώδικα του μοντέλου της εφαρμογής. Ο κώδικας του μοντέλου επιστρέφει την "`ActionForward`", μια λέξη που ενημερώνει τον ελεγκτή για τη σελίδα εξόδου που θα πρέπει να στείλει στο χρήστη. Οι πληροφορίες μεταφέρονται μεταξύ του μοντέλου και της όψης σε μορφή ειδικών αντικειμένων Java beans. Μια ειδική βιβλιοθήκη ετικετών (tag) για JSP σελίδες επιτρέπει την ανάγνωση και την εγγραφή του περιεχομένου των Java beans στο επιπεδο παρουσίασης της εφαρμογής χωρίς να χρειάζεται να υπάρχει ενσωματωμένος στις σελίδες κώδικας σε Java. Η βασική λειτουργία του Struts παρουσιάζεται στην Εικόνα 10.

Το Struts προσφέρει διάφορα πλεονεκτήματα μειώνοντας το χρόνο ανάπτυξης και καθιστώντας πιο εύκολη τη διαχείριση της εφαρμογής. Τα κυριότερα χαρακτηριστικά του είναι τα εξής:

- ***Κεντρική διαχείριση με αρχεία:*** Αντί να υπάρχει η πληροφορία μέσα στον κώδικα της εφαρμογής, πολλές τιμές του Struts αναπαρίστανται σε XML αρχεία και σε αρχεία ιδιοτήτων. Έτσι, μπορούν να γίνουν διάφορες αλλαγές χωρίς μετατροπές στον κώδικα, αλλά με απλή μετατροπή ενός μόνο αρχείου. Αυτή η προσέγγιση

επιτρέπει στους προγραμματιστές εφαρμογών διαδικτύου να εστιάσουν σε συγκεκριμένες λειτουργίες χωρίς να χρειάζεται να γνωρίζουν τη διάταξη ολόκληρου του συστήματος.



Εικόνα 10. Λειτουργία του Apache Struts.

- ***Beans για φόρμες:*** Πρόκειται για αντικείμενα Java beans, τα οποία αποθηκεύουν τα δεδομένα από τις φόρμες της εφαρμογής. Με αυτόν τον τρόπο ο προγραμματιστής αποκτά εύκολη πρόσβαση στις παραμέτρους που εισάγει ο χρήστης.
- ***Ετικέτες για Beans:*** Το Struts προσφέρει ένα σύνολο JSP ετικετών που επιτρέπουν την εύκολη παρουσίαση των παραμέτρων των διάφορων Java beans που χρησιμοποιεί η εφαρμογή.
- ***HTML ετικέτες:*** Το Struts προσφέρει επιπροσθέτως ένα σύνολο JSP ετικετών που σχετίζονται με τα Java beans. Ο προγραμματιστής μπορεί να θέσει τις τιμές των πεδίων της φόρμας από τις τιμές κάποιων πεδίων αντικειμένων ή να επανεμφανίσει την ίδια φόρμα με κάποιες τιμές συμπληρωμένες ανάλογα με την προηγούμενη είσοδο του χρήστη.
- ***Επικύρωση πεδίων φόρμας:*** Το Struts έχει ενσωματωμένη δυνατότητα ελέγχου για τις τιμές στις φόρμες, π.χ. αν μια ημερομηνία είναι σωστή. Αν κάποιες τιμές λείπουν ή είναι σε ακατάλληλη μορφή η φόρμα μπορεί να επανεμφανιστεί αυτόματα με τα κατάλληλα μηνύματα λάθους. Αυτή η επικύρωση μπορεί να γίνει είτε μόνο στην πλευρά του εξυπηρετητή , είτε και στον εξυπηρετητή και στον πελάτη.

2.3.2 Spring MVC

Το Spring [29] αποτελεί ένα framework ανοικτού κώδικα που δημιουργήθηκε για να αντιμετωπίσει την πολυπλοκότητα της ανάπτυξης επιχειρησιακών εφαρμογών. Ένα από τα βασικά του πλεονεκτήματα είναι η αρχιτεκτονική του που επεκτείνεται σε διάφορα επίπεδα, επιτρέποντας στο χρήστη να επιλέξει ποια από τα συστατικά του Spring θα χρησιμοποιήσει. Ένα από αυτά τα επίπεδα είναι το Spring MVC [31], το οποίο δημιουργήθηκε για να αντιμετωπίσει διάφορα μειονεκτήματα που εμφάνιζε το Struts και αντίστοιχα frameworks. Συγκεκριμένα, θεωρήθηκε ότι ο διαχώρισμος μεταξύ του επιπεδου παρουσίασης και του χειρισμού αιτήσεων, καθώς και μεταξύ του επιπέδου χειρισμού αιτήσεων και του μοντέλου ήταν ανεπαρκής.

Όπως και το Struts, το Spring MVC βασίζεται στις αιτήσεις που λαμβάνονται. Το Spring MVC ορίζει διεπαφές για όλες τις αρμοδιότητες που πρέπει να διαχειριστούν. Η αρμοδιότητα κάθε διεπαφής είναι αρκετά απλή και ξεκάθαρη, ώστε να είναι εύκολο για τους χρήστες να αναπτύξουν τις δικές τους υλοποιήσεις αν το επιθυμούν. Όλες οι διεπαφές συνδέονται στενά με το σύνολο βιβλιοθηκών Servlet API.

Το Spring Web MVC είναι σχεδιασμένο γύρω από έναν `DispatcherServlet` που αποστέλλει αιτήσεις στους χειριστές με τις παραμέτρους που απαιτούνται. Ο πιο απλός χειριστής αποτελεί στην ουσία μια πολύ απλή διεπαφή ελεγκτή. Οι βασικές διεπαφές που ορίζονται από το Spring MVC και οι αντίστοιχες αρμοδιότητές τους είναι οι εξής:

- ***Αντιστοίχιση χειριστή (`HandlerMapping`):*** επιλέγει αντικείμενα που χειρίζονται τις εισερχόμενες αιτήσεις βάσει οποιασδήποτε παραμέτρου ή κάποιας εσωτερικής ή εξωτερικής συνθήκης.
- ***Προσαρμογέας χειριστή (`HandlerAdapter`):*** είναι υπεύθυνος για την κλήση αντικειμένων που χειρίζονται τις εισερχόμενες αιτήσεις.
- ***Ελεγκτής (`Controller`):*** διαχειρίζεται τις εισερχόμενες αιτήσεις και τις ανακατευθύνει στην κατάλληλη απάντηση.
- ***Όψη (`View`):*** είναι υπεύθυνη για την επιστροφή της κατάλληλης απάντησης στο χρήστη.
- ***Αναλυτής όψεων (`ViewResolver`):*** επιλέγει μια όψη βάσει του λογικού ονόματος της όψης.
- ***Χειριστής λήψης (`HandlerInterceptor`):*** είναι υπεύθυνος για τη λήψη των εισερχόμενων αιτήσεων.
- ***Αναλυτής τοποθεσίας (`LocaleResolver`):*** είναι υπεύθυνος για την ανάλυση και προαιρετικά για την αποθήκευση της τοποθεσίας ενός χρήστη.

- ***Πολυμερής αναλυτής (`MultipartResolver`):*** διευκολύνει τη διαδικασία για την αποθήκευση (upload) αρχείων.

Το Spring MVC παρουσιάζει διάφορα πλεονεκτήματα που μπορούν επιπλέον να συνδυαστούν με τα γενικά πλεονεκτήματα του Spring (π.χ. Inversion of Control / IOC).

Το Spring Web Flow [30] [23] αποτελεί μια επέκταση του Spring MVC που επιτρέπει τον ορισμό ελεγκτών μέσω μιας γλώσσας ειδικού σκοπού. Η γλώσσα είναι σχεδιασμένη για τη μοντελοποίηση των αλληλεπιδράσεων με το χρήστη της εφαρμογής που μπορούν να αποτελούνται από διάφορες αιτήσεις στον εξυπηρετητή. Επιτρέπει την αναπαράσταση της διεπαφής του χρήστη με τρόπο απλό και κατανοητό.

2.4 Βιβλιογραφία

[1] Apache Ant Project, http://ant.apache.org/.

[2] Apache Axis, http://ws.apache.org/axis/.

[3] Apache Axis2 for Java, http://ws.apache.org/axis2/.

[4] Apache Cocoon, http://cocoon.apache.org/.

[5] Apache CXF, http://cxf.apache.org/.

[6] Apache Maven Project, http://maven.apache.org/.

[7] Apache Struts, http://struts.apache.org/.

[8] Apache WSIF (Web Services Invocation Framework), http://ws.apache.org/wsif/.

[9] ASP .NET MVC, http://www.asp.net/mvc/default.aspx?wwwaspnetrdirset=1.

[10] Avgeriou, P., Uwe Z., "Architectural patterns revisited:a pattern language", Proc. 10th European Conference on Pattern Languages of Programs (EuroPlop 2005), 2005, pp. 1-39.

[11] Camping, http://code.whytheluckystiff.net/camping/.

[12] Catalyst, http://www.catalystframework.org/.

[13] Csoap, http://csoap.sourceforge.net/.

[14] Django, http://www.djangoproject.com/.

[15] Fuse Framework, http://www.phpfuse.net/.

[16] Google Web Toolkit (GWT), http://code.google.com/webtoolkit/terms.html.

[17] gSoap, http://www.cs.fsu.edu/~engelen/soap.html.

[18] Hessian binary web service protocol, http://hessian.caucho.com/.

[19] Java BluePrints, “Model-View-Controller (MVC)”, http://java.sun.com/blueprints/patterns/MVC-detailed.html.

[20] Java Server Faces (JSF), http://java.sun.com/javaee/javaserverfaces/.

[21] JAX-WS 2.1, “The Java API for XML-Based Web Services“, Sun Microsystems, 2007, http://jcp.org/aboutJava/communityprocess/mrel/jsr224/index2.html

[22] .JSON, http://www.json.org/.

[23] Ladd, S., Donald, K., “Expert Spring MVC and Web Flows”, Apress, 2006, ISBN-13 (pbk): 978-1-59059-584-8.

[24] Maverick .NET, http://mavnet.sourceforge.net/.

[25] Monk, E., Wagner, B., “Concepts in Enterprise Resource Planning (Second ed.)”, Boston: Thomson Course Technology, 2006, ISBN 0-619-21663-8.

[26] NuSOAP, http://sourceforge.net/projects/nusoap/.

[27] OASIS Web Services Security (WSS) TC, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss.

[28] Pautasso, C., Zimmermann, O., Leymann, F., “RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision”, Proc. 17th International World Wide Web Conference (WWW2008), 2008, pp. 805-814.

[29] Spring Framework, http://www.springframework.org/.

[30] Spring Web Flow, http://www.springsource.org/webflow.

[31] Spring Web MVC framework, Chapter 13, Spring documentation, http://static.springframework.org/spring/docs/2.0.x/reference/mvc.html.

[32] Sun Microsystems, Java Web Services Developer Pack, http://java.sun.com/webservices/downloads/previous/index.jsp.

[33] Tapestry, http://tapestry.apache.org/

[34] Universal Description, Discovery and Integration (UDDI) Specification, 3.0.2, 2004, http://uddi.org/pubs/uddi-v3.0.2-20041019.htm.

[35] W3C, "SOAP version 1.2 Part 0: Primer (Second Edition)", April 2007, http://www.w3.org/TR/2007/REC-soap12-part0-20070427/.

[36] W3C, Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., "Web Services Description Language (WSDL) version 1.1", March 2001, http://www.w3.org/TR/wsdl.

[37] Wavemaker, http://dev.wavemaker.com/forums/.

[38] WebObject, http://developer.apple.com/tools/webobjects/.

[39] Windows Communication Foundation (WCF), http://msdn.microsoft.com/en-us/netframework/aa663324.aspx.

[40] Winer, D., "XML-RPC Specification, Update 3", June 1999, http://www.xmlrpc.com/spec/#update3.

[41] WSO2 Web Services Framework for C, http://wso2.com/products/wsfc/.

[42] XFire, http://xfire.codehaus.org/.

[43] XML Beans, http://xmlbeans.apache.org/.

[44] XML Interface for Network Services, http://xins.sourceforge.net/.

3ο Κεφάλαιο

Επισκόπηση Βιβλιογραφίας Προσαρμογής στο Πλαίσιο Χρήσης

Τα τελευταία χρόνια έχει παρατηρηθεί μεγάλο ενδιαφέρον για την έρευνα που σχετίζεται με τις εφαρμογές με επίγνωση του πλαισίου χρήσης με αναφορά σε διάφορα θέματα και επίπεδα του πλαισίου χρήσης, όπως είναι η ανάκτηση πληροφοριών πλαισίου χρήσης από το περιβάλλον (π.χ. [20]), ο σχεδιασμός και η υλοποίηση υπηρεσιών προσαρμοσμένων στο πλαίσιο χρήσης (π.χ. [25]), μελέτες εφαρμογής των υπηρεσιών (π.χ. [7]), κτλ. Στη διατριβή επικεντρωνόμαστε στην προσέγγιση της επίγνωσης πλαισίου χρήσης από την πλευρά των μηχανικών εφαρμογών και υπηρεσιών διαδικτύου, ενώ γίνεται και γενικότερη αναφορά σε υπηρεσίες που προσανατολίζονται στο κινητό διαδίκτυο. Επιπλέον, εστιάζουμε στη διαδικασία προσαρμογής των παρεχόμενων υπηρεσιών στο πλαίσιο χρήσης θεωρώντας ότι έχει προηγηθεί η διαδικασία ανάκτησης της πληροφορίας του πλαισίου χρήσης από το περιβάλλον με διάφορους μηχανισμούς (π.χ. μέσω αισθητήρων), η κατάλληλη μεταφορά και κάποια αρχική επεξεργασία της πληροφορίας, ώστε να υπάρχει διαθέσιμη σε μια κατανοητή για τις υπηρεσίες μορφή. Με τις παραπάνω προϋποθέσεις η πληροφορία πλαισίου χρήσης γίνεται διαθέσιμη μέσω κλήσεων σε υπηρεσίες (ίσως χαμηλότερου επιπέδου) που παρέχουν αυτή την πληροφόρηση και που μπορούν να συνδυαστούν με τεχνικές μηχανικής λογισμικού και υπηρεσιών. Μάλιστα η μηχανική υπηρεσιών (service engineering) μπορεί να οριστεί ως "*μια ειδική κατηγορία της μηχανικής λογισμικού που στοχεύει στην ανάπτυξη εφαρμογών που θα χρησιμοποιηθούν από τους τελικούς χρήστες / πελάτες*".

Στο παρόν Κεφάλαιο επιχειρείται μια πιο γενική παρουσίαση της βιβλιογραφίας που σχετίζεται με την παροχή υπηρεσιών με επίγνωση του πλαισίου χρήσης. Στα επόμενα

Κεφάλαια το βάρος δίνεται σε υπηρεσίες διαδικτύου, όπου θεωρείται ότι έχει προηγηθεί η διαδικασία ανάκτησης της πληροφορίας πλαισίου χρήσης μέσω κατάλληλων μηχανισμών, χωρίς να γίνεται αναφορά στις υποκείμενες τεχνικές συλλογής.

3.1 Κατηγοριοποίηση Λύσεων

Για την παροχή υπηρεσιών με επίγνωση του πλαισίου χρήσης διάφορες λύσεις έχουν προταθεί και διάφορα συστήματα έχουν υλοποιηθεί. Τα περισσότερα συστήματα είναι προσανατολισμένα σε συγκεκριμένες περιπτώσεις εφαρμογών και δεν έχουν καθολική εφαρμογή. Πράγματι, η υλοποίηση ενός συστήματος που να καλύπτει όλες τις περιπτώσεις είναι ένα πολύ δύσκολο – αν όχι αδύνατο – έργο. Σημαντικό είναι να υπάρχουν προσεγγίσεις που καλύπτουν όσο το δυνατόν περισσότερες περιπτώσεις και εφαρμογές.

Από τη μελέτη που πραγματοποιήθηκε λαμβάνοντας υπόψη τις προγραμματιστικές τεχνικές που χρησιμοποιούνται και τους μηχανισμούς που προτείνονται διακρίνονται οι εξής κατηγορίες στη βιβλιογραφία:

1. Ειδικές πλατφόρμες υπηρεσιών
2. Λύσεις σε επίπεδο πηγαίου κώδικα ή επεκτάσεις γλωσσών προγραμματισμού
3. “Σύλληψη”μηνυμάτων (message interception)

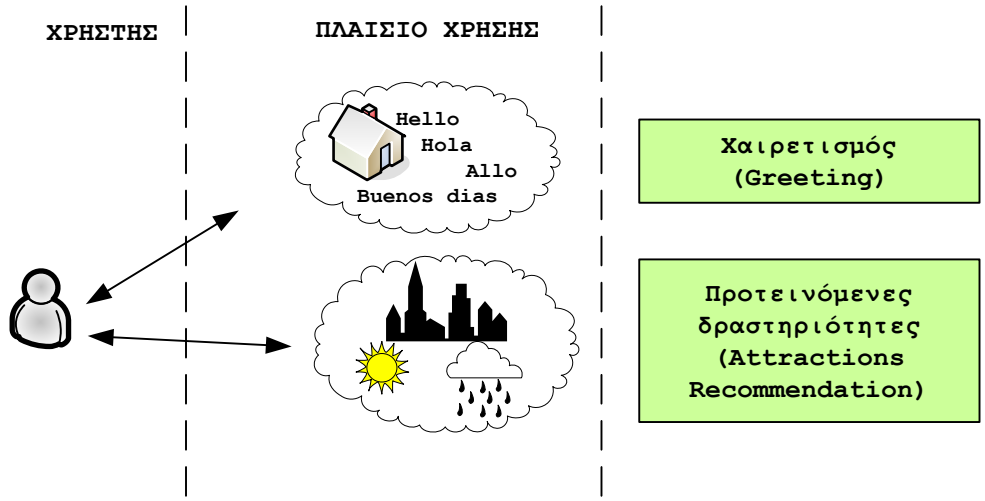
Η πρώτη κατηγορία που σχετίζεται με τις ειδικές πλατφόρμες υπηρεσιών έχει μελετηθεί και παρουσιάζεται αναλυτικά στην επισκόπηση της εργασίας [5]. Αν και οι λύσεις της συγκεκριμένης κατηγορίας δίνουν βάρος κυρίως στις διαδικασίες συλλογής και ανάκτησης της πληροφορίας πλαισίου χρήσης, παρουσιάζονται στο σημείο αυτό για λόγους πληρότητας. Όσον αφορά τη δεύτερη κατηγορία η διαχείριση της πληροφορίας πλαισίου χρήσης μπορεί να γίνει απευθείας σε επίπεδο κώδικα εμπλουτίζοντας τη λογική της εφαρμογής με κομμάτια κώδικα υπεύθυνα για το χειρισμό του πλαισίου χρήσης, παρέχοντας με αυτόν τον τρόπο τον κώδικα με την απαραίτητη συμπεριφορά προσαρμογής. Οι λύσεις σε επίπεδο πηγαίου κώδικα σχετίζονται συνήθως με την επέκταση υπάρχουσων γλωσσών προγραμματισμού είτε στο συντακτικό τους επιπεδο είτε μέσω της παροχής συμπληρωματικών εξωτερικών μηχανισμών. Για την τελευταία κατηγορία η διαχείριση με “σύλληψη” μηνυμάτων είναι εφικτή κυρίως για υπηρεσίες που χρησιμοποιούν ένα σύνολο μηνυμάτων αιτήσεων και απαντήσεων (π.χ. υπηρεσίες διαδικτύου, CORBA). Σε αυτή την περίπτωση η προσαρμογή στο πλαίσιο χρήσης επιτυγχάνεται συλλέγοντας και “αναχαιτώντας” τα εισερχόμενα και εξερχόμενα μηνύματα και τροποποιώντας τα κατάλληλα χωρίς να μεταβληθεί η κύρια εφαρμογή.

Μια άλλη ομάδα λύσεων θα μπορούσε να θεωρηθεί η προσαρμογή στο πλαίσιο χρήσης μέσω εξαγωγής συμπερασμάτων βάσει κανόνων (rule-based reasoning). Οι rule-based προσεγγίσεις βασίζονται σε συστήματα κανόνων (rule-based systems), που αποτελούν συνδυασμό ενός αριθμού κανόνων και ενός συνόλου συνθηκών ενεργοποίησης. Προσφέρουν μια λειτουργία παρόμοια με τις `if-then-else` δηλώσεις, αλλά με πιο περίτεχνο τρόπο. Τα συστήματα κανόνων παρέχουν μια υποδομή γνώσης που κατευθύνει την ενεργοποίηση μιας συμπεριφοράς βάσει της γνώσης του συστήματος που έχει ενσωματωθεί στους κανόνες και τα πρότυπα ενεργοποίησης. Μια ενδιαφέρουσα προσέγγιση έχει παρουσιαστεί στο [11]. Αν και η συγκεκριμένη κατηγορία παρουσιάζει ενδιαφέρον, δεν υπάρχει στη βιβλιογραφία μεγάλο πλήθος προσεγγίσεων σχετικά με την προσαρμογή υπηρεσιών στο πλαίσιο χρήσης.

Στη διατριβή επικεντρωνόμαστε κυρίως στην τελευταία κατηγορία της σύλληψης μηνυμάτων που συγκεντρώνει τα χαρακτηριστικά της προσέγγισης που επιχειρούμε. Τεχνικές αυτής της κατηγορίας επιχειρούν να διαχωρίσουν τη λογική της υπηρεσίας από το επίπεδο προσαρμογής στο πλαίσιο χρήσης. Στις ενότητες που ακολουθούν πραγματοποιείται μια ανάλυση των προσεγγίσεων των τριών κατηγοριών και παρουσιάζεται η επίδραση τους στην πορεία της διατριβής και στο σύστημα διαχείρισης του πλαισίου χρήσης για υπηρεσίες διαδικτύου.

Για να γίνει πιο κατανοητή η λειτουργία κάθε προσέγγισης εισάγεται το παράδειγμα της Εικόνας 11. Το παράδειγμα αναφέρεται σε μια τουριστική υπηρεσία που παρουσιάζει: 1) ένα μήνυμα χαιρετισμού στο χρήστη βάσει της εθνικότητας του (που καθορίζει τη γλώσσα που χρησιμοποιείται στο μήνυμα), της ώρας της ημέρας (που καθορίζει το είδος του χαιρετισμού που εμφανίζεται) και της τρέχουσας θέσης του χρήστη (που χρησιμοποιείται για να προστεθεί το όνομα της αντίστοιχης πόλης στο μήνυμα), και 2) μια πρόταση δραστηριοτήτων τουριστικού ενδιαφέροντος βάσει των προτιμήσεων του χρήστη (που μπορούν να ανακτηθούν από ένα αποθηκευμένο προφίλ για το χρήστη) και της τρέχουσας πληροφορίας για τον καιρό (που καθορίζει αν θα ληφθούν υπόψη και υπαίθριες δραστηριότητες στη λίστα δραστηριοτήτων).

Το συγκεκριμένο παράδειγμα αποτελεί μια απλή περίπτωση προσαρμογής. Ωστόσο, βασίζεται σε διάφορες πληροφορίες πλαισίου χρήσης που προέρχονται από το περιβάλλον του χρήστη. Μάλιστα, με αυτόν τον τρόπο παρουσιάζεται η διαχείριση της πληροφορίας πλαισίου χρήσης στο επίπεδο της εφαρμογής – και όχι σε υποκείμενα επίπεδα – που είναι και η οπτική σκοπιά που χρησιμοποιείται στη διατριβή.



Εικόνα 11. Παράδειγμα τουριστικής υπηρεσίας.

3.2 Ειδικές Πλατφόρμες Υπηρεσιών

Ένα από τα βασικότερα συστήματα της πρώτης κατηγορίας είναι η υπόδομή επίγνωσης πλαισίου χρήσης (Context-awareness sub-structure / CASS) που παρουσιάζεται στο [14]. Σκοπός της CASS είναι η υποστήριξη εφαρμογών με επίγνωση του πλαισίου χρήσης για κινητές συσκευές και άλλα είδη μικρών φορητών υπολογιστών. Στην αρχιτεκτονική του μεσισμικού της CASS περιλαμβάνεται ένας διερμηνέας (*Interpreter*), ένας συλλέκτης πληροφορίας πλαισίου χρήσης (*ContextRetriever*) και ένας ελεγκτής αισθητήρων (*SensorListener*). Ο τελευταίος είναι υπεύθυνος για την ανίχνευση αλλαγών από τους αισθητήρες που είναι τοποθετημένοι σε ένα κατανεμημένο σύστημα. Εν συνεχεία, η συγκεντρωμένη πληροφορία αποθηκεύεται σε μια βάση δεδομένων, ενώ ο *ContextRetriever* είναι υπεύθυνος για την ανάκτηση των αποθηκευμένων πληροφοριών. Και τα δύο αυτά συστατικά μπορούν να χρησιμοποιήσουν τις υπηρεσίες του διερμηνέα. Ο ελεγκτής αλλαγών (*ChangeListener*) που υπάρχει στην πλευρά της κινητής συσκευής παρέχει υπηρεσίες επικοινωνίας και επιτρέπει στα κινητά τερματικά να ενημερώνονται για τις αλλαγές του πλαισίου χρήσης. Οι κινητοί χρήστες συνδέονται με τον εξυπηρετητή μέσω ασύρματων δικτύων. Για να αποφευχθεί η επίδραση των διακοπτόμενων συνδέσεων στον εξυπηρετητή χρησιμοποιείται τοπική κρυφή μνήμη (cache) στην πλευρά του πελάτη.

Ένα ακόμα σύστημα είναι το προσανατολισμένο σε υπηρεσίες μεσισμικό επίγνωσης πλαισίου χρήσης (Service-oriented Context-Aware Middleware / SOCAM) [15]. Το SOCAM υποστηρίζει την παροχή εφαρμογών με επίγνωση του πλαισίου χρήσης και απαρτίζεται από τα εξής βασικά δομικά στοιχεία:

- ***Πάροχος πληροφορίας πλαισίου χρήσης (context provider):*** ανακτά την πληροφορία πλαισίου χρήσης από τις πηγές και την μετατρέπει σε κατάλληλες αναπαραστάσεις, ώστε να μπορούν να την κατανοήσουν και να την επεξεργαστούν οι υπόλοιπες οντότητες του συστήματος.
- ***Διερμηνέας πληροφορίας πλαισίου χρήσης (context interpreter):*** αποτελεί ένα κεντρικό εξυπηρετητή που επεξεργάζεται τις πληροφορίες.
- ***Βάση δεδομένων πληροφορίας πλαισίου χρήσης (context database):*** αποθηκεύει τις αναπαραστάσεις του πλαισίου χρήσης.
- ***Υπηρεσία εντοπισμού υπηρεσίας (service-locating service):*** μέσω αυτής της υπηρεσίας διαφημίζονται οι πάροχοι και οι διερμηνείς πληροφορίας πλαισίου χρήσης.

Σε υψηλότερο επίπεδο από τα παραπάνω τοποθετούνται οι υπηρεσίες στις οποίες παρέχεται η δυνατότητα ανάκτησης της εκάστοτε απαραίτητης πληροφορίας και της ανάλογης προσαρμογής.

Μια αξιόλογη λύση αποτελεί το WildCAT [12]. Το WildCAT είναι ένα επεκτάσιμο framework σε Java που έχει σκοπό να διευκολύνει τη δημιουργία εφαρμογών με επίγνωση του πλαισίου χρήσης. Παρέχει ένα απλό δυναμικό μοντέλο για την αναπαράσταση του πλαισίου χρήσης εκτέλεσης της εφαρμογής. Η πληροφορία πλαισίου χρήσης μπορεί να προσπελαστεί από τους προγραμματιστές μέσω δύο διεπαφών: σύγχρονες αιτήσεις (μηχανισμός pull) και ασύγχρονες ειδοποιήσεις ή notifications (μηχανισμός push). Εσωτερικά το WildCAT επιτρέπει διάφορα επίπεδα επεκτάσεων, από την απλή διαμόρφωση της γενικής υλοποίησης μέχρι τελείως νέες υλοποιήσεις για κάλυψη συγκεκριμένων αναγκών.

Ακόμα μια προσπάθεια σε αυτή την κατηγορία αποτελεί η αρχιτεκτονική μεσάζοντος πλαισίου χρήσης (Context Broker Architecture / CoBrA) [8] που είναι βασισμένη στην τεχνολογία των πρακτόρων (agent) για την υποστήριξη της παροχής υπηρεσιών προσαρμοσμένων στο πλαίσιο χρήσης σε “έξυπνους” χώρους (intelligent spaces). Η κεντρική οντότητα του συστήματος είναι ο *Context Broker*, ο οποίος ανακτά τις πληροφορίες πλαισίου χρήσης από ετερογενείς πηγές και τις προσαρμόζει βάσει της κατάλληλης αναπαράστασης προκειμένου να μπορούν να τις χρησιμοποιήσουν οι υπόλοιπες οντότητες του συστήματος, όπως π.χ. εφαρμογές που φιλοξενούνται στα κινητά τερματικά των χρηστών, υπηρεσίες που παρέχονται από συσκευές εντός του έξυπνου χώρου, κτλ. Ο *Context Broker* αποτελείται από τέσσερις λειτουργικές οντότητες: τη Βάση Γνώσης Πλαισίου Χρήσης (*Context Knowledge Base)*, τη Μηχανή Εξαγωγής Συμπερασμάτων

Πλαισίου Χρήσης (*Context Inference Engine)*, την Υπομονάδα Απόκτησης Πλαισίου Χρήσης (*Context Acquisition Module)* και την Υπομονάδα Διαχείρισης Ιδιωτικότητας (*Privacy Management Module)*.

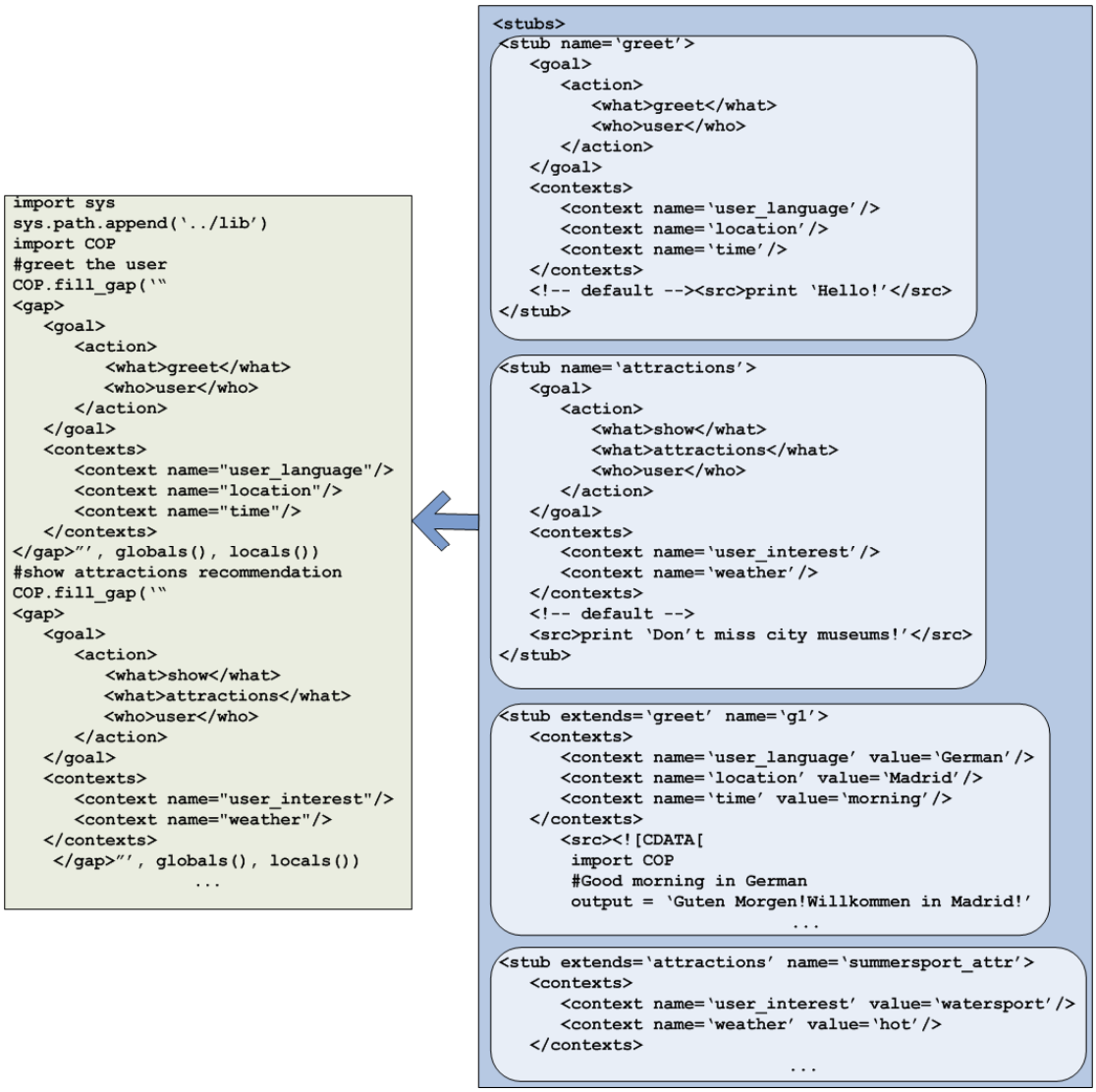
Κάποια άλλα παραδείγματα ειδικών πλατφόρμων για υπηρεσίες είναι τα [4], [30] και [16].

3.3 Προσαρμογή σε Επίπεδο Πηγαίου Κώδικα

Η πρώτη σημαντική προσπάθεια χειρισμού της πληροφορίας πλαισίου χρήσης σε επίπεδο πηγαίου κώδικα έγινε από τον Προσανατολισμένο στο Πλαίσιο Χρήσης Προγραμματισμό (Context-Oriented Programming / COP) [21]. Στον COP ο σκελετός της εφαρμογής (skeleton) παραμένει ανεξάρτητος και ανεπηρέαστος από την πληροφορία πλαισίου χρήσης, ενώ χωριστά κομμάτια κώδικα (stubs) που εξαρτώνται από το πλαίσιο χρήσης είναι υπεύθυνα για το χειρισμό του. Το πλαίσιο χρήσης αντιμετωπίζεται ως κομμάτι της βασικής δομής της γλώσσας προγραμματισμού, ενώ οι στόχοι (goals), οι "ανοιχτοί" όροι (open terms) και τα stub χρησιμοποιούνται σε συνδυασμό με το πλαίσιο χρήσης για να ενσωματώσουν την επίγνωση του πλαισίου χρήσης στη βασική εκτέλεση του σκελετού της εφαρμογής. Οι "ανοιχτοί" όροι αποτελούν κενά στη λογική του προγράμματος και αποτελούνται από το ρόλο μιας οντότητας (το οποίο αναφέρεται ως goal), την πληροφορία πλαισίου χρήσης και προαιρετικά ένα γεγονός που προκαλεί την εκτέλεση του "ανοιχτού" όρου. Η "γέμιση" πλαισίου χρήσης (context-filling) είναι η διαδικασία επιλογής του κατάλληλου stub και η ένωσή του με το αντίστοιχο κενό του προγράμματος, καθιστώντας με αυτόν τον τρόπο εφικτή την ενσωμάτωση της απαραίτητης συμπεριφοράς κατά την εκτέλεση της εφαρμογής. Αυτή η διαδικασία απαιτεί τη χρήση μιας εξωτερικής οντότητας, την αποθήκη των stub (stub repository), από την οποίο ανακτώνται τα κατάλληλα κομμάτια κώδικα για το σκελετό της εφαρμογής όταν πληρούνται συγκεκριμένες συνθήκες πλαισίου χρήσης. Ο COP έχει υλοποιηθεί ως μια επέκταση της γλώσσας προγραμματισμού Python [26], της οποίας τα χαρακτηριστικά την καθιστούν κατάλληλη για μια τέτοια λειτουργία προσαρμογής. Οι "ανοιχτοί" όροι και οι τύποι των stub αναπαρίστανται σε XML μορφοποίηση, ενώ η μεταξύ τους αντιστοίχιση βασίζεται σε αυτές τις XML αναπαραστάσεις.

Ένα παράδειγμα χρήσης του COP για την περίπτωση της τουριστικής υπηρεσίας που αναφέρθηκε παραπάνω παρουσιάζεται στην Εικόνα 12. Στην Εικόνα φαίνεται ο σκελετός της υπηρεσίας στη γλώσσα Python, όπου μια κλήση στη ρουτίνα `fill_gap()` ενεργοποιεί τη διαδικασία αντιστοίχισης με το πλαίσιο χρήσης. Ωστόσο, η υλοποίηση σε Python

παρουσιάζει κάποιους περιορισμούς καθώς μόνο οι δηλώσεις της γλώσσας (statements) θεωρούνται “ανοιχτοί” όροι.



Εικόνα 12. Σκελετός της υπηρεσίας με “ανοιχτούς” όρους στα αριστερά και τα stub για την αντιστοίχιση στα δεξιά.

Γενικά, η έννοια του COP χρησιμοποιήθηκε και σε άλλες ερευνητικές προσπάθειες για το χειρισμό του πλαισίου χρήσης. Μια τέτοια περίπτωση παρουσιάζεται στο [10], όπου οι συγγραφείς προτείνουν το σχεδιασμό συστημάτων με επίγνωση του πλαισίου χρήσης χρησιμοποιώντας μια προσέγγιση επιπέδων. Ο όρος “επίπεδο” (layer) αντιστοιχεί σε ένα σύνολο μερικής κλάσης και ορισμών μεθόδων του αντικειμενοστραφούς προγραμματισμού με συγκεκριμένη συμπεριφορά. Για να γίνει χρήση αυτών των επιπέδων, προστίθενται μικρά τμήματα κώδικα για να προκαλέσουν την ενεργοποίηση (“with”) ή την απενεργοποίησή (“without”) των επιπέδων. Με αυτόν τον τρόπο η συμπεριφορά του προγράμματος μπορεί να αλλάξει δυναμικά κατά την εκτέλεση. Σύμφωνα με αυτή την

προσέγγιση, οι εξαρτήσεις από το πλαίσιο χρήσης παραμένουν ανεξάρτητες από το βασικό ορισμό του προγράμματος. Αντί όμως να γίνεται χρήση ενός εξωτερικού μηχανισμού, όπως στην προηγούμενη περίπτωση, ο χειρισμός του πλαισίου χρήσης γίνεται από την ίδια τη γλώσσα προγραμματισμού που έχει δημιουργηθεί ως μια επέκταση του Common Lisp Object System (CLOS) [9] (ContextL). Σε πιο πρόσφατη εργασία ο ίδιος μηχανισμός επιπέδων παρουσιάζεται πιο αφαιρετικά, ώστε να είναι δυνατός ο συνδυασμός του με διάφορες γλώσσες προγραμματισμού, όπως οι γλώσσες Java και Squeak/Smalltalk [18]. Για το παράδειγμα της τουριστικής υπηρεσίας χρειάζεται να οριστούν διάφορα επίπεδα προκειμένου να εκφραστούν όλες οι περιπτώσεις προσαρμογής. Κάποια από αυτά παρουσιάζονται στην Εικόνα 13 στη γλώσσα ContextJ*, που αποτελεί το πρωτότυπο της προσέγγισης για Java (π.χ. το επίπεδο *GoodWeather* χρησιμοποιείται για την προσθήκη των υπαίθριων δραστηριοτήτων στη λίστα των δραστηριοτήτων όταν είναι ενεργό).

```
//prototype implementation library
import static
be.ac.vu.prog.contextj.ContextJ.*;
//File: Layers.java
public class Layers {
  public static final Layer Location = new
Layer();
  public static final Layer Spanish = new
Layer();
  . . .
  public static final Layer GoodWeather = new
Layer();
  . . .
}
//File: IAttractionsService.java
public interface IAttractionsService {
  public List recommendAttractions();
  public List indoorAttractions();
  public List outdoorAttractions();
}
//File: AttractionsService.java
public class AttractionsService implements
iAttractionsService {
  private LayerDefinitions<IAttractionsService>
layers =
    new LayerDefinitions<IAttractionsService>();
  public List recommendAttractions(){
    return
layers.select().recommendAttractions();
  }
  public List indoorAttractions() { ... }
  public List outdoorAttractions() { ... }
  {
    layers.define(RootLayer,
        new IAttractionsService () {
          public List recommendAttractions() {
            return indoorAttractions();
          }
        }
```

```
//File: IGreetingService.java
public interface IGreetingservice {
  public String greetUser();
}
//File: GreetingService.java
public class Greetingservice
implements iGreetingService{
  String userName; String cityName;
  private
LayerDefinitions<IGreetingService>  layers
=
    new
LayerDefinitions<IGreetingService>();
  public String greetUser() {
    return layers.select().greetUser();
  }

  {
    layers.define(RootLayer,
        new IGreetingService() {
          public String greetUser() {
            return "Hello " + userName;
          }
        }
    );
    layers.define(Layers.Location,
      new IGreetingService() {
        public String greetUser() {
return layers.next(this)+".Welcome in " +
cityName;
        }
      }
    );
    layers.define(Layers.Spanish,
      new IGreetingService() {
        public String greetUser() {
          return "Hola " + userName;
        }
      }
```

```
    );
    layers.define(Layers.GoodWeather,
          new IAttractionsService () {
              public List recommendAttractions() {
                 return
layers.next(this).addAll(outdoorAttractions());
              }
          }
    ); . . . } }
```

```
        );
    . . .
    }
}
```

Εικόνα 13. Ορισμοί επιπέδων για διάφορες περιπτώσεις της υπηρεσίας δραστηριοτήτων (αριστερά) και χαιρετισμού (δεξιά).

Στην περίπτωση ενός πιο πολύπλοκου σκελετού υπηρεσίας, η πληροφορία πλαισίου χρήσης απαιτείται σε διάφορα σημεία της εφαρμογής και μπορεί να ενεργοποιήσει την κλήση πρόσθετων μεθόδων. Υπό αυτό το πρίσμα ο χειρισμός της πληροφορίας πλαισίου χρήσης γίνεται μια υπόθεση (concern) που σχετίζεται και εμπλέκεται με διάφορα τμήματα της εφαρμογής κόβωντας ή διακόπτωντας σε διάφορα σημεία την εκτέλεση της (crosscutting). Μια προγραμματιστική προσέγγιση που στοχεύει στο χειρισμό αυτών των υποθέσεων που εμπλέκονται σε διάφορα κομμάτια της εφαρμογής και αναφέρονται ως aspects είναι ο Θεματοστρεφής Προγραμματσμός (Aspect Oriented Programming / AOP) [13]. Ο AOP έχει σχεδιαστεί σε πολλές περιπτώσεις ως επέκταση ευρέως διαδεδομένων γλωσσών προγραμματισμού (π.χ. AspectJ [3] [23], AspectC++ [2], κτλ.). Στις γλώσσες αυτές ορίζονται οι έννοιες των joinpoint (συνδετικών σημείων) που ορίζουν σημεία στο πρόγραμμα όπου μπορεί να εισαχθεί η πρόσθετη συμπεριφορά από τα aspect, των pointcut (σημείων διακοπής) που ορίζουν εκφρασεις για τον εντοπισμό των joinpoint μέσα στον κώδικα και των advice (συμβουλών) που περιγράφουν τον κώδικα που θα εφαρμοσθεί στα joinpoint (σε περίπτωση που πληρούνται οι συνθήκες). Τα aspect αποτελούνται από ένα pointcut και την αντίστοιχη advice.

Κάνοντας χρήση του AOP ο χειρισμός της πληροφορίας πλαισίου χρήσης μπορεί να γίνει μέσω των aspect που διακόπτουν την κύρια εκτέλεση της υπηρεσίας, ώστε να επιτύχουν την προσαρμογή της. Αυτή η προσέγγιση είναι ενδιαφέρουσα για τους μηχανικούς υπηρεσιών ειδικά όταν συνδυάζεται με άλλες μεθοδολογίες όπως την προσέγγιση Theme [6] που βοηθάει να εντοπιστούν όσο νωρίτερα γίνεται οι πιθανές λειτουργίες του συστήματος που αποτελούν aspects (αναφερόμενες ως “themes”) και οι βασικές λειτουργίες (αναφερόμενες ως “base themes”).

Στο [29] προτείνεται η προσαρμογή των aspect στο πλαίσιο χρήσης περιγράφοντας τα aspect με επίγνωση του πλαισίου χρήσης (context-aware aspects). Αυτό σημαίνει ότι η χρήση των aspect κατευθύνεται από το πλαίσιο χρήσης και κάθε aspect μπορεί να εκτελεστεί ή όχι ανάλογα με το πλαίσιο χρήσης. Ακόμα θεωρείται ότι η πληροφορία

πλαισίου χρήσης που κατευθύνει την κλήση των αντίστοιχων aspect μπορεί να έχει παραμέτρους (π.χ. ώρα ή τοποθεσία) ή να συνδυαστεί με άλλες τιμές πλαισίου χρήσης. Ωστόσο, η σημαντική συνεισφορά της συγκεκριμένης προσέγγισης είναι ότι η συμπεριφορά της υπηρεσίας μπορεί να επηρεαστεί όχι μόνο από τις τρέχουσες τιμές του πλαισίου χρήσης, αλλά και από παλαιότερες τιμές, που αναφέρονται ως "στιγμιότυπα" πλαισίου χρήσης (context snapshotting).

Τα context-aware aspect υποστηρίζονται από ένα framework που έχει υλοποιηθεί ως κομμάτι του Reflex [28]. Το Reflex αποτελεί μια ανοιχτή επέκταση της Java που παρέχει τις απαιτούμενες δομές για τη διευκόλυνση της χρήσης των αρχών του AOP. Επιτρέπει τη λειτουργία δυναμικού crosscutting, που επιτυγχάνεται μέσω των hookset του Reflex που λειτουργούν όπως τα pointcut του AOP και μπορούν να καθορίσουν τις συνθήκες για το πότε να σταλεί ένα συγκεκριμένο μήνυμα σε ένα μετα-αντικείμενο επιτυγχάνοντας με αυτόν τον τρόπο εκτέλεση τμημάτων κώδικα ή μεθόδων υπό συγκεκριμένες συνθήκες. Η πληροφορία πλαισίου χρήσης αναπαρίσταται μέσω αντικειμένων με μια μέθοδο που δηλώνει την κατάσταση ενεργοποίησής της (επιστρέφοντας μια `Boolean` τιμή). Στο Reflex είναι διαθέσιμες διάφορες δομές για την έκφραση συνθηκών ενεργοποίησης βάσει του τρέχοντος ή του παλαιότερου πλαισίου χρήσης, όπως οι αφαιρετικές κλάσεις `CtxActive` και `SnapshotCtxActive`.

Ως παράδειγμα χρήσης των context-aware aspect θεωρείται μια υπηρεσία, η οποία περιλαμβάνει ένα προεπιλεγμένο μήνυμα στα αγγλικά και κάποια aspect με άλλους χαιρετισμούς που εξαρτώνται από τη γλώσσα του χρήστη (π.χ. αν ένας χρήστης είναι από τη Μαδρίτη, καλείται ο αντίστοιχος χαιρετισμός στα ισπανικά). Σε αυτή την περίπτωση για την περαιτέρω προσαρμογή του μηνύματος χαιρετισμού μπορεί να χρησιμοποιηθεί ως παράμετρος η τοπική ώρα. Έτσι, μπορεί να θεωρηθεί ότι αυτή η παράμετρος του χρόνου εκφράζει την κατάσταση του πλαισίου χρήσης, καθώς η τιμή της δεν παραμένει ίδια ανάμεσα στις διαφορετικές κλήσεις της ίδιας υπηρεσίας (το ίδιο ισχύει αν χρησιμοποιηθεί ως παράμετρος η τοποθεσία του χρήστη). Ομοίως, η υπηρεσία προτεινόμενων δραστηριοτήτων μπορεί να επιστρέφει μια λίστα με όλες τις δραστηριότητες για εσωτερικούς χώρους στην τοποθεσία του χρήστη που μπορεί να εμπλουτιστεί και με τις δραστηριότητες για εξωτερικούς χώρους σε περίπτωση καλών καιρικών συνθηκών. Στην Εικόνα 14 φαίνονται οι ανάλογοι ορισμοί για τα context `SpanishGreetingCtx` και `GoodWeatherCtx`, καθώς και τα pointcut που εκφράζουν την εκτέλεση των αντίστοιχων aspect.

```
//time and city specify the context state
Class SpanishGreetingState extends ContextState {
```

```
   Time time;
   String city;
   Time getLocalTime() {
      . . .
      return time;
   }
   String getUserCity() {
      . . .
      return city;
   }
}

Class SpanishGreetingCtx extends Context {
//The control flow of the hookset matching requests to greetUser operation is exposed
   CFlow cf = CFlowFactory.get(new Hookset(MsgReceive.class,
         new NameCS("WebServiceRequest", true),
         new  NameOS(("greetUser", true)));

   Time time; //with getTime and setTime methods
   String city; // with getCity and setCity methods
   boolean active() {
      return cf.in();
    }

   ContextState getState() {
      if (!cf.in()) return null;
            return new SpanishGreetingState(time, city);
          }
}

Class GoodWeatherCtx extends Context {
   CFlow cf = CFlowFactory.get(new Hookset(MsgReceive.class,
         new NameCS("WebServiceRequest", true),
         new  NameOS(("showAttractions", true)));

   boolean active() {
      return cf.in();
    }

//Relevant pointcuts for the example: they detect the joinpoints of calls to greetUser
and showAttractions that match the SpanishGreetingCtx and GoodWeatherCtx
accordingly
pointcut userLoc(): execution(*
*..*.greetUser(..))&&inContext(SpanishGreetingCtx(time));

pointcut weather(): execution(*
     *..*.showAttractions(..))&&inContext(GoodWeatherCtx());

...
```

Εικόνα 14. Ορισμός κώδικα για το πλαίσιο χρήσης μέσω context-aware aspect.

Ένα κατάλληλο παράδειγμα για το χαρακτηριστικό του στιγμιοτύπου πλαισίου χρήσης που προτείνεται στα context-aware aspect είναι αυτό του διαδικτυακού καταστήματος. Θεωρείται ότι ένα νέο αντικείμενο καλαθιού αγοράς δημιουργείται κάθε φορά που ένας χρήστης εισάγει τα στοιχεία του στο σύστημα και ότι κάθε στιγμή υπάρχουν διάφορες ενεργές προσφορές στο κατάστημα (active contexts). Κάνοντας χρήση των

προηγούμενων καταστάσεων του πλαισίου χρήσης (και όχι της τρέχουσας κατάστασης), ειδικές τιμές μπορεί να προσφερθούν στους πελάτες βάσει του context snapshot τη στιγμή της σύνδεσής τους στο κατάστημα. Με άλλα λόγια, τα στοιχεία του πλαισίου χρήσης εγγράφονται ως ένα στιγμιότυπο τη στιγμή της σύνδεσης και οι τιμές που προσφέρονται δεν εξαρτώνται από τις ενεργές προσφορές, αλλά από τις ειδικές προσφορές που ήταν σε ισχύ τη στιγμή της σύνδεσης του χρήστη.

Η λειτουργία του στιγμιοτύπου πλαισίου χρήσης μπορεί να εμπλουτιστεί συνδυάζοντας την υπηρεσία με μια τεχνική ενεργοποίησης βάσει κανόνων. Αυτή η προσέγγιση έχει ακολουθηθεί στο [19] όπου γίνεται χρήση υβριδικών aspect (Hybrid Aspects) που συνδυάζουν χαρακτηριστικά του AOP και του rule-based reasoning. Τα υβριδικά aspect καθορίζουν joinpoint για τα σημεία διακοπής της κανονικής εκτέλεσης του προγράμματος και εκτέλεσης της αντίστοιχης advice. Η υβριδική advice είναι ανεξάρτητη από τη γλώσσα προγραμματισμού που χρησιμοποιείται για το βασικό κορμό του προγράμματος και υποστηρίζει την αποστολή μηνυμάτων, αλλά και την ενεργοποίηση κανόνων. Για την υλοποίηση των υβριδικών aspect έχει χρησιμοποιηθεί το AspectS framework για AOP [17].

Μια διαφορετική τεχνική προγραμματισμού που αποτελεί μια επέκταση του αντικειμενοστραφούς προγραμματισμού και ονομάζεται Ισότοπο Προγραμματιστικό Μοντέλο (Isotope Programming Model / IPM) παρουσιάζεται στο [27]. Σύμφωνα με το IPM ο αντικειμενοστραφής προγραμματισμός μπορεί να μεταβληθεί, ώστε η συμπεριφορά των αντικειμένων να μην είναι στατική, αλλά προσαρμόσιμη σε διαφορετικά πλαίσια χρήσης. Υπό αυτή την έννοια ο κώδικας αλλάζει βάσει του πλαισίου χρήσης, ενώ διατηρείται συγχρόνως ανεξάρτητος από αυτή την πληροφορία περιβάλλοντος. Τα αντικείμενα ορίζονται από ένα σύνολο ιδιοτήτων και προεπιλεγμένων (default) μεθόδων όπως στο παραδοσιακό αντικειμενοστραφές μοντέλο (main element) και ένα σύνολο ισότοπων στοιχείων (isotope elements). Τα ισότοπα στοιχεία περιγράφουν τη δομή του κώδικα σε διαφορετικά περιβάλλοντα. Αποτελούνται από ένα τμήμα πλαισίου χρήσης που καθορίζει τις συνθήκες ενεργοποίησης και ένα τμήμα συμπεριφοράς που περιλαμβάνει τις μεθόδους του ισότοπου στοιχείου. Το πιο πρόσφατο και το πιο κατάλληλο ισότοπο στοιχείο επιλέγεται σε κάθε εκτέλεση. Αν αυτό το στοιχείο δεν υπάρχει καλείται η προεπιλεγμένη μέθοδος (αν υπάρχει). Καθώς αλλάζει το περιβάλλον εκτέλεσης, νέα ισότοπα στοιχεία μπορούν να προστεθούν ή κάποια από τα υπάρχοντα να αφαιρεθούν.

3.4 Προσαρμογή μέσω "Σύλληψης" Μηνυμάτων

Η τεχνική της "σύλληψης" μηνυμάτων αποτελεί μια ιδιαίτερα ενδιαφέρουσα κατηγορία, καθώς μπορεί να εφαρμοσθεί σε υπηρεσίες διαδικτύου. Γενικά, για τη χρήση "σύλληψης" μηνυμάτων απαιτείται η ύπαρξη μηνυμάτων που να επηρεάζουν την εκτέλεση της υπηρεσίας. Κάτι τέτοιο συμβαίνει στις ευρέως διαδεδομένες υπηρεσίες διαδικτύου, αν και η τεχνική μπορεί να εφαρμοσθεί και σε άλλες τεχνολογίες (π.χ. CORBA) ή ακόμα και σε γλώσσες προγραμματισμού μέσω της "σύλληψης" αντικειμένων. Όπως έχει αναφερθεί, στη διατριβή μας ενδιαφέρει η περίπτωση των υπηρεσιών διαδικτύου στις οποίες και επικεντρωνόμαστε.

Οι αιτήσεις και οι απαντήσεις των υπηρεσιών διαδικτύου μεταφέρονται μεταξύ των πελατών και των παρόχων μέσα σε φακέλους που ακολουθούν τη μορφοποίηση του SOAP πρωτοκόλλου. Η διαχείριση της πληροφορίας πλαισίου χρήσης που σχετίζεται με μια συγκεκριμένη υπηρεσία μπορεί να γίνει σε πρώτο στάδιο μέσω της επέκτασης της σύνταξης των SOAP μηνυμάτων, ώστε να συμπεριληφθεί στα μηνύματα η πληροφορία για το πλαίσιο χρήσης, και εν συνεχεία μέσω της "σύλληψης" αυτών των μηνυμάτων κατά τη μεταφορά τους στο δίκτυο και της κατάλληλης επεξεργασίας της πληροφορίας που περιέχεται σε κάθε μήνυμα. Με τον τρόπο αυτό, η διαχείριση του πλαισίου χρήσης μπορεί να παραμείνει σε μεγάλο βαθμό ανεξάρτητη από τις ίδιες τις υπηρεσίες διαδικτύου. Αυτή η προσέγγιση ακολουθείται και παρουσιάζεται στο [22], όπου όλοι οι τύποι που περιγράφουν την πληροφορία πλαισίου χρήσης περιλαμβάνονται ως επέκταση της επικεφαλίδας των SOAP μηνυμάτων, αφού έχει προηγηθεί ο κατάλληλος ορισμός τους ως tModels.

Τα tModels είναι ένας τρόπος δημιουργίας ορισμών διεπαφών υπηρεσιών σε XML μορφοποίηση που μπορεί να χρησιμοποιηθεί για την ανακάλυψη υπηρεσιών σύμφωνα με το μοντέλο της UDDI. Οι επεκτάσεις περιλαμβάνουν τμήματα πλαισίου χρήσης που αποτελούνται από ένα μοναδικό αναγνωριστικό για τον τύπο της πληροφορίας και την αντίστοιχη τιμή. Ένα παράδειγμα με αυτή την προσέγγιση για την περίπτωση της τουριστικής υπηρεσίας παρουσιάζεται στην Εικόνα 15. Οι επεκτάσεις για το πλαίσιο χρήσης στην εικόνα περιλαμβάνουν πληροφορία για την τοποθεσία, την ώρα, τις προτιμήσεις του χρήστη και τις καιρικές συνθήκες.

Η διαχείριση της πληροφορίας πλαισίου χρήσης πραγματοποιείται αυτόματα μέσω χειριστών που είτε περιλαμβάνονται ως πρόσθετοι μηχανισμοί (plugin) στο framework, είτε παρέχονται απομακρυσμένα από ειδικές για το σκοπό αυτό υπηρεσίες διαδικτύου. Οι χειριστές "συλλαμβάνουν" τα SOAP μηνύματα που ανταλλάσσονται κατά την εκτέλεση της υπηρεσίας και μεταβάλλουν καταλλήλως τις επεκτάσεις της επικεφαλίδας για το πλαίσιο

χρήσης. Η διαχείριση των μηνυμάτων πραγματοποιείται είτε στην πλευρά του πελάτη είτε στην πλευρά της υπηρεσίας κάνοντας χρήση μιας ειδκής βιβλιοθήκης για το πλαίσιο χρήσης που υλοποιήθηκε για το σκοπό αυτό (Context API). Για να καθοριστεί ποιες οντότητες έχουν το δικαίωμα να τροποποιήσουν την πληροφορία πλαισίου χρήσης και επιπλεόν ποιους από τους τύπους της πληροφορίας παρέχονται κατάλληλες οδηγίες επεξεργασίας, οι οποίες περιέχουν αναφορές στις υπηρεσίες που είναι υπεύθυνες για αυτήν την επεξεργασία του πλαισίου χρήσης, αλλά και πρόσθετες πληροφορίες για τη διαδικασία της επεξεργασίας. Το βασικό πλεονέκτημα αυτής της προσέγγισης είναι ότι επιτυγχάνεται ο χειρισμός της πληροφορίας πλαισίου χρήσης που βρίσκεται στην επικεφαλίδα του SOAP μηνύματος ανεξάρτητα από τη βασική λειτουργία της υπηρεσίας διαδικτύου. Ωστόσο, δε γίνεται λόγος για προσαρμογή των υπηρεσιών, αλλά η προσέγγιση αφορά περισσότερο την επεξεργασία και τροποποίηση της ενσωματωμένης πληροφορίας πλαισίου χρήσης.

```
<!--Greeting Service-->
<soapenv:Envelope>
  …
 <soapenv:Header>
   <Context xmlns="http://sg.fmi.uni-
passau.de/context">
     <Language>German</Language>
     <Location>
       <address>
         <addressLine keyname="Street"
keyValue="60">San Miguel
18</addressLine>
    <addressLine keyname="City"
keyValue="40">Madrid</addressLine>
    <addressLine keyname="Country"
keyValue="20">Spain</addressLine>
       </address>
     </Location>
     <Time>
       <localTime>19:00</localTime>
     </Time>
   </Context>
 </soapenv:Header>
 <soapenv:Body>
...
 </soapenv:Body>
</soapenv:Envelope>
```

```
<!--Attractions Service-->
<soapenv:Envelope>
 …
 <soapenv:Header>
   <Context xmlns="http://sg.fmi.uni-
passau.de/context">
     <ActivityInterest>
       <pref1>History</pref1>
       <pref2>WaterSports</pref2>
     </ActivityInterest>
     <Weather>
       <localTemp>27C</localTemp>
       <rainLikely>0.005</rainLikely>
     </Weather >
   </Context>
 </soapenv:Header>
 <soapenv:Body>
...
 </soapenv:Body>
</soapenv:Envelope>
```

Εικόνα 15. Παραδείγματα SOAP φακέλων με επεκτάσεις για την πληροφορία πλαισίου χρήσης.

Μια παρόμοια προσέγγιση “σύλληψης” μηνυμάτων προτείνεται και στο [24]. Εδώ συνδυάζεται η χρήση του AOP με τη “σύλληψη” μηνυμάτων για να επιτευχθεί και πάλι η ανεξαρτητοποίηση της διαχείρισης του πλαισίου χρήσης από τη βασική λειτουργία της υπηρεσίας, ενώ η προσαρμογή πραγματοποιείται στην πλευρά του πελάτη.

Η προσέγγιση που χρησιμοπείται για το χειρισμό του πλαισίου χρήσης στη διατριβή έχει επηρεαστεί και από τις δύο παραπάνω προσεγγίσεις. Ωστόσο, επιχειρείται να αντιμετωπιστούν κάποια επιπλέον ζητήματα που δε λαμβάνοντα υπόψη στις παραπάνω περιπτώσεις, αλλά και να απλοποιηθεί η διαδικασία διαχείρισης οδηγώντας σε μια κατανεμημένη αρχιτεκτονική χωρίς περιττή επεξεργασία μηνυμάτων ούτε πολύπλοκους μηχανισμούς διαχείρισης. Έτσι, αποφεύγεται η προσθήκη πληροφοριών για το πλαίσιο χρήσης στην επικεφαλίδα, η οποία και χρησιμοποιείται μόνο για την εξακρίβωση της ταυτότητας του χρήστη όταν αυτό απαιτείται.

Ένα γενικό μοντέλο για την προσθήκη πληροφοριών πλαισίου χρήσης σε μηνύματα που δε σχετίζεται με συγκεκριμένες τεχνολογίες προτείνεται από το αφηρημένο εξαρτημένο από το πλαίσιο χρήσης Μοντέλο Ρόλων (Context-Dependent Role Model / CDR Model) που περιγράφεται στο [31]. Σε αυτή την προσέγγιση οι διάφορες οντότητες αντιδρούν σε αλλαγές στο πλαίσιο χρήσης υιοθετώντας ένα συγκεκριμένο ρόλο ανά περίπτωση. Αυτό επιτυγχάνεται μέσω ενός εξαρτημένου από το πλαίσιο χρήσης επιλογέα (context-dependent role selector), μια ειδική οντότητα που επιλέγει τον κατάλληλο ρόλο βάσει της πληροφορίας πλαισίου χρήσης που σχετίζεται με τον αποστολέα και τον παραλήπτη του μηνύματος. Η πληροφορία πλαισίου χρήσης εισάγεται στα μηνύματα από τον πληρεξούσιο αναφοράς πλαισίου χρήσης (context reference proxy), ο οποίος συλλαμβάνει τα εξερχόμενα μηνύματα και μπορεί να προσθέσει πληροφορία πλαισίου χρήσης σε αυτά προτού τα αποστείλει στον παραλήπτη. Αυτό γίνεται βάσει κανόνων που καθορίζουν υπό ποιες συνθήκες προστίθεται η πληροφορία πλαισίου χρήσης.

Δεν είναι μόνο σε προσεγγίσεις όπως στις παραπάνω όπου οι υπηρεσίες διαδικτύου χρησιμοποιούνται για το χειρισμό της πληροφορίας πλαισίου χρήσης. Στο [1] περιγράφεται η υλοποίηση της υπηρεσίας Πυρήνας πλαισίου χρήσης (Context Kernel), η οποία επιτρέπει στις εφαρμογές να αποθηκεύουν και να ανταλλάσσουν πληροφορία πλαισίου χρήσης. Άλλες υπηρεσίες διαδικτύου μπορούν να εκμεταλλευτούν τους μηχανισμούς ανάκτησης και αποθήκευσης του context kernel για να αποκτήσουν τις πληροφορίες και να τις ενσωματώσουν στη δική τους λειτουργία.

3.5 Σύνοψη

Κάθε κατηγορία από τις παραπάνω παρουσιάζει διάφορα πλεονεκτήματα και μειονεκτήματα. Η διαχείριση του πλαισίου χρήσης σε επίπεδο πηγαίου κώδικα προσδίδει μεγαλύτερη ελευθερία και κατ' επέκταση ευελιξία κατά την ανάπτυξη της υπηρεσίας, αλλά επίσης κληρονομεί τα δυνατά σημεία, αλλά και τους περιορισμούς της γλώσσας

προγραμματισμού που έχει χρησιμοποιηθεί για την υλοποίηση του μηχανισμού προσαρμογής. Το ίδιο ισχύει και για τις ειδικές πλατφόρμες, όπου πρέπει να ακολουθούνται οι σχεδιαστικές αρχές της πλατφόρμας κατά την παροχή των υπηρεσιών. Οι προσεγγίσεις με "σύλληψη" μηνυμάτων, τέλος, παρεμβαίνουν σε μικρότερο ποσοστό στην εργασία του προγραμματιστή όσον αφορά το σκελετό της υπηρεσίας, αλλά απαιτούν διάφορες αλλαγές στο σύστημα (π.χ. δημιουργία χειριστών SOAP μηνυμάτων) για να λειτουργήσουν πλήρως.

Στον παρακάτω πίνακα παρουσιάζονται συνοπτικά τα χαρακτηριστικά των πιο αντιπροσωπευτικών προσεγγίσεων των δύο τελευταίων κατηγοριών, οι οποίες είναι σχετικές με το πνεύμα της προσαρμογής που επιχειρείται στη διατριβή (Πίνακας 3).

Προσέγγιση	**Υποστήριξη Ιστορικών Δεδομένων**	**Υποκείμενη Πλατφόρμα/ Υλοποίηση**	**Σημείο Προσαρμογής**
COP [21]	όχι	Python	εξυπηρετητής
Layer Activation [18]	όχι	CLOS, Java, Squeak/Smalltalk	εξυπηρετητής
Context-aware aspects [29]	ναι	Reflex	εξυπηρετητής
IPM [27]	όχι	OOP	εξυπηρετητής
Hybrid aspects [19]	μερικώς	AspectS	–
Towards adaptable web services [22]	όχι	υπηρεσίες διαδικτύου	εξυπηρετητής, πελάτης
Context-Dependent Role Model [31]	όχι	AmbientTalk	εξυπηρετητής, πελάτης

Πίνακας 3. Σύνοψη βασικών χαρακτηριστικών αντιπροσωπευτικών προσεγγίσεων.

3.6 Βιβλιογραφία

[1] Arruda Jr, C. R. E., Neto, R. B., Pimentel, M. da G., "Open context-aware storage as a Web Service", Proc. International Conference on Distributed Systems Platforms and Open Distributed Processing, 2003, pp. 81-87.

[2] AspectC++, http://www.aspectc.org/.

[3] AspectJ, http://www.eclipse.org/aspectj/.

[4] Athanasopoulos, D., Issarny, V., Pitoura, E., Vassiliadis, P., Zarras, A., “Mobile Web Services for Context-Aware Pervasive Environments”, ACM Transactions on Internet Technology, December 2005.

[5] Baldauf, M., Dustdar, S., Rosenberg, F., “A Survey on Context-Aware Systems”, International Journal of Ad Hoc and Ubiquitous Computing, vol. 2(4), 2007, pp. 263-277.

[6] Baniassad, E., Clarke, S., “Theme: An Approach for Aspect-Oriented Analysis and Design”, Proc. International Conference on Software Engineering (ICSE’04), 2004, pp. 158-167.

[7] Chen, G., Kotz, D., “A Survey of Context-Aware Mobile Computing Research”, Technical Report: TR2000-381, Dartmouth College Hanover, NH, USA, 2000.

[8] Chen, H., Finin, T., Joshi, A., "A Context Broker for Building Smart Meeting Rooms", Proc. Knowledge Representation and Ontology for Autonomous Systems Symposium, (AAAI Spring Symposium’04), 2004, pp. 53-60.

[9] Common Lisp Object System (CLOS), http://www.dreamsongs.com/CLOS.html.

[10] Costanza, P., Hirschfeld, R., “Language Constructs for Context-oriented Programming: An overview of ContextL”, Proc. 2005 conference on Dynamic Languages Symposium, 2005, pp. 1-10.

[11] Daniele, L., Dockhorn Costa, P., Ferreira Pires, L., “Towards a Rule-Based Approach for Context-Aware Applications“, Proc. 13th Open European Summer School and IFIP TC6.6 Workshop (EUNICE’07), 2007, pp. 33-43.

[12] David, P.-C., Ledoux, T., “WildCAT: a generic framework for context-aware applications”, Proc. 3rd international workshop on Middleware for pervasive and ad-hoc computing (MPAC’05),2005, pp. 1-7.

[13] Elrad, T., Filman, R. E., Bader, A., ”Aspect-Oriented Programming”, Communications of the ACM, vol. 44(10), 2001, pp. 28-32.

[14] Fahy, P., Clarke, S., “CASS – Middleware for Mobile Context-Aware Applications”, Proc. Workshop on Context Awareness in MobiSys 2004, 2004.

[15] Gu, T., Pung, H. K., Zhang, D. Q., "A Middleware for Building Context-Aware Mobile Services", Proc. Vehicular Technology Conference (VTC'04), 2004, vol. 5, pp. 2656-2660.

[16] Han, B., Jia, W., Shen, J., Yuen, M.-C., "Context-Awareness in Mobile Web Services", Parallel and Distributed Processing and Applications, Springer-Verlag , 2008, pp. 519-528.

[17] Hirschfeld R., "AspectS - Aspect-Oriented Programming with Squeak", Lecture Notes in Computer Science: Objects, Components, Architectures, Services, and Applications for a Networked World: International Conference NetObjectDays (NODe POOP), 2002, pp. 216-232.

[18] Hirschfeld, R., Costanza, P., Nierstrasz, O., "Context-oriented Programming", Journal of Object Technology, vol. 7(3), March/April 2008, pp. 125-151.

[19] Hondt, M. D., Jonckers V., "Hybrid Aspects for Weaving Object-Oriented Functionality and Rule-Based Knowledge", Proc. 3rd international conference on Aspect-oriented software development (AOSD'04), 2004, pp. 132-140.

[20] Indulska, J., Sutton, P., "Location management in pervasive systems", Proc. Australasian Information Security Workshop (CRPITS'03), 2003, pp.143-151.

[21] Keays, R., Rakotonirainy, A., "Context-Oriented Programming", Proc. 3rd ACM international workshop on Data engineering for wireless and mobile access, 2003, pp. 9-16.

[22] Keidl, M., Kemper, A., "Towards Context-Aware Adaptable Web Services", Proc. 13th international World Wide Web conference (WWW´04), New York, NY, USA, 2004, pp. 55-65.

[23] Laddad, R., "AspectJ in Action, Practical Aspect-Oriented Programming Programming", Manning Publications Co., 2003.

[24] Prezerakos, G. N., Tselikas, N., Cortese, G., "Model-driven Composition of Context-aware Web Services Using ContextUML and Aspects", Proc. IEEE International Conference on Web Services 2007 (ICWS'07), 2007, pp. 320-329.

[25] Priyantha, N. B., Chakraborty, A., Balakrishnan, H., "The Cricket location-support system", Proc. Sixth Annual ACM International Conference on Mobile Computing and Networking, 2000, pp. 32-43.

[26] Python, http://www.python.org/.

[27] Saiyu, Q., Min, X., Yong, Q., "Isotope Programming Model for Context aware Application", International Journal of Software Engineering and Its Applications, vol. 1(1), July 2007, pp. 53-66.

[28] Tanter, E., Noye, J., "A Versatile Kernel for Multi-Language AOP", Proc. ACM SIGPLAN/SIGSOFT Conference on Generative Programming and Component Engineering (GPCE'05), LNCS, Springer-Verlag, 2005, pp. 173-188.

[29] Tanter, E., Gybels, K., Denker, M., Bergel, A., "Context-Aware Aspects", Proc. Software Composition 2006, LNCS 4089, Springer-Verlag, 2006, pp. 227-242.

[30] Truong, H. L., Juszczyk, L., Manzoor, A., Dustdar, S., "ESCAPE - An Adaptive Framework for Managing and Providing Context Information in Emergency Situations", Proc. Smart Sensing and Context, Second European Conference (EuroSSC'07), Springer-Verlag, 2007, pp. 207-222.

[31] Vallejos, J., Ebraert, P., Desmet, B., Cutsem, T. V., Mostinckx, S., Costanza, P., "The Context-Dependent Role Model", Proc. International Conference on Distributed Applications and Interoperable Systems (DAIS'07), 2007, LNCS 4531, Springer Verlag, vol. 4531, pp. 1-16.

4ο Κεφάλαιο

Προτεινόμενη Αρχιτεκτονική

Εχοντας ως αφετηρία το περιεχόμενο των προηγούμενων Κεφαλαίων η διατριβή περιλαμβάνει την υλοποίηση μιας αρχιτεκτονικής που προσφέρει ένα αποδοτικό χειρισμό της πληροφορίας πλαισίου χρήσης για τις υπηρεσίες διαδικτύου και έχει ως αποτέλεσμα την προσαρμογή των υπηρεσιών. Ιδιαίτερη βαρύτητα δίνεται στο διαχωρισμό της διαδικασίας προσαρμογής από τη λογική της υπηρεσίας διαδικτύου. Μάλιστα ο διαχωρισμός αυτός αποτέλεσε τη βασική σχεδιαστική αρχή για το προτεινόμενο σύστημα. Η προσαρμογή στο πλαίσιο χρήσης είναι στην ουσία ανεξάρτητη από τη βασική λειτουργία της υπηρεσίας και είναι σημαντικό να μπορεί μια εφαρμογή να λειτουργεί απρόσκοπτα είτε χωρίς είτε με προσαρμογή στο πλαίσιο χρήσης. Παρόλο που η αλληλεπίδραση μεταξύ του χρήστη και της υπηρεσίας μπορεί να προσαρμοστεί στις παραμέτρους του πλαισίου χρήσης, οι γενικοί στόχοι της λογικής της υπηρεσίας είναι ανεξάρτητοι από αυτό. Ως παράδειγμα μπορεί να θεωρηθεί μια υπηρεσία που δίνει τη δυνατότητα κράτησης αεροπορικών εισιτηρίων. Ανάλογα με τις παραμέτρους του πλαισίου χρήσης (π.χ. οικονομικοί περιορισμοί, αγαπημένες αεροπορικές εταιρίες, κτλ.) διάφορες μέθοδοι μπορούν να κληθούν. Σε κάθε περίπτωση όμως ο στόχος της υπηρεσίας παραμένει ο ίδιος ανεξάρτητα από το συγκεκριμένο πλαίσιο χρήσης.

Στη διατριβή ο χειρισμός της πληροφορίας πραγματοποιείται μέσω της “σύλληψης” των SOAP μηνυμάτων των υπηρεσιών και της κατάλληλης τροποποίησής τους. Η προσέγγιση αυτή έχει επηρεαστεί από τις αντίστοιχες προσεγγίσεις “σύλληψης” που παρουσιάστηκαν στο προηγούμενο Κεφάλαιο. Ωστόσο, επιδιώκεται η αποφυγή χρήσης πρόσθετης πληροφορίας στις SOAP επικεφαλίδες.

Στο παρόν Κεφάλαιο παρουσιάζεται η αρχιτεκτονική διαχείρισης του πλαισίου χρήσης για υπηρεσίες διαδικτύου [2]. Πραγματοποιείται μια περιγραφή της λειτουργίας της

αρχιτεκτονικής κατά την παροχή υπηρεσιών, η οποία ακολουθείται από την ανάλυση των στοιχείων που την αποτελούν. Ας σημειωθεί ότι η αρχιτεκτονική αφορά ένα μηχανισμό προσαρμογής που λαμβάνει χώρα στο επίπεδο της υπηρεσίας διαδικτύου. Η διαδικασία είναι ωστόσο εφαρμόσιμη και σε εφαρμογές διαδικτύου που απαρτίζονται από υπηρεσίες διαδικτύου, όπου η κάθε υπηρεσία προσαρμόζεται στο πλαίσιο χρήσης κατά την κλήση της. Με άλλα λόγια η εφαρμογή διαδικτύου αποτελεί ένα mashup υπηρεσιών. Γενικότερα, κάθε εφαρμογή που επικεντρώνεται στον τελικό χρήστη μπορεί να αναλυθεί σε ένα αριθμό βημάτων, όπου το καθένα είναι υπεύθυνο για συγκεκριμένο τμήμα της αλληλεπίδρασης χρήστη-εφαρμογής. Τα αντικείμενα στα οποία βασίζει τη λειτουργία της η εφαρμογή ενδέχεται να αποτελούνται από πελάτες για υπάρχουσες υπηρεσίες διαδικτύου, όπως θεωρείται στη διατριβή.

Η προτεινόμενη αρχιτεκτονική αντιμετωπίζει τα ερευνητικά θέματα που αναφέρθηκαν στο εισαγωγικό Κεφάλαιο, ενώ συνδέεται με τη μοντελοκεντρική μεθοδολογία για την ανάπτυξη υπηρεσιών διαδικτύου με επίγνωση του πλαισίου χρήσης που παρουσιάζεται σε επόμενα Κεφάλαια. Μάλιστα η μεθοδολογία ανάπτυξης καταλήγει ως περίπτωση χρήσης σε υλοποιήσεις εφαρμογών που προορίζονται για την αρχιτεκτονική που παρουσιάζεται.

4.1 Προσαρμογή Μηνυμάτων στο Πλαίσιο Χρήσης

Ο μηχανισμός προσαρμογής των υπηρεσιών υποστηρίζει τρεις διαφορετικές περιπτώσεις:

- ***Αντικατάσταση παραμέτρου (Parameter injection)***

  Αναφέρεται στην περίπτωση, όπου μια ή παραπάνω παράμετροι της μεθόδου που καλείται, όπως εκφράζονται στο μήνυμα αίτησης της υπηρεσίας, αλλάζουν σε τιμές που σχετίζονται με το πλαίσιο χρήσης. Για παράδειγμα, μια παράμετρος που αντιπροσωπεύει την τοποθεσία εκτέλεσης της υπηρεσίας αλλάζει στην τιμή της τρέχουσας θέσης του χρήστη ή μια παράμετρος που αντιπροσωπεύει τη γλώσσα που θα χρησιμοποιηθεί στην υπηρεσία αλλάζει σύμφωνα με τη γλώσσα του χρήστη που καθορίζεται σε κατάλληλο προφίλ.

- ***Επιλογή μεθόδου (Operation selection)***

  Στην περίπτωση που μια υπηρεσία αποτελείται από διάφορες μεθόδους που παρέχουν την ίδια λειτουργικότητα με διαφορετικό τρόπο ή με διαφορετικές παραμέτρους, ο πάροχος της υπηρεσίας μπορεί να παρέχει ένα μηχανισμό μέσω του οποίου να επιλέγεται και να καλείται η κατάλληλη μέθοδος βάσει της

πληροφορίας πλαισίου χρήσης που ανακτάται από τις πηγές. Αν για παράδειγμα είναι διαθέσιμοι σε μια υπηρεσία δυο διαφορετικοί μηχανισμοί πληρωμής (πληρωμή με μετρητά και πληρωμή με πιστωτική κάρτα) και ο καθένας εφράζεται ως μια χωριστή μέθοδος στην υπηρεσία, η προσαρμογή στο πλαίσιο χρήσης μπορεί να αλλάξει την προεπιλεγμένη μέθοδο βάσει των προτιμήσεων που αναφέρονται στο προφίλ του χρήστη, των συνθηκών ασφάλειας του περιβάλλοντος εκτέλεσης, κτλ.

- ***Χειρισμός απάντησης (Response manipulation)***

  Σε αυτή την περίπτωση προσαρμογής η απάντηση της υπηρεσίας τροποποιείται μετά από την ολοκλήρωση της εκτέλεση της υπηρεσίας. Η προσαρμογή μπορεί να περιλαμβάνει λειτουργίες ταξινόμησης ή φιλτραρίσματος (π.χ. στην περίπτωση μιας λίστας αποτελεσμάτων που μπορεί να αναφέρεται σε προτεινόμενες τουριστικές δραστηριότητες), μετασχηματισμού (π.χ. βάσει νομίσματος ή μονάδας θερμοκρασίας), κτλ.

Είναι δυνατόν να υπάρχουν και συνδυασμοί των παραπάνω κατηγοριών προσαρμογής, στους οποίους η μια προσαρμογή διαδέχεται την άλλη. Για παράδειγμα μια περίπτωση αντικατάστασης παραμέτρου μπορεί να ακολουθείται από μια περίπτωση τροποποίησης απάντησης. Αν μάλιστα απαιτούνται κατά την κλήσης μιας υπηρεσίας και οι τρεις τύποι, η σειρά προσαρμογής θα ολοκληρωθεί ως εξής:

1. Επιλογή μεθόδου
2. Αντικατάσταση παραμέτρου
3. Χειρισμός απάντησης

Οι προαναφερθείσες περιπτώσεις προσαρμογής έχουν νόημα για υπηρεσίες διαδικτύου, στις οποίες είναι αναγκαία η διατήρηση της ακεραιότητας των SOAP μηνυμάτων αιτήσεων και απαντήσεων. Όσον αφορά την τροποποίηση μιας αίτησης υπηρεσίας είναι δυνατό να μεταβληθούν είτε οι τιμές διαφόρων παραμέτρων του μηνύματος είτε η προεπιλεγμένη προς κλήση μέθοδος. Ωστόσο, δεν είναι δυνατό να αλλάξει εξ’ ολοκλήρου το SOAP μήνυμα ή να μεταβληθεί η αρχική διεύθυνση της υπηρεσίας που καλείται (target service address). Παρόμοιοι περιορισμοί ισχύουν και για τις SOAP απαντήσεις, οι οποίες όμως αφήνουν μεγαλύτερη ελευθερία όσον αφορά την εφαρμογή διαδικασιών τροποποίησης λόγω της φύσης των δεδομένων της απάντησης. Με αυτόν τον τρόπο διασφαλίζεται επιπλέον ο πλήρης διαχωρισμός της διαδικασίας προσαρμογής στο πλαίσιο χρήσης από τη λογική των εφαρμογών που απαρτίζονται από

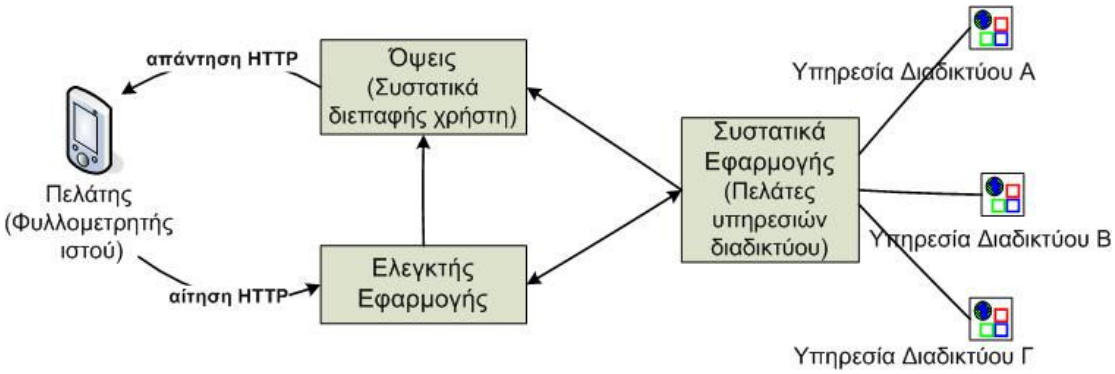
διάφορες υπηρεσίες διαδικτύου, καθώς η προσαρμογή σχετίζεται με τα μηνύματα των υπηρεσιών που συμμετέχουν.

4.2 Περιγραφή Λειτουργίας της Αρχιτεκτονικής

Η προτεινόμενη αρχιτεκτονική στοχεύει στην παροχή υπηρεσιών διαδικτύου με επίγνωση του πλαισίου χρήσης ή εφαρμογών που αποτελούνται από υπηρεσίες διαδικτύου. Οι υπηρεσίες διαδικτύου χρησιμοποιούνται ως δομικά συστασικά (components) για την κατασκευή της εφαρμογής, όπως συνηθίζεται σε πολλές περιπτώσεις όπου διαθέσιμα συστατικά επαναχρησιμοποιούνται και συνδυάζονται για την κατασκευή νέων, ελκυστικών εφαρμογών [1]. Η προσαρμογή στο πλαίσιο χρήσης πραγματοποιείται βάσει της ανάκτησης πληροφορίας από κατάλληλες πηγές πλαισίου χρήσης, η λειτουργία των οποίων είναι επίσης διαθέσιμη σε υπηρεσίες διαδικτύου, όπως θεωρείται στην προσέγγιση της διατριβής. Οι συγκεκριμένες υπηρεσίες αναφέρονται ως υπηρεσίες πλαισίου χρήσης (Context Web Services / CWSs) σε αντιδιαστολή με τις υπηρεσίες εφαρμογών του επιπέδου εφαρμογής (business layer) που παρέχουν τη λειτουργικότητά τους στη βασική εφαρμογή (Business Web Services / BWSs). Οι υπηρεσίες πλαισίου χρήσης καλούνται από την αρχιτεκτονική μέσω προσαρμογέων πλαισίου χρήσης (plugins) που “φορτώνονται” αυτόματα κατά την εκτέλεση της εφαρμογής από το προτεινόμενο σύστημα. Όπως έχει ήδη αναφερθεί το βασικό χαρακτηριστικό και πλεονέκτημα της προσέγγισης είναι ο πλήρης διαχωρισμός της βασικής λειτουργίας της εφαρμογής (που σχετίζεται με τη λειτουργία των ανεξάρτητων υπηρεσιών διαδικτύου) και της προσαρμογής στο πλαίσιο χρήσης (μέσω των εξωτερικών πήγών πληροφορίας πλαισίου χρήσης). Επιπλέον, υπάρχει η δυνατότητα ενεργοποίησης και απενεργοποίησης διαφόρων πηγών για την ίδια εφαρμογή διατηρώντας και πάλι αναλλοίωτη τη βασική της λειτουργία.

Η προσαρμογή μέσω της αρχιτεκτονικής μπορεί να εφαρμοσθεί σε υπηρεσίες διαδικτύου που βασίζονται στο πρωτόκολλο SOAP για την ανταλλαγή μηνυμάτων, αφού αυτά τα μηνύματα “συλλαμβάνονται” και τροποποιούνται κατάλληλα. Οι εφαρμογές που προκύπτουν από το συνδυασμό των υπηρεσιών διαδικτύου μέσω της αρχιτεκτονικής είναι ανεξάρτητες από το λογισμικό που υπάρχει στην πλευρά του πελάτη ή του τελικού χρήστη της εφαρμογής. Για να δοθεί μάλιστα έμφαση σε αυτή την ιδιότητα η αρχιτεκτονική χρησιμοποιείται για την προσαρμογή εφαρμογών διαδικτύου που έχουν ως διεπαφή για τον τελικό χρήστη ένα φυλλομετρητή ιστού. Με αυτόν τον τρόπο, δεν απαιτείται η ύπαρξη κάποιας συγκεκριμένης υποδομής στην πλευρά του πελάτη για την υποστήριξη των υπηρεσιών.

Συγκεκριμένα, οι εφαρμογές που προορίζονται για την αρχιτεκτονική προσαρμογής στο πλαίσιο χρήσης είναι σχεδιασμένες ως εφαρμογές που ακολουθούν το αρχιτεκτονικό πρότυπο μοντέλου-όψης-ελεγκτή. Το πλαίσιο μοντέλου-όψης-ελεγκτή είναι κατάλληλο για εφαρμογές αυτού του τύπου, καθώς εξασφαλίζει το διαχωρισμό του επιπέδου εφαρμογής από τη διεπαφή του χρήστη. Ο ελεγκτής διαχειρίζεται τις αιτήσεις των πελατών και είναι υπεύθυνος για τις κατάλληλες κλήσεις στα δομικά συστατικά του επιπέδου εφαρμογής που είναι ουσιαστικά πελάτες για υπηρεσίες διαδικτύου, ώστε να επιλέξει και να κατασκευάσει τις κατάλληλες όψεις βάσει των απαντήσεων που θα λάβει. Αυτός ο τρόπους λειτουργίας των εφαρμογών παρουσιάζεται στην Εικόνα 16, όπου οι αιτήσεις στην εφαρμογή εκκινούνται από τον πελάτη (μέσω HTTP αιτήσεων για την περίπτωση του φυλλομετρητή ιστού). Στο επίπεδο της λογικής της εφαρμογής ένα σύνολο πελατών για υπηρεσίες διαδικτύου υλοποιούν τοπικά τις διεπαφές που υπάρχουν στις WSDL περιγραφές των υπηρεσιών διαδικτύου που χρησιμοποιούνται στη εφαρμογή, ώστε να είναι δυνατή η κλήση των υπηρεσιών. Ένα πρόσθετο πλεονέκτημα του παραπάνω σχεδιασμού είναι ότι επιτρέπει την άμεση επαναχρησιμοποίηση των συστατικών του επιπέδου εφαρμογής, ενώ είναι εφικτή και η χρήση υπηρεσιών που προέρχονται από τρίτους παρόχους αρκεί να είναι γνωστός ο τρόπος κλήσης των υπηρεσιών.

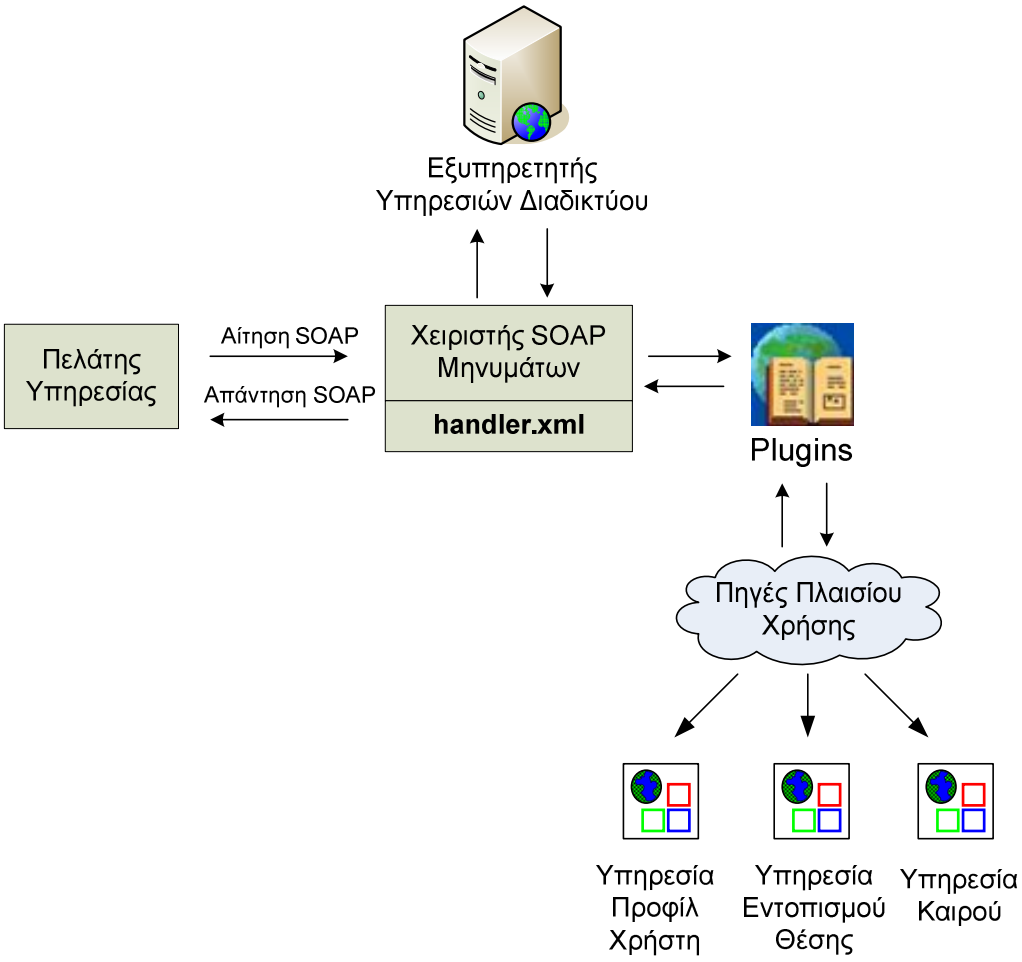


Εικόνα 16. Εφαρμογή με επίγνωση του πλαισίου χρήσης σύμφωνα με το πρότυπο μοντέλου-όψης-ελεγκτή.

Η προσαρμογή στο πλαίσιο χρήσης πραγματοποιείται στο επίπεδο διεπαφής της υπηρεσίας διαδικτύου και όχι εσωτερικά στη λειτουργία της. Αυτό πραγματοποιείται μέσω χειριστών, όπως ορίζονται σε διάφορα framework υπηρεσιών διαδικτύου. Οι χειριστές είναι μονάδες επεξεργασίας μηνυμάτων και μπορεί να σχετίζονται με διάφορες λειτουργίες κρυπτογράφησης ή καταγραφής των δραστηριότητων των υπηρεσιών διαδικτύου (logging). Μπορούν να βρίσκονται είτε στην πλευρά του πελάτη είτε του εξυπηρετητή, είναι οργανωμένοι σειριακά και μπορεί να αλληλεπιδρούν με τα εισερχόμενα ή τα εξερχόμενα μηνύματα ή και με τα δύο ανάλογα με τη διαμόρφωση που έχει επιλεχθεί. Στην προσέγγιση

που επιχειρείται στη διατριβή χρησιμοποιούνται χειριστές μηνυμάτων με την παραπάνω έννοια για την προσαρμογή του περιεχομένου των SOAP μηνυμάτων στις παραμέτρους του πλαισίου χρήσης.

Για να επιτευχθεί η προσαρμογή στο πλαίσιο χρήσης προτείνεται η αρχιτεκτονική που παρουσιάζεται στην Εικόνα 17. Η προσαρμογή βασίζεται στη "σύλληψη" των μηνυμάτων αιτήσεων και απαντήσεων, την ανάκτηση της απαραίτητης πληροφορίας πλαισίου χρήσης που σχετίζεται με τα μηνύματα, και την αντίστοιχη τροποποίηση του μηνύματος βάσει της πληροφορίας αυτής. Η τροποποίηση του μηνύματος πραγματοποιείται μέσω μιας βιβλιοθήκης προσαρμογέων. Οι προσαρμογείς, αφού επικοινωνήσουν με τις υπηρεσίες πλαισίου χρήσης που έχουν πρόσβαση στην πληροφορία πλαισίου χρήσης, τροποποιούν τα αντίστοιχα εισερχόμενα και εξερχόμενα μηνύματα για να αντικατοπτριστεί η τρέχουσα κατάσταση του χρήστη, της υπηρεσίας και του περιβάλλοντός.



Εικόνα 17. Μηχανισμός προσαρμογής στο πλαίσιο χρήσης.

Με αυτόν τον τρόπο η προσαρμογή στο πλαίσιο χρήσης παραμένει ανεξάρτητη από την υλοποίηση της υπηρεσίας διαδικτύου και τελείως χωριστή από τη λογική της, αφού πραγματοποιείται εν αγνοία της υπηρεσίας. Επιπλέον, η φάση προσαρμογής στο πλαίσιο

χρήσης παραμένει προαιρετική, αφού για κάποιες υπηρεσίες διαδικτύου δεν απαιτείται κάποιο είδος προσαρμογής. Η λογική της προσαρμογής μπορεί ακόμα να τροποποιηθεί οποιαδήποτε στιγμή χωρίς να επηρεαστεί η υπηρεσία ή η εφαρμογή διαδικτύου. Οι λειτουργίες που οδηγούν στην προσαρμογή των μηνυμάτων πραγματοποιούνται μέσω των plugin που "φορτώνονται" αυτόματα κατά την εκτέλεση βάσει του κατάλληλου αρχείου διαμόρφωσης. Έτσι, η επιλογή της ομάδας των κατάλληλων για κάθε μήνυμα προσαρμογέων γίνεται βάσει της υπάρχουσας διαμόρφωσης που μπορεί επίσης να ενημερωθεί ή να τροποποιηθεί οποιαδήποτε στιγμή.

Η διαδικασία προσαρμογής πραγματοποιείται χωριστά για κάθε υπηρεσία διαδικτύου και για κάθε κατεύθυνση μηνυμάτων (αιτήσεις και απαντήσεις). Οι αιτήσεις που διαχειρίζεται ο ελεγκτής της εφαρμογής οδηγούν στην κλήση των υπηρεσιών διαδικτύου από τις οντότητες του επιπέδου εφσρμογής. Τα αποτελέσματα των κλήσεων μεταβιβάζονται από τον ελεγκτή στον πελάτη μέσω της κατάλληλης διεπαφής (όψη στο φυλλομετρητή ιστού).

Η διαδικασία προσαρμογής στο πλαίσιο χρήσης μέσω της αρχιτεκτονικής αποτελείται από τα παρακάτω βήματα:

1. ***"Σύλληψη" του SOAP μηνύματος αίτησης***

   Ο χειριστής πλαισίου χρήσης "συλλαμβάνει" το SOAP μήνυμα αίτησης υπηρεσίας και ενδέχεται να το προσαρμόσει βάσει των περιεχομένων του αρχείου διαμόρφωσης *handler.xml*, που συνδέει την καλούμενη μέθοδο της υπηρεσίας με τον κατάλληλο προσαρμογέα ή προσαρμογείς όταν υπάρχει κάποια εξάρτηση από πλαίσιο χρήσης.

2. ***Φόρτωση των κατάλληλων προσαρμογέων***

   Τα plugin που σχετίζονται με την υπηρεσία (βάσει του αρχείου διαμόρφωσης) φορτώνονται από το χειριστή του μηχανισμού. Σε αυτό το σημείο ο φάκελος του SOAP μηνύματος παραδίδεται στον προσαρμογέα για την περαιτέρω επεξεργασία και προσαρμογή.

3. ***Τροποποίηση του μηνύματος***

   Τα plugin προσαρμόζουν κατάλληλα το περιεχόμενου του SOAP φακέλου βάσει της διαθέσιμης πληροφορίας πλαισίου χρήσης που ανακτάται από τις πηγές και επιστρέφουν το μήνυμα στο χειριστή. Πρόσβαση σε στοιχεία που σχετίζονται με το χρήστη (π.χ. προφίλ με προσωπικές πληροφορίες και προτιμήσεις του χρήστη) επιτυγχάνεται μέσω ενός μοναδικού χαρακτηριστικού χρήστη που προστίθεται

αυτόματα στην επικεφαλίδα του SOAP μηνύματος από τον πελάτη κατά την κατασκευή της αίτησης.

4. ***"Σύλληψη" του SOAP μηνύματος απάντησης***

   Η παραπάνω διαδικασία (βήματα 1 εώς 3) επαναλαμβάνεται για το μήνυμα απάντησης ("σύλληψη" μηνύματος, φόρτωση κατάλληλων προσαρμογέων, τροποποίηση μηνύματος).

Η λειτουργικότητα του χειριστή (δηλ. το κομμάτι της "σύλληψης" του μηνύματος και της φόρτωσης των κατάλληλων plugin) παραμένει η ίδια σε κάθε περίπτωση προσαρμογής στο πλαίσιο χρήσης. Η λογική της διαδικασίας προσαρμογής έγκειται στην ανάπτυξη της λειτουργίας των προσαρμογέων και του αρχείου διαμόρφωσης που είναι τελείως διαχωρισμένα από την εφαρμογή διαδικτύου.

4.3 Ανάλυση Στοιχείων Αρχιτεκτονικής

Σε αυτό το σημείο τα στοιχεία που απαρτίζουν την αρχιτεκτονική και η λειτουργία τους περιγράφονται με περισσότερες λεπτομέρειες.

4.3.1 Χειριστής Μηνυμάτων

Ο ρόλος του χειριστή είναι, όπως αναφέρθηκε και παραπάνω, να "συλλαμβάνει" τα SOAP μηνύματα και να φορτώνει τα κατάλληλα plugin. Κάθε υπηρεσία διαδικτύου μπορεί να σχετίζεται με ένα ή περισσότερα plugin. Ο συσχετισμός αυτός υποδηλώνει πως θα εφαρμοστούν μία ή παραπάνω περιπτώσεις προσαρμογής στο πλαίσιο χρήσης ανάλογα με τον αριθμό των plugin που συνδέονται με την υπηρεσία. Ο συσχετισμός αναπαρίσταται στο αρχείο διαμόρφωσης σε XML μορφοποίηση, το *handler.xml*.

Το βασικό τμήμα του XML σχήματος του αρχείου διαμόρφωσης παρουσιάζεται στην Εικόνα 18. Στο αρχείο απαριθμούνται οι διαθέσιμοι προσαρμογείς (στοιχεία <*plugin*> στο σχήμα) και οι συσχετισμοί τους με τις υπηρεσίες διαδικτύου (στοιχεία <*association*>). Κάθε plugin, που προσδιορίζεται από ένα μοναδικό αναγνωριστικό (μεταβλητή ιδιότητας *id*), περιγράφεται από το όνομα της κλάσης που το υλοποιεί (attribute *classname*) μαζί με το πακέτο (package) και το jar αρχείο όπου βρίσκεται (μεταβλητές ιδιοτήτων *pckg* και *jarfile* αντίστοιχα). Σε κάθε στοιχείο συσχετισμού ένα συγκεκριμένο plugin που προσδιορίζεται από το αναγνωριστικό του (στοιχείο <*pluginEnd*>) συσχετίζεται με μια κλήση μεθόδου (attribute *operation* στο στοιχείο <*serviceEnd*>) μιας συγκεκριμένης υπηρεσίας (attribute *name*). Αυτοί οι συσχετισμοί δείχνουν ότι είναι απαραίτητη κάποια διαδικασία προσαρμογής. Το αρχείο διαμόρφωσης, καθώς και τα plugin που υλοποιούν τη λογική

προσαρμογής, μπορούν να ενημερωθούν ή να αντικατασταθούν κατά την εκτέλεση χωρίς να διακόψουν τη διαθεσιμότητα της εφαρμογής.

```
<xs:element name="plugin" maxOccurs="unbounded">
 <xs:complexType>
   <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="id" type="xs:string" use="required"/>
      <xs:attribute name="pckg" type="xs:string" use="required"/>
      <xs:attribute name="classname" type="xs:string" use="required"/>
      <xs:attribute name="jarfile" type="xs:string" use="required"/>
    </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>
<xs:element name="association" maxOccurs="unbounded">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="serviceEnd">
 <xs:complexType>
      <xs:simpleContent>
       <xs:extension base="xs:string">
        <xs:attribute name="name" type="xs:string" use="required"/>
        <xs:attribute name="operation" type="xs:string" use="required"/>
       </xs:extension>
      </xs:simpleContent>
  </xs:complexType>
  </xs:element>
  <xs:element name="pluginEnd">
   <xs:complexType>
    <xs:simpleContent>
     <xs:extension base="xs:string">
      <xs:attribute name="id" type="xs:string" use="required"/>
      <xs:attribute name="direction" type="xs:string" use="required"/>
     </xs:extension>
    </xs:simpleContent>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
 </xs:complexType>
</xs:element>
```

Εικόνα 18. XML σχήμα για το αρχείο συσχετισμού της υπηρεσίας με τα plugin.

Ένα απλό παράδειγμα αρχείου διαμόρφωσης που προκύπτει από το παραπάνω XML σχήμα παρουσιάζεται στην Εικόνα 19.

```
<?xml version="1.0" encoding="UTF-8"?>
 <plugins>
  <plugin id="id1" pckg="path.to.main.plugin.class"
classname="PluginBasicClassName.class" jarfile="pluginJarFile.jar"/>
  <plugin .../>
  <association>
   <serviceEnd name="ServiceName" operation="serviceOperationName"/>
  <pluginEnd id="id1" direction="in"/>
 </association>
 <association>
  <serviceEnd name="ServiceName" operation="serviceOperationName"/>
 <pluginEnd id="id1" direction="out"/>
```

```
</association>
<association>
  ...
</association>
</plugins>
```

Εικόνα 19. Παράδειγμα αρχείου hander.xml.

4.3.2 Προσαρμογείς στο Πλαίσιο Χρήσης

Διάφοροι προσαρμογείς στο πλαίσιο χρήσης είναι διαθέσιμοι στην αντίστοιχη βιβλιοθήκη των plugin. Τα διαθέσιμα plugin χωρίζονται στις κατηγορίες των *inPlugins* που χρησιμοποιούνται για την τροποποίηση των αιτήσεων και των *outPlugins* που τροποποιούν τις απαντήσεις. Αυτό εκφράζεται με τη μεταβλητή ιδιότητας *direction* στο στοιχείο *<pluginEnd>* στο *<association>* κομμάτι του *handler.xml* αρχείου. Οι προσαρμογείς είναι το σημείο όπου λαμβάνει ουσιαστικά χώρα η τροποποίηση του SOAP μηνύματος. Τα plugin που έχουν φορτωθεί επικοινωνούν με τις πηγές πλαισίου χρήσης και επιστρέφουν το τροποποιημένο μήνυμα στο χειριστή, ώστε να συνεχιστεί η πορεία του είτε προς το άκρο της υπηρεσίας διαδικτύου (για τις αιτήσεις), είτε προς την πλευρά του πελάτη (για τις απαντήσεις).

4.3.3 Πηγές Πληροφορίας Πλαισίου Χρήσης

Στην προτεινόμενη αρχιτεκτονική θεωρείται ότι οι πηγές πλαισίου χρήσης είναι και αυτές υπηρεσίες διαδικτύου. Έτσι, η ανάκτηση του πλαίσιο χρήσης πραγματοποιείται μέσω των συγκεκριμένων CWS υπηρεσιών. Με αυτόν τον τρόπο η υποκείμενη υποδομή μέσω της οποίας υπάρχει άμεση πρόσβαση στις τρέχουσες τιμές του πλαισίου χρήσης (π.χ. αισθητήρες για ενδείξεις θερμοκρασίας, συστήματα διαχείρισης βάσεων δεδομένων για το προφίλ του χρήστη, κτλ.) είναι διαθέσιμη και στα plugin που τροποποιούν τα μηνύματα. Μάλιστα η διατριβή δε διαχειρίζεται τα υποκείμενα στρώματα της συλλογής, διανομής και επεξεργασίας της πληροφορίας πλαισίου χρήσης, αλλά εστιάζει στο επίπεδο εφαρμογής για την προσαρμογή της υπηρεσίας βάσει της διαθέσιμης πληροφορίας.

4.4 Υλοποίηση Αρχιτεκτονικής

Για την υλοποίηση της αρχιτεκτονικής και του μηχανισμού προσαρμογής χρησιμοποιήθηκε το framework Apache Axis2 και συγκεκριμένα η έκδοσή του σε Java. Ωστόσο, οι αρχές που παρουσιάστηκαν μπορούν να εφαρμοστούν και σε οποιοδήποτε άλλο framework αρκεί αυτό να υποστηρίζει ιεραρχία χειριστών ή ασύγχρονη κλήση

χειριστών (π.χ. Apache CXF). Ο χειριστής για το πλαίσιο χρήσης έχει τοποθετηθεί στην πλευρά του παροχέα υπηρεσιών, οπότε θεωρείται ότι αυτό το άκρο της επικοινωνίας πελάτη-εξυπηρετητή είναι υπεύθυνο για την απαραίτητη προσαρμογή των υπηρεσιών.

Η βασική λειτουργία του χειριστή πλαισίου χρήσης υλοποιήθηκε σε δύο κλάσεις, μία ανά κατεύθυνση μηνυμάτων: `InAspectHandler` και `OutAspectHandler` για αιτήσεις και απαντήσεις αντίστοιχα. Κατά την εκτέλεση της υπηρεσίας ο κάθε χειριστής αναζητάει στη λίστα των διαθέσιμων προσαρμογέων αυτούς που σχετίζονται με την καλούμενη υπηρεσία και μέθοδο (από τα στοιχεία <*association*> του αρχείου διαμόρφωσης) και φορτώνει τους κατάλληλους. Η φόρτωση των plugin από το χειριστή πραγματοποιείται μέσω της φόρτωσης των αντίστοιχων κλάσεων των plugin (classloading) με χρήση της κλάσης `PluginClassLoader`.

Για τη διαδικασία της δημιουργίας της λίστας των διαθέσιμων plugin (`PluginList`), η οποία προηγείται της διαδικασίας συσχέτισης της υπηρεσίας με τα κατάλληλα plugin, διαβάζεται και ερμηνεύεται από το χειριστή πλαισίου χρήσης το αρχείο διαμόρφωσης *handler.xml*. Η διαδικασία ανάγνωσης του αρχείου πραγματοποιείται μέσω της αντίστοιχης βιβλιοθήκης σε Java: τη *handler.jar* που έχει κατασκευαστεί για το σκοπό αυτό κάνοντας χρήση του εργαλείου Apache XML Beans. Βάσει των στοιχείων του αρχείου, κατασκευάζονται περιγραφείς για κάθε plugin (`PluginDescription`) και αυτοί προστίθενται εν συνεχεία στη λίστα. Καθώς υπάρχει ο διαχωρισμός μεταξύ των plugin χειρισμού μηνυμάτων αιτήσεων και των plugin χειρισμού μηνυμάτων απαντήσεων, διατηρούνται στην ουσία δύο λίστες από plugin, μία ανά κατεύθυνση (λίστες `inPlugins` και `outPlugins` αντιστοίχως).

Από τη στιγμή που φορτωθεί μέσω της παραπάνω διαδικασίας ένα plugin για μια υπηρεσία, αναλαμβάνει το ίδιο τον έλεγχο για την προσαρμογή του μηνύματος και εκτελεί κάποια από τις περιπτώσεις προσαρμογής στο πλαίσιο χρήσης καλώντας τις αντίστοιχες πηγές πληροφορίας. Βασική προϋπόθεση είναι να είναι διαθέσιμη η υπηρεσία πλαισίου χρήσης που καλείται και να είναι επιπλέον δυνατός ο συνδυασμός της με τη συγκεκριμένη υπηρεσία εφαρμογών. Η προσαρμογή συνεχίζεται με αυτόν τον τρόπο και από τα υπόλοιπα plugin που πιθανόν σχετίζονται με την υπηρεσία. Φυσικά δε φορτώνεται κανένα plugin για περιπτώσεις υπηρεσιών, όπου δεν απαιτείται κάποιο είδος προσαρμογής.

Κάθε νέο plugin υλοποιείται ως επέκταση της βασικής κλάσης `AbstractPlugin`. Περιλαμβάνει συνήθως πελάτες για τις υπηρεσίες διαδικτύου που αποτελούν πηγές πλαισίου χρήσης, ώστε να ανακτηθεί η απαραίτητη πληροφορία, και διάφορες μεθόδους προσαρμογής ανάλογα με την περίπτωση.

Με τον παραπάνω τρόπο η λίστα με τα απαραίτητα plugin φορτώνεται κατά την εκτέλεση της υπηρεσίας παρέχοντας στο σύστημα ευελιξία όσον αφορά αλλαγές στη λίστα των προσαρμογέων. Καινούρια plugin μπορούν να προστεθούν ή κάποια από τα υπάρχοντα να τροποποιηθούν ή να αφαιρεθούν βάσει των εκάστοτε αναγκών της εφαρμογής. Σε κάθε περίπτωση η λογική της υπηρεσίας παραμένει ακέραια, αφού δεν υπάρχει καμία επέμβαση στον κώδικά της. Το ίδιο ισχύει και για την εφαρμογή διαδικτύου που συνδέει τις διάφορες υπηρεσίες.

Στην ιεραρχία χειριστών που είναι ήδη διαθέσιμοι στο Axis2 για διάφορες λειτουργίες προστέθηκε ως νέο module ο χειριστής που σχετίζεται με την προσαρμογή στο πλαίσιο χρήσης στη νέα φάση *contextPhase* του αρχείο διαμόρφωσης *axis2.xml*. Ο νέος χειριστής καλείται τόσο για τα εισερχόμενα (inFlow) όσο και για τα εξερχόμενα (outFlow) μηνύματα (καθώς και για τα ρεύματα σφαλμάτων InFaultFlow και OutFaultFlow). Για να κληθεί ο χειριστής του πλαισίου χρήσης (μεταξύ των άλλων διαθέσιμων χειριστών) απαιτείται η κατάλληλη τροποποίηση της περιγραφής της υπηρεσίας. Στην περίπτωση του Axis2 απαιτείται η ακόλουθη προσθήκη στο αρχείο services.xml που περιγράφει την υπηρεσία:

```
<module ref="contextPhase"/>
```

Με τη δήλωση αυτή δημιουργείται μια αναφορά προς το module το οποίο περιλαμβάνει τους χειριστές που θέλουμε να συνδέσουμε με την υπηρεσία, ώστε να κληθούν σε κάποια από τις φάσεις που προβλέπονται στο *axis2.xml*.

4.5 Βιβλιογραφία

[1] Brown, A. W., Wallnan, K. C., "Engineering of Component-Based Systems", Component-Based Software Engineering: Selected Papers from the Software Engineering Institute, Los Alamitos, CA: IEEE Computer Society Press, 1996, pp. 7-15.

[2] Kapitsaki, G. M., Kateros, D. A., Venieris, I. S., “Architecture for Provision of Context-aware Web Applications based on Web Services”, IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC’08), Cannes, France, September 14-18, 2008.

5ο Κεφάλαιο

Μοντελοκεντρική Ανάπτυξη Υπηρεσιών

Στο εισαγωγικό Κεφάλαιο δόθηκε μια περιγραφή της μοντελοκεντρικής ανάπτυξης λογισμικού, στην οποία βασίζεται η διατριβή για την ανάπτυξη εφαρμογών, οι οποίες αποτελούνται από υπηρεσίες διαδικτύου με επίγνωση του πλαισίου χρήσης. Στο παρόν Κεφάλαιο προσεγγίζεται το θέμα της μοντελοποίησης της πληροφορίας πλαισίου χρήσης και γίνεται μια περιγραφή των προτεινόμενων λύσεων της διεθνούς βιβλιογραφίας. Ακολούθως, γίνεται μια επισκόπηση προτεινόμενων προσεγγίσεων και τεχνικών που σχετίζονται με τη μοντελοκεντρική ανάπτυξη υπηρεσιών και, τέλος, περιγράφονται κάποια γενικά εργαλεία μοντελοκεντρικής ανάπτυξης λογισμικού για να δοθεί μια πληρέστερη εικόνα της περιοχής.

5.1 Μοντελοποίηση Εφαρμογών και Πλαισίου Χρήσης

Για να ακολουθήσει η ανάπτυξη μιας εφαρμογής τη μοντελοκεντρική προσέγγιση απαιτείται η σχεδίαση του μοντέλου της εφαρμογής. Η γλώσσα μοντελοποίησης που χρησιμοποιείται στις περισσότερες περιπτώσεις είναι η UML, η οποία αποτελεί την πιο ευρέως διαδεδομένη γλώσσα για τη μοντελοποίηση αντικειμένων. Το μοντέλο της εφαρμογής βασίζεται με τη σειρά του στο δικό του μοντέλο ή στο μεταμοντέλο της εφαρμογής (metamodel), όπως αναφέρεται. Έτσι, για να πραγματοποιηθεί η μοντελοποίηση μιας εφαρμογής συγκεκριμένης χρήσης, χρειάζεται να ορίστει αρχικά το κατάλληλο μεταμοντέλο που να περιλαμβάνει όλα τα χαρακτηριστικά της κλάσης εφαρμογών που θα αναπτυχθούν. Οι εφαρμογές βασίζονται έπειτα σε αυτό για τη μοντελοποίησή τους. Στα πλαίσια της διατριβής ένα σημαντικό ζήτημα είναι η περιγραφή του πλαισίο χρήσης με κατάλληλα μοντέλα που να μπορούν να συνδυαστούν με τα

αντίστοιχα μοντέλα περιγραφής της υπηρεσίας προκειμένου να προκύψει μια λειτουργική υπηρεσία με επίγνωση του πλαισίου χρήσης.

Για τη μοντελοποίηση ενός συστήματος ή εφαρμογής χρησιμοποιούνται και άλλες γλώσσες μοντελοποίησης, όπως οντολογίες ή γλώσσες ειδικού πεδίου (Domain Specific Languages / DSL). Μια κατηγοριοποίηση και επισκόπηση μοντέλων πλαισίου χρήσης συμπεριλαμβάνοντας μεταξύ άλλων μοντέλα και των παραπάνω κατηγοριών έχει παρουσιαστεί στο [37], ενώ στο [7] δίνεται μια παρουσίαση μοντέλων που έχουν προταθεί σε διάφορα context-aware συστήματα. Τεχνικές μοντελοποίησης μπορούν να χρησιμοποιηθούν και στο στάδιο της ανάλυσης των απαιτήσεων των εφαρμογών με επίγνωση του πλαισίου χρήσης. Αυτό προτείνεται στην προσανατολισμένη στο πλαίσιο χρήσης Ανάλυση Πεδίου (Context-Oriented Domain Analysis / CODA) [13]. Η CODA αποτελεί μια συστηματική προσέγγιση για τη συλλογή απαιτήσεων συστημάτων με επίγνωση του πλαισίου χρήσης κάνοντας χρήση μοντέλων που μπορούν εν συνεχεία να αντιστοιχηθούν σε πίνακες αποφάσεων.

Στη διατριβή επικεντρωνόμαστε κυρίως στη μοντελοποίηση σε UML που χρησιμοποιείται και στην προτεινόμενη λύση. Η επιλογή αυτή έγινε λόγω του καθολικού ρόλου της γλώσσας, η οποία είναι ανεξάρτητη συστήματος. Η UML πέραν του ότι αποτελεί την πιο ευρέως διαδεδομένη γλώσσα σχεδιασμού και μοντελοποίησης εφαρμογών, αποτελεί και την καλύτερη επιλογή για τη μοντελοκεντρική ανάπτυξη καθώς επιτρέπει στους μηχανικούς λογισμικού να χρησιμοποιήσουν διαθέσιμα περιβάλλοντα μοντελοποίησης σε UML για την απεικόνιση και τροποποίηση του μοντέλου της εφαρμογής χωρίς να απαιτείται χρήση ή εγκατάσταση ειδικού λογισμικού. Με αυτόν τον τρόπο η μεθοδολογία ανάπτυξης απευθύνεται σε ένα ευρύτερο κοινό. Εξάλλου, στόχος της διατριβής και της μεθοδολογίας που προτείνεται είναι η υιοθέτηση ευρέως αποδεκτών προδιαγραφών για τη διαδικασία ανάπτυξης. Ωστόσο, στην παρούσα ενότητα περιλαμβάνεται και μια σύντομη επισκόπηση άλλων συστημάτων μοντελοποίησης.

5.1.1 Μοντελοποίηση σε UML

5.1.1.1 Unified Modeling Language

Η γλώσσα UML είναι μια γραφική γλώσσα για την απεικόνιση και κατασκευή των στοιχείων ενός σύστήματος λογισμικού και είναι μάλιστα η πρότυπη γλώσσα μοντελοποίησης στη μηχανική λογισμικού. Προσφέρει έναν τρόπο για τη δημιουργία του σχεδίου ή του πρωτοτύπου ενός συστήματος συμπεριλαμβανομένων λογικών συστατικών, όπως επιχειρησιακές διεργασίες και διαδικασίες του συστήματος, αλλά και συγκεκριμένων

στοιχείων, όπως δηλώσεις μιας γλώσσας προγραμματισμού, σχημάτων βάσεων δεδομένων και επαναχρησιμοποιήσιμων συστατικών λογισμικού.

Η UML μπορεί να χρησιμοποιηθεί σε διάφορες φάσεις ανάπτυξης – από την ανάλυση απαιτήσεων ως τον έλεγχο ενός ολοκληρωμένου συστήματος – και με διαφορετικές τεχνολογίες υλοποίησης. Ως UML μεταμοντέλο έχει οριστεί επίσημα από την OMG ένα MOF μεταμοντέλο, ενώ η τελευταία εκδοσή της γλώσσας είναι η 2.2. Η UML επιτρέπει στους μηχανικούς λογισμικού να επικεντρωθούν στη σχεδίαση και την αρχιτεκτονική των συστημάτων, ενώ τα μοντέλα της UML μπορούν να μετασχηματιστούν σε άλλες αναπαραστάσεις μέσω γλωσσών μετασχηματισμού συμβατών με την πρότυπη γλώσσα μετασχηματισμού μοντέλων QVT.

Η UML 2.0 ορίζει 13 διαφορετικούς τύπους διαγραμμάτων που χωρίζονται σε τρεις κατηγορίες: έξι από αυτά αναπαριστούν τη δομή μιας εφαρμογής, τρία γενικούς τύπους συμπεριφοράς και τέσσερα διαφορετικά είδη αλληλεπιδράσεων. Τα βασικά διαγράμματα που χρησιμοποιούνται είναι:

- ***Διαγράμματα δομής (Structure diagrams)***
  - *Διαγράμματα κλάσεων (Class diagram)*: Περιγράφουν τη δομή ενός συστήματος και αποτελούνται από κλάσεις ή σύνολα κλάσεων και τις μεταξύ τους συσχετίσεις.
  - *Διαγράμματα συστατικών (Component diagram):* Συμβολίζουν τα συστατικά που αποτελούν το σύστημα λογισμικού και τις μεταξύ τους σχέσεις.
  - *Διαγράμματα σύνθετης δομής (Composite structure diagram):* Περιγράφουν την εσωτερική δομή μιας κλάσης.
  - *Διαγράμματα ανάπτυξης (Deployment diagram):* Χρησιμοποιούνται για τη μοντελοποίηση του υλικού που χρησιμοποιείται στην υλοποίηση των συστημάτων, των συστατικών του υλικού και των μεταξύ τους συσχετίσων.
  - *Διαγράμματα αντικειμένων (Object diagram):* Παρουσιάζουν μια πλήρη ή μερική όψη της δομής του συστήματος που μοντελοποιείται σε μια δεδομένη στιγμή.
  - *Διαγράμματα πακέτων (Package diagram):* Δείχνουν πώς το σύστημα χωρίζεται σε λογικές ομάδες αναπαριστώντας τις εξαρτήσεις μεταξύ των ομάδων.
- ***Διάγραμματα συμπεριφοράς (Behavior diagrams)***
  - *Διαγράμματα δραστηριοτήτων (Activity diagram):* Περιγράφουν πώς παρέχεται η γενική λειτουργία μιας οντότητας βάσει απλούστερων λειτουργιών που την απαρτίζουν και δείχνουν τη ροή του ελέγχου.
  - *Διαγράμματα καταστάσεων (State machine diagram):* Αναπαριστούν – μεταξύ άλλων χρήσεων – την τρέχουσα κατάσταση μιας οντότητας, λειτουργίας ή

συστήματος και τις πιθανές αλλαγές στην κατάσταση που προκαλούνται από άλλες λειτουργίες ή συνθήκες.

- *Διαγράμματα περιπτώσεων χρήσης (Use case diagrams):* Χρησιμοποιούνται για να περιγράψουν ποιες υπηρεσίες ή λειτουργίες προσφέρει ένα σύστημα και σε ποιους, καθώς και ποιες οντότητες χρειάζεται να συνεργαστούν για αυτή την παροχή λειτουργιών.

- ***Διαγράμματα αλληλεπίδρασης*** *(Interaction diagrams* – Αποτελούν υποομάδα των διαγραμμάτων συμπεριφοράς)
  - *Ακολουθιακά διαγράμματα (Sequence diagrams):* Περιγράφουν την πλήρη εκτέλεση μιας ορισμένης εργασίας στην οποία λαμβάνουν μέρος συγκεκριμένα αντικείμενα των κλάσεων που η εργασία απαιτεί να συνεργάζονται.
  - *Διαγράμματα επικοινωνίας (Communication diagram):* Δείχνουν τις αλληλεπιδράσεις μεταξύ αντικειμένων ή τμημάτων μέσω διαδοχικών μηνύματων. Αποτελούνται από ένα συνδυασμό πληροφοριών που χρησιμοποιούνται και στα διαγράμματα κλάσεων, τα ακολουθιακά διαγράμματα και τα διαγράμματα περιπτώσεων χρήσης περιγράφοντας τόσο τη στατική δομή όσο και τη δυναμική συμπεριφορά ενός συστήματος.
  - *Διαγράμματα γενικής αλληλεπίδρασης (Interaction overview diagram):* Είναι ένας τύπος διαγραμμάτων δραστηριοτήτων όπου οι κόμβοι αναπαριστούν διαγράμματα αλληλεπίδρασης.
  - *Χρονικά Διαγράμματα (Timing diagrams):* Αποτελούν ένα ειδικό τύπο διαγραμμάτων αλληλεπίδρασης που επικεντρώνονται σε χρονικούς περιορισμούς.

Η γλώσσα UML επιτρέπει τη χρήση μεταμοντέλων τα οποία επεκτείνουν τη σύνταξη της γλώσσας για να γίνει εφικτή η χρήση της σε διαφορετικά πεδία ενδιαφέροντος. Η δημιουργία προφιλ (profiling) είναι μια ειδική τεχνική δημιουργίας μεταμοντέλου που επιτρέπει την επέκταση των μετακλάσεων (metaclasses) ενός υπάρχοντος μεταμοντέλου για την προσαρμογή τους σε διαφορετικές χρήσεις. Τα προφίλ αποτελούνται από “στερότυπα” (stereotypes) που επεκτείνουν τις υπάρχουσες μετακλάσεις και από ετικέτες (tags) που εκφράζουν μεταχαρακτηριστικά. Με άλλα λόγια τα στερεότυπα μας δίνουν τη δυνατότητα να προσδώσουμε μια καινούρια σημασιολογία σε κάποια σύμβολα της UML, ενώ οι ετικέτες εμπλουτίζουν τα στοιχεία με πρόσθετες πληροφορίες αναλόγως την περιοχή στην οποία θέλουμε να εφαρμοστεί το προφίλ που ορίζεται. Έχοντας ως σημείο εκκίνησης τα προφίλ που έχουν οριστεί, συμβατά μοντέλα εφαρμογών μπορούν να κατασκευαστούν.

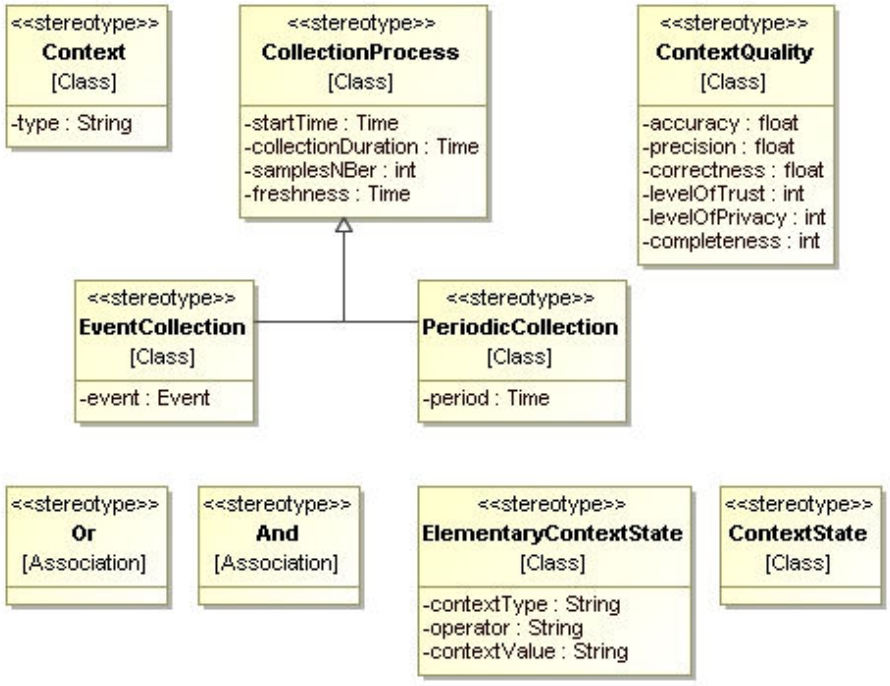
Στη μοντελοκεντρική μηχανική τα προφίλ μπορούν να λειτουργήσουν και ως οδηγός για την παραγωγή κώδικα.

5.1.1.2 UML Μοντέλα

Μία από τις πρώτες και πιο ενδιαφέρουσες προσεγγίσεις για τη μοντελοποίηση του πλαισίου χρήσης και των αλληλεπιδράσεων μεταξύ του context και της υπηρεσίας είναι η ContextUML [35]. Η ContextUML αποτελεί ένα UML μεταμοντέλο που επεκτείνει τη σύνταξη της UML εισάγοντας κατάλληλα στοιχεία για τη δημιουργία μοντέλων υπηρεσιών με επίγνωση του πλαισίου χρήσης. Τα μοντέλα που προκύπτουν από τη ContextUML αποτελούνται από διαγράμματα κλάσεων, όπου οι κλάσεις αντιστοιχούν σε πληροφορίες πλαισίου χρήσης, καθώς και από διαγράμματα υπηρεσιών, ενώ διάφορες UML εξαρτήσεις και σχέσεις εκφράζουν την αλληλεπίδραση μεταξύ των κλάσεων του πλαισίου χρήσης και των κλάσεων των υπηρεσιών. Η ContextUML μπορεί να χρησιμοποιηθεί για να εκφράσει περιπτώσεις προσαρμογής που περιλαμβάνουν την αντικατάσταση των παραμέτρων των μεθόδων των υπηρεσιών βάσει του πλαισίου χρήσης, αλλά και τη διαχείριση των απαντήσεων των υπηρεσιών με βάση και πάλι το πλαίσιο χρήσης.

Ένα πιο γενικό προφίλ σε UML έχει προταθεί στο [4] (Εικόνα 20). Το προφίλ υποστηρίζει τη δομική προσαρμογή, αρχιτεκτονική προσαρμογή και προσαρμογή συμπεριφοράς του σχεδιασμού της υπηρεσίας στο πλαίσιο χρήσης. Οι δομικές προσαρμογές σχετίζονται με την επέκταση των κλάσεων της υπηρεσίας με πρόσθετες μεταβλητές και μεθόδους, ενώ η αρχιτεκτονική προσαρμογή αφορά την επιλεκτική δημιουργία αντικειμένων. Οι προσαρμογές συμπεριφοράς από την άλλη επεκτείνουν τα ακολουθιακά διαγράμματα της UML, ώστε να υποστηρίζεται η σχεδίαση ροών που ενεργοποιούνται επιλεκτικά.

Ένα άλλο UML profile σχεδιασμένο για τη μοντελοκεντρική ανάπτυξη εφαρμογών με επίγνωση του πλαισίου χρήσης περιγράφεται στο [18]. Σε αυτό το προφίλ το πλαίσιο χρήσης χωρίζεται σε context κατάστασης (state-based context) που χαρακτηρίζει την τρέχουσα κατάσταση μιας οντότητας και context γεγονότος (event-based context) που αναπαριστά αλλαγές στην κατάσταση μιας οντότητας. Περιορισμοί χρησιμοποιούνται και στους δύο τύπους για να οδηγήσουν σε διάφορες κλήσεις μεθόδων: οι περιορισμοί κατάστασης αναφέρονται σε συγκεκριμένες χρονικές στιγμές, ενώ οι περιορισμοί γεγονότων χρησιμοποιούν ιστορικά δεδομένα γεγονότων που σχετίζονται με το πλαίσιο χρήσης.



Εικόνα 20. Διάγραμμα UML προφίλ των Ayed και Berbers.

Άλλες ενδιαφέρουσες προτάσεις για τη μοντελοποίηση του πλαισίου χρήσης σε UML προτείνονται στα [1] και [16].

Τα μοντέλα υπηρεσιών και πλαισίου χρήσης που σχεδιάζονται με κάποιον από τους παραπάνω τρόπους μοντελοποίησης μπορούν να μετασχηματιστούν σε κώδικα από εργαλεία ανοικτού λογισμικού που επιτρέπουν τη μετατροπή του μοντέλου σε εκτελέσιμο κώδικα. Ιδιαίτερο ενδιαφέρον για τις υπηρεσίες διαδικτύου από τις παραπάνω προσεγγίσεις αποτελεί η ContentUML. Για αυτό το λόγο για τη μοντελοποίηση του πλαισίου χρήσης και τις εξαρτήσεις με τις υπηρεσίες διαδικτύου στη διατριβή έχει χρησιμοποιηθεί μια επέκταση της ContextUML μέσω της ανάπτυξης ενός προφίλ που περιλαμβάνει όλες τις απαραίτητες πληροφορίες για τη διευκόλυνση του μετασχηματισμού σε κώδικα.

5.1.2 Οντολογίες και DSL

Αν και η επιλογή της UML ως γλώσσας μοντελοποίησης είναι συχνή στη μοντελοκεντρική προσέγγιση, πολλές φορές το πλαίσιο χρήσης μιας υπηρεσίας, εφαρμογής ή συστήματος μοντελοποιείται με οντολογίες ή γλώσσες ειδικού πεδίου που παρουσιάζονται συνοπτικά στην παρούσα υποενότητα.

Οι οντολογίες [40] χρησιμοποιούν τις αρχές του σημασιολογικού ιστού (Semantic Web) [5] για τη δημιουργία αναπαραστάσεων που περιγράφουν την πληροφορία πλαισίου χρήσης και παρέχουν τη δυνατότητα εξαγωγής συμπερασμάτων από τη διαθέσιμη

πληροφορία. Συνήθως χρησιμοποιούν το Πλαίσιο Περιγραφής Πόρων (Resource Description Framework / RDF) [42] και τη Γλώσσα Οντολογιών Διαδικτύου (Web Ontology Language / OWL) [43] και συνδυάζονται με τεχνολογίες μεσισμικού για να παρέχουν πλήρη διαχείριση του πλαισίου χρήσης. Από την άλλη, οι γλώσσες ειδικού πεδίου [24] είναι γλώσσες ειδικά σχεδιασμένες για συγκεκριμένους τύπους εφαρμογών ή κλάσεις προβλημάτων και δεν έχουν εφαρμογή εκτός του πεδίου που αφορούν. Η UML σε αντίθεση με τις γλώσσες DSL θεωρείται μια γλώσσα μοντελοποίησης γενικού ενδιαφέροντος (general-purpose modeling language).

Μια οντολογία συνοδεύει την αρχιτεκτονική του SOCAM (Service-oriented Context-Aware Middleware), όπου το πλαίσιο χρήσης αναπαριστάται στη γλώσσα OWL υποστηρίζοντας με αυτόν τον τρόπο ταξινόμηση του context, σημασιολογικές μεθόδους και εξαγωγή συμπερασμάτων με εστίαση σε περιβάλλοντα “έξυπνων” σπιτιών [45]. Μέσω της οντολογίας διευκολύνεται η ενημέρωση για την πληροφορία πλαισίου χρήσης μεταξύ διαφόρων οντοτήτων (ατόμων, συσκευών και υπηρεσιών) και εισάγεται ένα είδος κατηγοριοποίησης εκχωρώντας συγκεκριμένα χαρακτηριστικά σε διάφορα υποπεδία χρήσης (π.χ. πεδίο σπιτιού, πεδίο γραφείου, κτλ.). Σημαντικά χαρακτηριστικά αυτής της προσέγγισης αποτελούν η εξαγωγή συμπερασμάτων για το πλαίσιο χρήσης βασισμένη σε συσχετισμένες τιμές πληροφορίας context, καθώς και η ποιότητα του πλαισίου χρήσης, η οποία εισάγει ανάλογους περιορισμούς (π.χ. πόσο πρόσφατη ή πόσο ακριβής είναι η πληροφορία). Διάφορα στοιχεία εισήχθησαν στην OWL για να εκφράσουν την κατηγοριοποίηση του πλαισίου χρήσης και τις αλληλεξαρτήσεις μεταξύ των διάφορων τιμών, ενώ το μοντέλο της οντολογίας υποστηρίζεται από την αρχιτεκτονική SOCAM που είναι προσανατολισμένη σε περιβάλλοντα σπιτιών και περιλαμβάνει τις κατάλληλες υποδομές για τις πηγές, τις υπηρεσίες και τις μηχανές εξαγωγής συμπερασμάτων.

Μια άλλη οντολογία προτείνεται με την αρχιτεκτονική COBRA (Context Broker Architecture), η οποία αναπαριστάται σε XML και ορίζει μερικές συνήθεις σχέσεις και ιδιότητες που σχετίζονται με ανθρώπους, τοποθεσίες και δραστηριότητες. Άλλες προτάσεις για τη μοντελοποίηση της πληροφορίας πλαισίου χρήσης μέσω οντολογιών δίνονται από την επεκτάσιμη οντολογία πλαισίου χρήσης (extensible CONtext ONtology / CONON) [44], καθώς και από τις εργασίες [28] και [17].

Σε μια πιο εξειδικευμένη προσέγγιση μοντελοποίησης που αναφέρεται σε εφαρμογές διαδικτύου προτείνεται ένα framework που υποστηρίζει το σχεδιασμό πολυκαναλικών εφαρμογών διαδικτύου με επίγνωση του πλαισίου χρήσης [11]. Ακολουθώντας την προτεινόμενη μέθοδο οι εφαρμογές μπορούν να προσαρμόζονται αυτόματα στο πλαίσιο

χρήσης χωρίς την παρέμβαση του χρήστη. Ο σχεδιασμός των δεδομένων (application data) διαχωρίζεται από το σχεδιασμό του υπερκειμένου (hypertext), του κώδικα δηλ. της ιστοσελίδας (application front-end). Με αυτόν τον τρόπο τα δεδομένα της εφαρμογής εμπλουτίζονται με πληροφορία πλαισίου χρήσης και ο κώδικας της ιστοσελίδας προσαρμόζεται εν συνεχεία στο context. Η πληροφορία πλαισίου χρήσης προστίθεται στα δεδομένα της εφαρμογής σε μορφή μεταδεδομένων, ενώ η δυνατότητα επίγνωσης του πλαισίου χρήσης εισάγεται στην ιστοσελίδα μέσω της προσθήκης context-aware σελίδων (C-pages). Οι C-pages είναι υπεύθυνες για την ανάκτηση της πληροφορίας πλαισίου χρήσης και τις αντίστοιχες ενεργοποίησεις. Η διαδικασία μοντελοποίησης της εφαρμογής γίνεται σε μια ειδική DSL γλώσσα, τη WebML, και συνοδεύεται από ένα CASE εργαλείο, το WebRatio [10].

Μια παρουσίαση της διαδικασίας μοντελοποίησης που απευθύνεται συγκεκριμένα σε εφαρμογές διαδικτύου υπάρχει στο [23].

5.2 Προσεγγίσεις Μοντελοκεντρικής Ανάπτυξης για Υπηρεσίες με Επίγνωση του Πλαισίου Χρήσης

Σε πολλές περιπτώσεις η παράδοση μιας υπηρεσίας απαιτεί μια πολύπλοκη διαδικασία μηχανικής λογισμικού που περνάει από τα στάδια της ανάλυσης και του σχεδιασμού μέχρι να φτάσει στην υλοποίηση της υπηρεσίας. Ένα θέμα που είναι ιδιαίτερο κρίσιμο είναι η διαχείριση της πληροφορίας πλαισίου χρήσης σε αυτά τα στάδια ανάπτυξης μιας εφαρμογής. Το μοντέλο της εφαρμογής μπορεί να χρησιμοποιηθεί μόνο για λόγους τεκμηρίωσης ή – μέσω των κατάλληλων εργαλείων – μπορεί να χρησιμοποιηθεί για να παράγει κάποιο τμήμα ή ακόμα ολόκληρη την εφαρμογή ακολουθώντας τις αρχές της μοντελοκεντρικής ανάπτυξης. Η μοντελοκεντρική ανάπτυξη έχει επεκταθεί και υιοθετηθεί επιτυχώς από πολλούς ερευνητές τα τελευταία χρόνια. Γι’ αυτό το λόγο παρατηρείται ένας σημαντικός αριθμός προσφάτων προσεγγίσεων διαχείρισης πλαισίου χρήσης που επικεντρώνονται στη μοντελοκεντρική ανάπτυξη εφαρμογών με επίγνωση του πλαισίου χρήσης.

Στο [30] προτείνεται ο συνδυασμός της μοντελοκεντρικής ανάπτυξης με τις αρχές του AOP. Συγκεκριμένα έχει επεκταθεί η χρήση της ContextUML για τη μοντελοποίηση του πλαισίου χρήσης, ενώ γίνεται χρήση UML διαγραμμάτων δραστηριοτήτων για τη μοντελοποίηση της ροής της εφαρμογής. Το μοντέλο της εφαρμογής μετατρέπεται εν συνεχεία από την XMI αναπαράστασή του στην AOP έκδοση της Java, την AspectJ. Και σε

άλλες μελέτες γίνεται αναφορά στα πλεονεκτήματα του συνδυασμού της μοντελοκεντρικής ανάπτυξης με το πρότυπο του AOP [9].

Στο UML προφίλ που προτείνεται στο [18] η πληροφορία πλαισίου χρήσης μπορεί να χρησιμοποιηθεί για να αντιστοιχηθεί με συγκεκριμένες τιμές ή για να τροποποιήθει η δομή ή η συμπεριφορά της εφαρμογής. Στο προφίλ περιγράφονται τα ρυθμιστικά στοιχεία (monitor elements) που είναι υπεύθυνα για την ανάκτηση της πληροφορίας πλαισίου χρήσης και οι προσαρμογείς (adapters) που προσαρμόζουν την εφαρμογή βάσει αυτής της πληροφορίας, προκαλώντας π.χ. την εκτέλεση πρόσθετων ενεργειών, κτλ. Το UML μοντέλο που δημιουργείται με αυτόν τον τρόπο μεταφράζεται έπειτα και σε αυτή την προσέγγιση σε κώδικα σε AspectJ, όπου τα στοιχεία των adapter και monitor υλοποιούνται ως aspects.

Σε μια πιο πρόσφατη εργασία [41] προτείνεται η υιοθέτηση των αρχών της μοντελοκεντρικής αρχιτεκτονικής κατά τη σχεδίαση εφαρμογών σε συνδυασμό με παραμέτρους, των οποίων οι τιμές επηρεάζουν τους μετασχηματισμούς που πραγματοποιούνται κατά τη διαδικασία ανάπτυξης. Οι παράμετροι που χρησιμοποιούνται εκφράζουν πληροφορία πλαισίου χρήσης (*Context Property* και *Context Data Type*), ενώ ο εντοπισμός των στοιχείων του μοντέλου που ενδέχεται να χρήζουν προσαρμογή στο πλαίσιο χρήσης γίνεται μέσω “οδηγών” (templates) που μαρκάρουν τα αντίστοιχα στοιχεία με το σύμβολο της δίεσης (*#)*. Με άλλα λόγια στη συγκεκριμένη προσέγγιση η γλώσσα μετασχηματισμού επηρεάζεται από το πλαίσιο χρήσης, αλλά δεν περιγράφεται η μετατροπή σε κώδικα. Μια παρόμοια προσέγγιση που προσπαθεί να ενοποιήσει τη UML και τη μοντελοκεντρική αρχιτεκτονική με γλώσσες περιγραφής οντολογιών περιγράφεται στο [27].

Για την περίπτωση των εφαρμογών διαδικτύου στο [11] έχει υλοποιηθεί μια επέκταση του εργαλείου που βασίζεται στη WebML για την υποστήριξη επίγνωσης πλαισίου χρήσης. Μετά από το σχεδιασμό της εφαρμογής σε WebML ο αντίστοιχος κώδικας των σελίδων παράγεται αυτόματα ακολουθώντας τις αρχές του μοντέλου. Αξίζει να αναφερθεί στο σημείο αυτό και η παλαιότερη μέθοδος σχεδιασμού Object Oriented Hypermedia Design Method (OO-HDM) για την ανάπτυξη εφαρμογών διαδικτύου [34]. Η OO-HDM είναι μια αντικειμενοστραφής μέθοδος που βασίζεται σε διάφορα μοντέλα για την περιγραφή της πληροφορίας, της πλοήγησης (navigation) και των στοιχείων διεπαφής των εφαρμογών. Κατά τη διαδικασία παραγωγής κώδικα χρησιμοποιείται η αντιστοίχιση των μοντέλων σε εκτελέσιμες εφαρμογές για διάφορα περιβάλλοντα. Το HyperDE είναι ένα περιβάλλον ανοικτού κώδικα που υποστηρίζει τη μέθοδο OO-HDM και είναι υλοποιημένο στο Ruby on Rails [33]. Μια ενδιαφέρουσα παρατήρηση για τις δύο παραπάνω

προσεγγίσεις της WebML και της OO-HDM είναι ότι εστιάζουν περισσότερο στη μοντελοποίηση πληροφορίας παρά στη μοντελοποίηση της λειτουργίας της εφαρμογής διαδικτύου.

Οι μοντελοκεντρικές προσεγγίσεις ωφελούνται από τα πλεονεκτήματα ανάπτυξης που προσφέρει η μοντελοκεντρική μηχανική, αλλά απαιτούν αρχικά μια καλή σχεδίαση του συνολικού συστήματος, όπου συμπεριλαμβάνονται τόσο η λογική της υπηρεσίας όσο και η πληροφορία πλαισίου χρήσης. Φυσικά όσον αφορά την προσαρμογή στο πλαίσιο χρήσης απαιτείται να υπάρχει η κατάλληλη υποδομή για την προσαρμογή στο επίπεδο της υλοποίησης. Στις περισσότερες από τις παραπάνω προσεγγίσεις γίνεται λόγος για μετατροπές μεταξύ μοντέλων χωρίς να υπάρχει εκτενή αναφορά στη διαδικασία μετατροπής σε εκτελέσιμο κώδικα για ρεαλιστικές εφαρμογές. Η λύση της διατριβής φιλοδοξεί στη μετατροπή κώδικα λειτουργικών εφαρμογών, όπου το μοντέλο της εφαρμογής οδηγεί την παραγωγή του κώδικα διευκολύνοντας την εργασία του μηχανικού λογισμικού. Με τον τρόπο αυτό η διαδικασία διευκολύνεται σημαντικά, καθώς παράγεται το μεγαλύτερο κομμάτι του κώδικα και ο προγραμματιστής ενδέχεται να χρειαστεί να επέμβει μόνο σε συγκεκριμένα τμήματα.

5.3 Εργαλεία Μοντελοκεντρικής Ανάπτυξης

Διάφορα MDE εργαλεία χρησιμοποιούνται στη μοντελοκεντρική ανάπτυξη, στα οποία η μοντελοποίηση πραγματοποιείται συνήθως σε UML ή σε κάποια άλλη γλώσσα που ορίζει το εργαλείο. Σε αυτό το σημείο δίνεται μια σύντομη περιγραφή εργαλείων μοντελοκεντρικής ανάπτυξης. Στον πίνακα που ακολουθεί (Πίνακας 4) παρουσιάζεται μια λίστα των διαθέσιμων εργαλείων ανάπτυξης, όπου φαίνεται η γλώσσα μοντελοποίησης που υιοθετούν και το αν αποτελούν λύσεις ανοικτού κώδικα μαζί με μια σύντομη περιγραφή του εργαλείου.

MDE Εργαλείο	Γλώσσα Μοντελοποίησης	Ανοικτού Κώδικα	Σύντομη Περιγραφή
AndroMDA [2]	UML	Ναι	Στηρίζει τη λειτουργία του στη συνεργασία διαφόρων εργαλείων. Μπορεί να λειτουργήσει ως plugin για διάφορα εργαλεία ανάπτυξης ή για ολοκληρωμένα περιβάλλοντα ανάπτυξης.

OpenArchitecture Ware [25]	EMF, UML2, XML, JavaBeans	Ναι	Είναι ένας MDA/MDD παραγωγός κώδικα υλοποιημένος σε Java. Υποστηρίζει μια γλώσσα για τον έλεγχο και τη μετατροπή μοντέλων, καθώς και για την παραγωγή κώδικα βάσει αυτών.
BluAge [6]	EMF, UML	Όχι	Μετατρέπει μοντέλα σε Agile Java Enterprise Edition και .NET εφαρμογές.
ArcStyler [3]	UML	Όχι	Περιλαμβάνει ένα UML περιβάλλον μοντελοποίησης και εργαλεία παραγωγής κώδικα και χρησιμοποιείται για την παραγωγή εφαρμογών J2EE/.NET υποστηρίζοντας γνωστούς εξυπηρετητές εφαρμογών (BEA, IBM, κτλ.).
iQGen [20]	UML	Όχι	Παράγει κώδικα από UML μοντέλα και χρησιμοποιεί ως “οδηγούς” μετατροπής JSP σελίδες.
iUML [21]	Executable UML (xUML)	Όχι	Αποτελείται από ένα περιβάλλον μοντελοποίησης για πολλαπλούς χρήστες (iUML Modeller) και ένα εργαλείο εκτέλεσης και δοκιμών εφαρμογών (iUML Simulator).
Jamda [22]	UML	Ναι	Χρησιμοποιείται για την κατασκευή μετατροπέων κώδικα εφαρμογών που παράγουν Java κώδικα από μοντέλα.
OpenMDX [26]	Διαγράμματα κλάσεων συμβατά με MOF	Ναι	Αποτελεί ένα περιβάλλον μοντελοκεντρικής αρχιτεκτονικής, που μπορεί να συνεργαστεί με αρκετά άλλα εργαλεία χρησιμοποιώντας το XMI. Υποστηρίζει τη δημιουργία κώδικα για

			επιλεγμένες πλατφόρμες (J2EE, .NET).
PathMATE [29]	UML	Όχι	Χρησιμοποιείται για την παραγωγή κώδικα από UML μοντέλα βάσει "οδηγών" σε PathMATE.
Rhapsody [32]	UML 2.1, SysML 1.0 [38], κτλ.	Όχι	Χρησιμοποιεί διάφορες γλώσσες μοντελοποίησης και επιτρέπει στους χρήστες να επεκτείνουν το περιβάλλον μοντελοποίησης προσθέτοντας δυνατότητες γλωσσών ειδικού πεδίου.
CodeGenie [12]	UML	Όχι	Εστιάζει στη μετατροπή UML μοντέλων σε λογισμικό και μπορεί να προσαρμοστεί πλήρως για να υποστηρίξει τις διεργασίες λογισμικού, την αρχιτεκτονική λογισμικού και τα εργαλεία διαχείρισης διαμόρφωσης που επιλέγει ο πελάτης.
SmartGenerator [36]	UML	Όχι	Η XMI αναπαράσταση του μοντέλου μετατρέπεται από το περιβάλλον μοντελοποίησης που επιλέγει ο χρήστης σε κώδικα.
UMT-QVT [39]	UML/XMI	Ναι	Παρέχει ένα περιβάλλον όπου μπορούν να εισαχθούν νέοι μετατροπείς σε κώδικα. Το περιβάλλον είναι υλοποιημένο σε Java και οι μετατροπείς υλοποιούνται είτε σε XSLT είτε σε Java.

Πίνακας 4. Εργαλεία μοντελοκεντρικής ανάπτυξης.

Η πλειοψηφία των εργαλείων παρέχει πλήρη υποστήριξη για το Eclipse Modelling Framework (EMF) [15] του ολοκληρωμένου περιβάλλοντος ανάπτυξης (Integrated Development Environment / IDE) Eclipse [14], το οποίο έχει χρησιμοποιηθεί και στη λύση που προτείνεται στη διατριβή και περιγράφεται στη συνέχεια. Ακόμα δίνεται μια πιο

αναλυτική περιγραφή του AndroMDA που αποτελεί ένα από τα πιο αντιπροσωπευτικά εργαλεία μοντελοκεντρικής ανάπτυξης.

5.3.1 Eclipse EMF

Το Eclipse αποτελεί ένα περιβάλλον ανάπτυξης ανοικτού κώδικα για διάφορες γλώσσες προγραμματισμού, αν και ο βασικός του στόχος είναι η Java. Παρέχει διάφορα εργαλεία που διευκολύνουν τη γρήγορη ανάπτυξη εφαρμογών, ενώ υποστηρίζει και ένα μηχανισμό plugin που επιτρέπουν στο χρήστη να επεκτείνει τις δυνατότητες του περιβάλλοντος εγκαθιστώντας plugin που έχουν αναπτυχθεί από το Eclipse ή από άλλους παρόχους.

Το EMF [8] αποτελεί ένα εργαλείο μοντελοποίησης και παραγωγής κώδικα για την κατασκευή εφαρμογών βάσει ενός δομημένου μοντέλου δεδομένων. Ξεκινώντας από την προδιαγραφή του μοντέλου σε XMI μορφοποίηση, το EMF παρέχει ένα σύνολο εργαλείων υποστήριξης για την παραγωγή ενός συνόλου κλάσεων σε Java για το μοντέλο, καθώς και ένα σύνολο κλάσεων προσαρμογής που επιτρέπουν αλλαγές στο μοντέλο (editing) μαζί με ένα περιβάλλον τροποποίησης του μοντέλου (editor). Το EMF παρέχει τη βάση για διαλειτουργικότητα με άλλα βασισμένα στο EMF εργαλεία και εφαρμογές.

Το EMF στοχεύει στη μετατροπή των μοντέλων σε αποδοτικό κώδικα. Η μοντελοποίηση σε EMF αποτελεί ουσιαστικά ένα υποσύνολο των δομών που μπορούν να οριστούν με τη γλώσσα UML (απλοί ορισμοί κλάσεων, ιδιοτήτων και των μεταξύ τους σχέσεων). Τα μοντέλα μπορούν να καθοριστούν με διάφορους τρόπους προκειμένου να εισαχθούν στο EMF:

- Απευθείας κατασκευή της XMI αναπαράστασης του μοντέλου μέσω ενός περιβάλλοντος δημιουργίας XML αρχείων
- Εξαγωγή του XMI αρχείου από κάποιο εργαλείο μοντελοποίησης, όπως το Rational Rose [31]
- Χρήση κώδικα Java με υπομνηματισμούς (annotated Java) για την περιγραφή των στοιχείων του μοντέλου
- Χρήση XML σχημάτων για την περιγραφή της μορφής μιας σειριοποίησης του μοντέλου

Ακολούθως ο EMF μετατροπέας δημιουργεί βάσει του μοντέλου το αντίστοιχο σύνολο κλάσεων υλοποίησης. Οι κλάσεις μπορούν να τροποποιηθούν από το χρήστη για να προστεθούν μέθοδοι, ενώ αν επαναληφθεί η διαδικασία παραγωγής κώδικα διατηρούνται

οι αλλαγές που προηγήθηκαν (εκτός αν ο κώδικας που προστέθηκε εξαρτάται από κάποια αλλαγή στο μοντέλο).

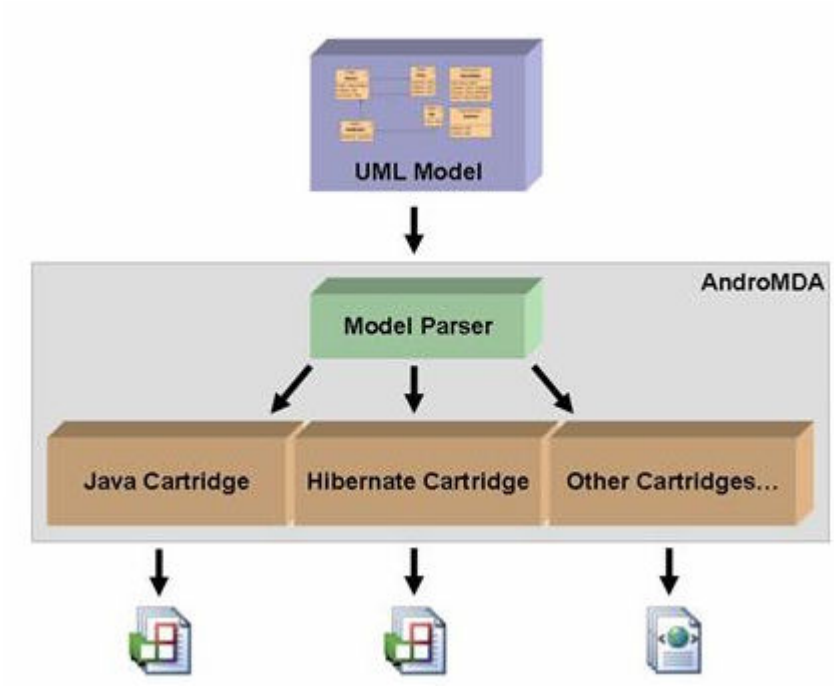
Το EMF αποτελείται από τρία βασικά μέρη:

- Το κύριο περιβάλλον (EMF Core) που περιλαμβάνει ένα μεταμοντέλο (Ecore) για την περιγραφή και την υποστήριξη των μοντέλων συμπεριλαμβανομένων και ειδοποιήσεων για αλλαγές που πραγματοποιούνται, βιβλιοθηκών για το χειρισμό EMF αντικειμένων, κτλ.
- EMF.Edit που περιλαμβάνει γενικές, επαναχρησιμοποιήσιμες κλάσεις για την κατασκευή ενός περιβάλλοντος τροποποίησης για EMF μοντέλα.
- EMF.Codegen που είναι υπεύθυνο για την παραγωγή των απαραίτητων αρχείων για την κατασκευή ενός πλήρους περιβάλλοντος τροποποίησης για EMF μοντέλα. Περιλαμβάνει μια γραφική διεπαφή χρήστη, από την οποία μπορούν να καθοριστούν διάφορες επιλογές ώστε να ξεκινήσει η παραγωγή κώδικα.

5.3.2 AndroMDA

Το AndroMDA αποτελεί ένα εργαλείο ανοικτού λογισμικού γραμμένο σε Java που επιτρέπει τη μετατροπή UML μοντέλων σε συστατικά διαφόρων πλατφόρμων. Το AndroMDA στηρίζει τη λειτουργία του στη συνεργασία διαφόρων εργαλείων: μπορεί να λειτουργήσει ως plugin για διάφορα εργαλεία ανάπτυξης, όπως το Apache Ant και το Apache Maven, ή για ολοκληρωμένα περιβάλλοντα ανάπτυξης, όπως το Eclipse. Το μοντέλο της εφαρμογής παρέχεται στην XMI αναπαράστασή του και μπορεί να χρησιμοποιηθεί ως βάση για τη διαδικασία μετασχηματισμού. Στις πρόσθετες λειτουργίες του AndroMDA περιλαμβάνεται ο έλεγχος των εισαγόμενων στο εργαλείο μοντέλων, οι μετατροπές μεταξύ μοντέλων και η παραγωγή κειμένου (π.χ. για αρχεία διαμόρφωσης).

Το AndroMDA ενσωματώνει τη λειτουργία του με διάφορες πλατφόρμες μέσω ειδικών plugin που αναφέρονται ως cartridges. Μέσω των cartridge υποστηρίζονται μετασχηματισμοί για διάφορες τεχνολογίες (EJB, Hibernate [19], Java, Spring, υπηρεσίες διαδικτύου, XML σχήματα, κτλ.), όπως φαίνεται στην Εικόνα 21.



Εικόνα 21. Αλυσίδα εργαλείων του AndroMDA.

5.4 Βιβλιογραφία

[1] Anagnostopoulos, C., Tsounis, A., Hadjiefthymiades, S., “Context management in pervasive computing environments”, Proc. International Conference on Pervasive Services (ICPS '05), 2005, pp. 421-424.

[2] AndroMDA, http://www.andromda.org/.

[3] ArcStyler, http://www.interactive-objects.com/products/.

[4] Ayed, D., Berbers, Y., “UML profile for the design of a platform-independent context-aware applications”, Proc. 1st Workshop on Model Driven Development for Middleware (MODDM '06), Melbourne, Australia, 2006, pp. 1-5.

[5] Berners-Lee, T., Hendler, J., Lassila, O., “The Semantic Web”, Scientific American, 2001, pp. 34-43.

[6] BluAge, http://www.bluage.com/.

[7] Bolchini, C., Curino, C. A., Quintarelli, E., Schreiber, F. A., Tanca, L., “A data-oriented survey of context models”, ACM SIGMOD, vol. 36(4), December 2007, pp. 19-26.

[8] Budinsky, F., Steinberg, D., Merks, E., Ellersick, R., Grose, T. J., "Eclipse Modeling Framework", The Eclipse Series, Addison Wesley Professional, 2003.

[9] Carton, A., Clarke, S., Senart, A., Cahill, V., "Aspect-Oriented Model-Driven Development for Mobile Context-Aware Computing", Proc.First International Workshop on Software Engineering for Pervasive Computing Applications, Systems, and Environments, (SEPCASE '07), 2007, pp. 191.

[10] Ceri, S., Fraternali, P., Matera, M., "Conceptual Modeling of Data Intensive Web Applications, IEEE Internet Computing, vol. 6(4), 2002, pp. 20-30.

[11] Ceri, S., Daniel, F., Matera, M., "Model-Driven Development of Context-Aware Web Applications", ACM Transactions of Internet Technology, vol. 7(1), article no. 2, 2007.

[12] CodeGenie, http://www.ooagenerator.com/codegenie.htm.

[13] Desmet, B., Vallejos, J., Costanza, P., Meuter, W. D., Hondt, T. D'., "Context-Oriented Domain Analysis", Proc. Sixth International and Interdisciplinary Conference on Modeling and Using Context, (CONTEXT'07), Denmark, Springer Verlag, 2007, pp. 178-191.

[14] Eclipse IDE, http://www.eclipse.org/.

[15] Eclipse Modeling Framework (EMF), http://www.eclipse.org/modeling/emf/.

[16] Farias, C. R. G. de, Preto, R., Leite, M. M., Calvi, C. Z., Pessoa, R. M., Pereira Filho, J. G., "A MOF metamodel for the development of context-aware mobile applications", Proc. 2007 ACM symposium on Applied computing, 2007, pp. 947-952.

[17] de Freitas Bulcao Neto, R., da Graca Campos Pimentel, M., "Toward a domain-independent semantic model for context-aware computing", Proc. Third Latin American Web Congress, (LA-WEB'05), 2005.

[18] Grassi, V., Sindico, A., "Towards Model Driven Design of Service-Based Context-Aware Applications", Proc. International workshop on Engineering of software services for pervasive environments: in conjunction with the 6th ESEC/FSE joint meeting, 2007, pp. 69-74.

[19] Hibernate, http://www.hibernate.org/.

[20] iQGen, http://www.innoq.com/iqgen/.

[21] iUML, http://www.kc.com/products/iuml.php.

[22] Jamda, http://jamda.sourceforge.net/.

[23] Kaltz, W. J., Ziegler J., Lohmann, S., "Context-aware Web Engineering: Modeling and Applications", Revue d'Intelligence Artificielle (RIA), Special Issue on Applying Context-Management, vol. 19(3), 2005, pp. 439-458.

[24] Mernik, M., Heering, J., Sloane, A. M., "When and how to develop domain-specific languages", ACM Computing Surveys, vol. 37(4), December 2005, pp. 316-344.

[25] OpenArchitectureWare, http://www.openarchitectureware.org/.

[26] OpenMDX, http://www.openmdx.org/.

[27] Ou S., Georgalas N., Azmoodeh M., Yang K., Sun X., "A Model Driven Integration Architecture for Ontology-Based Context Modelling and Context-Aware Application Development", Proc. European Conference on Model Driven Architecture – Foundations and Applications, Springer-Verlag Berlin Heidelberg, 2006, pp.188–197.

[28] Ötztürk, P., Aamodt, A., "Towards a model of context for case-based diagnostic problem solving", Proc. interdisciplinary conference on modeling and using context (Context-97), 1997, pp. 198-208.

[29] PathMATE, http://www.pathfindermda.com/.

[30] Prezerakos, G. N., Tselikas, N., Cortese, G., "Model-driven Composition of Context-aware Web Services Using ContextUML and Aspects", Proc. IEEE International Conference on Web Services 2007 (ICWS'07), 2007, pp. 320-329.

[31] Rational Rose data modeler, http://www-01.ibm.com/software/awdtools/developer/datamodeler/.

[32] Rhapsody, http://modeling.telelogic.com/products/rhapsody/index.cfm.

[33] Ruby On Rails, http://rubyonrails.org/.

[34] Schwabe, D., Rossi, G., "Developing Hypermedia Applications using OOHDM", Proc. Workshop on Hypermedia Development Processes, Methods and Models, Pittsburgh, USA, 1998.

[35] Sheng, Q. Z., Benatallah, B., "ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services", Proc. International Conference on Mobile Business (ICMB'05), 2005, pp. 206-212.

[36] SmartGenerator, http://www.bitplan.com/com/bitplan/web/index.php?topic=products/smartgenerator.

[37] Strang, T., Linnhoff-Popien, C., "A Context Modeling Survey", Proc. UbiComp 1st International Workshop on Advanced Context Modelling, Reasoning and Management, Nottingham, 2004, pp. 34-41.

[38] Systems Modeling Language (SysML), http://www.sysml.org/.

[39] UMT-QVT, http://umt-qvt.sourceforge.net/.

[40] Uschold, M., Gruninger, M., "Ontologies: Principles, methods and applications", Knowledge Engineering Review, vol. 11(2), 1996, pp. 93-155.

[41] Vale, S., Hammoudi, S., "Context-aware Model Driven Development by Parameterized Transformation", Proc. Model Driven Interoperability for Sustainable Information Systems (MDISIS'08) held in conjunction with CAiSE'08, 2008, pp. 121-133.

[42] W3C, "RDF/XML Syntax Specification (Revised)", 2004, http://www.w3.org/TR/rdf-syntax-grammar/.

[43] W3C, "OWL Web Ontology Language Overview", 2004, http://www.w3.org/TR/owl-features/.

[44] Wang, X. H., Zhang, D. G., Gu, T., Pung, H. K., "Ontology Based Context Modeling and Reasoning using OWL", Proc. Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW'04), 2004, pp. 18-22.

[45] Ying, X., Fu-yuan, X., "Research on Context Modeling Based on Ontology", Proc. International Conference on Computational Intelligence for Modelling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, 2006, pp. 188-188.

6ο Κεφάλαιο

Μεθοδολογία Μοντελοκεντρικής Ανάπτυξης Εφαρμογών (ContextWS-MDE)

Στο παρόν Κεφάλαιο πραγματοποιείται η παρουσίαση της μοντελοκεντρικής μεθοδολογίας για την ανάπτυξη υπηρεσιών διαδικτύου με επίγνωση του πλαισίου χρήσης. Με αυτόν τον τρόπο η διαδικασία προσαρμογής στο πλαίσιο χρήσης εισάγεται σε όλα τα στάδια ανάπτυξης. Μάλιστα η στρατηγική που προτείνεται επιτρέπει το διαχωρισμό του χειρισμού της πληροφορίας πλαισίου χρήσης από την κύρια λογική της υπηρεσίας, καθώς αυτή η προσέγγιση οδηγεί σε απλούστερα μοντέλα ανάπτυξης λογισμικού ειδικά στα στάδια της ανάλυσης και του σχεδιασμού. Ως περίπτωση χρήσης η διαδικασία ανάπτυξης καταλήγει σε εφαρμογές που είναι συμβατές με την προτεινόμενη αρχιτεκτονική διαχείρισης πλαισίου χρήσης.

Η μεθοδολογία έχει πάρει την ονομασία ContextWS-MDE (Context Web Services – Model-Driven Engineering) και βασίζεται σε συγκεκριμένα προφίλ, ενώ η γλώσσα μοντελοποίησης που έχει επιλεχθεί είναι η UML για λόγους που έχουν αναφερθεί και στο προηγούμενο Κεφάλαιο. Η UML αποτελεί γλώσσα ευρείας εφαρμογής για διάφορες μοντελοποιήσεις, οπότε είναι προτιμότερη η χρήση της έναντι κάποιας γλώσσας με πιο περιορισμένη εφαρμογή. Ένα επιπλέον πλεονέκτημα αποτελεί η ύπαρξη διαθέσιμων βιβλιοθηκών και εργαλείων που διευκολύνουν την προσπέλαση του UML μοντέλου.

Στα πλαίσια της μεθοδολογίας προτείνεται ακόμα μια διαδικασία σύνθεσης υπηρεσιών διαδικτύου για τη δημιουργία μιας εφαρμογής διαδικτύου που βρίσκεται σε συνεχή αλληλεπίδραση με το χρήστη. Εφαρμογές που έχουν ως επίκεντρο το χρήστη (user-centric) εμφανίζονται συχνά το τελευταίο διάστημα. Η ευρεία υιοθέτηση του προτύπου μοντέλου-όψης-ελεγκτή για την ανάπτυξη εφαρμογών διαδικτύου παρέχει επαρκή

αποσύνθεση της εφαρμογής σε ένα σύνολο δομικών συστατικών με συγκεκριμένο ρόλο [14]. Τα περιβάλλοντα μοντέλου-όψης-ελεγκτή επιτρέπουν την άμεση αντιστοίχιση της ροής της διεπαφής χρήστη με τα περιεχόμενα των πηγών της πληροφορίας της εφαρμογής και των αρχείων διαμόρφωσης. Με αυτόν τον τρόπο η εφαρμογή στο επίπεδο σχεδίασης μπορεί να αποτελείται από ένα διάγραμμα που σχετίζεται με καταστάσεις, ενέργειες που εκτελούνται στις καταστάσεις και γεγονότα που ενεργοποιούν τις μεταβάσεις μεταξύ των καταστάσεων, όπου οι ενέργειες που εκτελούνται σχετίζονται με αλληλεπιδράσεις με υπηρεσίες διαδικτύου. Σκοπός είναι να προκύψει μια εφαρμογή αποτελούμενη από διάφορες υπηρεσίες διαδικτύου, όπου η μετάβαση από τη μια υπηρεσία στην άλλη εξαρτάται από τις ενέργειες του χρήστη. Σε αυτή τη διαδικασία λαμβάνεται υπόψη το πλαίσιο χρήσης, καθώς η κάθε υπηρεσία προσαρμόζεται κατά την εκτέλεσή της για τον εκάστοτε πελάτη. Η διαδικασία σύνθεσης βασίζεται σε μοντελοκεντρικές μεθόδους που διευκολύνουν αισθητά την εργασία του προγραμματιστή για αυτή την περίπτωση user-centric εφαρμογών [10] [9].

6.1 Γενική Περιγραφή Μεθοδολογίας

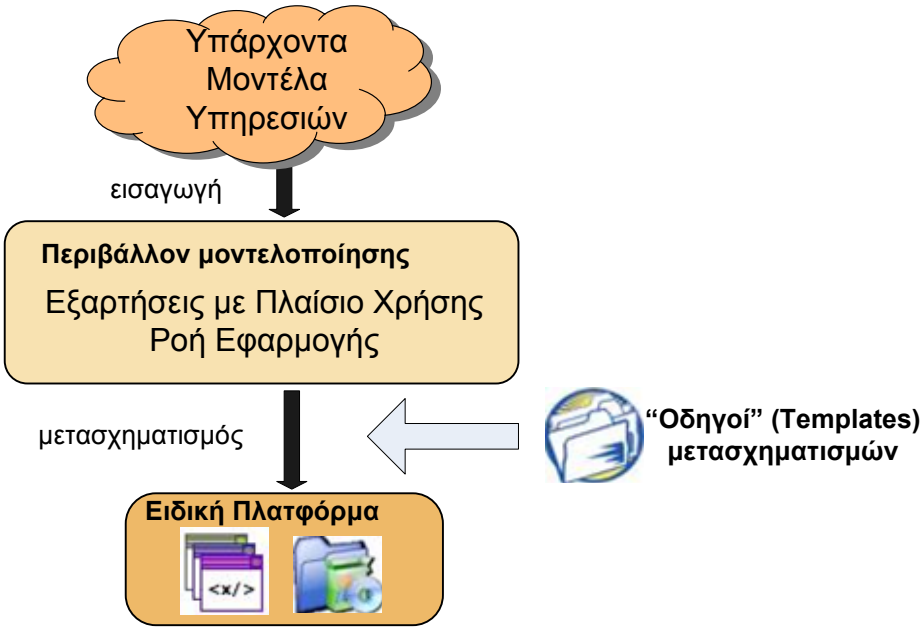
Η προτεινόμενη λύση μοντελοκεντρικής ανάπτυξης επιτρέπει στους μηχανικούς λογισμικού να επαναχρησιμοποιήσουν υπάρχουσες υπηρεσίες λογισμικού ως δομικά συστατικά για τη δημιουργία σύνθετων εφαρμογών. Η προσαρμογή των υπηρεσιών στο πλαίσιο χρήσης κατά την εκτέλεσή είναι ανεξάρτητη από τις ίδιες τις υπηρεσίες, ιδιότητα που διατηρείται και κατά το στάδιο του σχεδιασμού. Η σχεδίαση της εφαρμογής ακολουθείται από τη μετατροπή του μοντέλου της εφαρμογής σε κώδικα, η οποία επιτυγχάνεται μέσω μοντελοκεντρικών εργαλείων ανάπτυξης και μετασχηματισμών μοντέλων, όπως παρουσιάζεται στην Εικόνα 22.

Η διαδικασία ανάπτυξης αποτελείται από τις εξής φάσεις:

- ***Εισαγωγή υπαρχόντων μοντέλων υπηρεσιών διαδικτύου σε UML αναπαράσταση***: αυτό το βήμα επιτρέπει την επαναχρησιμοποίηση UML μοντέλων υπηρεσιών στην εφαρμογή που αναπτύσσεται.
- ***Σχεδίαση της εφαρμογής στο περιβάλλον σχεδιασμού:*** η εφαρμογή μοντελοποιείται σε UML κάνοντας χρήση και των μοντέλων που εισήχθησαν στο προηγούμενο βήμα. Οι εξαρτήσεις μεταξύ της πληροφορίας πλαισίου χρήσης και κάθε υπηρεσίας της εφαρμογής, καθώς και η ροή της εφαρμογής, η οποία καθορίζει τη σειρά με την οποία καλούνται οι διάφορες μέθοδοι των υπηρεσιών,

μοντελοποιούνται στο σημείο αυτό, ώστε να προκύψει τελικά το πλήρες μοντέλο της εφαρμογή διαδικτύου.

- ***Μετασχηματισμός στον κώδικα της επιλεγμένης πλατφόρμας:*** το αποτέλεσμα του ενδιάμεσου σταδίου σχεδίασης μετατρέπεται στα απαραίτητα για την εκτέλεση της εφαρμογής αρχεία (εκτελέσιμος κώδικας, αρχεία διαμόρφωσης, κτλ.) μέσω των κατάλληλων μετασχηματισμών μοντέλου βάσει των ορισμένων "οδηγών" μετασχηματισμών.



Εικόνα 22. Βασικά βήματα μεθοδολογίας μοντελοκεντρικής ανάπτυξης.

Ιδιαίτερη έμφαση κατά την παραπάνω διαδικασία δίνεται στον ορισμό των κατάλληλων UML προφίλ για τη μοντελοποίηση των εφαρμογών. Τα προφίλ σχετίζονται τόσο με το πλαίσιο χρήσης και τις υπηρεσίες διαδικτύου όσο και με το επίπεδο παρουσίασης της εφαρμογής και περιγράφονται αναλυτικά στην ενότητα που ακολουθεί. Εκφράζουν διάφορα απαραίτητα για την εφαρμογή στοιχεία που πρέπει να αντικατοπτριστούν και στον τελικό κώδικα (εξαρτήσεις από το πλαίσιο χρήσης, όψεις της εφαρμογής, δυνατότητα πλοήγησης μεταξύ των όψεων της εφαρμογής, κτλ.). Τα εισαχθέντα μοντέλα της πρώτης φάσης χρειάζεται να είναι και αυτά συμβατά με το αντίστοιχο προφίλ μοντελοποίησης που σχετίζεται με τις υπηρεσίες διαδικτύου.

6.2 Προφίλ Μοντελοποίησης

Για το στάδιο της σχεδίασης της εφαρμογής με επίγνωση του πλαισίου χρήσης ορίζονται τρία διαφορετικά UML προφίλ:

- Προφίλ υπηρεσιών διαδικτύου
- Μεταμοντέλο πλαισίου χρήσης
- Προφίλ παρουσίασης

Χρησιμοποιώντας διαφορετικά μοντέλα προσδίδεται ευελιξία στη διενέργεια διαφόρων συνδυασμών υπηρεσιών διαδικτύου, πληροφορίας πλαισίου χρήσης και τρόπων παρουσίασης και πλοήγησης για την εφαρμογή. Η αλληλεπίδραση των μοντέλων συνίσταται σε λίγες συσχετίσεις που επιτρέπουν στο μηχανικό λογισμικού να κάνει αλλαγές σε κάποιο από τα μοντέλα ή ακόμα και να το αντικαταστήσει με κάποιο άλλο χωρίς να απαιτείται ο επανασχεδιάσμος ολόκληρης της εφαρμογής.

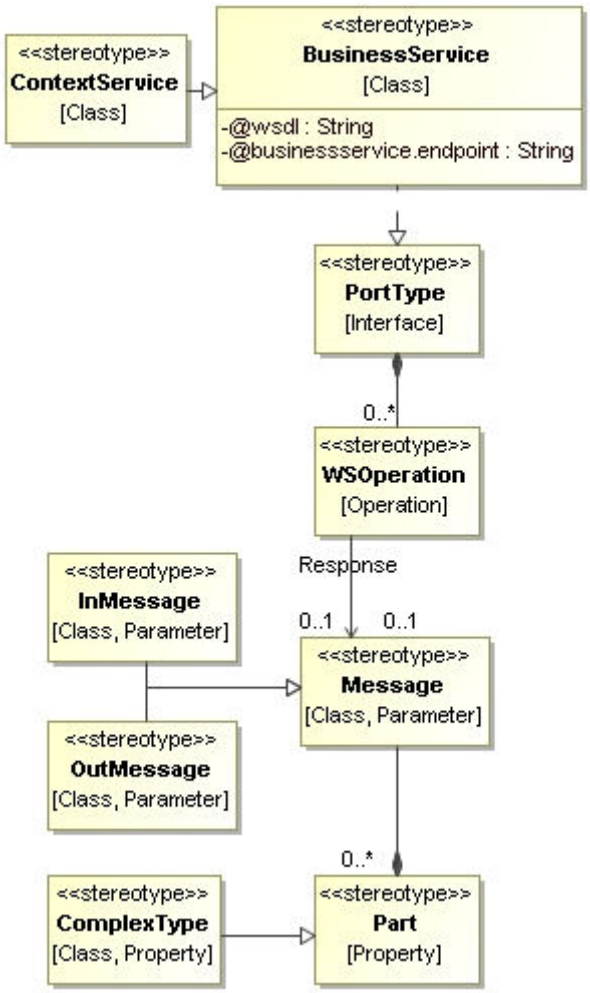
6.2.1 Προφίλ Υπηρεσιών Διαδικτύου

Η WSDL γλώσσα περιγραφής των υπηρεσιών διαδικτύου μπορεί να μετατραπεί σε UML αναπαράσταση εισάγοντας στερεότυπα που αντιστοιχούν στα στοιχεία του WSDL αρχείου. Με αυτόν τον τρόπο προκύπτει μια αντιστοιχία μεταξύ του μοντέλου και των στοιχείων της υπηρεσίας στο WSDL αρχείο όσον αφορά τα τερματικά σημεία, τις μεθόδους, τα μηνύματα, κτλ. που περιγράφονται. Στην περίπτωση που δεν είναι διαθέσιμο το μοντέλο της υπηρεσίας διαδικτύου σε UML αυτή η αντιστοιχία επιτρέπει την κατασκευή ή και την παραγωγή του μοντέλου από το WSDL αρχείο μέσω κατάλληλων μετασχηματισμών μοντέλου σε μοντέλο.

Το προφίλ, το οποίο παρουσιάζει ομοιότητες με προσεγγίσεις που έχουν προταθεί και σε άλλες εργασίες για την περιγραφή υπηρεσιών διαδικτύου (όπως στην εργασία [13], όπου προτείνεται η μοντελοκεντρική ανάπτυξη υπηρεσιών διαδικτύου), παρουσιάζεται στην Εικόνα 23. Οι τιμές που εμφανίζονται μέσα σε αγκύλες σε κάθε στερεότυπο του προφίλ δείχνουν τις μετα-κλάσεις του μετα-μοντέλου της UML που επεκτείνει. Στο προφίλ χρησιμοποιούνται στερεότυπα για την αναπαράσταση:

- των τερματικών σημείων της υπηρεσίας διαδικτύου (`<<PortType>>`),
- των μεθόδων που υποστηρίζει (`<<WsOperation>>`),
- των μηνυμάτων που δέχεται η υπηρεσία για αιτήσεις ως εισερχόμενα και που στέλνει ως απαντήσεις ως εξερχόμενα μηνύματα (στερεότυπα `<<InMessage>>` και `<<OutMessage>>` αντίστοιχα που κληρονομούν τα χαρακτηριστικά του στερεοτύπου `<<Message>>`),
- των τμημάτων των μηνυμάτων (`<<Part>>`) που αναπαριστούν τις ιδιότητες ή τις παραμέτρους που περιέχονται στα μηνύματα της υπηρεσίας, και, τέλος,

- των πολύπλοκων τύπων δεδομένων (<<ComplexType>>) που μπορούν να χρησιμοποιηθούν ως τμήματα μηνυμάτων.

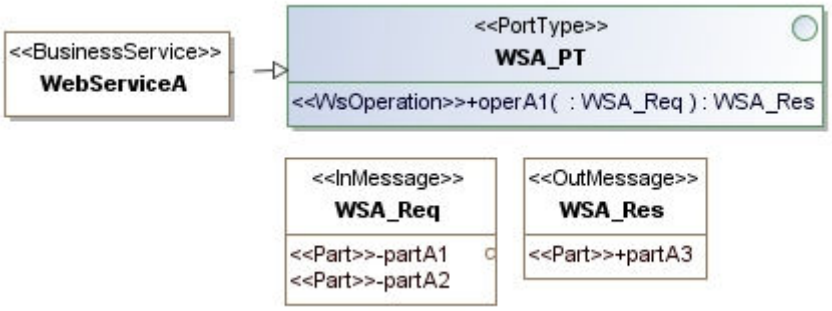


Εικόνα 23. Προφίλ για την περιγραφή των υπηρεσιών διαδικτύου.

Το προφίλ χρησιμοποιείται για τη σχεδίαση μοντέλων που αναπαριστούν τις υπηρεσίες εφαρμογών BWS, αλλά και την πιο ειδική κατηγορία υπηρεσιών που παρέχουν την πληροφορία πλαισίου χρήσης ως υπηρεσίες πλαισίου χρήσης CWS (το αντίστοιχο στερεότυπο <<ContextService>> κληρονομεί τα χαρακτηριστικά του <<BusinessService>>). Οι ετικέτες wsdl και businessservice.endpoint που χρησιμοποιούνται επιπλέον για την περιγραφή των υπηρεσιών σχετίζονται αντίστοιχα με το WSDL αρχείο από το οποίο έχει προέλθει το UML μοντέλο και το ενιαίο προσδιοριστικό πόρου (Uniform Resource Identifier / URI), μέσω του οποίου είναι διαθέσιμη η υπηρεσία διαδικτύου.

Ένα απλό παράδειγμα μοντέλου για μια υπηρεσία διαδικτύου βάσει του παραπάνω προφίλ παρουσιάζεται στην Εικόνα 24, όπου η υπηρεσία WebServiceA υλοποιεί τη

μέθοδο `operA1` με τα αντίστοιχα μηνύματα αίτησης και απάντησης (`WSA_Req` και `WAS_Res`).



Εικόνα 24. Παράδειγμα UML μοντελοποίησης υπηρεσίας διαδικτύου.

6.2.2 Μεταμοντέλο Πλαισίου Χρήσης

Το προφίλ περιγραφής των υπηρεσιών διαδικτύου μπορεί να συνδυαστεί με ένα μεταμοντέλο πλαισίου χρήσης για το σχηματισμό ενός ολοκληρωμένου προφίλ που επιτρέπει τη μοντελοποίηση της πληροφορίας πλαισίου χρήσης, αλλά και:

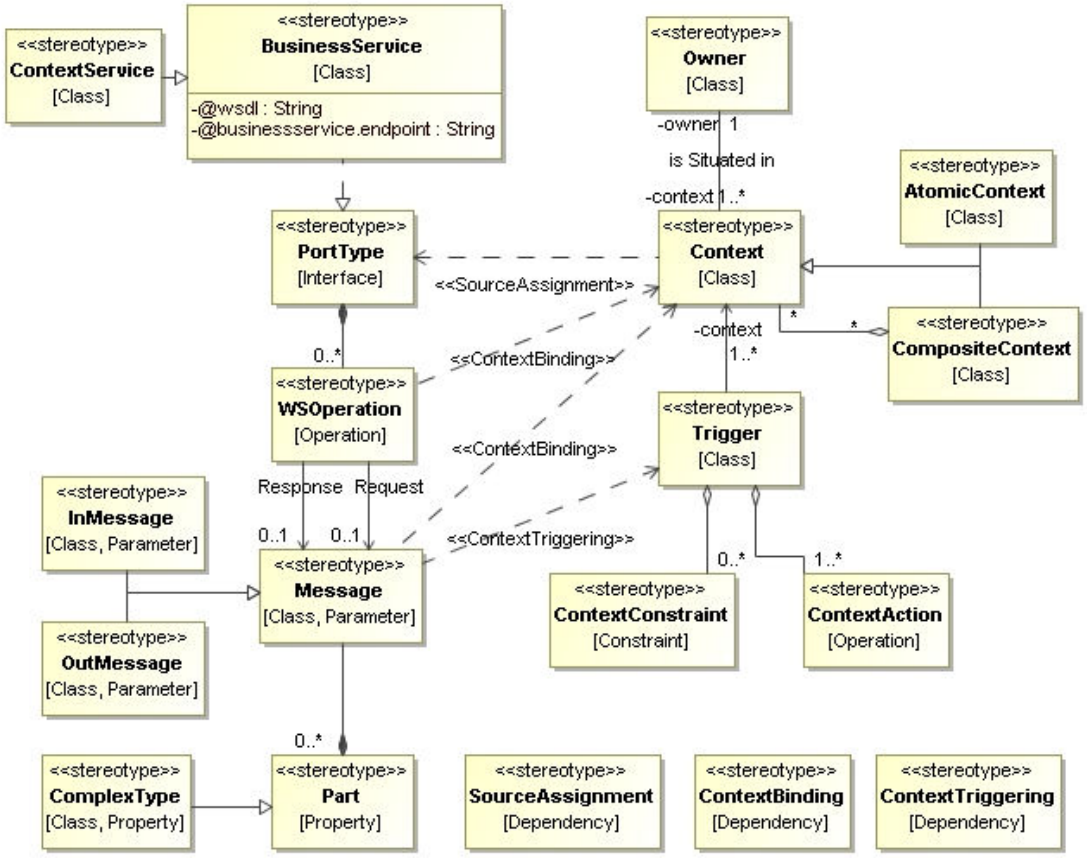
- των συνδέσεων της με την κατάλληλη υπηρεσία πλαισίου χρήσης, από την οποία ανακτάται η πληροφορία, και
- των συσχετισμών μεταξύ των υπηρεσιών εφαρμογών και του πλαισίου χρήσης.

Για τη μοντελοποίηση του πλαισίου χρήσης προτείνεται ένα μεταμοντέλο που βασίζεται σε μια τροποποιήμένη έκδοση του μεταμοντέλου της ContextUML (ExtContextUML). Το μεταμοντέλο παρουσιάζεται στη δεξιά πλευρά της Εικόνας 25. Στην Εικόνα φαίνονται και οι συσχετίσεις με το προφίλ των υπηρεσιών διαδικτύου.

Το μεταμοντέλο πλαισίου χρήσης επεκτείνει τη σύνταξη της UML μέσω ενός συνόλου στερεοτύπων. Βασικό στερεότυπο του μεταμοντέλου αποτελεί το `<<Context>>` που χρησιμοποιείται για την περιγραφή γενικής πληροφορίας πλαισίου χρήσης και διακρίνεται σε δύο υποτύπους:

- την ατομική πληροφορία `<<AtomicContext>>`, η οποία αναπαριστά πληροφορία με απλές τιμές που παρέχεται απευθείας από κάποια πηγή πλαισίου χρήσης (π.χ. ένδειξη θερμοκρασίας, γεωγραφικές συντεταγμένες μιας τοποθεσίας, κτλ.), και
- τη σύνθετη πληροφορία `<<CompositeContext>>`, η οποία αποτελεί συνάθροιση τιμών που προέρχονται από άλλες ατομικές ή σύνθετες τιμές (π.χ. η ημερομηνία γέννησης του χρήστη χρησιμοποιείται για να προκύψει η τιμή για το αν είναι ανήλικος ή ενήλικος). Με άλλα λόγια οι σύνθετες τιμές κατασκευάζονται

χρησιμοποιώντας τιμές που προέρχονται από άλλα δεδομένα. Για παράδειγμα, μια υψηλή τιμή για τη θερμοκρασία και μια χαμηλή τιμή για την ένταση των ανέμων μπορούν να συνδυαστούν σε μια τιμή που περιγράφει καλές καιρικές συνθήκες.



Εικόνα 25. Μεταμοντέλο πλαισίου χρήσης και συσχετίσεις με το προφίλ των υπηρεσιών διαδικτύου.

Το stereotype <<Owner>> που συνδέεται με την κλάση <<Context>> αναφέρεται στον ιδιοκτήτη της πληροφορίας πλαισίου χρήσης ή αλλιώς στην οντότητα με την οποία σχετίζεται η πληροφορία. Ιδιοκτήτης της πληροφορίας μπορεί να θεωρηθεί ο χρήστης όταν πρόκειται για τα προσωπικά του δεδομένα, μια κινητή συσκευή όταν περιγράφονται χαρακτηριστικά υλικού και λογισμικού, κτλ.

Μεταξύ των κλάσεων που αναπαριστούν το πλαίσιο χρήσης και των κλάσεων που αναπαριστούν υπηρεσίες διαδικτύου υπάρχουν διάφορες εξαρτήσεις. Η εξάρτηση με στερεότυπο <<SourceAssignment>> συνδέει την πληροφορία με τις αντίστοιχες πηγές πλαισίου χρήσης, οι οποίες παρέχουν τις τιμές για τις ιδιότητες που περιγράφονται στις κλάσεις <<Context>>. Η αντίστοιχη σύνδεση φαίνεται στο παραπάνω διάγραμμα μεταξύ των στερεοτύπων <<Context>> και <<PortType>>, αν και είναι δυνατόν η εξάρτηση

να καταλήγει και στην πλευρά του <<Operation>> της υπηρεσίας για να εκφράσει τη συγκεκριμένη μέθοδο που πρέπει να κληθεί για την ανάκτηση της πληροφορίας.

Οι μηχανισμοί μέσω των οποίων αναπαριστάται η εξάρτηση της υπηρεσίας από το πλαίσιο χρήσης φαίνονται μέσω των σχέσεων με στερεότυπα <<ContextBinding>> και <<ContextTriggering>>. Η σχέση <<ContextBinding>> μοντελοποιεί την άμεση συσχέτιση της πληροφορίας πλαισίου χρήσης με τα στοιχεία των υπηρεσιών που χρειάζονται προσαρμογή (μέθοδοι και μηνύματα ή παράμετροι) και δηλώνει πως υπάρχει ανάγκη για αντικατάσταση της τιμής κάποιου στοιχείου με την τιμή που αντιστοιχεί στο πλαίσιο χρήσης. Το στερεότυπο <<ContextTriggering>> από την άλλη μοντελοποιεί την προσαρμογή που σχετίζεται με την τροποποίηση της λειτουργίας της υπηρεσίας βάσει της πληροφορίας πλαισίου χρήσης. Μια εξάρτηση <<ContextTriggering>> καταλήγει σε μια κλάση <<Trigger>> που αποτελείται από δύο μέρη:

- Ένα στοιχείο που μέσω του στερεοτύπου <<ContextConstraint>> εκφράζει τους περιορισμούς πλαισίου χρήσης που περιγράφονται στη γλώσσα Object Constraint Language (OCL) [12] της UML.
- Μια δράση (στερεότυπο <<ContextAction>>) που εκτελείται όταν πληρούνται όλοι οι περιορισμοί που ορίζονται. Η δράση αναφέρεται συνήθως σε κάποια μέθοδο που μπορεί να ανήκει είτε σε υπηρεσία διαδικτύου που υλοποιεί την απαιτούμενη λειτουργικότητα είτε σε κάποια άλλη λειτουργία της εφαρμογής που μοντελοποιείται στο περιβάλλον σχεδίασης.

Η OCL αποτελεί τυπική γλώσσα για την έκφραση περιορισμών σε UML διαγράμματα. Μάλιστα είναι μια γλώσσα που μπορεί να χρησιμοποιηθεί σε οποιοδήποτε MOF μοντέλο ή μεταμοντέλο για να εκφράσει περιορισμούς που δε μπορούν να εκφραστούν μέσω κάποιας διαφορετικής αναπαράστασης στο διάγραμμα του μοντέλου. Μια OCL έκφραση κατασκευάζεται από τέσσερα τμήματα:

1. Ένα πλαίσιο χρήσης (context[1]) που δηλώνει την περιορισμένη κατάσταση για την οποία είναι έγκυρη η συνθήκη
2. Μια ιδιότητα που αναπαριστά κάποια χαρακτηριστικά του πλαισίου χρήσης (π.χ. αν το πλαίσιο χρήσης της έκφρασης αναφέρεται σε μία κλάση, η συγκεκριμένη ιδιότητα μπορεί να αναφέρεται σε μια ιδιότητα της κλάσης)

[1] Η έννοια του context που χρησιμοποιείται εδώ είναι γενικότερη και δε σχετίζεται αποκλειστικά με τον ορισμό πλαισίου χρήσης που αφορά υπηρεσίες, όπως ορίζεται στο εισαγωγικό Κεφάλαιο της διατριβής.

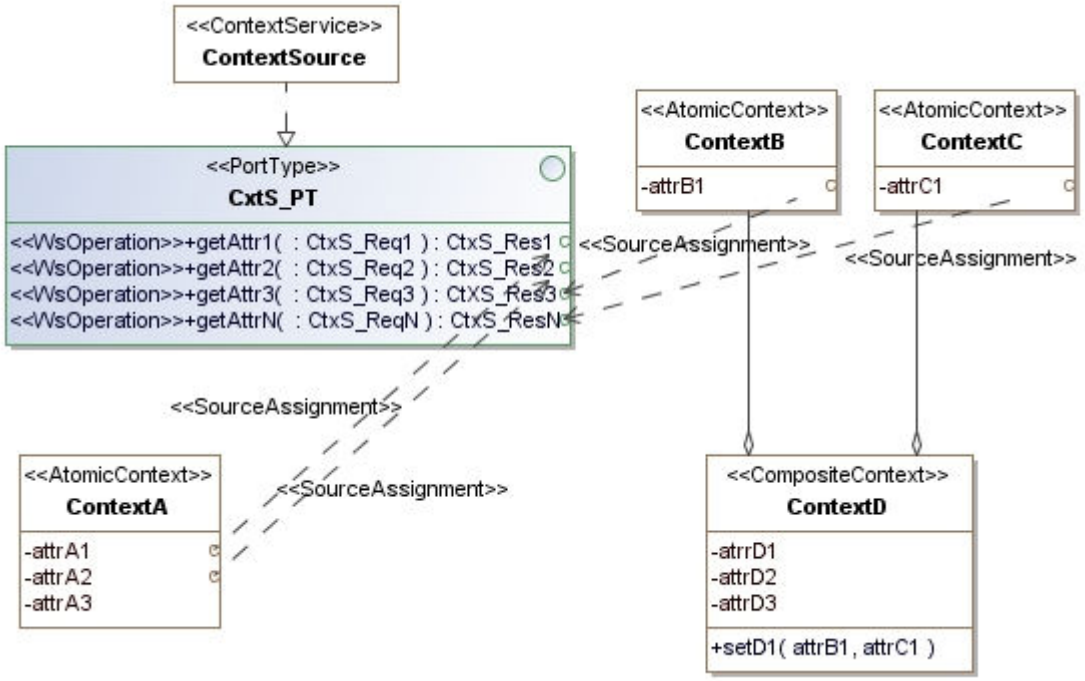
3. Μια μέθοδο (π.χ. αριθμητική) που χειρίζεται ή τροποποιεί μια ιδιότητα
4. Λέξεις κλειδιά (π.χ. if, then, else, and, or, not, implies) που χρησιμοποιούνται για να καθορίσουν συνθήκες

Οι OCL εκφράσεις που ορίζουν περιορισμούς είναι τριών ειδών:

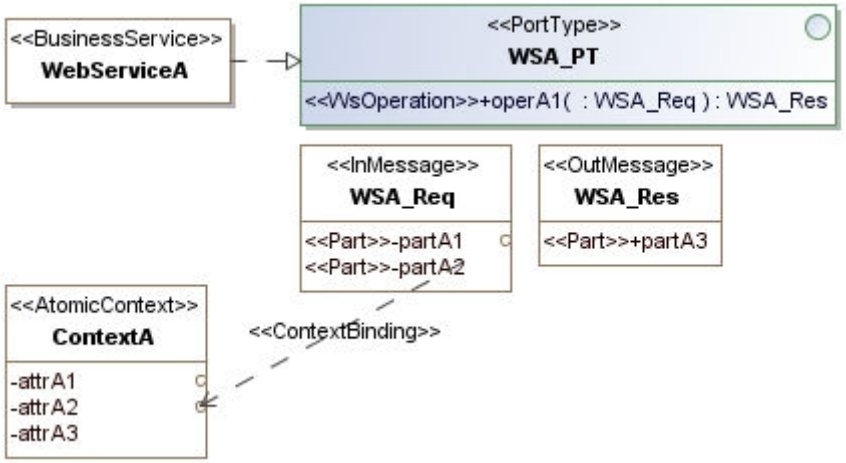
- Αναλλοίωτες συνθήκες (invariant conditions)
- Προαπαιτούμενες συνθήκες (pre-conditions)
- Συνθήκες αποτελέσματος (post-conditions)

Σύμφωνα με τα παραπάνω το μεταμοντέλο του πλαισίου χρήσης μπορεί να χρησιμοποιηθεί σε συνδυασμό με το προφίλ των υπηρεσιών διαδικτύου για να εκφράσει τις τρεις διαφορετικές περιπτώσεις προσαρμογής που παρουσιάστηκαν στο Κεφάλαιο 4. Η περίπτωση αντικατάστασης παραμέτρου, καθώς και η περίπτωση επιλογής μεθόδου, (περιπτώσεις 1 και 2) εκφράζονται μέσω των σχέσεων `<<ContextBinding>>`. Όταν αυτό το στερεότυπο ενώνει κάποια κλάση πλαισίου χρήσης ή κάποια ιδιότητα της κλάσης με κάποιο μήνυμα ή τμήμα μηνύματος μιας υπηρεσίας διαδικτύου πρόκειται για προσαρμογή στην οποία απαιτείται τροποποίηση παραμέτρου, ενώ όταν μια τέτοια σχέση καταλήγει σε κλάση διεπαφής που υλοποιεί μια υπηρεσία απαιτείται προσαρμογή επιλογής μεθόδου. Η σχέση `<<ContextTriggering>>` από την άλλη αναφέρεται στην τροποποίηση της απάντησης και συνδέει το μήνυμα απάντησης της υπηρεσίας με κάποια κλάση `<<Trigger>>` που υλοποιεί την απαραίτητη τροποποίηση. Η κλάση `<<Trigger>>` μπορεί επίσης να συνδέεται με την κατάλληλη κλάση πληροφορίας πλαισίου χρήσης (υπό των συνθηκών που εκφράζονται στον περιορισμό `<<ContextConstraint>>`).

Στην Εικόνα που ακολουθεί (Εικόνα 26) παρουσιάζεται ένα παράδειγμα μοντελοποίησης πληροφορίας πλαισίου χρήσης και αντίστοιχων πηγών. Η πληροφορία μοντελοποιείται μέσω των κλάσεων `ContextA`, `ContextB`, `ContextC` και `ContextD`, όπου η τιμή του `<<CompositeContext>>` `ContextD` προκύπτει από το συνδυασμό των τιμών των `ContextB` και `ContextC`. Όλες οι τιμές των ιδιοτήτων της πληροφορίας πλαισίου χρήσης ανακτώνται από διαφορετικές μεθόδου της ίδιας πηγής (`ContextSource`). Στη συνέχεια στην Εικόνα 27 παρουσιάζεται και μια περίπτωση μοντελοποίησης της εξάρτησης αντικατάστασης παραμέτρου. Η εξάρτηση αφορά την υπηρεσία διαδικτύου `WebServiceA` και την ιδιότητα `attrA2` της κλάσης πληροφορίας πλαισίου χρήσης `ContextA`.



Εικόνα 26. Παράδειγμα μοντελοποίησης πληροφορίας πλαισίου χρήσης και πηγών.

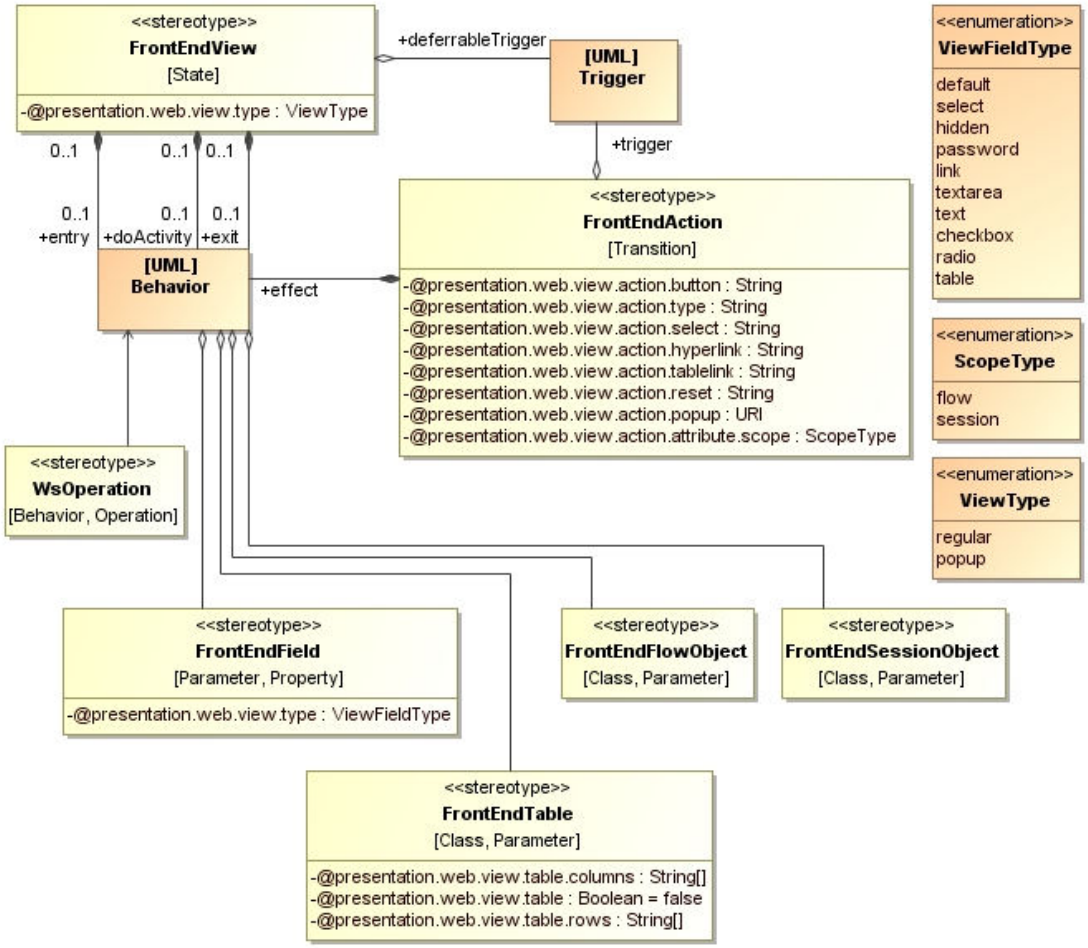


Εικόνα 27. Παράδειγμα μοντελοποίησης εξάρτησης αντικατάστασης παραμέτρου.

6.2.3 Προφίλ Παρουσίασης Εφαρμογής

Το προφίλ για την παρουσίαση της εφαρμογής και των υπηρεσιών που την απαρτίζουν (presentation profile) σχετίζεται με την αναπαράσταση του επιπέδου παρουσίασης της εφαρμογής διαδικτύου και χρησιμοποιείται για τη μοντελοποίηση της διεπαφής του χρήστη, της ροής της εφαρμογής (μέσω της εναλλαγής των διαφόρων όψεων) και των λειτουργιών πλοήγησης του χρήστη στην εφαρμογή. Η ροή ενσωματώνει επαναχρησιμοποιήσιμα και διαδοχικά βήματα που μπορούν να εκτελούνται υπό

διαφορετικές συνθήκες. Το προφίλ περιέχει τις αρχές για το σχεδιασμό των στοιχείων των όψεων (views) της εφαρμογής, ενώ είναι τελείως ανεξάρτητο από τα προηγούμενα προφίλ των υπηρεσιών διαδικτύου και του πλαισίου χρήσης που παρουσιάστηκαν και μπορεί κάλλιστα να χρησιμοποιηθεί ανεξάρτητα για τη μοντελοποίηση εφαρμογών διαδικτύου χωρίς την παρουσία κάποιου μηχανισμού προσαρμογής στο πλαίσιο χρήσης. Το προφίλ, το οποίο παρουσιάζει χαρακτηριστικά που υπάρχουν και σε ένα προφίλ παρουσίασης που χρησιμοποιείται στο εργαλείο μοντελοκεντρικής ανάπτυξης AndroMDA [2], παρουσιάζεται στην Εικόνα 28 μαζί με ένα υποσύνολο του διαγράμματος κλάσεων για την περιγραφή των διαγραμμάτων καταστάσεων της προδιαγραφής της UML. Όσες κλάσεις ορίζονται από το μεταμοντέλο της UML και όχι από το παρόν προφίλ συμβολίζονται με `[UML]` πριν από το όνομά τους.



Εικόνα 28. Προφίλ για την παρουσίαση της εφαρμογής.

Χρησιμοποιώντας το παραπάνω προφίλ η εφαρμογή διαδικτύου μοντελοποιείται μέσω ενός UML διαγράμματος καταστάσεων. Τα διαγράμματα καταστάσεων μπορούν να χρησιμοποιηθούν για τη μοντελοποίηση ευδιάκριτων καταστάσεων των αντικειμένων μιας

εφαρμογής, καθώς και των εξαρτήσεων μεταξύ αυτών των καταστάσεων Είναι ιδανικά για την περιγραφή της συμπεριφοράς ενός μόνο αντικειμένου που στην προκειμένη περίπτωση είναι η υπό ανάπτυξη εφαρμογή. Ωστόσο, σε περιπτώσεις όπου μας ενδιαφέρουν περισσότερα του ενός αντικείμενα τα διαγράμματα δραστηριοτήτων ή τα διαγράμματα αλληλεπίδρασης που παρουσιάζουν και αυτά χρήσιμα χαρακτηριστικά μπορούν να αποδειχτούν καλύτερη επιλογή [7]. Επιπλέον, τα διάγραμματα καταστάσεων περιγράφουν πιο αποτελεσματικά τα χαρακτηριστικά της εφαρμογής που υπάρχουν στις αρχιτεκτονικές μοντέλου-όψης-ελεγκτή και τις αντίστοιχες υλοποιήσεις, όπως τα Apache Struts και Spring MVC. Ειδικά όσον αφορά την επέκταση Spring Web Flow του Spring MVC η σημασιολογία των στοιχείων που το απαρτίζουν για την περιγραφή της ροής του ελεγκτή μπορεί να αντιστοιχηθεί απευθείας σε χαρακτηριστικά που εκφράζονται από τα διαγράμματα καταστάσεων. Διαγράμματα καταστάσεων χρησιμοποιούνται και από το εργαλείο AndroMDA για τη μοντελοποίηση της ροής της εφαρμογής που μπορεί να αντιστοιχηθεί στις τεχνολογίες Struts ή Java Server Faces (JSF) [8] κατά την παραγωγή κώδικα.

Στα πλαίσια των εφαρμογών διαδικτύου που ακολουθούν το MVC πρότυπο ο σκοπός του προφίλ παρουσίασης είναι να βοηθήσει στην παραγωγή της λειτουργίας του ελεγκτή της εφαρμογής, καθώς και των στοιχείων που αποτελούν τις όψεις της εφαρμογής. Το προφίλ αποτελείται από ένα σύνολο στερεοτύπων και ετικετών με διάφορες λειτουργίες (π.χ. κάποια στοιχεία χρησιμοποιούνται για να εκφράσουν τον τρόπο με τον οποίο κάθε απάντηση αντιστοιχίζεται σε στοιχεία της διεπαφής χρήστη, όπως κουμπιά / buttons, πλαίσια κειμένου, κτλ.). Οι ετικέτες που έχουν οριστεί ακολουθούν τη γενική μορφή:

```
@presentation.web.view.type = text/enumeration_type
```

, όπως φαίνονται στο παραπάνω διάγραμμα του προφίλ μαζί με κάποιους τύπους απαριθμήσεων (enumeration) που χρησιμοποιούνται για τις τιμές των ετικετών.

Το `<<FrontEndView>>` είναι το βασικό στερεότυπο που μπορεί να χρησιμοποιηθεί στις καταστάσεις του διαγράμματος. Αναφέρεται σε στοιχεία που αντιστοιχίζονται σε συστατικά των όψεων της εφαρμογής. Διακρίνονται δύο είδη όψεων ή front end views από τον τύπο `ViewType`: `regular` για τις απλές όψεις και `popup` για τις αναδυόμενες. Οι καταστάσεις του διαγράμματος που εκφράζουν τις όψεις της εφαρμογής συνδέονται με μεθόδους των υπηρεσιών διαδικτύου που εισάγονται ως δραστηριότητες (activities) στις όψεις (π.χ. δραστηριότητες Entry, doActivity ή Exit όπως περιγράφονται στην προδιαγραφή των διαγραμμάτων καταστάσεων). Αυτό φαίνεται στο

διάγραμμα του προφίλ μέσω της σχέσης με την κλάση Behavior που συνδέεται με κλήσεις μεθόδων (κλάση WsOperation).

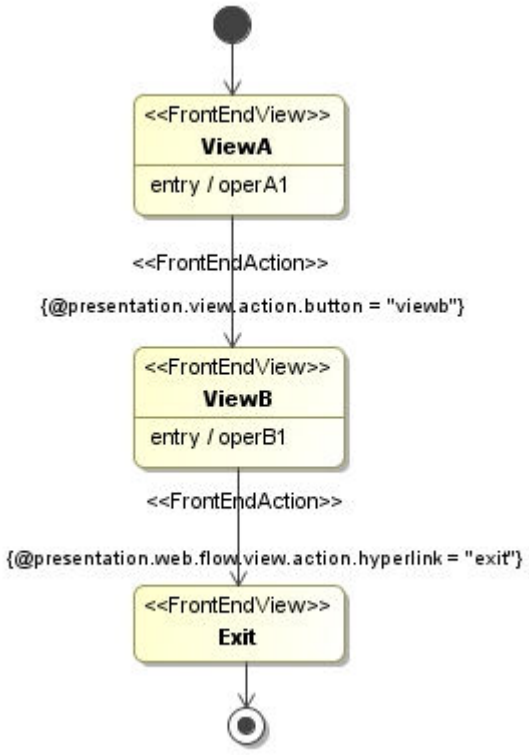
Το στερεότυπο <<FrontEndAction>> χρησιμοποιείται στις μεταβάσεις μεταξύ των καταστάσεων και μοντελοποιεί μια ενέργεια που πραγματοποιείται στην όψη (συνήθως κάποια ενέργεια του χρήστη της εφαρμογής), η οποία και προκαλεί τη μετάβαση σε μια άλλη νέα κατάσταση (και κατ' επέκταση μια νέα όψη). Τέτοιες ενέργειες μπορεί να αποτελούν η επιλογή ενός κουμπιού, ένας υπερσύνδεσμος (hyperlink), η επιλογή ενός συγκεκριμένου πεδίου (field) στην όψη, κτλ. Κάτι τέτοιο εκφράζεται μέσω των τιμών των ετικετών @presentation.web.view.action.XXX που συνοδεύουν το στερεότυπο <<FrontEndAction>>. Για παράδειγμα η ετικέτα @presentation.web.view.action.hyperlink λαμβάνει ως τιμή το όνομα του υπερσυνδέσμου που επιλέχθηκε στην όψη και που εκκίνησε τη διαδικασία εκτέλεσης της σειράς ενεργειών που ενδεχομένως ορίζονται ως Exit activities στην κατάσταση, ώστε να προκύψει τελικά η νέα όψη και κατάσταση (αφού προηγηθεί και η εκτέλεση των Entry δραστηριοτήτων της νέας κατάστασης).

Η ετικέτα @presentation.web.view.action.attribute.scope που μπορεί να πάρει την τιμή flow ή session αναφέρεται στο κομμάτι της εφαρμογής για το οποίο έχουν ισχύ οι ιδιότητες που σχετίζονται με την ενέργεια. Ο διαχωρισμός σε flow (ροή) και session (συνεδρία) γίνεται για να εκφράσει την εμβέλεια μιας πράξης. Η συνεδρία αναφέρεται ως διάρκεια σε ολόκληρη τη διαδικασία επικοινωνίας και αλληλεπίδρασης του χρήστη με την εφαρμογή, ενώ η ροή σχετίζεται συνήθως με ένα υποσύνολο της συνεδρίας και έχει νόημα για την περίπτωση που μια εφαρμογή αποτελείται από υποεφαρμογές ή υποτμήματα που εκφράζονται σε χωριστές ροές. Φυσικά σε πολλές περιπτώσεις η ροή και η συνεδρία συμπίπτουν.

Το στερεότυπο <<FrontEndField>> αναπαριστά ένα γενικό πεδίο της όψης και σχετίζεται με κάποιο τύπο που εμφανίζεται συνήθως στις διαδικτυακές σελίδες ή σε κάποια γραφική διεπαφή χρήστη, όπως κείμενο (text), radio button, check box, κτλ., όπως ορίζεται στον τύπο ViewFieldType. Για την αναπαράσταση πινάκων χρησιμοποιείται το στερεότυπο <<FrontEndTable>> με ορισμένο αριθμό γραμμών (rows) και στηλών (columns) για τον πίνακα. Οι τιμές που χρησιμοποιούνται είτε ως τίτλοι για τον πίνακα είτε ως περιεχόμενο για τα στοιχεία στα πεδία του πίνακα ορίζονται από τις τιμές ετικετών @presentation.web.view.table.rows και @presentation.web.view.table.columns αντίστοιχα. Αυτές οι πληροφορίες εισάγονται μέσω των στοιχείων των ενεργειών (Behavior) των καταστάσεων του διαγράμματος.

Εκτός από τη μοντελοποίηση των όψεων και των μεταβάσεων μεταξύ αυτών υπάρχουν περιπτώσεις όπου είναι επιθυμητή η διατήρηση των τιμών συγκεκριμένων αντικειμένων καθ' όλη τη διάρκεια της συνόδου του χρήστη ή της ροής της εφαρμογής (session και flow, όπως ορίστηκαν παραπάνω). Τα στερεότυπα `<<FrontEnd SessionObject>>` και `<<FrontEndFlowObject>>` χρησιμοποιούνται σε αυτές τις περιπτώσεις με το δεύτερο να έχει μικρότερη εμβέλεια από το πρώτο. Μάλιστα το `<<FrontEndFlowObject>>` χρειάζεται συνήθως για να αποθηκεύονται τιμές στοιχείων που χρειάζεται να διατηρηθούν μεταξύ ενός μικρού αριθμού όψεων (που συνήθως εκφράζουν και μια διαφορετική περίπτωση χρήσης της εφαρμογής) και όχι καθ' όλη τη συνεδρία με το χρήστη, ο οποίος ενδέχεται να ενδιαφέρεται για διάφορες υπολειτουργίες της ίδιας εφαρμογής.

Στην Εικόνα που ακολουθεί παρουσιάζεται μια απλή περίπτωση μοντελοποίησης της ροής μιας εφαρμογής βάσει του προφίλ παρουσίασης (Εικόνα 29). Η εφαρμογή αποτελείται από τρεις όψεις (`ViewA`, `ViewB` και `Exit`), από τις οποίες οι δύο σχετίζονται με κλήσεις υπηρεσιών (`operA1` και `operB1` αντίστοιχα) που μπορεί να ανήκουν στην ίδια υπηρεσία διαδικτύου ή σε διαφορετικές. Στην προκειμένη περίπτωση πρόκειται για διαφορετικές υπηρεσίες, όπως φαίνεται στο δέντρο της Εικόνας 30. Η μετάβαση από την πρώτη όψη στη δεύτερη πραγματοποιείται μέσω του κουμπιού `view2`, ενώ από τη δεύτερη στην τρίτη μέσω του κουμπιού `exit`, όπως ορίζεται στις τιμές ετικετών των αντίστοιχων μεταβάσεων.



Εικόνα 29. Παράδειγμα μοντελοποίησης ροής εφαρμογής.

Εικόνα 30. Δέντρο διαθέσιμων μοντέλων υπηρεσιών.

6.3 Βήματα Μεθοδολογίας Ανάπτυξης

Οι φάσεις από τις οποίες αποτελείται η διαδικασία ανάπτυξης αναφέρθηκαν στη γενική περιγραφή της μεθοδολογίας στην αρχή του Κεφαλαίου. Στην παρούσα ενότητα δίνεται μια πιο αναλυτική περιγραφή της κάθε φάσης ανάπτυξης και των αντίστοιχων βημάτων.

6.3.1 Μοντελοποίηση Εφαρμογής

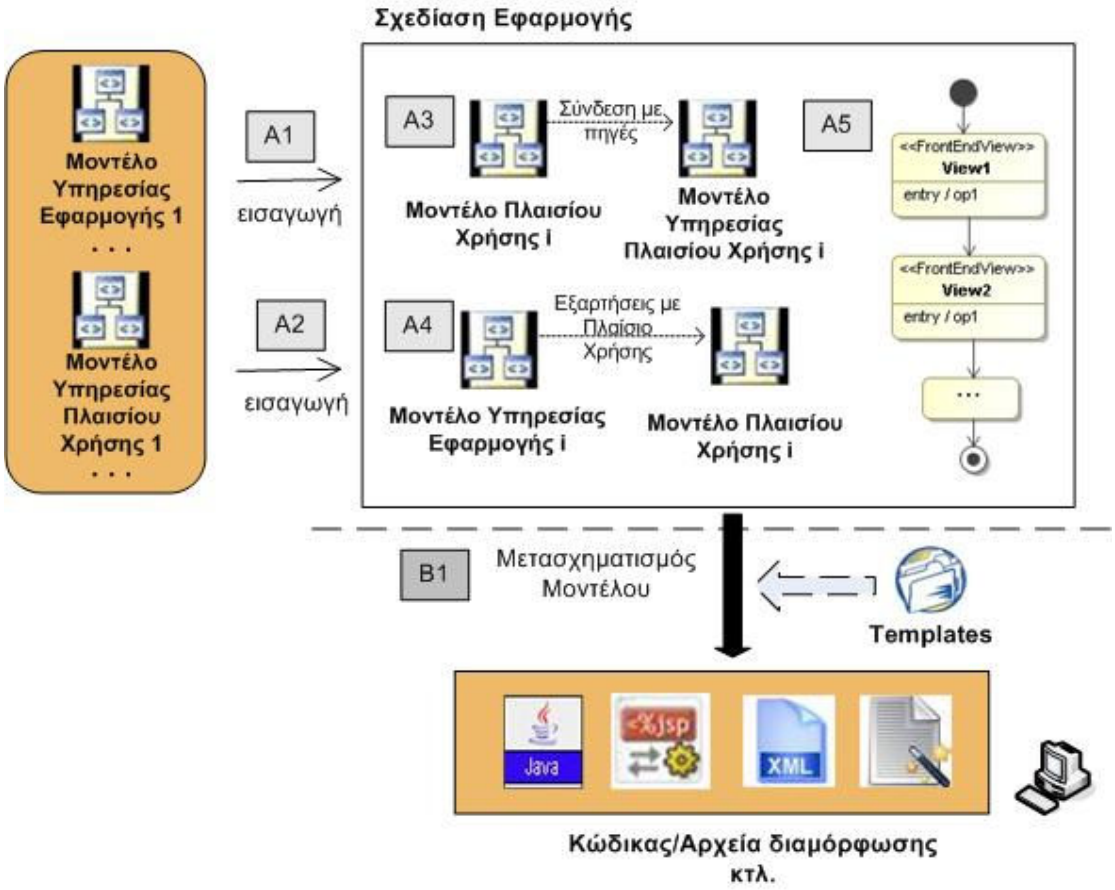
Στο στάδιο μοντελοποίησης ο μηχανικός λογισμικού κατασκευάζει σε διάφορα – συμβατά με τα ορισμένα προφίλ – UML διαγράμματα τα μοντέλα που περιγράφουν την επιθυμητή λειτουργία της εφαρμογής. Ακολούθως, το μοντέλα χρησιμοποιούνται στη διαδικασία μετασχηματισμού και την παραγωγή του κώδικα.

Η διαδικασία μοντελοποίησης μπορεί να διασπαστεί στις ακόλουθες ενέργειες, οι οποίες παρουσιάζονται στο άνω τμήμα της Εικόνας 31:

A1. *Εισαγωγή των διαθέσιμων μοντέλων των υπηρεσιών διαδικτύου που θα χρησιμοποιηθούν στη λειτουργία της εφαρμογής*. Σε περίπτωση που τα μοντέλα δεν είναι διαθέσιμα, οι περιγραφές των υπηρεσιών σε WSDL αντιστοιχίζονται στην αναπαράσταση τους σε UML ακολουθώντας τη δομή που ορίζεται στο προφίλ υπηρεσιών διαδικτύου. Κάτι τέτοιο είναι εφικτό μέσω της αντίστροφης διαδικασίας μετασχηματισμού από τη διαδικασία που προτείνεται στο [13] για την παραγωγή της WSDL περιγραφής της υπηρεσίας από το UML μοντέλο.

A2. *Εισαγωγή των μοντέλων των υπηρεσιών πλαισίου χρήσης (δηλ. των πηγών πλαισίου χρήσης), αν υπάρχουν*. Σε περίπτωση που τα μοντέλα δεν είναι διαθέσιμα οι περιγραφές των υπηρεσιών πλαισίου χρήσης σε WSDL αντιστοιχίζονται στην αναπαράσταση τους σε UML ακολουθώντας τη δομή που

ορίζεται στο προφίλ υπηρεσιών διαδικτύου, όπως και στην προηγούμενη περίπτωση.



Εικόνα 31. Βασικά βήματα της διαδικασίας ανάπτυξης.

A3. *Εισαγωγή (αν είναι ήδη διαθέσιμο) ή σχεδίαση του μοντέλου του πλαισίου χρήσης και των συνδέσεων με τις αντίστοιχες πηγές.* Σε αυτό το σημείο σχεδιάζεται το μοντέλο του πλαισίου χρήσης, καθώς και οι συνδέσεις μεταξύ των στοιχείων του πλαισίου χρήσης και των υπηρεσιών διαδικτύου που προσφέρουν πρόσβαση στην πληροφορία (συσχετίσεις `<<Source Assignment>>`), όπως σχεδιάστηκαν στο προηγούμενο βήμα.

A4. *Μοντελοποίηση των συσχετίσεων μεταξύ των δύο βασικών μοντέλων: της λογικής της λειτουργίας της εφαρμογής και του πλαισίου χρήσης.* Αρχικά χρειάζεται να καθοριστεί το είδος της προσαρμογής που θα πραγματοποιηθεί για κάθε υπηρεσία (για την περίπτωση που απαιτείται κάποια προσαρμογή). Εν συνεχεία οι αντίστοιχες εξαρτήσεις εισάγονται στο περιβάλλον μοντελοποίησης μέσω των σχέσεων `<<ContextBinding>>` και `<<Context Triggering>>`. Οι κλάσεις `<<Trigger>>` και οι αντίστοιχες μέθοδοι που

χρειάζονται για την τροποποίηση των απαντήσεων και δε βρίσκονται στα ήδη υπάρχοντα μοντέλα υπηρεσιών πρέπει επίσης να εισαχθούν ή να σχεδιαστούν.

A5. *Σχεδίαση της ροής της εφαρμογής σε ένα UML διάγραμμα καταστάσεων βάσει των αρχών σχεδίασης του προφίλ παρουσίασης.* Τα μοντέλα των υπηρεσιών διαδικτύου συνδυάζονται για το σχηματισμό μιας εφαρμογής διαδικτύου που αποτελείται από καταστάσεις και μεταβάσεις. Κάθε κατάσταση στο διάγραμμα αντιστοιχεί σε μια όψη της εφαρμογής που σχετίζεται με μία ή περισσότερες κλήσεις υπηρεσιών που εκφράζονται ως δραστηριότητες της κατάστασης (π.χ. ενέργεια κλήσης μεθόδου / call operation action). Οι μέθοδοι υπηρεσιών ανήκουν στα μοντέλα των υπηρεσιών της εφαρμογής που εισήχθησαν στο περιβάλλον στο πρώτο βήμα. Ορισμένες δραστηριότητες σχετίζονται με παραμέτρους που περιγράφουν ποιες απαντήσεις των μεθόδων θα πρέπει να αποτελέσουν κομμάτι των όψεων της εφαρμογής και με ποιον τρόπο (όπως καθορίζεται από τις ετικέτες `<<FrontEndViewField>>` και `<<Front EndTable>>`). Με τον ίδιο τρόπο κάποιες παράμετροι σχετίζονται με τις μεταβάσεις μεταξύ των καταστάσεων ως δραστηριότητες αποτελέσματος (Effect activities). Αυτές οι παράμετροι δείχνουν ποιες τιμές ιδιοτήτων χρειάζεται να διατηρηθούν κατά τη διάρκεια της ροής ή της συνεδρίας (παράμετροι με στερεότυπα `<<FrontEndFlowObject>>` και `<<FrontEndSession Object>>` αντίστοιχα), ενώ οι ετικέτες στις μεταβάσεις αναπαριστούν τα στοιχεία της όψης μέσω των οποίων ενεργοποιείται η μετάβαση στην επόμενη όψη.

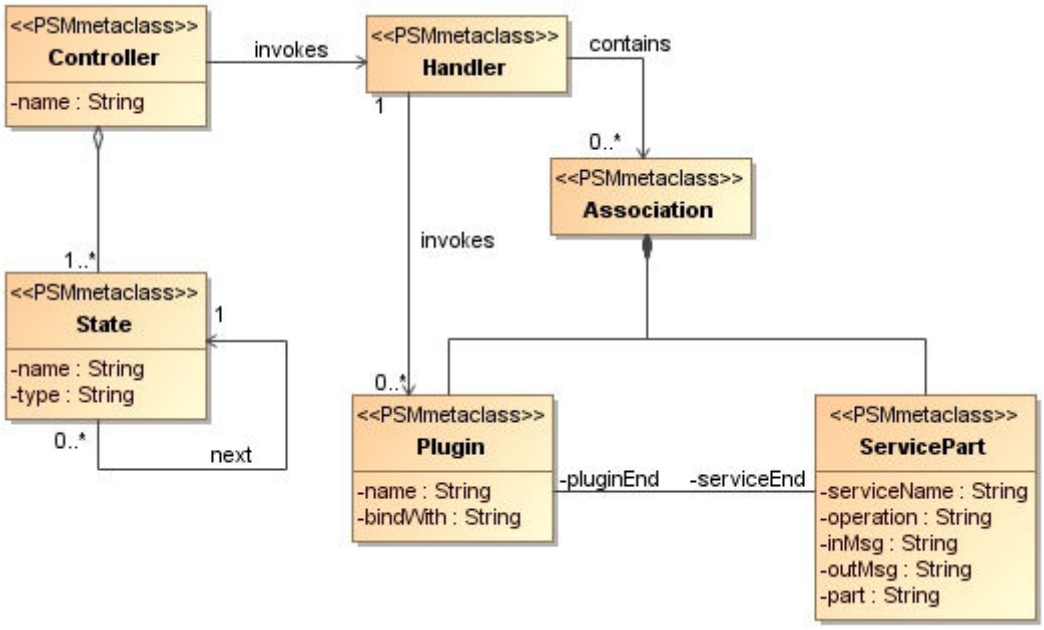
Σημειώνεται πως ως υπηρεσίες πλαισίου χρήσης μπορούν να χρησιμοποιηθούν και υπηρεσίες που υποστηρίζονται από τρίτους παρόχους ως συστατικά για την εφαρμογή αρκεί να εισαχθούν τα αντίστοιχα μοντέλα στο περιβάλλον σχεδίασης. Επίσης, η ροή στο διάγραμμα καταστάσεων μπορεί να περιλαμβάνει καταστάσεις που σχετίζονται με υπηρεσίες διαδικτύου, αλλά και καταστάσεις που σχετίζονται με άλλες λειτουργίες της διαδικτυακής εφαρμογής (π.χ. διαδικασία εισαγωγής του χρήστη στο σύστημα, πρόσθετα μηνύματα, κτλ.). Το διάγραμμα καταστάσεων ενδέχεται σε κάποια σημεία να περιέχει σύνθετες καταστάσεις (composite states) – όπως ορίζεται και στην αντίστοιχη προδιαγραφή της UML. Μάλιστα οι σύνθετες καταστάσεις συνηθίζονται όταν υπάρχουν περιπτώσεις προσαρμογής στο πλαίσιο χρήσης που σχετίζονται με επιλογή μεθόδου, αφού δίνουν τη δυνατότητα αναπαράστασης παραπάνω από μίας κλήσης μεθόδου στην ίδια σύνθετη κατάσταση.

Τα παραπάνω βήματα μπορούν να πραγματοποιηθούν σε κάποιο εργαλείο μοντελοποίησης που υποστηρίζει τη δημιουργία UML διαγραμμάτων και την εισαγωγή UML προφίλ. Προκειμένου να συνδυαστεί το περιβάλλον με την προτεινόμενη διαδικασία ανάπτυξης προϋπόθεση είναι η υποστήριξη της εξαγωγής του μοντέλου σε EMF μορφοποίηση (συνήθως στα εργαλεία μοντελοποίησης η προεπιλεγμένη μορφή αποθήκευσης του μοντέλο είναι η XMI αναπαράσταση). Κάποια παραδείγματα εργαλείων μοντελοποίησης αποτελούν τα: Acceleo [1], ArgoUML [4], MagicDraw [11] και Eclipse UML2 Tools [6], ενώ στη διατριβή έχει χρησιμοποιηθεί κυρίως το εργαλείο MagicDraw στην έκδοση Personal Edition που παρέχεται δωρεάν στα πανεπιστήμια που γίνονται μέλη του ακαδημαϊκού προγράμματος του MagicDraw.

6.3.2 Μετασχηματισμός σε Κώδικα

Αφού ολοκληρωθεί η διαδικασία μοντελοποίησης, ακολουθεί η διαδικασία μετασχηματισμού για την παραγωγή του κώδικα της εφαρμογής. Ο μετασχηματισμός που πραγματοποιείται χρησιμοποιεί στη λειτουργία του τα στερεότυπα και τις ετικέτες που έχουν συμπεριληφθεί στη σχεδίαση των υπηρεσιών διαδικτύου, του πλαισίου χρήσης και της παρουσίασης της εφαρμογής. Με αυτόν τον τρόπο το μοντέλο αντιστοιχίζεται στον πηγαίο κώδικα και στα αρχεία διαμόρφωσης που σχετίζονται με την υλοποίηση για την παροχή μιας λειτουργικής εφαρμογής. Η διαδικασία βασίζεται στους “οδηγούς” που έχουν οριστεί και κατευθύνουν τους μετασχηματισμούς, όπως φαίνεται στο κάτω τμήμα της Εικόνας 31 (βήμα Β1).

Για αυτό το στάδιο έχει αναπτυχθεί ένα εργαλείο μετατροπής που αναλύει όλα τα διαγράμματα που απαρτίζουν το μοντέλο της εφαρμογής και παράγει τον απαιτούμενο κώδικα. Ως περίπτωση χρήσης ο μετατροπέας έχει ως στόχο την αρχιτεκτονική προσαρμογής στο πλαίσιο χρήσης που έχει αναπτυχθεί στα πλαίσια της διατριβής χρησιμοποιώντας ως framework υπηρεσιών διαδικτύου το Apache Axis2, ενώ οι εφαρμογές διαδικτύου ακολουθούν τις αρχές των Spring MVC και Spring Web Flow (Εικόνα 32). Ωστόσο, όσον αφορά το είδος των εφαρμογών που παράγονται η αντιστοίχιση σε κώδικα μπορεί να πραγματοποιηθεί και για διαφορετικές τεχνολογίες ή πλατφόρμες (π.χ. Apache Struts, JSF), καθώς οι βασικές αρχές ανάπτυξης παραμένουν οι ίδιες σε κάθε περίπτωση. Επιπλέον τα προφίλ που έχουν οριστεί είναι επαρκώς γενικά, ώστε να υποστηρίζουν και άλλα είδη μετασχηματισμών. Αυτό που απαιτείται είναι ο ορισμός νέων “οδηγών” μετασχηματισμών που να λαμβάνουν υπόψη τα χαρακτηριστικά της εκάστοτε τεχνολογίας ή πλατφόρμας.



Εικόνα 32. Χαρακτηριστικά στοιχεία επιπέδου υλοποίησης.

Οι βιβλιοθήκες EMF και UML2 που προσφέρει το Eclipse για την υλοποίηση του UML 2.0 μεταμοντέλου [5] (συγκεκριμένα οι εκδόσεις EMF 2.3.1 και UML2 2.1.1) χρησιμοποιούνται στη διαδικασία ανάλυσης (parsing) του μοντέλου σε συνδυασμό με διάφορα άλλα εργαλεία που αναπτύχθηκαν στα πλαίσια της διατριβής. Συγκεκριμένα, για την ανάλυση του μοντέλου όλα τα διαγράμματα διαβάζονται στην EMF αναπαράστασή τους που μπορεί να εξαχθεί είτε από το περιβάλλον μοντελοποίησης είτε από την αντίστοιχη XMI αναπαράσταση. Η διαδικασία πραγματοποιείται από τη βασική κλάση `UML2ModelParser` του εργαλείου μετασχηματισμού, ενώ η προετοιμασία του μοντέλου υλοποιείται στην κλάση `UML2ModelSetup` μέσω της εισαγωγής των στοιχείων των προφίλ.

Εν συνεχεία, η παρουσία συγκεκριμένων στερεοτύπων στα στοιχεία του μοντέλου, όπως αυτά διαβάζονται κατά τη διαδικασία ανάλυσης από την κλάση `UML2ModelParser` (`<<BusinessService>>`, `<<FrontEndView>>`, κτλ.), προκαλεί την εκτέλεση διαδικασιών που σχετίζονται με την παραγωγή κώδικα. Η διαδικασία αυτή βασίζεται στη χρήση κατάλληλων "οδηγών" του Apache Velocity [3], οι οποίοι είναι υπεύθυνοι για την παραγωγή του κώδικα και των άλλων αρχείων διαμόρφωσης.

Το Apache Velocity παρέχει μια πληθώρα εργαλείων, αλλά λειτουργεί στη βάση του ως μια μηχανή “οδηγών” (template engine) που χρησιμοποιείται ευρέως για την παραγωγή κώδικα διαφόρων μορφών (πηγαίο κώδικα, HTML και XML αρχεία, σχήματα βάσεων δεδομένων, PostScript, κτλ.). Το Velocity είναι υλοποιημένο σε Java και δίνει τη δυνατότητα

στον προγραμματιστή να προσθέσει στο πλαίσιό του – που αναφέρεται ως Velocity Context – ένα σύνολο Java αντικειμένων, ώστε να μπορούν ακολούθως τα αντικείμενα αυτά να προσπελαστούν (όσον αφορά την κλήση μεθόδων και ιδιοτήτων των αντικειμένων) από τους Velocity "οδηγούς". Συνήθως χρησιμοποιείται από τους προγραμματιστές στην ανάπτυξη εφαρμογών διαδικτύου λόγω του διαχωρισμού του κώδικα της εφαρμογής από τις σελίδες διαδικτύου που παρέχει. Γενικά, μπορεί να χρησιμοποιηθεί είτε ως ανεξάρτητο εργαλείο για την παραγωγή κώδικα, είτε ως ενσωματωμένο συστατικό σε άλλα συστήματα. Αν και η μηχανή του Velocity είναι απλή, παραμένει αρκετά ισχυρή και υποστηρίζει όλες τις δομές ελέγχου που υπάρχουν στις παραδοσιακές γλώσσες προγραμματισμού (π.χ. if… then…else, for…, κτλ.).

Στα πλαίσια της μεθοδολογίας διάφορα template έχουν σχεδιαστεί για τα στοιχεία υλοποίησης που χρειάζεται να παραχθούν:

- αρχεία JSP σελίδων
- πελάτες υπηρεσιών διαδικτύου
- κλάσεις μηνυμάτων υπηρεσιών διαδικτύου
- plugin προσαρμογής στο πλαίσιο χρήσης
- αρχεία διαμόρφωσης του χειριστή και της εφαρμογής μοντέλου-όψης-ελεγκτή

Εισάγοντας σε αυτούς τους "οδηγούς" αντίστοιχες μεθόδους που ανακτούν πληροφορία από το μοντέλο της εφαρμογής, το εργαλείο μετασχηματισμού μπορεί να προσθέσει στον κώδικα και τα αρχεία διαμόρφωσης πληροφορία που υπάρχει διαθέσιμη στις κλάσεις του μοντέλου, τις διεπαφές, τις ιδιότητες και τις μεθόδους, τις εξαρτήσεις και συσχετίσεις, τα στερεότυπα και τις τιμές ετικετών. Για παράδειγμα η τιμή ετικέτας `@presentation .web.view.table.columns` χρησιμοποιείται για να παρέχει στην όψη της εφαρμογής τα ονόματα και τις πληροφορίες που θα χρησιμοποιηθούν στον πίνακα που θα συμπεριληφθεί στην όψη.

Για την περίπτωση χρήσης που μελετάμε η μετατροπή του μοντέλου σε κώδικα αποτελείται από τα παρακάτω στοιχεία. Ο κώδικας του χειριστή πλαισίου χρήσης είναι ανεξάρτητος από την εφαρμογή διαδικτύου και δεν περιλαμβάνεται στη διαδικασία μετατροπής.

- ***Παραγωγή κλάσεων μηνυμάτων (Beans)***

  Κλάσεις σε Java με μεθόδους ανάκτησης και τροποποίησης των ιδιοτήτων των αντικειμένων παράγονται για κάθε μήνυμα (κλάσεις με στερεότυπα `<<InMessage>>` και `<<OutMessage>>`) που συναντάται κατά την ανάλυση του μοντέλου. Οι αντίστοιχες κλάσεις παράγονται επίσης για τους πολύπλοκους τύπους

δεδομένων που περιλαμβάνονται στις παραμέτρους των μεθόδων των υπηρεσιών διαδικτύου και τις απαντήσεις (<<ComplexType>>). Αυτή η πληροφορία για τα μηνύματα υπάρχει στην WSDL περιγραφή των υπηρεσιών και κατ' επέκταση και στα διαγράμματα κλάσεων των υπηρεσιών που υπάρχουν στο περιβάλλον σχεδίασης. Το αντίστοιχο Velocity template που περιλαμβάνει την απαραίτητη πληροφορία για την παραγωγή των κλάσεων (class.vm) παρουσιάζεται στο Παράρτημα Α' της διατριβής ως παράδειγμα της διαδικασίας. Στο template γίνεται κλήση διαφόρων μεθόδων που παρέχονται από τα εργαλεία που έχουν υλοποιηθεί, καθώς και από τις βιβλιοθήκες EMF και UML2.

- ***Πελάτες υπηρεσιών διαδικτύου***

  Για κάθε κλάση <<BusinessService>> που υπάρχει στο μοντέλο παράγεται ένας πελάτης για την κλήση των μεθόδων της υπηρεσίας διαδικτύου από την πλευρά της εφαρμογής. Οι πελάτες παράγονται για να χρησιμοποιηθούν κατά την εκτέλεση της εφαρμογής για τις κλήσεις των κατάλληλων μεθόδων, όπως ορίζεται μέσω των δραστηριοτήτων που περιλαμβάνονται στις καταστάσεις και τις μεταβάσεις του διαγράμματος καταστάσεων. Με αυτόν τον τρόπο οι μέθοδοι των υπηρεσιών είναι διαθέσιμες ως τοπικές μέθοδοι στην εφαρμογή.

- ***Κώδικας Plugin προσαρμογής***

  Τα κατάλληλα plugin παράγονται για κάθε κλάση με στερεότυπο <<BusinessService>> βάσει των έμμεσων εξαρτήσεων που υπάρχουν μεταξύ των στοιχείων της υπηρεσίας (μηνύματα αιτήσεων και απαντήσεων, μέθοδοι, κτλ.) και των αντίστοιχων πηγών πλαισίου χρήσης. Φυσικά η διαδικασία αυτή ακολουθείται μόνο για τις υπηρεσίες που χρειάζονται κάποια προσαρμογή. Τα plugin παράγονται ως παιδιά της κλάσης AbstractQueryPlugin, που είναι η βασική κλάση που παρέχεται από το μηχανισμό της αρχιτεκτονικής προσαρμογής. Προκειμένου να γίνει η παραγωγή του κατάλληλου κώδικα αναλύονται όλα τα τερματικά σημεία που σχετίζονται με την υπηρεσία, όπως έχουν μοντελοποιηθεί στο περιβάλλον σχεδίασης μέσω διεπαφών. Εντοπίζονται οι εξαρτήσεις πλαισίου χρήσης για τα στοιχεία των υπηρεσιών που υλοποιούν τα τερματικά σημεία και έπειτα κατασκευάζονται τα κατάλληλα plugin για κάθε περίπτωση προσαρμογής ως ακολούθως:

  - Αρχεία κλάσεων Java με όνοματα που ακολουθούν τη μορφή *<ONOMA_PORT_TYPE><ONOMA_ΜΕΘΟΔΟΥ>InPlugin.java* παράγονται για κάθε περίπτωση αντικατάστασης παραμέτρου, όταν εντοπιστεί η αντίστοιχη εξάρτηση

`<<ContextBinding>>` μεταξύ μιας παραμέτρου μηνύματος και ιδιότητας πλαισίου χρήσης κατά την ανάλυση του μοντέλου. Για κάθε παράμετρο του μηνύματος αίτησης που χρειάζεται προσαρμογή ανακτάται η αντίστοιχη τιμή πλαισίου χρήσης καλώντας την πηγή πλαισίου χρήσης και εν συνεχεία η νέα τιμή αντικαθιστάται στο μήνυμα.

- Αρχεία κλάσεων Java με όνοματα που ακολουθούν τη μορφή *<ONOMA_PORT_TYPE>InPlugin.java* παράγονται για κάθε περίπτωση επιλογής μεθόδου που ανιχνεύεται στο μοντέλο μέσω της εξάρτησης που υπάρχει μεταξύ της διεπαφής που εκφράζει το τερματικό σημείο και κάποιου στοιχείου πλαισίου χρήσης. Η προς κλήση μέθοδος που περιλαμβάνεται στο SOAP μήνυμα αίτησης αντικαθιστάται με τη νέα μέθοδο που χρειάζεται να κληθεί βάσει της εξάρτησης πλαισίου χρήσης. Στην περίπτωση αλλαγής της μεθόδου, ενημερώνονται κατάλληλα και οι παράμετροι του SOAP μηνύματος όπως ορίζονται από το μοντέλο της υπηρεσίας και γίνεται ανάθεση αρχικών τιμών για τις παραμέτρους.
- Αρχεία κλάσεων Java με όνοματα που ακολουθούν τη μορφή *<ONOMA_PORT_TYPE><ONOMA_ΜΕΘΟΔΟΥ>OutPlugin.java* παράγονται για κάθε περίπτωση τροποποίησης μηνύματος απάντησης που εκφράζεται μέσω της σχέσης `<<ContextTriggering>>` στα στοιχεία του μοντέλου. Τα στοιχεία του SOAP μηνύματος απάντησης τροποποιούνται μέσω της κλήσης της μεθόδου τροποποίησης `<<Trigger>>`. Στην περίπτωση απάντησης σε μορφή λίστας ή πίνακα η τροποποίηση σχετίζεται συνήθως με ενέργειες ταξινόμησης ή φιλτραρίσματος βάσει συγκεκριμένων περιορισμών σε OCL εκφράσεις.

- ***Αρχεία διαμόρφωσης***

  Τα αρχεία διαμόρφωσης αφορούν τα ειδικά χαρακτηριστικά της υλοποίησης της εφαρμογής διαδικτύου και στην περίπτωση που μελετάμε σχετίζονται με τα χαρακτηριστικά των Spring MVC και Spring Web Flow. Κατά την ανάλυση του διαγράμματος καταστάσεων παράγεται το αρχείο *flow.xml* που αναπαριστά τις καταστάσεις και μεταβάσεις της εφαρμογής και το αρχείο *flow-beans.xml* που περιγράφει τα χαρακτηριστικά των αντικειμένων που χρειάζονται στην εφαρμογή (όπως ορίζει το Spring Web Flow).

  Ακόμα για κάθε plugin που παράγεται εισάγονται τα κατάλληλα στοιχεία στο αρχείο *handler.xml* που περιλαμβάνει τους ορισμούς των plugin και τις συσχετίσεις με τις υπηρεσίες. Αυτή η διαδικασία πραγματοποιείται μέσω της κλάσης `Handler ConfigGenerator`. Τέλος, παράγονται διάφορα άλλα αρχεία διαμόρφωσης (π.χ.

web.xml για την εφαρμογή διαδικτύου, *application-webflow-config.xml* και *application-servlet-config.xml* για τη διαμόρφωση του Spring) χρησιμοποιώντας αντίστοιχους Velocity "οδηγούς" ή εισάγοντας το σταθερό περιεχόμενο που ορίζει το Spring. Πολλές φορές η προσπέλαση και κατασκευή των αρχείων διαμόρφωσης διευκολύνεται από βιβλιοθήκες που έχουν δημιουργηθεί με χρήση του εργαλείου XML Beans.

- ***Αρχεία όψεων***

  Οι όψεις της εφαρμογής υλοποιούνται μέσω JSP σελίδων. Ένα χωριστό JSP αρχείο παράγεται για κάθε κατάσταση του διαγράμματος καταστάσεων με στερεότυπο `<<FrontEndView>>`. Το αρχείο της όψης περιέχει αναφορές σε αντικείμενα που εκφράζουν δεδομένα του επιπέδου εφαρμογής που προωθούνται από τον ελεγκτή στην όψη, καθώς και γεγονότα που σχετίζονται με τις μεταβάσεις που έχουν οριστεί στην εφαρμογή (π.χ. ως κουμπία ή υπερσύνδεσμοι). Αυτή η πληροφορία ανακτάται κυρίως μέσω των τιμών ετικετών που υπάρχουν στα στοιχεία των καταστάσεων και των μεταβάσεων του διαγράμματος καταστάσεων. Αξίζει ωστόσο να σημειωθεί πως η αναλυτική κατασκευή των όψεων της εφαρμογής δεν ανήκει στους βασικούς στόχους της προτεινόμενης λύσης. Οι όψεις που παράγονται περιλαμβάνουν τα ελάχιστα απαραίτητα στοιχεία για τη σωστή λειτουργία της εφαρμογής. Εν συνεχεία οι όψεις μπορούν να εμπλουτιστούν από κάποιο σχεδιαστή σελίδων διαδικτύου.

Η παραπάνω προσέγγιση δίνει τη δυνατότητα μετατροπής του μοντέλου σε κώδικα χρησιμοποιώντας τις βιβλιοθήκες που παρέχει η μεθοδολογία (*handler.jar* για το χειριστή προσαρμογής στο πλαίσιο χρήσης, *contextws-mde.jar* για τις βασικές κλάσεις ανάλυσης του μοντέλου, κτλ.) και τους διάφορους Velocity "οδηγούς", ενώ απαιτείται ο καθορισμός των παραμέτρων διαμόρφωσης του εργαλείου μετατροπής *model.properties* (Εικόνα 33).

```
#PROFILES TO BE IMPORTED
PROFILE_DATATYPES = /data/datatypes.profile.uml2
PROFILE_CONTEXTUML =
<ΜΟΝΟΠΑΤΙ_ΑΡΧΕΙΟΥ_ΜΟΝΤΕΛΟΥ>/</gr.ntua.icbnet.contextmda.profile.ExtContextUML.
profile.uml2
PROFILE_PRESENTATION =
<ΜΟΝΟΠΑΤΙ_ΑΡΧΕΙΟΥ_ΜΟΝΤΕΛΟΥ>/</gr.ntua.icbnet.contextmda.profile.Presentation.prof
ile.uml2

#APPLICATION MODEL PATHS
SERVICE_AND_CTX_MODELS =
<ΜΟΝΟΠΑΤΙ_ΑΡΧΕΙΟΥ_ΜΟΝΤΕΛΟΥ>/<ΟΝΟΜΑ_ΜΟΝΤΕΛΟΥ>.uml2
XMI_MODEL = <ΜΟΝΟΠΑΤΙ_ΑΡΧΕΙΟΥ_ΜΟΝΤΕΛΟΥ>/<ΟΝΟΜΑ_ΜΟΝΤΕΛΟΥ>.xml

#MAIN LIBRARIES
umlPluginPath =
```

<MONOΠΑΤΙ_ΒΙΒΛΙΟΘΗΚΩΝ>/</org.eclipse.uml2.uml_2.1.1.v200707311200.jar
umlResourcePath =
<MONOΠΑΤΙ_ΒΙΒΛΙΟΘΗΚΩΝ/org.eclipse.uml2.uml.resources_2.1.0.v200706251652.jar

Εικόνα 33. Διαμόρφωση εργαλείου μετατροπής μοντέλου

6.4 Βιβλιογραφία

[1] Acceleo, http://www.acceleo.org.

[2] AndroMDA BPM4Struts Cartridge Profile, http://galaxy.andromda.org/docs/andromda-bpm4struts-cartridge/profile.html.

[3] Apache Velocity, http://velocity.apache.org/.

[4] ArgoUML, http://argouml.tigris.org/.

[5] Eclipse Model Development Tools (MDT), http://www.eclipse.org/modeling/mdt/?project=uml2.

[6] Eclipse UML2 Tools, http://www.vogella.de/articles/UML/article.html.

[7] Fengler, O., Fengler, W., Duridanova, V., "Extending the Modeling Efficiency of the UML Activity Diagram for the Design of Distributed Systems", Proc. 2nd International Workshop on Innovative Internet Computing Systems, Springer-Verlag, vol. 2346, 2002, pp.51-62.

[8] Java Server Faces (JSF), http://java.sun.com/javaee/javaserverfaces/.

[9] Kapitsaki, G. M., Kateros, D. A., Pappas, C., Tselikas, N. D., Venieris, I. S., "Model-Driven Development of Composite Web Applications", Proc. 10th International Conference on Information Integration and Web-based Applications and Services (iiWAS'08), 2008, pp. 399-402.

[10] Kateros, D. A., Kapitsaki, G. M., Tselikas, N. D., Venieris, I. S., "A Methodology for Model-Driven Web Application Composition", Proc. 2008 IEEE International Conference on Services Computing (SCC'08), 2008, vol. 2, pp. 489-492.

[11] MagicDraw, http://www.magicdraw.com/.

[12] Object Management Group (OMG), Object Constraint Language OMG Available Specification, v.2.0, 2006, http://www.omg.org/docs/formal/06-05-01.pdf.

[13] Skogan, D., Gronmo, R., Solheim, I., Oldevik, J., “Model-driven web services development”, Proc. IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-04), 2004, pp. 42-45.

[14] Tai, H., Mitsui, K., Nerome, T., Abe, M., Ono, K., Hori, M., “Model-driven development of large-scale Web applications”, IBM Journal of research and Development, vol. 48(5/6), 2004, pp. 797-809.

7ο Κεφάλαιο

Παραδείγματα Χρήσης και Αξιολόγηση Μεθοδολογίας Ανάπτυξης

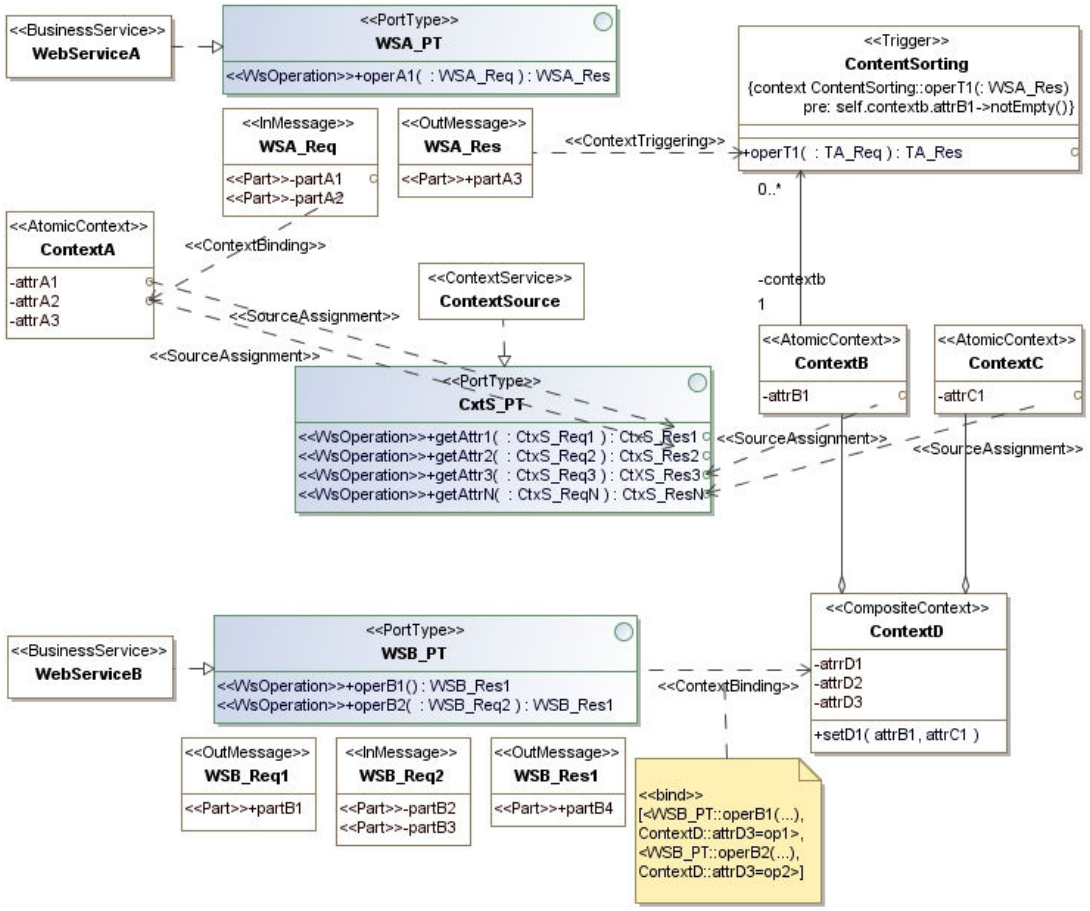
Προκειμένου να γίνει μεν κατανοητή η χρήση της μεθοδολογίας και να αναδειχθεί δε η χρησιμότητα της δίνονται στο παρόν Κεφάλαιο κάποια παραδείγματα χρήσης για την αξιολόγησή της. Παρουσιάζεται συνοπτικά μια απλή γενική εφαρμογή, ενώ αναλυτικά περιγράφεται η ανάπτυξη ενός πιο πολύπλοκου σεναρίου για μια ρεαλιστική εφαρμογή. Ακολούθως παρουσιάζονται τα βασικά πλεονεκτήματα της λύσης μέσω διαφόρων μετρικών αξιολόγησης.

7.1 Απλή Εφαρμογή

Σε αυτό το σημείο περιγράφεται εν συντομία η ανάπτυξη μιας εφαρμογής που αποτελείται από αφαιρετικές υπηρεσίες διαδικτύου που δεν παρέχουν συγκεκριμένη λειτουργικότητα (π.χ. υπηρεσία διαδικτύου Α με μεθόδους Α εώς Ε).

Συγκεκριμένα θεωρείται η ανάπτυξη μιας εφαρμογής που αποτελείται από τρεις υπηρεσίες διαδικτύου: δύο υπηρεσίες εφαρμογών BWS (`WebServiceA` και `WebServiceB`) και μια υπηρεσία πλαισίου χρήσης CWS (`ContextSource`). Το μοντέλο των υπηρεσιών μαζί με το αντίστοιχο του πλαισίου χρήσης και οι μεταξύ τους εξαρτήσεις παρουσιάζονται σε ένα κοινό διάγραμμα στην Εικόνα 34. Στο διάγραμμα είναι ορατές τρεις περιπτώσεις προσαρμογής, μία ανά περίπτωση: αντικατάσταση παραμέτρου και τροποποίηση απάντησης υπάρχει στα στοιχεία της υπηρεσίας `WebServiceA`, ενώ επιλογή μεθόδου στα στοιχεία της υπηρεσίας `WebServiceB`. Το διάγραμμα καταστάσεων για τη ροή της εφαρμογής αποτελείται από τρεις όψεις, από τις οποίες οι δύο σχετίζονται με

κλήσεις μεθόδων των δύο υπηρεσιών εφαρμογών. Το διάγραμμα έχει παρουσιαστεί στην Εικόνα 29 του προηγούμενου Κεφαλαίου.

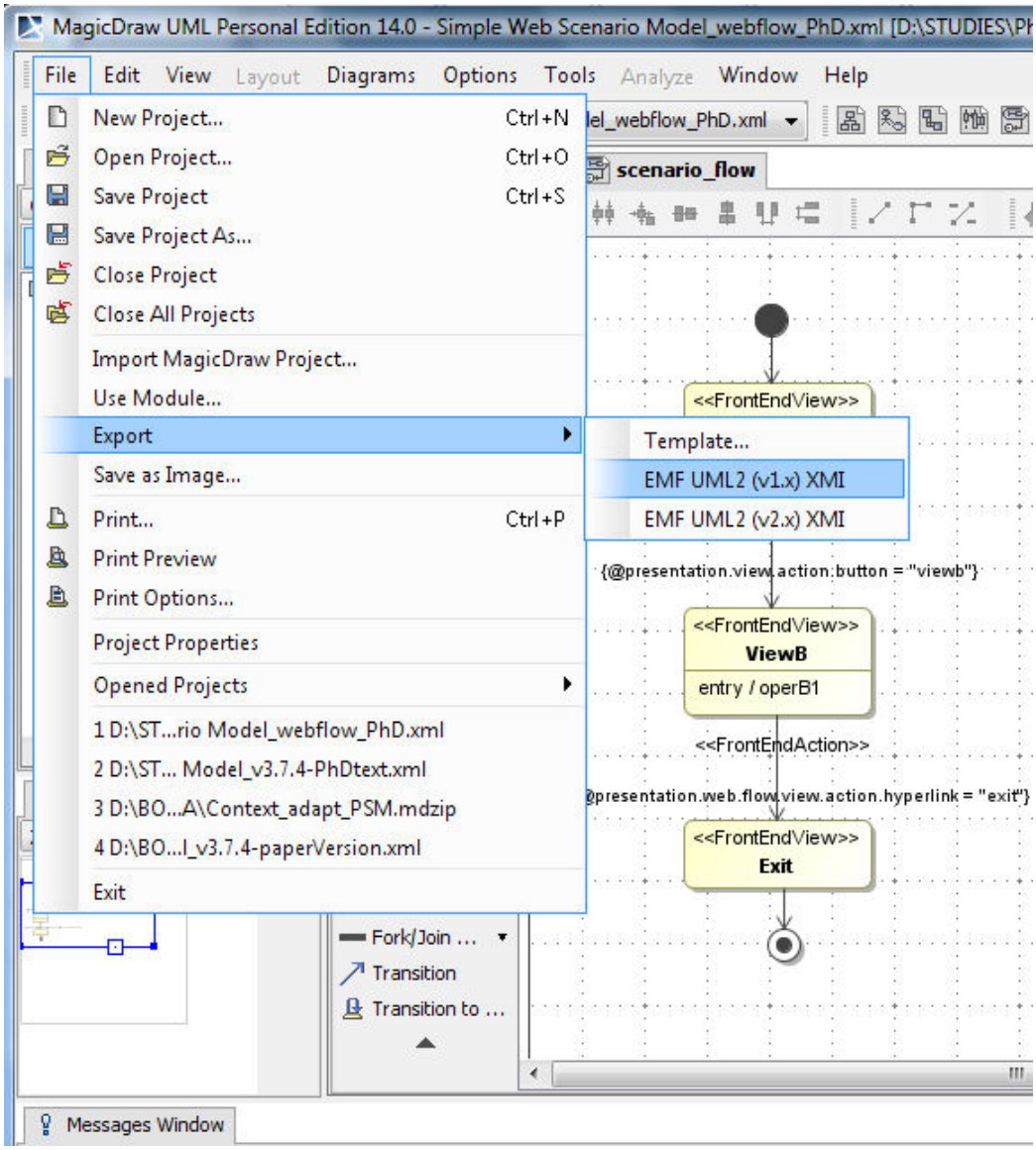


Εικόνα 34. Διάγραμμα υπηρεσιών αφαιρετικής εφαρμογής και συσχετίσεων με το πλαίσιο χρήσης.

Στη συσχέτιση `<<ContextBinding>>` μεταξύ του τερματικού σημείου `WSB_PT` και της πληροφορίας πλαισίου χρήσης `ContextD` χρησιμοποιείται για λόγους απλούστευσης το στερεότυπο `<<bind>>`. Το `<<bind>>` έχει προταθεί στο [9] και εισάγεται ως σχόλιο στη συσχέτιση για να είναι ορατό κατά τη διαδικασία ανάλυσης του μοντέλου. Το ίδιο στερεότυπο μπορεί να χρησιμοποιηθεί και στις υπόλοιπες συσχετίσεις `<<ContextBinding>>` που εκφράζουν πολλαπλές περιπτώσεις προσαρμογής στο πλαίσιο χρήσης, αλλά και στην περίπτωση πολλαπλών `<<SourceAssignment>>` εξαρτήσεων με πηγές για να αποφευχθεί η χρήση διαφορετικών γραμμών εξαρτήσεων με την αντίστοιχη ιδιότητα ή μέθοδο για κάθε ιδιότητα της πληροφορίας πλαισίου χρήσης. Το περιεχόμενο του σχολίου αναλύεται κατά τη διαδικασία μετατροπής για την παραγωγή των κατάλληλων πελατών υπηρεσιών διαδικτύου (στην περίπτωση της συσχέτισης

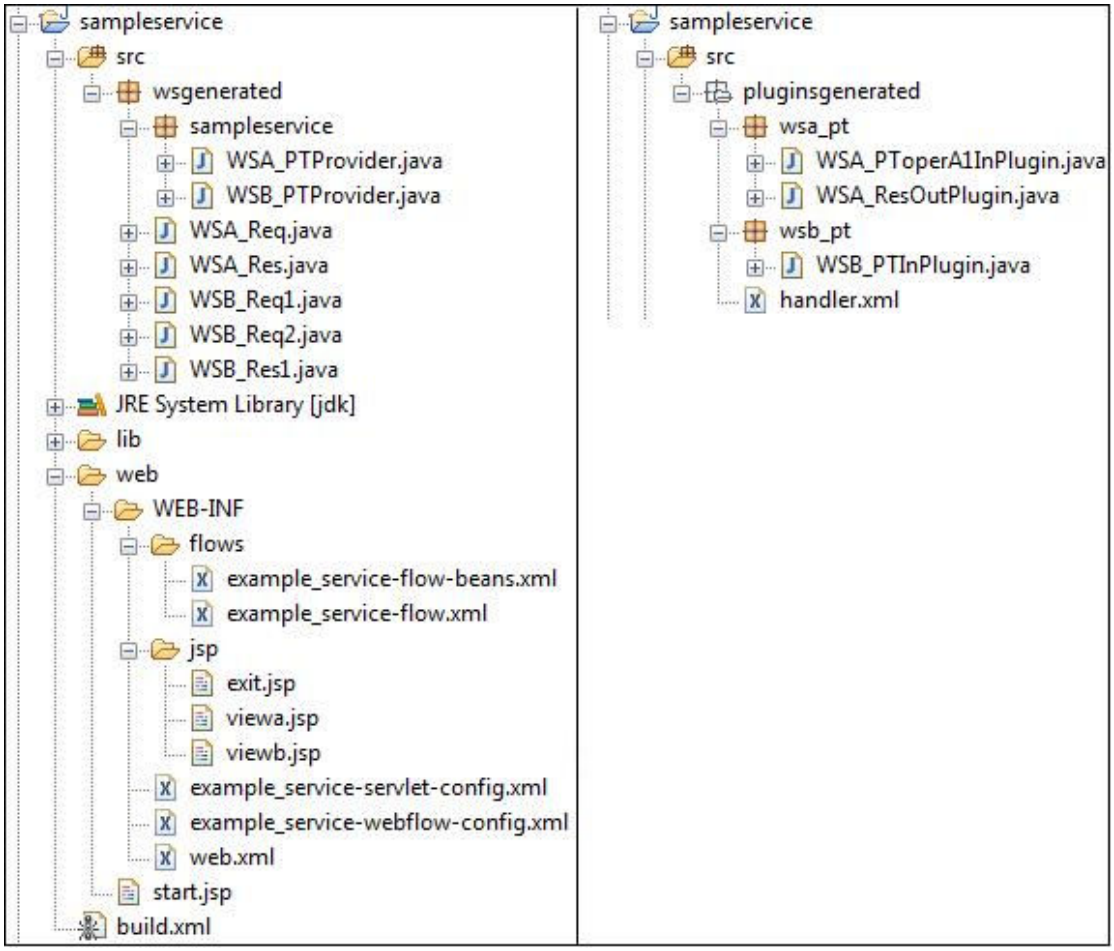
<<SourceAssignment>>) ή των κατάλληλων plugin προσαρμογής (στις περιπτώσεις <<ContextBinding>>).

Για να εκκινηθεί η διαδικασία μετατροπής το μοντέλο εξάγεται από το περιβάλλον μοντελοποίησης στην EMF αναπαράστασή του (Εικόνα 35).



Εικόνα 35. Εξαγωγή του μοντέλου της εφαρμογής σε EMF αναπαράσταση.

Με τη χρήση του εργαλείου μετατροπής του μοντέλου σε κώδικα, παράγονται συνολικά 21 αρχεία για τις κλάσεις των μηνυμάτων, τους πελάτες των υπηρεσιών διαδικτύου, τα plugin, τα αρχεία των όψεων και τα αρχεία διαμόρφωσης (Εικόνα 36). Στην Εικόνα 37 φαίνεται αναλυτικά η αντιστοίχιση του διαγράμματος καταστάσεων της εφαρμογής στο αρχείο διαμόρφωσης *sampleservice-flow.xml*.

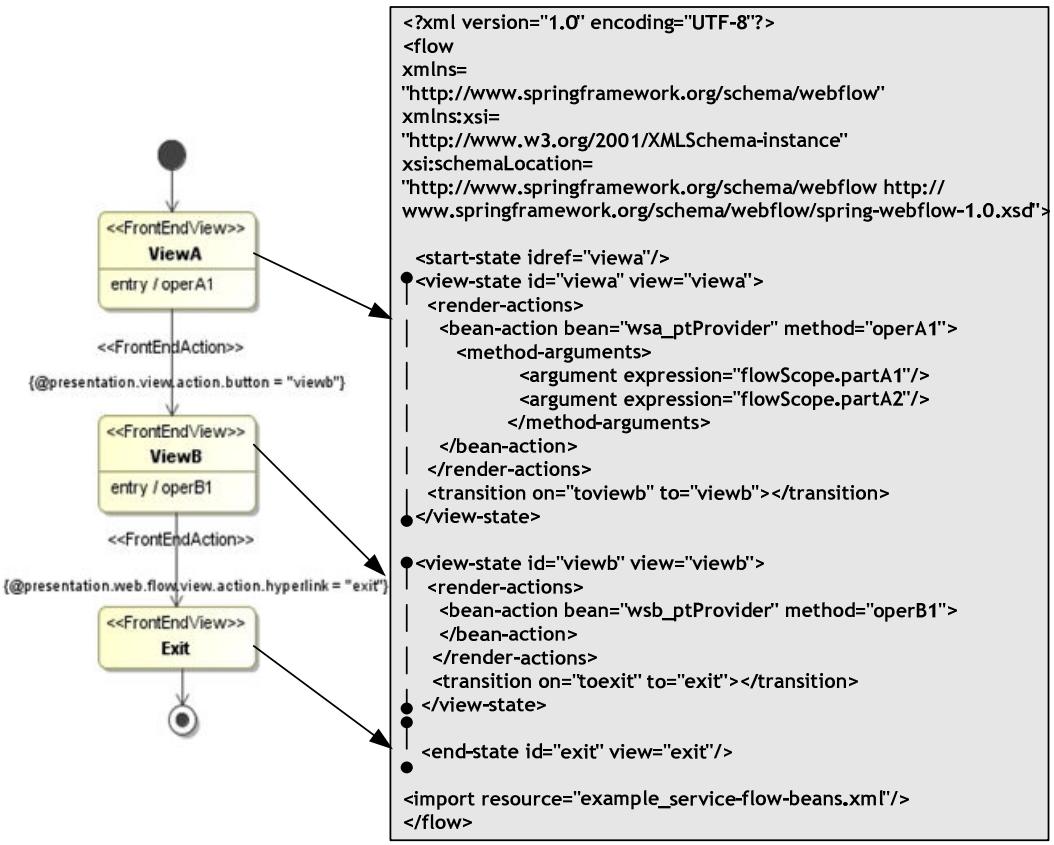


Εικόνα 36. Κώδικας που έχει παραχθεί για την απλή εφαρμογή διαδικτύου και τα plugin προσαρμογής.

7.2 Σενάριο Χρήσης

7.2.1 Περιγραφή Σεναρίου

Θεωρείται ότι έχει ανατεθεί σε ένα μηχανικό λογισμικού η ανάπτυξη μιας εφαρμογής που θα προσφέρεται σε όλους τους επισκέπτες ενός ομίλου πολυχώρων με κινηματογραφικές αίθουσες. Σκοπός είναι η παροχή μιας εφαρμογής διαδικτύου με επίγνωση του πλαισίου χρήσης που να περιλαμβάνει παρουσίαση των ταινιών που προβάλλονται και κράτηση θέσεων για την παρακολούθηση κάποιας ταινίας. Ο μηχανικός λογισμικού έχει στη διάθεσή του υπηρεσίες διαδικτύου που παρέχονται στο ενδοδίκτυο (intranet) του πολυχώρου (ή εφαρμογές που μπορούν να μετατραπούν σε υπηρεσίες διαδικτύου).



Εικόνα 37. Αντιστοίχιση διαγράμματος καταστάσεων στο αρχείο διαμόρφωσης flow.xml.

Στην εφαρμογή ενσωματώνονται τρεις υπηρεσίες διαδικτύου:

- Μια υπηρεσία χαιρετισμού που χρησιμοποιείται για την εμφάνιση ενός μηνύματος στο χρήστη (`GreetingService`)
- Μια υπηρεσία παρουσίασης των διαθέσιμων ταινιών που προβάλλονται και πληροφορίες για κάθε ταινία (`MovieListService`)
- Μια υπηρεσία κρατήσεων που επιτρέπει στο χρήστη να πραγματοποιήσει πληρωμές για εισιτήρια των ταινιών (`BookingService`)

Η λειτουργία των παραπάνω υπηρεσιών προσαρμόζεται στο πλάισιο χρήσης στα πλαίσια δημιουργίας μιας σύνθετης εφαρμογής για τους επισκέπτες. Συγκεκριμένα, η υπηρεσία χαιρετισμού λαμβάνει υπόψη το όνομα του χρήστη, τη μητρική του γλώσσα και την τρέχουσα θέση του. Οι δύο πρώτες τιμές πληροφορίας ανακτώνται από το προφίλ που είναι αποθηκευμένο για το συγκεκριμένο χρήστη στο σύστημα βάσης δεδομένων του πολυχώρου, ενώ η τρίτη τιμή παρέχεται μέσω μιας υπηρεσίας καθορισμού θέσης (π.χ. σύστημα GPS αν η κινητή συσκευή του χρήστη διαθέτει πομπό ή κάποια υπηρεσία υπολογισμού της θέσης του χρήστη μέσω της εξακρίβωσης της IP διεύθυνσης του υπολογιστικού συστήματος που χρησιμοποιεί). Η υπηρεσία παρουσίασης των διαθέσιμων

ταινιών προσαρμόζεται λαμβάνοντας υπόψη αρχικά την ηλικία του αιτούντος (ώστε να εμφανιστούν μόνο οι ταινίες που επιτρέπεται να παρακολουθήσει) και εν συνεχεία μέσω της τροποποίησης του μηνύματος απάντησης, ώστε να ταξινομηθούν οι ταινίες βάσει των προτιμήσεων σε είδη ταινιών που καθορίζονται από το χρήστη στο προφίλ του. Τέλος, η υπηρεσία πληρωμής προσαρμόζεται βάσει της επιλογής της κατάλληλης για το χρήστη μεθόδου, καθώς υλοποιεί δύο μηχανισμούς: μια μέθοδο για πληρωμή με μετρητά (μέθοδος `payWithCash`) και μια δεύτερη για πληρωμή μέσω πιστωτικής κάρτας (μέθοδος `payWithCreditCard`). Επιπλέον, η τελευταία υπηρεσία λαμβάνει υπόψη στη λειτουργία της τη μητρική γλώσσα του χρήστη, όπως και στην περίπτωση της υπηρεσίας χαιρετισμού.

Οι τρεις υπηρεσίες απαιτούν διαφορετικές περιπτώσεις προσαρμογής στο πλαίσιο χρήσης, καθώς και συνδυασμούς τους: η υπηρεσία `GreetingService` σχετίζεται με την αντικατάσταση παραμέτρου (όνομα χρήστη, γλώσσα χρήστη και τρέχουσα θέση), στη `MovieListService` υπάρχει προσαρμογή βάσει αντικατάστασης παραμέτρου (ηλικία χρήστη) και τροποποίησης απάντησης (ταξινόμηση λίστας ταινιών). Τέλος, στη `BookingService` συναντάμε και πάλι αντικατάσταση παραμέτρου (γλώσσα χρήστη) και επιλογή μεθόδου (βάσει της προτίμησης πληρωμής του χρήστη).

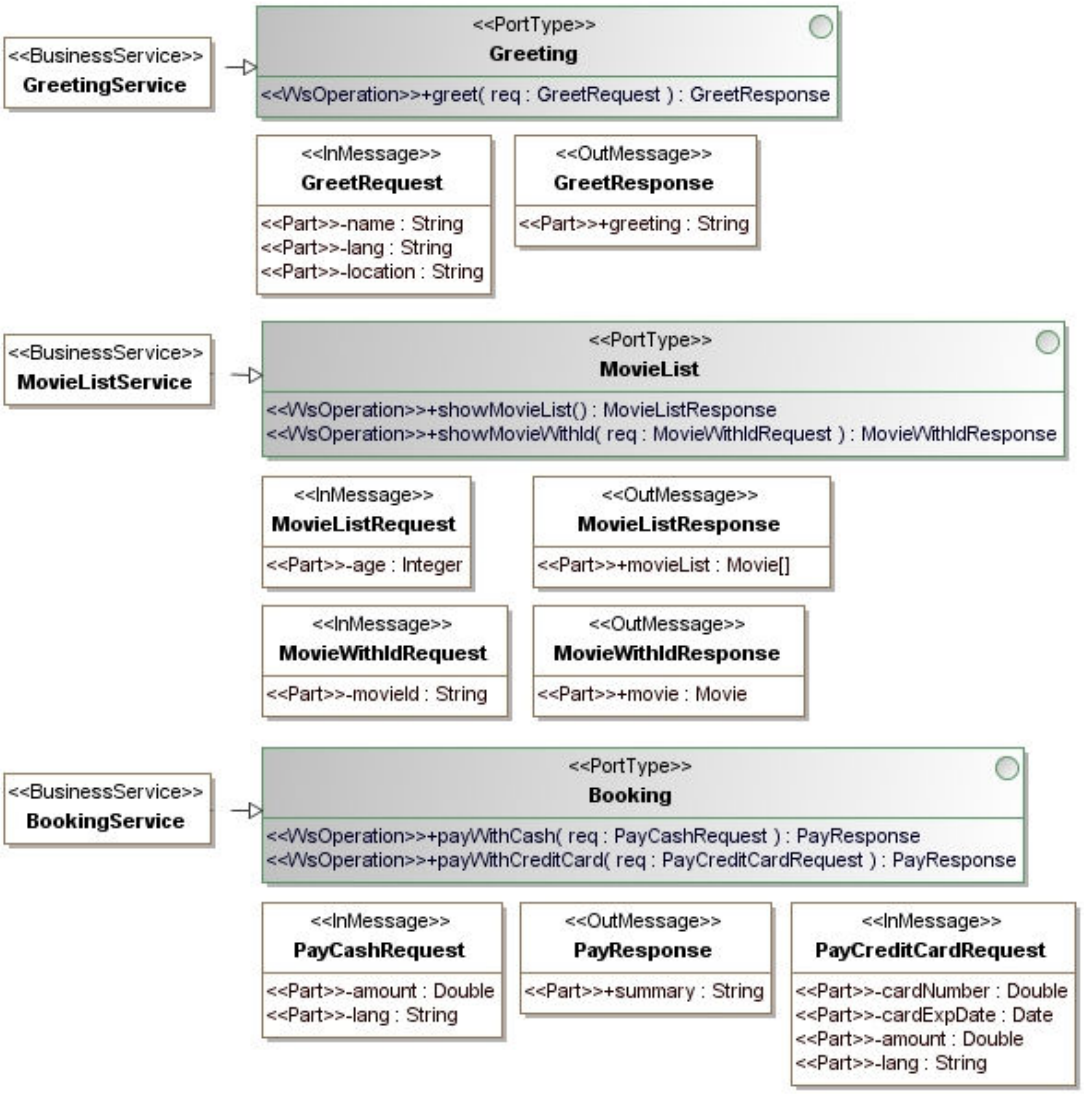
Στις επόμενες υποενότητες παρουσιάζεται η ανάπτυξη της υπηρεσίας χρησιμοποιώντας τη μοντελοκεντρική μέθοδο ανάπτυξης, καθώς και η εκτέλεση της προκύπτουσας εφαρμογής.

7.2.2 Ανάπτυξη Εφαρμογής

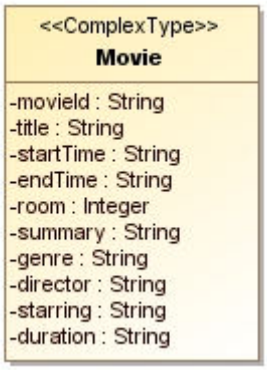
7.2.2.1 Μοντελοποίηση σεναρίου

Στη φάση μοντελοποίησης η εφαρμογή σχεδιάστηκε στο περιβάλλον MagicDraw. Τα UML μοντέλα των τριών υπηρεσιών διαδικτύου που χρησιμοποιούνται στο σενάριο (`GreetingService`, `MovieListService` και `BookingService`) με τις μεθόδους που υλοποιούν και τις αντίστοιχες παραμέτρους παρουσιάζονται στο διάγραμμα κλάσεων της Εικόνας 38, όπως μπορούν να εξαχθούν και από τις WSDL περιγραφές των υπηρεσιών. Στο περιβάλλον σχεδίασης χρειάζεται να εισαχθούν και οι πολύπλοκοι τύποι δεδομένων που ενδέχεται να περιλαμβάνονται στις περιγραφές των υπηρεσιών. Ο πολύπλοκος τύπος `Movie` που περιλαμβάνεται στις απαντήσεις των μεθόδων `showMovieList` και `showMovieWithId` εισάγεται στο μοντέλο με το στερεότυπο `<<ComplexType>>`, όπως φαίνεται στην Εικόνα 39. Και αυτή η πληροφορία μπορεί να εξαχθεί από τη WSDL

περιγραφή και είναι απαραίτητη για τη σωστή αντιστοίχιση των στοιχείων του τύπου στις όψεις της εφαρμογής διαδικτύου στη συνέχεια της διαδικασίας.



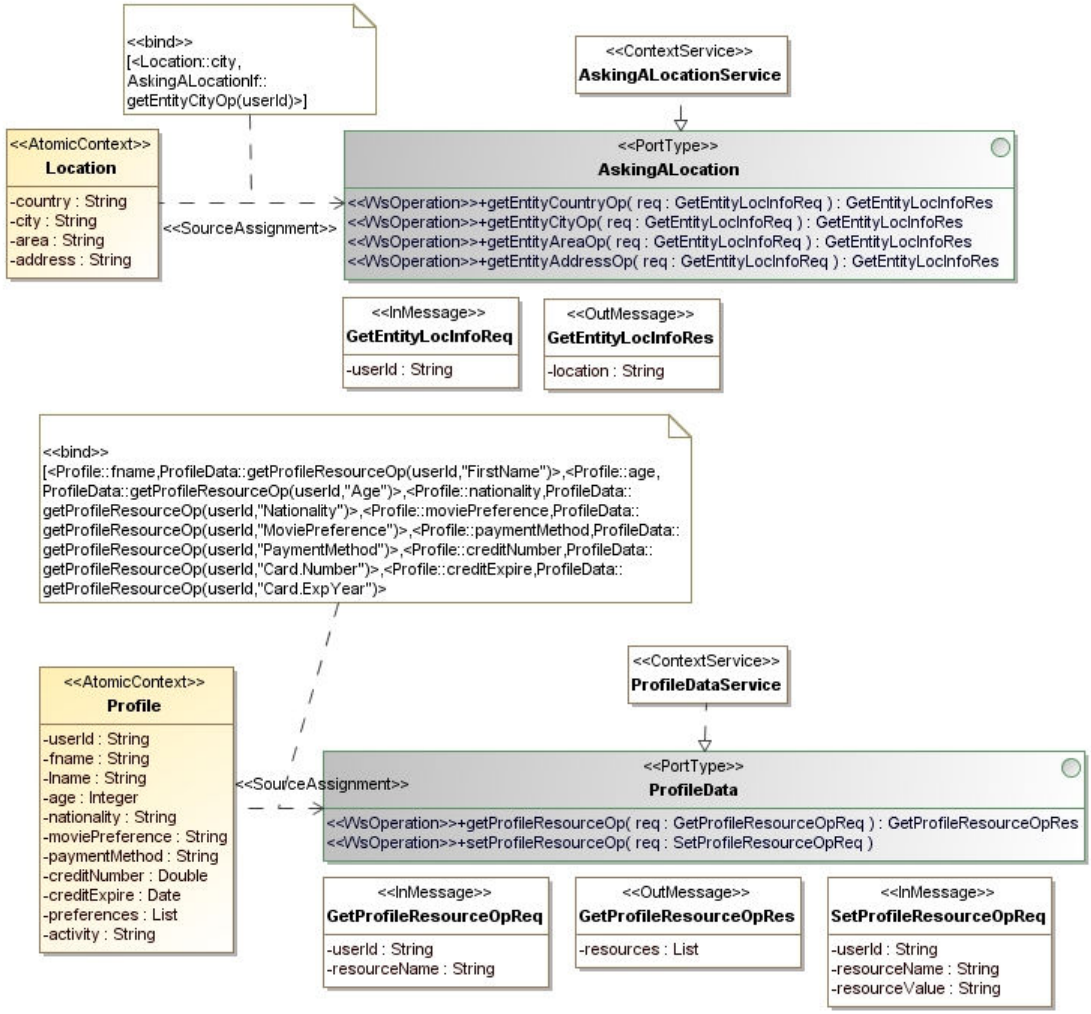
Εικόνα 38. Διάγραμμα κλάσεων των υπηρεσιών εφαρμογών (σενάριο πολυχώρου).



Εικόνα 39. Πολύπλοκος τύπος δεδομένων Movie.

Η ίδια διαδικασία εισαγωγης ακολουθείται για το μοντέλο των υπηρεσιών πλαισίου χρήσης CWS (στερεότυπο `<<ContextService>>`) και εν συνεχεία εισάγεται το μοντέλο πλαισίου χρήσης, το οποίο συσχετίζεται με τις πηγές μέσω των σχέσεων `<<SourceAssignment>>`. Το μοντέλο πλαισίου χρήσης που αποτελείται από τις εξής πληροφορίες εμφανίζεται στην Εικόνα 40 μαζί με τις συσχετίσεις με τις πηγές:

- **Πληροφορία θέσης (`Location`):** χρησιμοποιείται για να εκφράσει πληροφορία για μια τοποθεσία και περιλαμβάνει στοιχεία για τη χώρα, την πόλη, την περιοχή και τη διεύθυνση της τοποθεσίας.
- **Προφίλ χρήστη (`Profile`):** περιλαμβάνει πληροφορίες για το χρήστη και τις προτιμήσεις του (π.χ. χώρα προέλευσης, ηλικία, κτλ.). Το προφίλ που χρησιμοποιήθηκε βασίστηκε στο Generic User Profile (GUP) [2] της 3GPP (3rd Generation Partnership Project) σε XML μορφοποίηση.

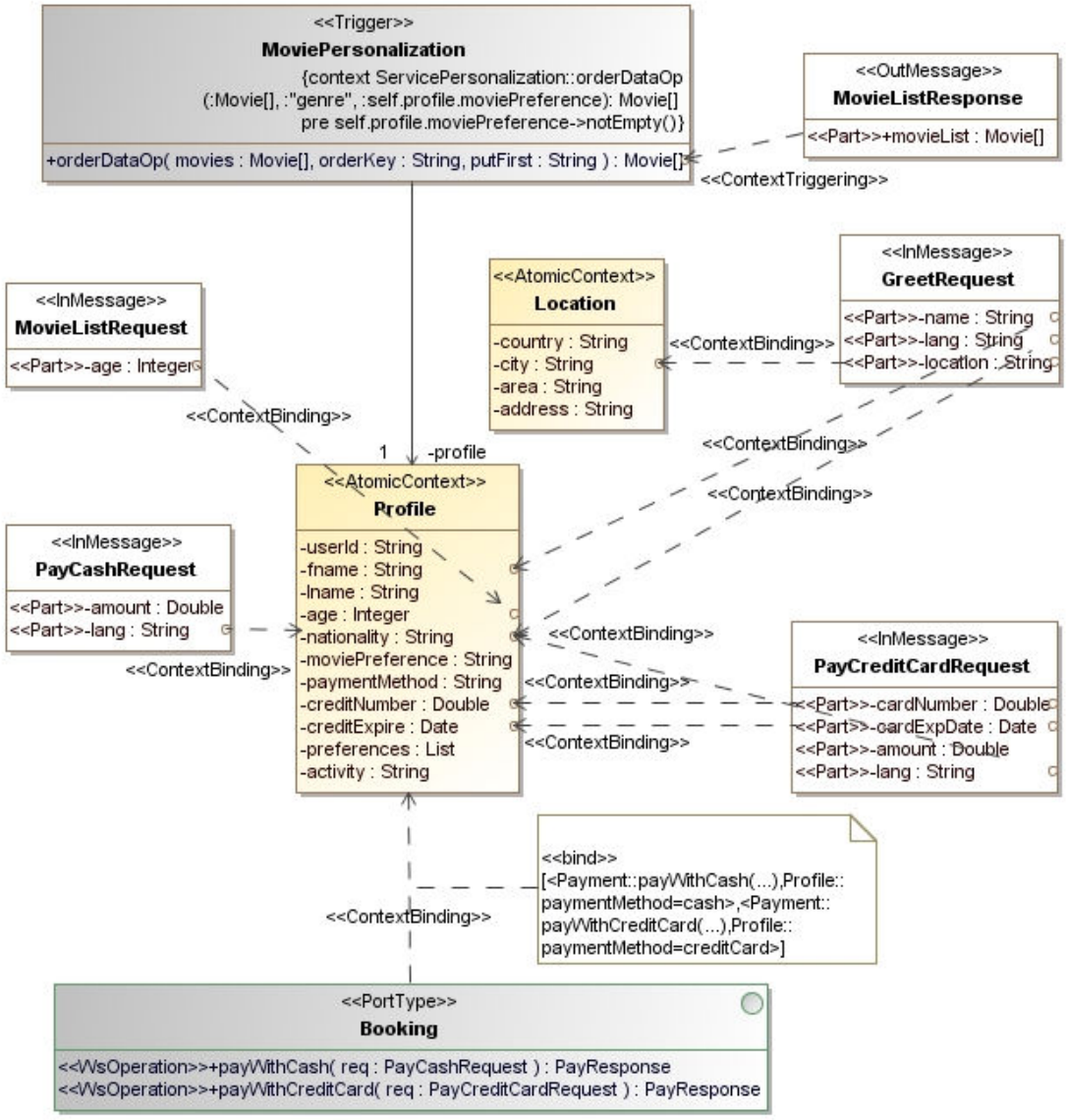


Εικόνα 40. Διάγραμμα κλάσεων πλαισίου χρήσης και συσχετίσεις με υπηρεσίες πλαισίου χρήσης (σενάριο πολυχώρου).

Οι ετικέτες `@businessservice.endpoint` που αναπαριστούν το URI, όπου είναι διαθέσιμη η υπηρεσία διαδικτύου, δεν εμφανίζονται στα παραπάνω διαγράμματα για λόγους παρουσίασης. Ένα παράδειγμα μιας τέτοιας τιμής ετικέτας είναι το εξής:

`@businessservice.endpoint=`
`http://contextmda.icbnet.ntua.gr:8080/axis2/services/MovieListService`

Το αποτέλεσμα της μοντελοποίησης των εξαρτήσεων προσαρμογής στο πλαίσιο χρήσης μεταξύ των δύο βασικών μοντέλων παρουσιάζεται στην Εικόνα 41. Στο διάγραμμα φαίνονται περιπτώσεις προσαρμογής και των τριών κατηγοριών, καθώς και αντίστοιχων συνδυασμών:



Εικόνα 41. Εξαρτήσεις μεταξύ του μοντέλου υπηρεσιών εφαρμογών και του μοντέλου πλαισίου χρήσης (σενάριο πολυχώρου).

- Αντικατάσταση παραμέτρου φαίνεται στις συσχετίσεις `<<ContextBinding>>` που εξέρχονται από τα μηνύματα αιτήσεων που ανήκουν στις τρεις υπηρεσίες: `GreetRequest`, `MovieListRequest`, `PayCashRequest` και `PayCreditCardRequest`. Οι τιμές αντικαθίστανται κατά περίπτωση με τις τιμές της ιδιότητας city του πλαισίου χρήσης `Location` και των ιδιοτήτων fname, nationality, age, creditNumber και creditExpire του `Profile`.
- Επιλογή μεθόδου συναντάται στην εξάρτηση μεταξύ της υπηρεσίας κράτησης θέσεων `BookingService` και της πληροφορίας πλαισίου χρήσης `Profile`. Η μέθοδος που θα επιλεχθεί μέσω της προσαρμογής πρέπει να αντιστοιχεί στην τιμή της ιδιότητας paymentMethod που βρίσκεται στο προφίλ του χρήστη, όπως εκφράζεται μέσω του σχολίου `<<bind>>`: για τιμή προτίμησης πληρωμής cash καλείται η μέθοδος `payWithCash`, ενώ για τιμή creditCard η μέθοδος `payWithCreditCard`.
- Η περίπτωση της τροποποίησης απάντησης εμφανίζεται στην εξάρτηση `<<ContextTriggering>>` που εξέρχεται από το μήνυμα απάντησης `MovieListResponse`. Το άλλο άκρο της συσχέτισης οδηγεί στην κλάση ενεργοποίησης `MoviePersonalization` και συγκεκριμένα στη μέθοδο `orderDataOp` της κλάσης. Η κλάση αυτή (όπως και κάθε κλάση `<<Trigger>>`) θα πρέπει να σχεδιαστεί και να υλοποιηθεί από το μηχανικό λογισμικού, καθώς είναι το μόνο κομμάτι του μοντέλου που δε συμπεριλαμβάνεται στη διαδικασία παραγωγής κώδικα. Αυτό συμβαίνει, διότι δεν είναι εφικτό να εκφραστεί όλη η λογική που υλοποιεί η κλάση `<<Trigger>>` στην αφαιρετική UML αναπαράσταση. Η μέθοδος `orderDataOp` καλείται μόνο στην περίπτωση που η ιδιότητα `moviePreference` του προφίλ του χρήστη έχει μη-μηδενική τιμή, όπως εκφράζεται μέσω της συνθήκης precondition στην OCL έκφραση:

  ```
  pre self.profile.moviePreference->notEmpty()
  ```

  Το περιεχόμενο του OCL περιορισμού αναλύεται κατά τη μετατροπή του μοντέλου – όπως και τα σχόλια `<<bind>>` – και χρησιμοποιείται στην παραγωγή του περιεχομένου του κώδικα του αντίστοιχου plugin για να καθορίσει τις περιπτώσεις κλήσης της κλάσης `MoviePersonalization`. Μέσω της συγκεκριμένης προσαρμογής τα είδη ταινιών που προτιμά ο χρήστης τοποθετούνται πρώτα στη λίστα των διαθέσιμων ταινιών που θα εμφανιστούν κατά την εκτέλεση της εφαρμογής.

Στο τελευταίο βήμα σχεδίασης μοντελοποιείται η ροή της εφαρμογής, όπως φαίνεται στην Εικόνα 42. Η πρώτη όψη στην εφαρμογή σχετίζεται με τη λειτουργία της πιστοποίησης της ταυτότητας του χρήστη (κατάσταση `User Login`). Όταν πατηθεί το πλήκτρο login, όπως φαίνεται στη μετάβαση μεταξύ της πρώτης και δεύτερης κατάστασης, καλείται μια μέθοδος για την πιστοποίηση του χρήστη και την εισαγωγή του στο σύστημα των υπηρεσιών του πολυχώρου (authenticate), όπως φαίνεται στη δραστηριότητα exit της κατάστασης `User Login`. Η μέθοδος authenticate δε σχετίζεται με κάποια από τις υπηρεσίες διαδικτύου που εισήχθησαν στο περιβάλλον σχεδίασης, αλλά με κάποια ανεξάρτητη λειτουργία της εφαρμογής. Σε περίπτωση που η πιστοποίηση του χρήστη είναι επιτυχής, όπως φαίνεται από τον αντίστοιχο OCL περιορισμό στη μετάβαση, ενεργοποιείται η επόμενη κατάσταση για το χαιρετισμό του χρήστη (κατάσταση `Greet the User`). Η μέθοδος χαιρετισμού `greet` της υπηρεσίας `GreetingService` καλείται πριν την είσοδο στην όψη (δραστηριότητα entry) και το μήνυμα απάντησης εμφανίζεται ως κείμενο στην οθόνη του χρήστη, όπως φαίνεται στις τιμές ετικετών της παραμέτρου greeting που έχει προστεθεί στη δραστηριότητα (Εικόνα 43).

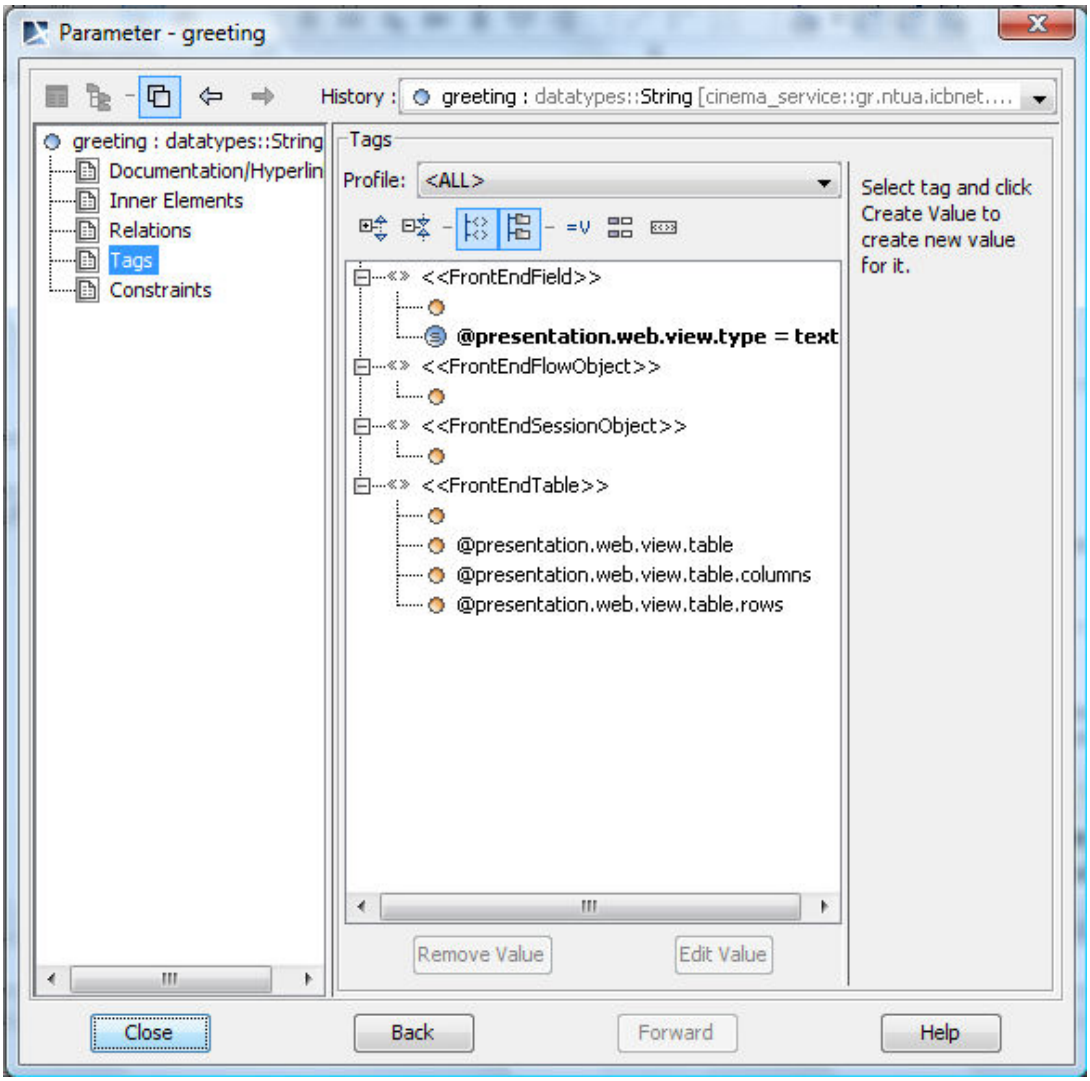
Στο σημείο αυτό ορίζονται δύο καταστάσεις για τη συνέχεια της εφαρμογής: με το πάτημα του υπερσυνδέσμου movies καλείται η μέθοδος `showMovieList` και πραγματοποιείται η μετάβαση στην επόμενη κατάσταση (κατάσταση `Show Movie List`), ενώ μια μετάβαση στην κατάσταση εξόδου (κατάσταση `Exit`) εκκινείται όταν ο χρήστης πατήσει τον υπερσύνδεσμο της εξόδου (exit link). Στην κατάσταση `Show Movie List` εμφανίζεται η λίστα των διαθέσιμων ταινιών σε μορφή πίνακα, όπως εκφράζεται από τις τιμές ετικετών της παραμέτρου movieList που έχει συμπεριληφθεί στη δραστηριότητα κλήσης της μεθόδου `showMovieList`:

```
@presentation.web.view.type = table
@presentation.web.view.table.columns = [Title, Start Time, End Time, Room, Details, Book]
```

Όσο ο χρήστης βρίσκεται στην όψη που περιλαμβάνει τη λίστα με τις διαθέσιμες ταινίες είναι εφικτή μια μετάβαση προς την όψη που εμφανίζει πιο αναλυτικές πληροφορίες για μια συγκεκριμένη ταινία (κατάσταση `Show Movie Details`) ή προς μια σύνθετη κατάσταση που σχετίζεται με την κράτηση θέσεων (κατάσταση `Show Transaction Summary`) ή προς την όψη εξόδου. Στην περίπτωση που επιλεχθεί η παρουσίαση των πληροφοριών για κάποια ταινία ο κωδικός της ταινίας μεταφέρεται από τη λίστα των ταινιών ως παράμετρος στην επόμενη όψη και μάλιστα χρησιμοποιείται κατά την κλήση της μεθόδου `showMovieWithId` (τιμή ετικέτας

`@presentation.web.view.action.tablelink = movies`). Από την όψη των πληροφοριών της ταινίας ο χρήστης μπορεί να επιστρέψει στην προηγούμενη όψη της λίστας των ταινιών.

Εικόνα 42. Διάγραμμα καταστάσεων για τη ροή της εφαρμογής (σενάριο πολυχώρου).



Εικόνα 43. Χαρακτηριστικά παραμέτρου greeting.

Η σύνθετη κατάσταση κράτησης περιλαμβάνει δύο υπο-καταστάσεις με τις δύο μεθόδους της `BookingService`, των οποίων η κλήση εξαρτάται από την επιλογή μεθόδου της προσαρμογής στο πλαίσιο χρήσης (καταστάσεις `Pay With Cash` και `Pay With CreditCard`).

Για την πλήρη μοντελοποίηση της ροής της εφαρμογής απαιτούνται διάφορες τιμές ετικετών, π.χ. για τα αντικείμενα που μεταφέρονται μέσω των μεταβάσεων. Οι περισσότερες από αυτές τις ετικέτες δεν είναι ορατές στην Εικόνα 42, καθώς το περιβάλλον μοντελοποίησης δεν εμφανίζει όλες τις ιδιότητες των UML στοιχείων στο διάγραμμα για λόγους παρουσίασης. Ωστόσο, αυτές οι ιδιότητες είναι ορατές στους πίνακες των χαρακτηριστικών του κάθε στοιχείου.

7.2.2.2 Μετατροπή σε κώδικα

Έχοντας ως είσοδο τα παραπάνω διαγράμματα για την εφαρμογή του σεναρίου εκκινείται η διαδικασία παραγωγής κώδικα. Κατά την ανάλυση και μετατροπή του μοντέλου εμφανίζονται αντίστοιχα μηνύματα που πληροφορούν το χρήστη για τις ενέργειες που πραγματοποιούνται και τα αρχεία που παράγονται. Τμήμα της εξόδου της κονσόλας μετατροπής εμφανίζεται στην Εικόνα 44.

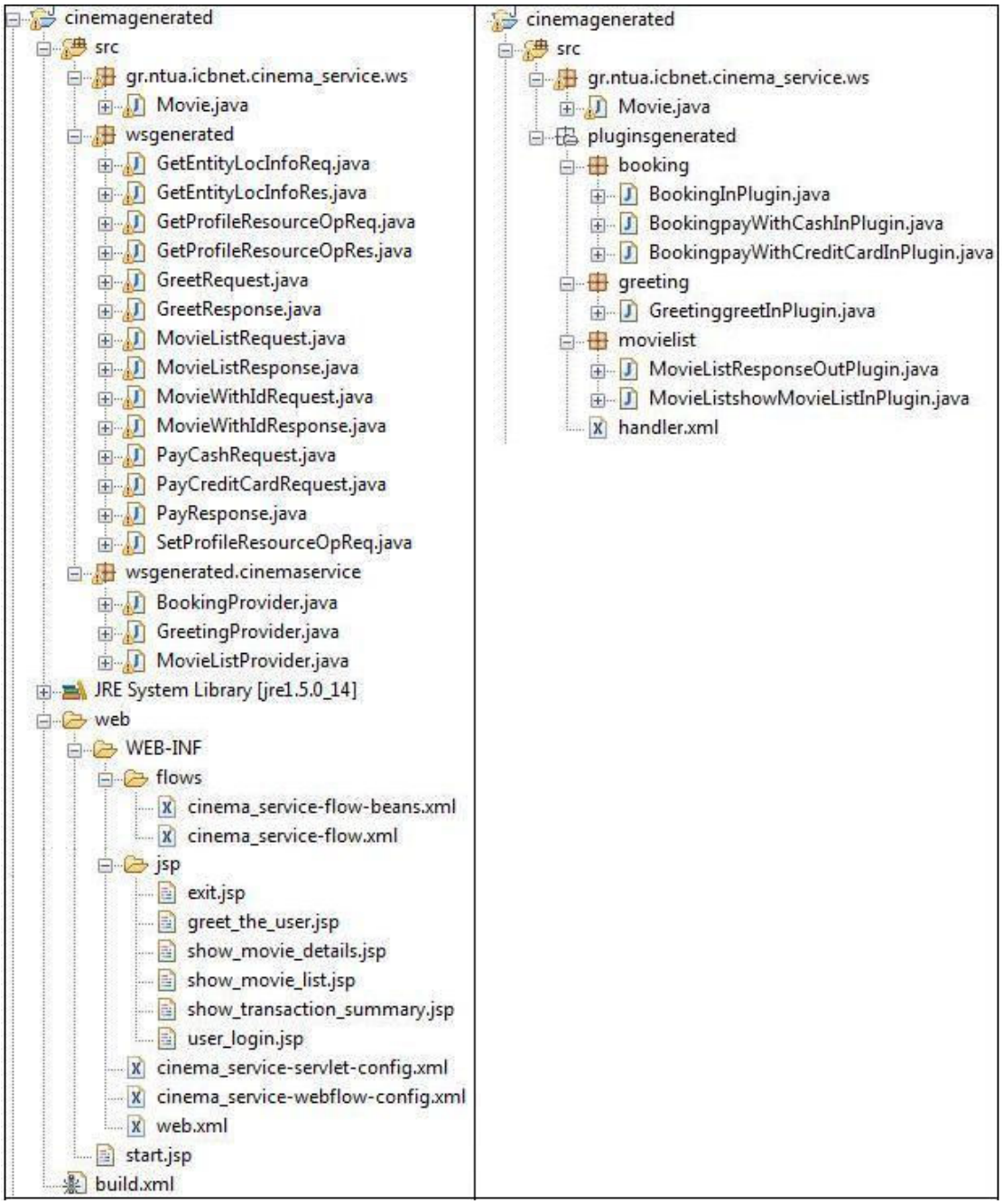
```
D:/Development/workspace/projects/ContextWS-MDE-Test/data/cinema/Simple_Web_Scenario_Model_cinema.uml2
[UML2ModelParser] ********parsing class MoviePersonalization with stereotype ExtContextUML__Trigger
. . .
[UML2ModelParser] ********parsing class GreetRequest with stereotype ExtContextUML__InMessage
[UML2ModelParser] GreetRequest.java class GENERATED
[UML2ModelParser] ********parsing class GreetResponse with stereotype ExtContextUML__OutMessage
[UML2ModelParser] GreetResponse.java class GENERATED
[UML2ModelParser] MovieListProvider.java web service client GENERATED
[UML2ModelParser] ********parsing class MovieListResponse with stereotype ExtContextUML__OutMessage
[UML2ModelParser] MovieListResponse.java class GENERATED
[UML2ModelParser] BookingProvider.java web service client GENERATED
[UML2ModelParser] ********parsing class PayCashRequest with stereotype ExtContextUML__InMessage
[UML2ModelParser] PayCashRequest.java class GENERATED
[UML2ModelParser] ********parsing class PayCreditCardRequest with stereotype ExtContextUML__OutMessage
[UML2ModelParser] PayCreditCardRequest.java class GENERATED
[UML2ModelParser] ********parsing class PayResponse with stereotype ExtContextUML__OutMessage
[UML2ModelParser] PayResponse.java class GENERATED
. . .
[UML2ModelParser] ********parsing class Movie with stereotype ExtContextUML__ComplexType
[UML2ModelParser] Movie.java class GENERATED
. . .
 [UML2ModelParser] ANT build.xml file GENERATED
 [UML2ModelParser] ********parsing class GreetRequest with stereotype ExtContextUML__InMessage
. . .
[UML2ModelParser] ********parsing service Greeting for method selection context dependency
[UML2ModelParser] ********parsing service Greeting for parameter injenction dependency
[UML2ModelParser] ********parsing service Greetingoutput for response manipulation dependency
[UML2ModelParser] GreetinggreetInPlugin.java parameter injection plugin GENERATED
[UML2ModelParser] ********parsing service MovieList for method selection context dependency
[UML2ModelParser] ********parsing service MovieList for parameter injenction dependency
[UML2ModelParser] ********parsing service MovieListoutput for response manipulation dependency
[UML2ModelParser] MovieListResponseOutPlugin.java response manipulation plugin GENERATED
[UML2ModelParser] ********parsing service MovieListoutput for response manipulation dependency
```

```
[UML2ModelParser] ********parsing service Booking for method selection context dependency
[UML2ModelParser] BookingInPlugin.java operation selection plugin GENERATED
[UML2ModelParser] ********parsing service Booking for parameter injenction dependency
[UML2ModelParser] ********parsing service Bookingoutput for response manipulation dependency
[UML2ModelParser] ********parsing service Bookingoutput for response manipulation dependency
. . .
[UML2ModelParser] ******parsing state machine with name scenario_flow to generate config files
[UML2ModelParser] cinema_service-flow.xml file GENERATED
[UML2ModelParser] cinema_service-flow-beans.xml file GENERATED
[UML2ModelParser] web.xml file GENERATED
[UML2ModelParser] cinema_service-servlet-config.xml file GENERATED
[UML2ModelParser] cinema_service-webflow-config.xml file COPIED
[UML2ModelParser] greet_the_user.jsp view GENERATED
[UML2ModelParser] show_movie_list.jsp view GENERATED
. . .
[UML2ModelParser] show_transaction_summary.jsp view GENERATED
[UML2ModelParser] show_movie_details.jsp view GENERATED
[UML2ModelParser] exit.jsp view GENERATED
. . .
```

Εικόνα 44. Τμήμα μηνυμάτων εξόδου του εργαλείου μετατροπής μοντέλου.

Αφού ολοκληρωθεί η διαδικασία, παράγονται ο κώδικας και τα αρχεία που παρουσιάζονται στην Εικόνα 45 τόσο για την εφαρμογή διαδικτύου όσο και για τα plugin προσαρμογής. Ο κώδικας περιλαμβάνει τους πελάτες για τις τρεις υπηρεσίες διαδικτύου (`GreetingProvider`, `MovieListProvider` και `BookingProvider`), τις κλάσεις για τα μηνύματα των υπηρεσιών και τους πολύπλοκους τύπους δεδομένων (packages `wsgenerated` και `gr.ntua.icbnet.cinema_service.ws` αντίστοιχα), τις όψεις που θα εμφανίζονται κατά την εκτέλεση (φάκελος *jsp*) και τα αρχεία διαμόρφωσης για την εκτέλεση της εφαρμογής, όπως απαιτείται από τα Spring MVC και Spring Web Flow, (φάκελοι *flows*, *web* και *WEB-INF*). Από την πλευρά των plugin περιλαμβάνεται το αρχείο με τις συσχετίσεις των plugin με τις υπηρεσίες (*handler.xml*) και, τέλος, τα ίδια τα plugin που χρειάζονται ανά υπηρεσία (π.χ. `MovieListshowMovieListInPlugin` και `MovieListResponseOutPlugin` για την υπηρεσία `BookingService`).

Το αρχείο διαμόρφωσης *handler.xml* που έχει παραχθεί παρουσιάζεται στην Εικόνα 46. Οι αναφορές που υπάρχουν προς jar αρχεία σχετίζονται με το αποτέλεσμα της εκτέλεσης του αντίστοιχου *build.xml* script του Ant που επίσης παράγεται μέσω του εργαλείου για να διευκολύνει την εγκατάσταση της εφαρμογής και των plugin.



Εικόνα 45. Κώδικας και αρχεία που παράγονται για την εφαρμογή διαδικτύου και τα plugins προσαρμογής στο πλαίσιο χρήσης (σενάριο πολυχώρου).

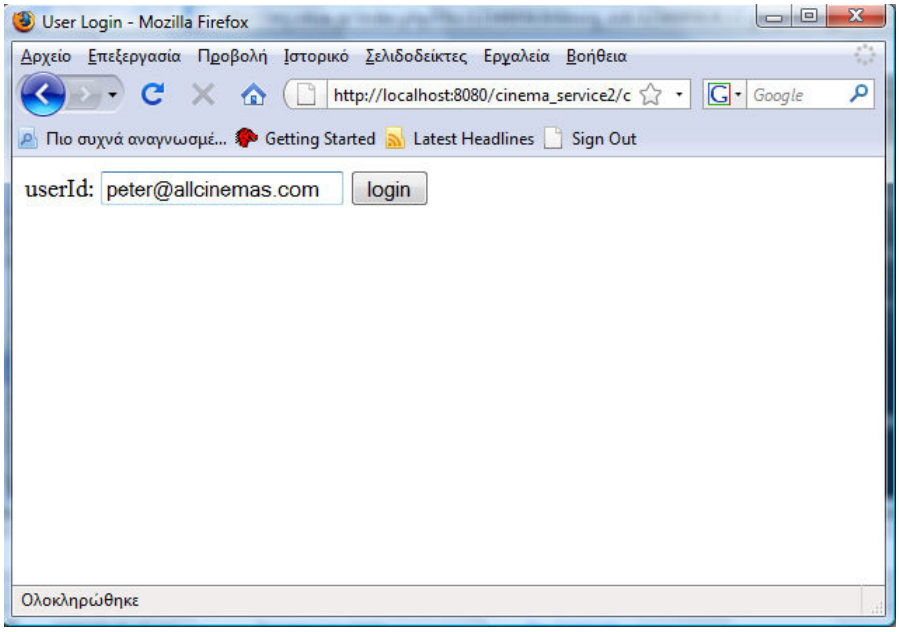
```
<?xml version="1.0" encoding="UTF-8"?>
<plugins>
	<plugin id="parinj1" pckg="pluginsgenerated.greeting"
classname="GreetinggreetInPlugin" jarfile="GreetingPl.jar"/>
	<plugin id="resman2" pckg="pluginsgenerated.movielist"
classname="MovieListResponseOutPlugin" jarfile="MovieListPl.jar"/>
	<plugin id="parinj3" pckg="pluginsgenerated.movielist"
classname="MovieListshowMovieListInPlugin" jarfile="MovieListPl.jar"/>
	<plugin id="opsel4" pckg="pluginsgenerated.booking" classname="BookingInPlugin"
jarfile="BookingPl.jar"/>
```

```
        <plugin id="parinj5" pckg=" pluginsgenerated.booking"
classname="BookingpayWithCashInPlugin" jarfile="BookingPl.jar"/>
        <plugin id="parinj6" pckg=" pluginsgenerated.booking"
classname="BookingpayWithCreditCardInPlugin" jarfile="BookingPl.jar"/>
        <association>
          <serviceEnd name="GreetingService" operation="greet"/>
          <pluginEnd id="parinj1" direction="in"/>
        </association>
        <association>
          <serviceEnd name="MovieListService" operation="showMovieList"/>
          <pluginEnd id="resman2" direction="in"/>
        </association>
        <association>
          <serviceEnd name="MovieListService" operation="showMovieList"/>
          <pluginEnd id="resman2" direction="out"/>
        </association>
        <association>
          <serviceEnd name="MovieListService" operation="showMovieList"/>
          <pluginEnd id="parinj3" direction="in"/>
        </association>
        <association>
          <serviceEnd name="BookingService" operation="payWithCash"/>
          <pluginEnd id="opsel4" direction="in"/>
        </association>
        <association>
          <serviceEnd name="BookingService" operation="payWithCash"/>
          <pluginEnd id="parinj5" direction="in"/>
        </association>
        <association>
          <serviceEnd name="BookingService" operation="payWithCreditCard"/>
          <pluginEnd id="parinj6" direction="in"/>
        </association>
</plugins>
```

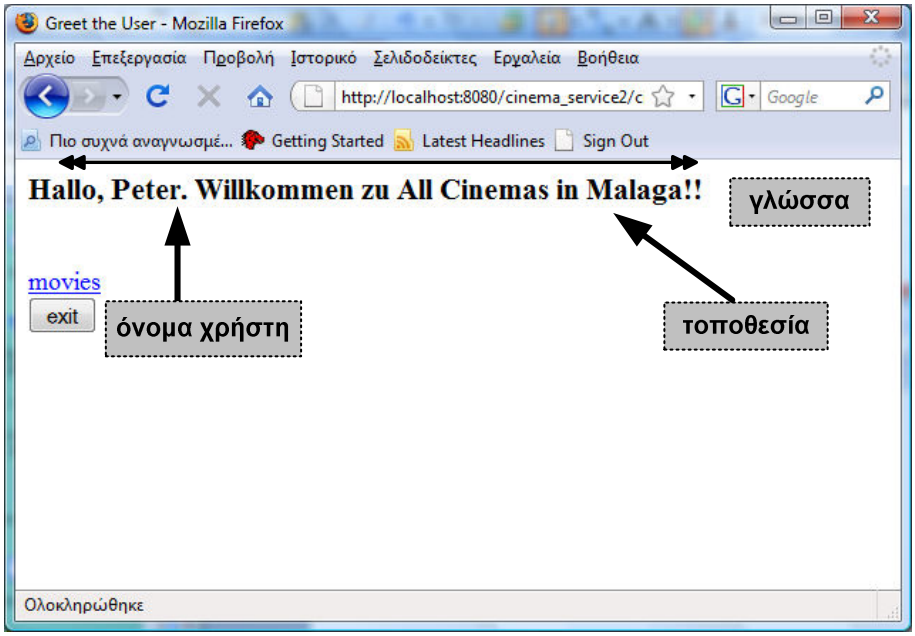
Εικόνα 46. Αρχείο διαμόρφωσης handler.xml που παράγεται (σενάριο πολυχώρου).

7.2.3 Εκτέλεση σεναρίου

Στις εικόνες που ακολουθούν παρουσιάζεται ένα παράδειγμα εκτέλεσης του σεναρίου για τον πολυχώρο με όνομα "AllCinemas". Μετά την είσοδο του χρήστη στην εφαρμογή (Εικόνα 47) εμφανίζεται το μήνυμα χαιρετισμού ως απάντηση από την κλήση της υπηρεσίας `GreetingService` (Εικόνα 48). Το μήνυμα εμφανίζεται στη γλώσσα του χρήστη (Γερμανικά στην προκειμένη περίπτωση) λαμβάνοντας υπόψη και το όνομά του (Peter) και την τοποθεσία στην οποία βρίσκεται (Malaga), όπως προκύπτει από την μετατροπή που πραγματοποιείται από το plugin `GreetinggreetInPlugin`.



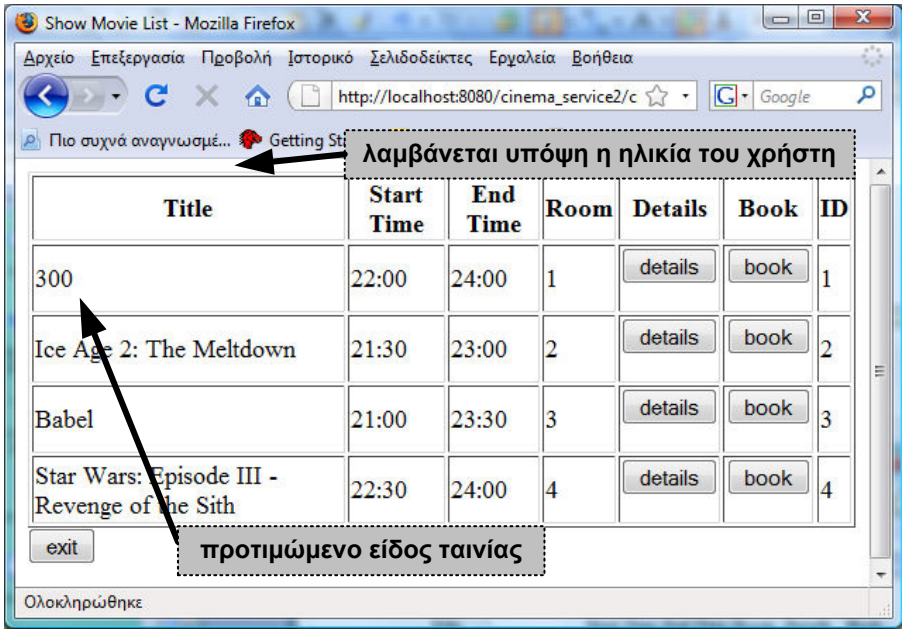
Εικόνα 47. Εισαγωγή χρήστη στο σύστημα (εκτέλεση σεναρίου πολυχώρου).



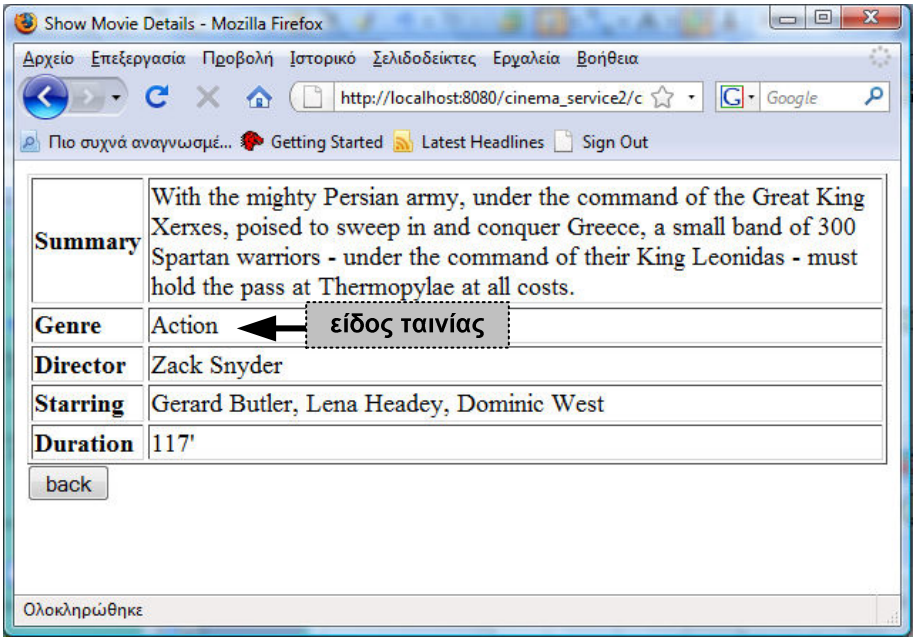
Εικόνα 48. Σελίδα χαιρετισμού χρήστη (εκτέλεση σεναρίου πολυχώρου).

Όταν ο χρήστης προχωρησει στην επόμενη σελίδα η λίστα με τις διαθέσιμες ταινίες ανακτάται από την αντίστοιχη υπηρεσία διαδικτύου (Εικόνα 49) λαμβάνοντας υπόψη την ηλικία του χρήστη (άνω των 18) και παρουσιάζεται ταξινομημένη βάσει των προτιμήσεών του για τα είδη ταινιών (οι ταινίες δράσης / action στην προκειμένη περίπτωση εμφανίζονται πρώτες). Το αποτέλεσμα αυτό προκύπτει από τη δράση των plugin `MovieListshowMovieListInPlugin` και `MovieListResponseOutPlugin`. Από τη βασική σελίδα των ταινιών ο χρήστης μπορεί να δει περισσότερες πληροφορίες για

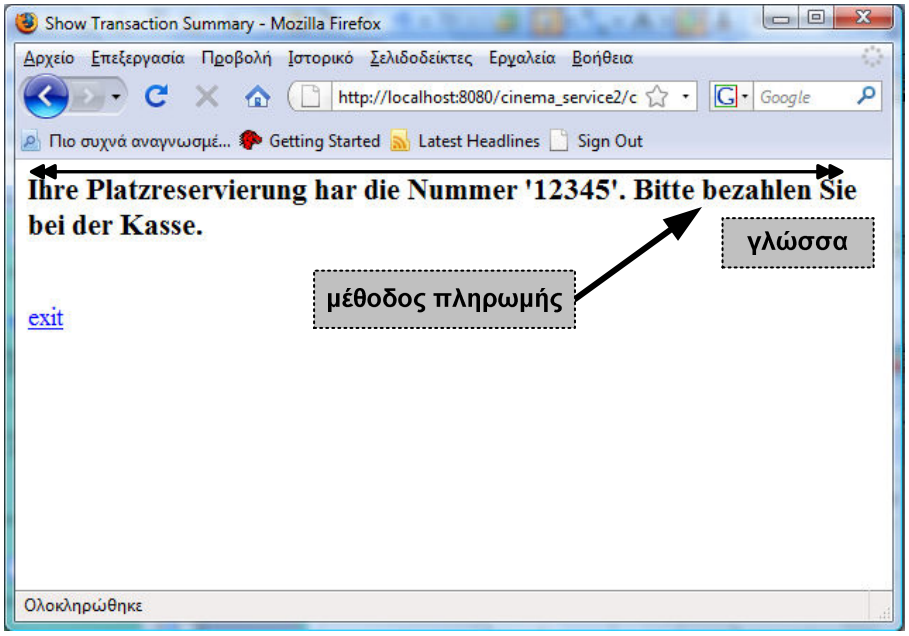
όποια ταινία από τη λίστα επιλέξει (Εικόνα 50) ή να κάνει κράτηση για ένα εισιτήριο για την προβολή κάποιας ταινίας (Εικόνα 51) χρησιμοποιώντας τη μέθοδο πληρωμής που προτιμά (μετρητά στο παράδειγμα). Το μήνυμα που επιστρέφεται από την υπηρεσία κράτησης παρουσιάζεται και αυτό στη γλώσσα του χρήστη (Γερμανικά).



Εικόνα 49. Λίστα με τις διαθέσιμες ταινίες (εκτέλεση σεναρίου πολυχώρου).



Εικόνα 50. Πληροφορίες για την πρώτη ταινία της λίστας (εκτέλεση σεναρίου πολυχώρου).



Εικόνα 51. Κράτηση θέσης για την ταινία (εκτέλεση σεναρίου πολυχώρου).

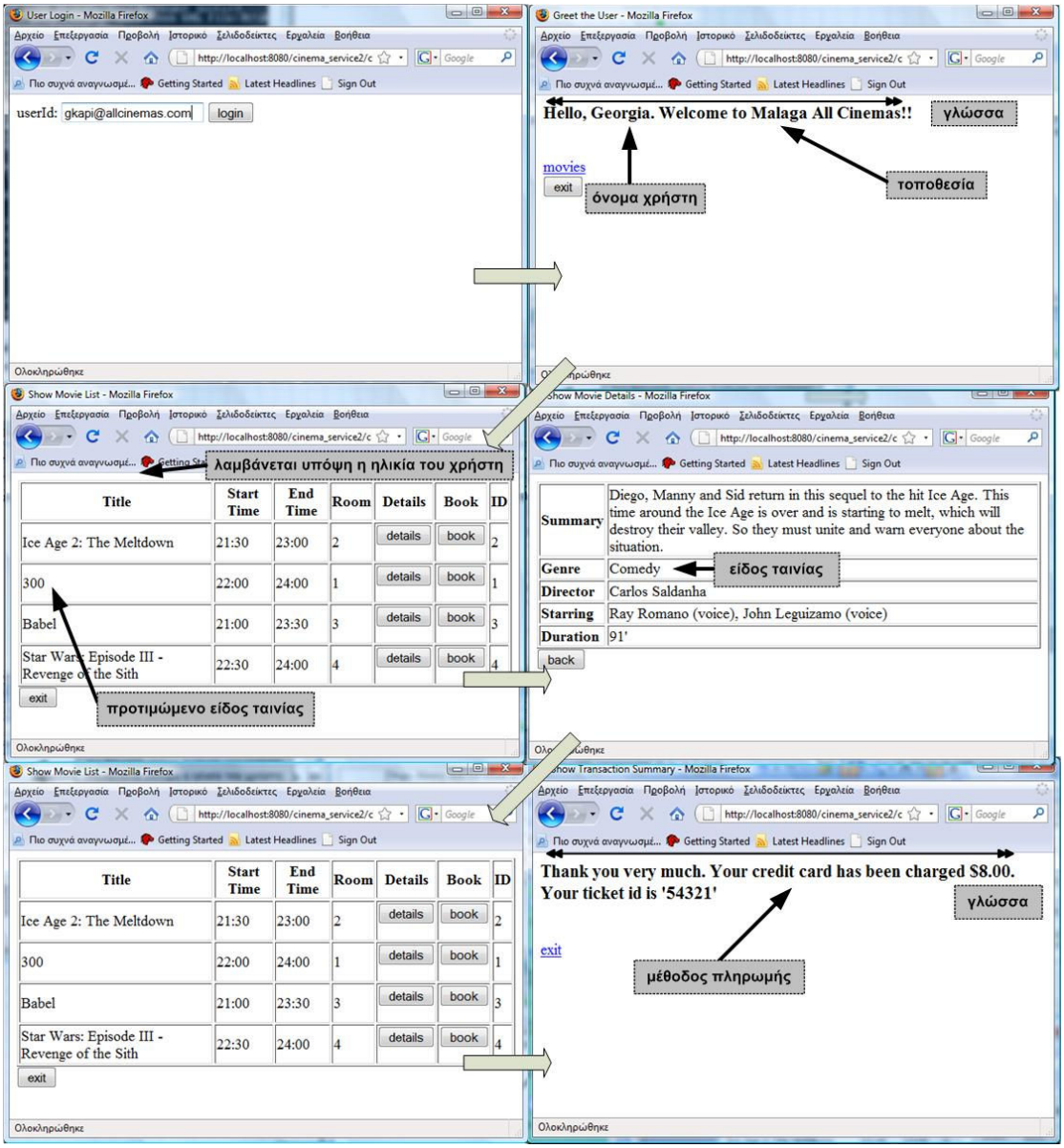
Στην Εικόνα 52 εμφανίζεται άλλη μια περίπτωση εκτέλεσης για κάποιον άλλο εγγεγραμμένο χρήστη του συστήματος. Σημειώνεται άλλη μια φορά πως οι όψεις που παρουσιάζονται στη μορφή που παράγονται από το εργαλείο παραγωγής κώδικα είναι απλές και περιλαμβάνουν μόνο τα βασικά στοιχεία. Μπορούν να εμπλουτιστούν περισσότερο από κάποιο σχεδιαστή σελίδων διαδικτύου.

7.3 Αξιολόγηση Προτεινόμενης Μεθοδολογίας

Για τον έλεγχο της σωστής λειτουργίας και την αποτίμηση της μοντελοκεντρικής μεθοδολογίας δοκιμάστηκαν διάφορες περιπτώσεις χρήσης κατά τον ορισμό των προφίλ και την ανάπτυξη του εργαλείου παραγωγής σε μια επαναληπτική διαδικασία. Στον αρχικό έλεγχο για την ολοκληρωμένη μεθοδολογία περιλήφθηκαν περιπτώσεις αφαιρετικών υπηρεσιών διαδικτύου χωρίς συγκεκριμένες λειτουργίες (όπως στην απλή εφαρμογή που παρουσιάστηκε). Ακολούθως, δοκιμάστηκε η σωστή παραγωγή κώδικα και η προσαρμογή για μεμονωμένες υπηρεσιές διαδικτύου (για τις υπηρεσίες του παραπάνω σεναρίου, αλλά και πρόσθετες υπηρεσίες).

Στο επόμενο στάδιο συμπεριλήφθηκαν στη διαδικασία ελέγχων σύνθετες εφαρμογές με διάφορες πολυπλοκότητες όσον αφορά τον αριθμό των υπηρεσιών εφαρμογών και υπηρεσιών πλαισίου χρήσης που συμμετείχαν, τον αριθμό στοιχείων του μοντέλου πλαισίου χρήσης και τον αριθμό των εξαρτήσεων των υπηρεσιών από την πληροφορία πλαισίου χρήσης. Τέλος, πραγματοποιήθηκαν δοκιμές για πολύπλοκα σενάρια χρήσης,

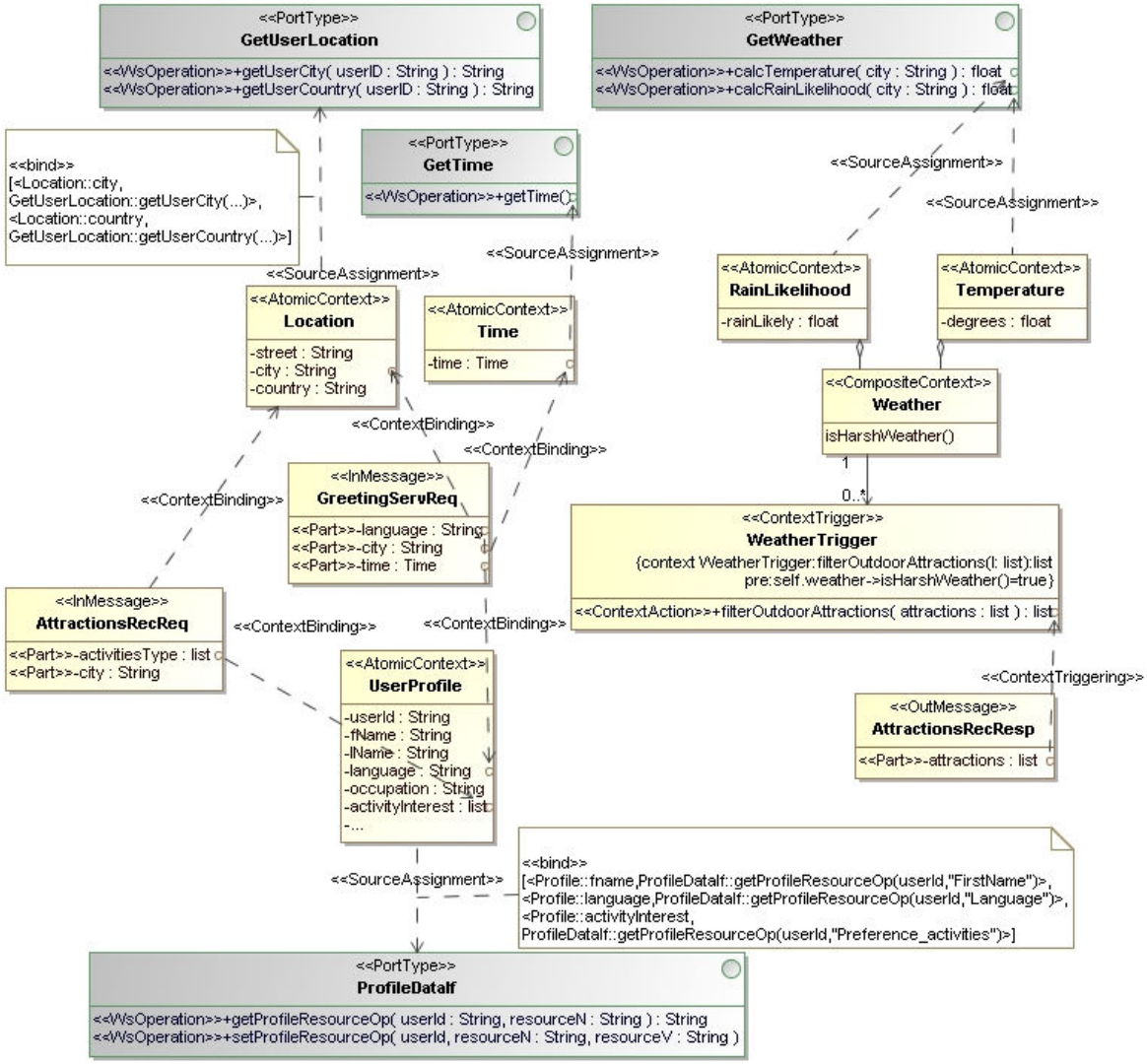
όπως το σενάριο του πολυχώρου που αναλύθηκε και η εφαρμογή τουρισμού που φαίνεται στην Εικόνα 53 (διάγραμμα κλάσεων για τις συσχετίσεις πλαισίου χρήσης).



Εικόνα 52. Εκτέλεση σεναρίου πολυχώρου για άλλο χρήστη του συστήματος.

Η ποιοτική δοκιμή μιας μεθοδολογίας μηχανικής λογισμικού όπως αυτή που προτείνεται στη διατριβή σχετίζεται συχνά με διάφορες μελέτες περίπτωσης (case studies) [7], που ορίζονται ως:

"Οι μελέτες περίπτωσης περιλαμβάνουν την αξιολόγηση μιας μεθοδολογίας / ενός εργαλείου μετά από τη χρησιμοποίησή της / του σε ρεαλιστικά έργα ανάπτυξης λογισμικού."



Εικόνα 53. Μοντέλο πλαισίου χρήσης και συσχετίσεις (εφαρμογή τουρισμού).

Ο βασικός έλεγχος που πραγματοποιήθηκε σε αυτές τις περιπτώσεις περιελάμβανε τη σωστή λειτουργία της προκύπτουσας εφαρμογής. Επιπλέον, η τρέχουσα υλοποίηση του εργαλείου που συνοδεύει τη μεθοδολογία είναι διαθέσιμη στο διαδίκτυο [4].

Στη βιβλιογραφία συναντώνται διάφορες απαιτήσεις και ποιοτικές μετρικές αξιολόγησης για τη μηχανική λογισμικού γενικότερα [6] [7] και τη μοντελοκεντρική ανάπτυξη και αντίστοιχα εργαλεία μοντελοκεντρικής αρχιτεκτονικής ειδικότερα [3] [1] [5] [8]. Η προτεινόμενη λύση δεν αποτελεί ένα νέο γενικό εργαλείο μοντελοκεντρικής ανάπτυξης ή μια νέα υλοποίηση της MDA προδιαγραφής με όλα τα βήματα που ορίζονται, αλλά μια μεθοδολογία προσέγγισης της ανάπτυξης μιας ειδικής ομάδας εφαρμογών (εφαρμογών που αποτελούνται από υπηρεσίες διαδικτύου με επίγνωση του πλαισίου χρήσης) που βασίζεται σε μοντελοκεντρικές τεχνικές και εργαλεία. Έτσι, στην παρούσα ενότητα επιλέχθηκε ένα υποσύνολο των προτεινόμενων μετρικών και απαιτήσεων, οι

οποίες μπορούν να εφαρμοστούν στην προτεινόμενη λύση και να αντιστοιχηθούν με τις λειτουργίες της. Η αντιστοίχιση παρουσιάζεται στους πίνακες που ακολουθούν, όπου εμφανίζονται τα πλεονεκτήματα της μεθοδολογίας, αλλά και κάποιες ελλείψεις που παρουσιάζει και που κυρίως σχετίζονται με απαιτήσεις που πρέπει να πληρούνται από γενικά εργαλεία μοντελοκεντρικής ανάπτυξης και όχι από πιο ειδικές λύσεις, όπως αυτή που προτείνεται στη διατριβή.

Οι απαιτήσεις κατηγοριοποιούνται σε τρεις ομάδες (Πίνακες 5 μέχρι 7):

- ***Χαρακτηριστικά μοντελοκεντρικής ανάπτυξης*** [3] [1] [5]: αποτιμούν το βαθμό στον οποίο ακολουθούνται οι αρχές της μοντελοκεντρικής ανάπτυξης. Τα κριτήρια αυτής της ομάδας απορρέουν κυρίως από την προδιαγραφή της μοντελοκεντρικής αρχιτεκτονικής της OMG.
- ***Ποιότητα και Χρήση*** [6] [7]: αποτιμούν την ποιότητα του εργαλείου συμπεριλαμβανομένων της αποδοτικότητάς του, της ευκολίας υλοποίησης, κτλ., καθώς και την αλληλεπίδραση με τους χρήστες.
- ***Παραγωγικότητα*** [1]: αποτιμούν τα οφέλη που επιτυγχάνονται μέσω της χρήσης του μοντελοκεντρικού εργαλείου.

Α/Α	**Κριτήριο**	**Λειτουργία Προτεινόμενης Λύσης**
Ομάδα Α: Χαρακτηριστικά μοντελοκεντρικής ανάπτυξης		
Α.1	Υποστήριξη PIM και πολλαπλών PSM (υποστήριξη τουλάχιστον ενός PIM και διαφορετικών PSM)	✓ Χρήση αφαιρετικού PIM υπηρεσιών διαδικτύου (απευθείας αντιστοίχιση σε κώδικα μέσω των κατάλληλων template) ✓ Νοητή αντιστοίχιση σε PSM για την αρχιτεκτονική προσαρμογής στο πλαίσιο χρήσης μέσω handler – Έλλειψη πλήρως αφαιρετικού επιπέδου μοντελοποίησης – Έλλειψη αντιστοίχισης σε πολλαπλά PSM (στην παρούσα υλοποίηση μεθοδολογίας), αλλά εφικτή
Α.2	Μετατροπές (είδη μετατροπών που υποστηρίζονται)	✓ Υποστήριξη μετατροπών μοντέλου σε κώδικα – Έλλειψη μετατροπής μοντέλου σε μοντέλο
Α.3	Εισαγωγή/εξαγωγή μοντέλων	✓ Εισαγωγή διαθέσιμων μοντέλων (υπηρεσιών εφαρμογών και υπηρεσιών πλαισίου χρήσης)

		✓ Επαναχρησιμοποίηση μοντέλων που προκύπτουν κατά τη σχεδίαση
Α.4	Πρότυπα και γενίκευση (χρήση έτοιμων προτύπων στις μετατροπές)	✓ Ορισμός προτύπων μετατροπών για όλες τις πτυχές της εφαρμογής
Α.5	UML προφίλ και τύποι διαγραμμάτων	✓ Υιοθέτηση UML ως γλώσσα μοντελοποίησης ✓ Χρήση διαγραμμάτων κλάσεων, καταστάσεων και περιπτώσεων χρήσης
Α.6	Υποστήριξη OCL	✓ Υποστήριξη OCL εκφράσεων
Α.7	Χρήση τυποποιήσεων (προδιαγραφές που υποστηρίζονται)	✓ Χρήση UML, XML, MOF
Α.8	Συνεργασία με περιβάλλοντα μοντελοποίησης	✓ Το μοντέλο μπορεί να σχεδιαστεί σε διάφορα περιβάλλοντα και εν συνεχεία να τροφοδοτήσει τη διαδικασία μετατροπής
Α.9	Αντίστροφη μηχανική (εξαγωγή μοντέλων από κώδικα / reverse engineering)	– Δεν υποστηρίζεται (στην παρούσα υλοποίηση μεθοδολογίας), αλλά είναι εφικτή

Πίνακας 5. Αντιστοίχιση κριτηρίων μοντελοκεντρικής ανάπτυξης: χαρακτηριστικά μοντελοκεντρικής ανάπτυξης.

Α/Α	**Κριτήριο**	**Λειτουργία Προτεινόμενης Λύσης**
Ομάδα Β: Ποιότητα και Χρήση		
Β.1	Ευκολία χρήσης[2] (Simplicity) και Κατανοησιμότητα (Understandability)	✓ Κατανόηση από χρήστες και διαλειτουργικότητα με διάφορα εργαλεία (αρχές μοντελοκεντρικής ανάπτυξης) ✓ Παροχή οδηγιών χρήσης [4]

[2] Βλ. και ειδικά κριτήρια για εφαρμογές με επίγνωση του πλαισίου χρήσης.

Β.2	Δυνατότητα παραγωγής των αναμενόμενων αποτελεσμάτων	✓ Υποστήριξη ανάπτυξης διαφόρων στοιχείων της εφαρμογής διαδικτύου (διεπαφή χρήστη, λογική εφαρμογής, προσαρμογή στο πλαίσιο χρήσης) ✓ Παραγωγή έτοιμης προς εκτέλεση εφαρμογής με ελάχιστα σημεία, όπουενδέχεται να απαιτείται παρέμβαση
Β.3	Πληρότητα (Completeness)	✓ Υποστήριξη σχεδιαστικών στόχων (ανεξάρτησία χειρισμού πλαισίου χρήσης, επαναχρησιμοποίηση υπηρεσιών, κτλ.)
Β.4	Δυνατότητα μάθησης (Learnability)	✓ Απλοποίηση της διαδικασίας μέσω της χρήσης ευρέως αποδεκτών προτύπων (π.χ. χρήση γλώσσας μοντελοποίησης γενικού σκοπού)

Πίνακας 6. Αντιστοίχιση κριτηρίων μοντελοκεντρικής ανάπτυξης: ποιότητα και χρήση.

Α/Α	**Κριτήριο**	**Λειτουργία Προτεινόμενης Λύσης**
Ομάδα Γ: Παραγωγικότητα		
Γ.1	Μειωμένος χρόνος ανάπτυξης	✓ Διευκόλυνση διαδικασίας ανάπτυξης μέσω των επαναχρησιμοποιήσιμων μοντέλων και της αυτόματης παραγωγής κώδικα (που διαρκεί κάποια δευτερόλεπτα)
Γ.2	Μειωμένη πολυπλοκότητα υλοποίησης	✓ Τα διάφορα επίπεδα της εφαρμογής (επίπεδο παρουσίασης, επίπεδο εφαρμογής, προσαρμογή στο πλαίσιο χρήσης) διαφοροποιούνται και στη σχεδίαση και στον παραγόμενο κώδικα ✓ Σαφώς διαχωρισμένες λειτουργίες και ρόλοι
Γ.3	Ποιότητα κώδικα	✓ Επαναχρησιμοποιήσιμα συστατικά μέρη ✓ Αποφυγή περιττών ελέγχων και επαναλήψεων – Δεν έχει πραγματοποιηθεί εκτεταμένος ποσοτικός έλεγχος για την ποιότητα κώδικα του εργαλείου μετατροπής

Γ.4	Μειωμένο επίπεδο ικανότητων	✓ Υποστήριξη αφαιρετικότητας λειτουργιών ✓ Δε χρειάζεται ο μηχανικός λογισμικού να γνωρίζει όλες τις λεπτομέρειες υλοποίησης (π.χ. ακριβής λειτουργία μηχανισμού προσαρμογής στο πλαίσιο χρήσης)

Πίνακας 7. Αντιστοίχιση κριτηρίων μοντελοκεντρικής ανάπτυξης: παραγωγικότητα.

Συνδυάζοντας την παραπάνω βιβλιογραφία με τα χαρακτηριστικά των εφαρμογών πλαισίου χρήσης και τη γενικότερη μελέτη της ερευνητικής περιοχής προκύπτουν κάποια πρόσθετα ειδικότερα κριτήρια που σχετίζονται με την ανάπτυξη, αλλά και την παροχή υπηρεσιών με επίγνωση του πλαισίου χρήσης και είναι τα εξής:

- ***Ευελιξία:*** σχετίζεται με το βαθμό ελευθερίας που παρέχει στο μηχανικό λογισμικού ο μηχανισμός προσαρμογής στο πλαίσιο χρήσης.
- ***Δυνατότητα μετατροπών:*** δείχνει πόσο εύκολος είναι ο επανασχεδιασμός της εφαρμογής σε περίπτωση που πραγματοποιηθούν αλλαγές είτε στη λογική της εφαρμογής είτε στην απαιτούμενη προσαρμογή στο πλαίσιο χρήσης.
- ***Ευκολία χρήσης μηχανισμού:*** δείχνει πόσο εύκολο είναι για ένα μηχανικό λογισμικού μη εξοικειωμένο με το συγκεκριμένο μηχανισμό προσαρμογής στο πλαίσιο χρήσης να το χρησιμοποιήσει.
- ***Ενσωμάτωση σε υπάρχον σύστημα:*** δείχνει πόσο εύκολη ή δύσκολη είναι η ενσωμάτωση του μηχανισμού προσαρμογής σε ένα υπάρχον σύστημα.
- ***Είδος μοντελοποίησης πλαισίου χρήσης:*** δείχνει το είδος της μοντελοποίησης που χρησιμοποιείται για την πληροφορία πλαισίου χρήσης.
- ***Ανεξαρτησία από τη βασική λογική της εφαρμογής:*** δείχνει ποιος είναι ο βαθμός ανεξαρτησίας που επιτυγχάνεται μεταξύ της λογικής της υπηρεσίας και του πλαισίου χρήσης.

Στα πλαίσια της προτεινόμενης λύσης ο Πίνακας 8 παρουσιάζει το βαθμό στον οποίο ικανοποιούνται τα παραπάνω κριτήρια και τα αντίστοιχα χαρακτηριστικά.

Α/Α	**Κριτήριο/ Παράμετρος**	**Βαθμός Ικανοποίησης**	**Σχόλια**
Ομάδα Δ: Χαρακτηριστικά εφαρμογών με επίγνωση του πλαισίου χρήσης			
Δ.1	Ευελιξία	Μέτριος	Ο μηχανισμός προσαρμογής καλύπτει πολλές περιπτώσεις. Ωστόσο, δεν αφήνει στο μηχανικό λογισμικού πλήρη ελευθερία για οποιοδήποτε

			είδος προσαρμογής λόγω της απαίτησης διαχωρισμού του μηχανισμού από τις υπηρεσίες διαδικτύου.
Δ.2	Δυνατότητα μετατροπών	Υψηλός	Οι μετατροπές μπορούν να πραγματοποιηθούν σε επίπεδο μοντέλου χωρίς να επηρεάζεται ολόκληρη η εφαρμογή.
Δ.3	Ευκολία χρήσης μηχανισμού	Υψηλός	Ο μηχανικός λογισμικού δε χρειάζεται να γνωρίζει λεπτομέρειες για τη διαδικασία προσαρμογής στο πλαίσιο χρήσης. Η διαδικασία αυτοματοποιείται σε μεγάλο βαθμό.
Δ.4	Ενσωμάτωση σε υπάρχον σύστημα	Υψηλός	Ο μηχανισμός προσαρμογής τοποθετείται πάνω από το μεσισμικό που χρησιμοποιείται (υπηρεσίες διαδικτύου) χωρίς να επηρεάζει τη βασική λειτουργία των υπηρεσιών.
Δ.5	Είδος μοντελοποίησης πλαισίου χρήσης	UML μοντέλα	Με χρήση της γενικού σκοπού γλώσσας μοντελοποίησης UML διευκολύνεται η διαδικασία της ανάπτυξης λόγω της εξοικείωσης των μηχανικών και της ύπαρξης ποικίλων υποστηρικτικών εργαλείων μοντελοποίησης.
Δ.6	Ανεξαρτησία από τη βασική λογική της εφαρμογής	Υψηλός	Η διαδικασία προσαρμογής στο πλαίσιο χρήσης διατηρείται ανεξάρτητη από τη λογική της εφαρμογής και των υπηρεσιών διαδικτύου που την απαρτίζουν τόσο κατά την ανάπτυξη όσο και κατά την εκτέλεση της εφαρμογής.

Πίνακας 8. Αντιστοίχιση κριτηρίων για εφαρμογές με επίγνωση του πλαισίου χρήσης.

Όσο αφορά την αποτίμηση της μεθοδολογίας με ποσοτικές μετρικές πραγματοποιήθηκαν μετρήσεις για το χρόνο παραγωγής που απαιτείται για διάφορα επίπεδα πολυπλοκότητας μοντέλων (επεξεργαστής: Intel Core 2 Duo 1,74 GHz). Οι μετρήσεις αφορούν τον απόλυτο χρόνο παραγωγής (absolute time) σε σχέση με το πλήθος των στοιχείων που συμμετέχουν στο μοντέλο της εφαρμογής, αλλά και το σχετικό χρόνο (relative time), όπου λαμβάνεται επιπλέον υπόψη το πλήθος των αρχείων που παράγονται. Ο σχετικός χρόνος αποτελεί πιο ακριβή ένδειξη του χρόνου που απαιτείται ανά αρχείο.

Το πλήθος των στοιχείων προκύπτει από το σύνολο:

$$|\text{στοιχεία}| = |\text{υπηρεσίες\_διαδικτύου}| + |\text{στοιχεία\_πλαισίου\_χρήσης}| + |\text{εξαρτήσεις\_πλαισίου\_χρήσης}| + |\text{κλάσεις\_trigger}| + |\text{όψεις\_εφαρμογής}|$$

Ο απόλυτος χρόνος αναφέρεται στο συνολικό χρόνο παραγωγής των αρχείων της εφαρμογής, ενώ ο σχετικός χρόνος προκύπτει από το πηλίκο:

$$\text{σχετικός\_χρόνος} = \frac{\text{απόλυτος\_χρόνος}}{|\text{παραχθέντα\_αρχεία}|}$$

Τα αντίστοιχα διαγράμματα απεικονίζονται στις Εικόνες 54 και 55, όπου στον οριζόντιο άξονα εμφανίζεται το πλήθος των στοιχείων των μοντέλων και στον κατακόρυφο οι χρόνοι σε δευτερόλεπτα (secs). Από τα διαγράμματα παρατηρείται πως ο χρόνος παραγωγής κώδικα σαφώς και αυξάνεται ανάλογα με την πολυπλοκότητα του μοντέλου, καθώς χρειάζεται ανάλυση περισσότερων στοιχείων στο μοντέλο για τον εντοπισμό των εξαρτήσεων και την εφαρμογή των “οδηγών” μετατροπής. Ωστόσο, η αύξηση που παρατηρείται δεν είναι ιδιαίτερα σημαντική και ο χρόνος διατηρείται σε χαμηλά επίπεδα και σε αποδεκτά πλαίσια για το μηχανικό λογισμικού που χρησιμοποιεί τη μεθοδολογία (<30 secs για την παραγωγή ολόκληρης της εφαρμογής).

Αν και ο χρόνος παραγωγής κώδικα ενδέχεται να είναι μεγαλύτερος σε σύγκριση με εμπορικά εργαλεία μοντελοκεντρικής ανάπτυξης, δεν είναι εφικτό να πραγματοποιηθεί κάποια άμεση σύγκριση, καθώς δεν είναι διαθέσιμο κάποιο εργαλείο που να στοχεύει σε περιπτώσεις μοντελοκεντρικής ανάπτυξης εφαρμογών με τα χαρακτηριστικά που διαχειρίζεται η διατριβή (επίγνωση πλαισίου χρήσης, κτλ.).

Εικόνα 54. Απόλυτος χρόνος παραγωγής για διάφορες πολυπλοκότητες μοντέλων (πλήθος στοιχείων).

Εικόνα 55. Σχετικός χρόνος παραγωγής για διάφορες πολυπλοκότητες μοντέλων (πλήθος στοιχείων).

7.4 Βιβλιογραφία

[1] "An Evaluation of Compuware OptimalJ Professional Edition as an MDA tool", King's College London and University of York, white paper, September 2003.

[2] 3GPP, "Technical Specification Group Services and System Aspects; Generic AuthenticationArchitecture (GAA); Generic bootstrapping architecture", TS 33.220, 2007, Sophia-Antipolis.

[3] Calic, T., Dascalu, S. Egbert, D., "Tools for MDA Software Development: Evaluation Criteria and Set of Desirable Features", Proc. Fifth International Conference on Information Technology (ITNG '08), 2008, pp. 44-50.

[4] ContextWS-MDE, http://icbnet.ntua.gr/context-ws-mde.

[5] Janssen, J. W., "Evaluation of current tool support for the Model Driven Architecture", MS thesis, University of Twente, Enschede, Netherlands. January 2004.

[6] Kan, S. H., "Metrics and Models in Software Quality Engineering", 2nd Edition, Addison-Wesley Professional, 2002, ISBN-10: 0-201-72915-6.

[7] Kitchenham, B., “DESMET: A Method for Evaluating Software Engineering Methods and Tools”, Technical Report TR96-09, University of Keele, UK. 1996.

[8] Mohagheghi, P. and Aagedal, J., “Evaluating Quality in Model-Driven Engineering”, Proc. International Workshop on Modeling in Software Engineering (MISE '07), 2007, pp. 6-6.

[9] Tandon, G., Ghosh, S., “Using subject-oriented modeling to develop Jini applications”, Proc. Eighth IEEE International Enterprise Distributed Object Computing Conference (EDOC 2004), IEEE Computer Society Press, 2004, pp. 111-122.

8ο Κεφάλαιο

Εντοπισμός Υπηρεσιών Πλαισίου Χρήσης

Η επαναχρησιμοποίηση διαθέσιμων υπηρεσιών για την κατασκευή μιας νέας εφαρμογής διαδικτύου μπορεί να πραγματοποιηθεί τόσο σε επίπεδο υπηρεσιών εφαρμογών BWS που προσφέρουν συγκεγκριμένη λειτουργικότητα (π.χ. κράτηση αεροπορικών εισιτηρίων, ανάκτηση συναλλαγματικών ισοτιμιών), όπως παρουσιάστηκε στα προηγούμενα Κεφάλαια της διατριβής, αλλά και σε επίπεδο υπηρεσιών πλαισίου χρήσης CWS μέσω της επαναχρησιμοποίησης διαθέσιμων υπηρεσιών διαδικτύου που προσφέρουν πρόσβαση σε πηγές πλαισίου χρήσης. Υπό αυτό το πρίσμα ιδιαίτερο ενδιαφέρον παρουσιάζει η αντιστοίχιση (matching) των διαθέσιμων υπηρεσιών εφαρμογών με κατάλληλες υπηρεσίες πλαισίου χρήσης. Στο παρόν Κεφάλαιο παρουσιάζεται το τελευταίο κομμάτι της διατριβής που αφορά αφενός την κατάλληλη περιγραφή υπηρεσιών διαδικτύου και αφετέρου τη διαδικασία εύρεσης συμβατών υπηρεσιών πλαισίου χρήσης για συνδυασμό με υπηρεσίες εφαρμογών στα πλαίσια της προσαρμογής στο πλαίσιο χρήσης. Οι περιπτώσεις εντοπισμού των κατάλληλων υπηρεσιών πλαισίου χρήσης αφορούν τις τρεις περιπτώσεις προσαρμογής μηνυμάτων που παρουσιάστηκαν στο Κεφάλαιο 4 της διατριβής. Προκειμένου να πραγματοποιηθεί η διαδικασία εντοπισμού χρειάζεται να υπάρχουν κατάλληλες περιγραφές των χαρακτηριστικών των υπηρεσιών που συμμετέχουν στη διαδικασία, οι οποίες αποθηκεύονται σε κεντρικές ή κατανεμημένες αποθήκες υπηρεσιών (service repositories).

8.1 Σύντομη Επισκόπηση Βιβλιογραφίας Περιγραφής και Εύρεσης Υπηρεσιών

Στο σημείο αυτό πραγματοποιείται μια σύντομη επισκόπηση ερευνητικών λύσεων που σχετίζονται με την περιγραφή και την ανακάλυψη υπηρεσιών διαδικτύου. Οι προσεγγίσεις για την περιγραφή των υπηρεσιών επιχειρούν τον εμπλουτισμό μέσω πρόσθετων χαρακτηριστικών που δεν περιέχονται στα υπάρχοντα πρωτόκολλα των υπηρεσιών διαδικτύου. Από την άλλη πλευρά, η εύρεση υπηρεσιών αφορά είτε το κομμάτι της ανακάλυψης κατά την παροχή υπηρεσιών στον τελικό χρήστη είτε την εύρεση υπηρεσιών για την επαναχρησιμοποίησή τους στη δημιουργία νέων εφαρμογών. Σε αυτό το σκέλος της διατριβής επικεντρωνόμαστε στο δεύτερο κομμάτι εύρεσης εκ των δύο.

Οι προσεγγίσεις για την αποτελεσματική περιγραφή υπηρεσιών διαδικτύου σχετίζονται συνήθως με επεκτάσεις της WSDL γλώσσας ή της UDDI προδιαγραφής. Σε πολλές περιπτώσεις επιχειρείται η χρήση των τεχνολογιών και των γλωσσών του σημασιολογικού ιστού, ενώ σημασιολογικά εμπλουτισμένες περιγραφές υπηρεσιών διαδικτύου παρέχονται μέσω της ειδικής γλώσσας OWL-S [16]. Η συγχώνευση των WSDL και OWL προτείνεται στην Καθολική Γλώσσα Περιγραφής Υπηρεσιών (Universal Service Description Language / USDL) [1], όπου μέσω της WSDL περιγράφεται η συντακτική δομή, ενώ η OWL αναπαριστά τις έννοιες που περικλείει η υπηρεσία. Οι Σημασιολογικοί Υπομνηματισμοί για τη WSDL (Semantic Annotations for WSDL / SAWSDL) [17] αποτελούν έναν ακόμα μηχανισμό για την εισαγωγή σημασιολογικών στοιχείων στις υπηρεσίες διαδικτύου. Περιλαμβάνει τρόπους για την εισαγωγή υπομνηματισμών στα διάφορα τμήματα ενός WSDL αρχείου, όπως στις δομές των μηνυμάτων αιτήσεων και απαντήσεων, τις μεθόδους και τα τερματικά σημεία.

Στα πλαίσια των αποθηκών υπηρεσιών διαδικτύου η ανάγκη για σημασιολογικές αποθήκες μελετάται στο [13], όπου πραγματοποιείται η ανάλυση μητρώων υπηρεσιών (registries) διαθέσιμων στο διαδίκτυο και προτείνεται η ενσωμάτωση σημασιολογικών στοιχείων. Μια άλλη ενδιαφέρουσα προσέγγιση προτείνεται στην Προδιαγραφή Επαναχρησιμοποιήσιμων Πόρων (Reusable Asset Specification / RAS) [9] της OMG, επέκταση της οποίας προτείνεται στα πλαίσια της διατριβής. Μια μελέτη περίπτωσης για το RAS έχει πραγματοποιηθεί από την ομάδα επαναχρησιμοποίησης λογισμικού συστημάτων δεδομένων της NASA (NASA Earth Science Data Systems Software Reuse Working Group), η οποία υλοποίησε μια διαδικτυακή πύλη με επαναχρησιμοποιήσιμους πόρους [4]. Μια επέκταση της προδιαγραφής για την πλήρη επαναχρησιμοποίηση λογισμικού αναφερόμενη

ως Συστατικό Πολλαπλών Φάσεων (Multiphased Component / MPC) παρουσιάζεται στο [11], όπου έχει εφαρμοστεί για την υλοποίηση ενός στρατιωτικού εφοδιαστικού συστήματος.

Στο πεδίο της ανακάλυψης υπηρεσιών πολλές προσεγγίσεις σχετίζονται με την αναζήτηση συγκεκριμένων λέξεων-κλειδιών στις εγγραφές των UDDI καταλόγων ή με την εξαγωγή πληροφορίας μέσω της υιοθέτησης αλγορίθμων από τη γραμμική άλγεβρα για την ανάκτηση πληροφοριών (π.χ. [12] [14]). Αναφορικά με WSDL περιγραφές υπηρεσιών έχουν προταθεί διάφορες τεχνικές που σχετίζονται με συγκρίσεις κειμένων ή με πιο ειδικές διαδικασίες (π.χ. [3]). Μια ενδιαφέρουσα προσέγγιση ανάκτησης που αποτελείται από διάφορα στάδια καταλήγοντας στη δομική αντιστοίχιση WSDL προδιαγραφών προτείνεται στο [18]. Οι τύποι δεδομένων, τα μηνύματα, οι μέθοδοι και οι υπηρεσίες συγκρίνονται διαδοχικά μέσω διαφόρων αλγορίθμων και τα τελικά αποτελέσματα ταξινομούνται βάσει της γενικής βαθμολογίας που συγκεντρώνεται. Ένα ακόμα σύστημα κανόνων για την εύρεση υπηρεσιών βάσει των επιθυμητών ιδιοτήτων που εισάγονται από το χρήστη παρουσιάζεται στο [23].

Κάποιες από τις παραπάνω λύσεις βασίζονται στην ύπαρξη UDDI καταλόγων. Ωστόσο, λόγω της περιορισμένης χρήσης της UDDI προδιαγραφής, η οποία ταξινομεί τις υπηρεσίες σε κατηγορίες, στη διατριβή επικεντρωνόμαστε σε μηχανισμούς που στοχεύουν στις WSDL περιγραφές και οι οποίοι προσφέρουν μεγαλύτερη ευελιξία από την απλή αναζήτηση καταλόγων, που μπορεί να αποδειχθεί ανεπαρκής. Οι προσεγγίσεις που αναφέρθηκαν παραπάνω είναι κατάλληλες για περιπτώσεις, όπου οι περιγραφές των υπηρεσιών συγκρίνονται έναντι συγκεκριμένων παραμέτρων ή όπου τα συμβατά μέρη δύο περιγραφών συγκρίνονται για την εξακρίβωση της μεταξύ τους ομοιότητας. Για την περίπτωση εύρεσης πηγών πλαισίου χρήσης χρειάζεται μια διαφορετική προσέγγιση σύγκρισης, την οποία επιχειρεί να καλύψει η λύση της διατριβής.

8.2 Αποθήκη Υπηρεσιών Διαδικτύου

Μέσω του διαδικτύου οι μηχανικοί λογισμικού έχουν τη δυνατότητα εύρεσης υπηρεσιών διαδικτύου από τα διάφορα μητρώα υπηρεσιών, τους καταλόγους των παρόχων, κτλ. Ωστόσο, σε περιπτώσεις ανακάλυψης υπηρεσιών, πέρα από τις πληροφορίες που σχετίζονται με τη λειτουργικότητα της υπηρεσίας, είναι χρήσιμη η ανάκτηση μη λειτουργικών πληροφοριών, όπως είναι η τοπική διαθεσιμότητα της υπηρεσίας, πιθανοί περιορισμοί σε πρωτόκολλα επικοινωνίας, στα χαρακτηριστικά της συσκευής του χρήστη, κτλ. Βάσει των παραπάνω προτείνεται στα πλαίσια της διατριβής η δομή μιας αποθήκης

υπηρεσιών που επεκτείνει τη RAS προδιαγραφή: το σημασιολογικό Προφίλ Υπηρεσιών Διαδικτύου (semantic Web Service Profile / sWSP). Το προφίλ στοχεύει στην ενσωμάτωση διαφόρων σημασιολογικών στοιχείων περιγραφής που σχετίζονται με την ανάπτυξη και την παροχή υπηρεσιών και χρησιμοποιείται στη διαδικασία εντοπισμού πηγών πλαισίου χρήσης. Εναλλακτικα, η διαδικασία εύρεσης μπορεί να εφαρμοσθεί μόνο στα WSDL αρχεία, όταν δεν υπάρχει κάποια άλλη διαθέσιμη πληροφορία, προσδίδοντας με αυτόν τον τρόπο περισσότερη ευελιξία στην προτεινόμενη λύση.

8.2.1 Reusable Asset Specification

Σύμφωνα με τον ορισμό που δίνεται από την OMG, πόρος (asset) *"είναι μια συλλογή τεχνημάτων (artifact) που παρέχουν λύση σε ειδικές ομάδες προβλημάτων"*. Η προδιαγραφή επαναχρησιμοποιήσιμων πόρων της OMG αποτελεί μια πλήρη λύση που επιτρέπει το συνδυασμό διαφόρων τύπων αρχείων για την περιγραφή μιας υπηρεσίας. Οι πόροι απαρτίζονται από πολλαπλά τεχνήματα που σχετίζονται με διαφορετικές πτυχές της περιγραφής. Τα τεχνήματα είναι αρχεία διαφόρων ειδών (π.χ. XML αρχεία, αρχεία κειμένου, UML διαγράμματα, εκτελέσιμος κώδικας, κτλ.). Μέσω του προτυποποιημένου τρόπου οργάνωσης και περιγραφής των διαφορετικών τεχνημάτων μπορεί να επιτευχθεί μια αποτελεσματική διαχείριση των επαναχρησιμοποιήσιμων πόρων.

Τα διάφορα τεχνήματα τοποθετούνται μαζί με το αρχείο manifest για να δημιουργήσουν έναν ολοκληρωμένο πόρο. Το αρχείο manifest περιλαμβάνει τα μεταδεδομένα της υπηρεσίας σε XML μορφοποίηση (αναγνωριστικό υπηρεσίας, όνομα και έκδοση, ονόματα τεχνημάτων, κτλ.). Απαιτούνται τουλάχιστον ένα αρχείο manifest και ένα τέχνημα για τη δημιουργία ενός πόρου. Η OMG παρέχει ένα βασικό προφίλ (Default Profile) για την προδιαγραφή, το οποίο περιλαμβάνει τα βασικά στοιχεία που ορίζονται, ενώ επιτρέπεται η εισαγωγή νέων προφίλ που σχετίζονται με ειδικούς περιορισμούς ως επεκτάσεις του βασικού προφίλ. Διαθέσιμο είναι ακόμα ένα βασικό προφίλ συστατικών (Default Component Profile) και ένα βασικό προφίλ υπηρεσιών διαδικτύου (Default Web Service Profile).

Τα κυριότερα στοιχεία που περιέχονται στο βασικό προφίλ της RAS προδιαγραφής παρουσιάζονται στην Εικόνα 56 και είναι τα εξής:

- ***Πόρος (asset):*** αποτελεί το πρώτο στοιχείο στο αρχείο manifest και σχετίζεται με κλάσεις που παρέχουν περισσότερες πληροφορίες για τον πόρο.
- ***Ταξινόμηση (classification):*** χρησιμοποιείται για την ταξινόμηση των πόρων και υποστηρίζει τη χρήση διάφορων όψεων για τους πόρους. Η ταξινόμηση

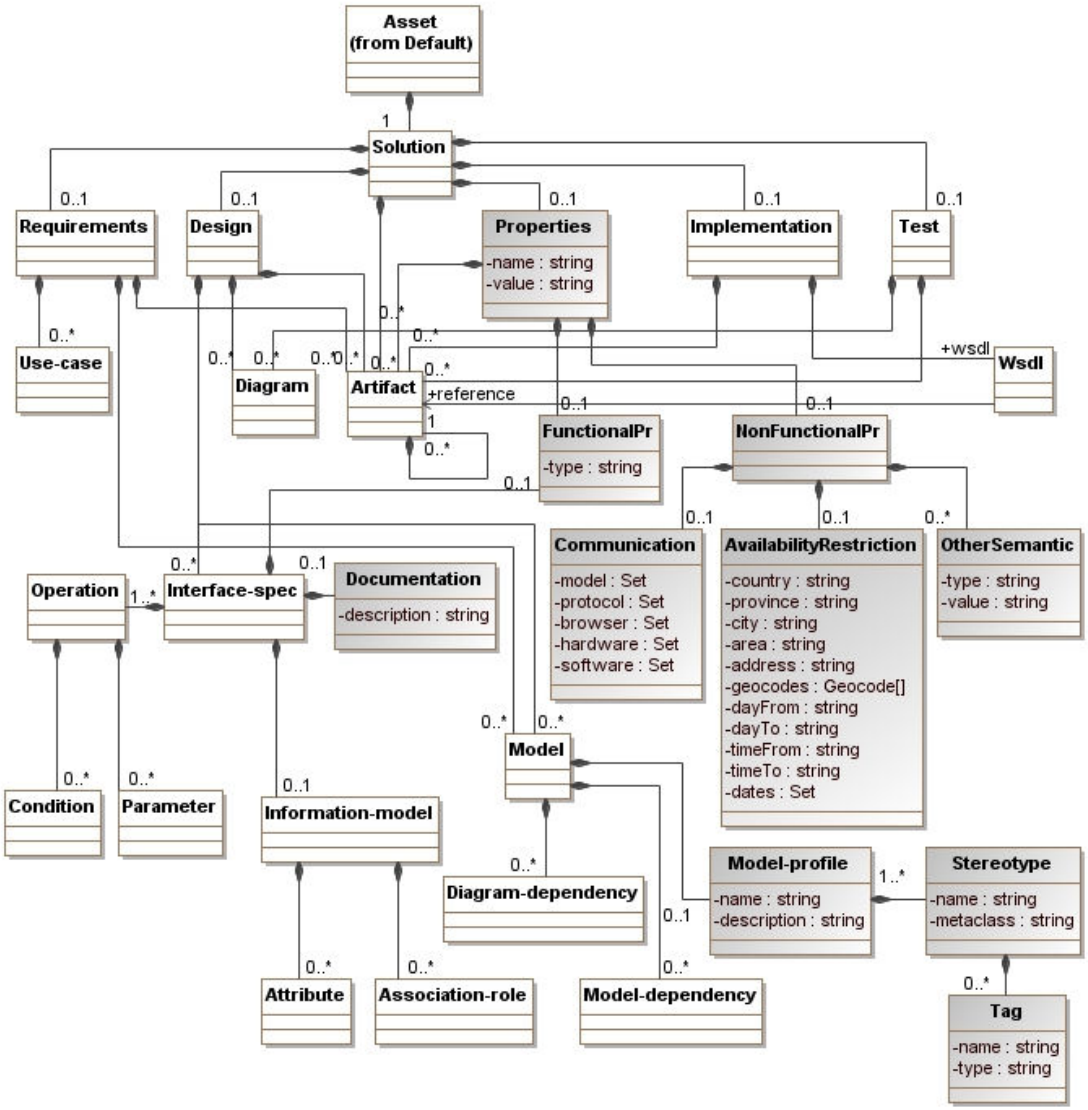
συνδυάζεται με το στοιχείο πλαίσιο (*context*) που ορίζει έννοιες για την κατανόηση της σημασίας των στοιχείων του πόρου.

- ***Λύση (solution):*** περιλαμβάνει τη συλλογή των τεχνημάτων που αποτελούν τον πόρο.
- ***Χρήση (usage):*** παρέχει καθοδήγηση για τη χρήση και την προσαρμογή των πόρων.
- ***Σχετικοί πόροι (related assets):*** δείχνει συσχετίσεις με άλλους πόρους.
- ***Προφίλ (profile):*** περιγράφει τον τύπο του πόρου. Όπως προαναφέρθηκε, τα προφίλ προέρχονται από άλλα προφίλ μέσω της προσθήκης στοιχείων και ιδιοτήτων στο βασικό προφίλ. Ωστόσο, δε μπορούν να καταργηθούν τα στοιχεία ή οι ιδιότητες του αρχικού προφίλ.

Εικόνα 56. Βασικά στοιχεία της RAS προδιαγραφής.

8.2.2 Προτεινόμενη Επέκταση

Προκειμένου να εισαχθούν στην περιγραφή των υπηρεσιών διαδικτύου όλες οι απαραίτητες πληροφορίες προτείνεται μια επέκταση της RAS προδιαγραφής. Μια παραλλαγή του προτεινόμενου προφίλ για την περιγραφή μιας διαφορετικής ομάδας υπηρεσιών παρουσιάζεται στο [6]. Το προτεινόμενο σημασιολογικό Προφίλ Υπηρεσιών Διαδικτύου αποτελεί μια επέκταση του βασικού προφίλ υπηρεσιών διαδικτύου της OMG, καθώς περιλαμβάνει ιδιότητες των υπηρεσιών διαδικτύου που δε διατίθενται στην περιγραφή του αρχικού προφίλ. Τα στοιχεία του προφίλ sWSP παρουσιάζονται στο διάγραμμα κλάσεων της Εικόνας 57, όπου οι βασικές προσθήκες εμφανίζονται με γκρι χρώμα.



Εικόνα 57. Διάγραμμα κλάσεων της προτεινόμενης επέκτασης του βασικού προφίλ υπηρεσιών διαδικτύου για το sWSP.

Οι ιδιότητες της υπηρεσίας διαδικτύου είναι απαραίτητες για τη σωστή της λειτουργία και χρήση και στενά συνυφασμένες με τη σχεδίαση και υλοποίησή της. Γενικά, μπορούν να διακριθούν σε λειτουργικές και μη λειτουργικές ιδιότητες. Οι λειτουργικές ιδιότητες (κλάση `FunctionalPr`) αναφέρονται σε χαρακτηριστικά διαμόρφωσης που σχετίζονται με την εκτέλεση της υπηρεσίας, όπως περιέχονται συνήθως στο αρχείο της WSDL περιγραφής. Από την άλλη πλευρά, οι μη λειτουργικές ιδιότητες (non-functional / NF) αναφέρονται κυρίως σε περιορισμούς επί των λειτουργιών της υπηρεσίας, όπως είναι η διάθεση της σε συγκεκριμένες τοποθεσίες ή για συγκεκριμένη χρονική διάρκεια, χαρακτηριστικά ποιότητας υπηρεσίας, κτλ. Αν και πληροφορίες αυτού του είδους περιέχονται εν μέρει στα τμήματα *Context* και *Classification* του βασικού προφίλ, η κλάση `NonFunctionalPr` προστέθηκε στο προφίλ για την παροχή πιο αναλυτικών περιγραφών.

Μάλιστα, οι μη λειτουργικές ιδιότητες έχουν ιδιαίτερη σημασία, καθώς τα συμπεράσματα που μπορούν να εξαχθούν από την ανάλυσή τους μειώνουν το σύνολο των συμβατών υπηρεσιών που ανακτώνται μέσω διαδικασιών εύρεσης υπηρεσιών.

Συγκεκριμένα, τα στοιχεία των μη λειτουργικών ιδιοτήτων των υπηρεσιών διαδικτύου περιλαμβάνουν πληροφορίες για:

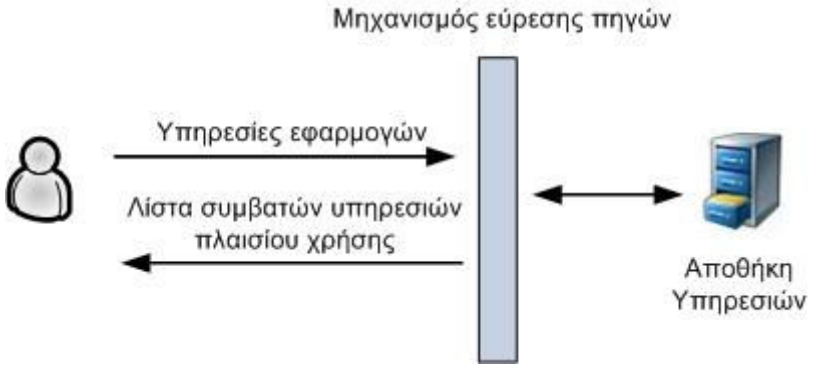
- ***Διαθεσιμότητα υπηρεσίας (κλάση `AvailabilityRestriction`):*** χωρική διαθεσιμότητα (π.χ. μια υπηρεσία μετεωρολογικών προβλέψεων μπορεί να είναι διαθέσιμη μόνο σε μια συγκεκριμένη χώρα) ή ακόμα και περιορισμοί χρονικής διαθεσιμότητας της υπηρεσίας (π.χ. για συγκεκριμένες ώρες της ημέρας).
- ***Απαιτήσεις επικοινωνίας (κλάση `Communication`):*** περιορισμοί στα πρωτόκολλα επικοινωνίας (π.χ. έκδοση SOAP πρωτοκόλλου), χαρακτηριστικά υλικού, λογισμικού και δικτύου που σχετίζονται με τη συσκευή του τελικού χρήστη, κτλ.
- ***Άλλες πληροφορίες (κλάση `OtherSemantic`):*** εδώ περιλαμβάνονται διάφορα άλλη είδη πληροφοριών. Παραδείγματα αποτελούν οι εγγυήσεις ασφάλειας, καθώς και τα πρωτόκολλα ασφάλειας και προστασίας προσωπικών δεδομένων που υποστηρίζονται, η κρυπτογράφηση, χαρακτηριστικά ποιότητας, κτλ.

Τα παραπάνω στοιχεία μπορούν να εισαχθούν είτε ως ανεξάρτητα τεχνήματα είτε ως χωριστά τμήματα σε ένα κοινό τέχνημα. Για λόγους καλύτερης διαχείρισης των ιδιοτήτων επιλέχθηκε η δεύτερη λύση μέσω της κατασκευής ενός κατάλληλου XML σχήματος. Το βασικό τμήμα του σχήματος παρουσιάζεται στο Παράρτημα Β.

Τα στοιχεία που σχετίζονται με το μοντέλο της υπηρεσίας, δηλ. οι κλάσεις προφίλ μοντέλου (`Model-profile`), στερεοτύπου (`Stereotype`) και ετικέτας (`Tag`), αναφέρονται σε πληροφορίες των UML μεταμοντέλων ή προφίλ που ακολουθούνται για τη σχεδίαση της υπηρεσίας. Τα προφίλ μπορούν να συμπίπτουν με τα UML προφίλ που προτείνονται στη διατριβή ή να ακολουθούν τη δομή άλλων προφίλ.

8.3 Διαδικασία Εύρεσης Υπηρεσιών Πλαισίου Χρήσης

Η διαδικασία εύρεσης πηγών πλαισίου χρήσης περιλαμβάνει τη σύγκριση των αντίστοιχων περιγραφών των υπηρεσιών διαδικτύου. Στόχος είναι ο εντοπισμός υπηρεσιών που μπορούν να συνδυαστούν με τις αντίστοιχες περιγραφές των υπηρεσιών εφαρμογών και να επαναχρησιμοποιηθούν στη διαδικασία ανάπτυξης εφαρμογών (Εικόνα 58). Σε αυτή την ενότητα παρουσιάζονται οι αρχές εντοπισμού και η αντίστοιχη διαδικασία.



Εικόνα 58. Διαδικασία εύρεσης πηγών πλαισίου χρήσης.

8.3.1 Περιπτώσεις Αντιστοίχισης

Δεδομένης μιας αίτησης *R* με ιδιότητες (*r1*, *r2*, *r3*, ...) *Є R* και μιας προσφοράς *O* με ιδιότητες (*o1*, *o*2, *o3*, …) *Є O* η βαθμολογία (score) που σχετίζεται με τη σημασιολογική ομοιότητα των δύο εννοιών υπολογίζεται μέσω μιας κατάλληλης συνάρτησης *sem_score(R, O)* που αναλύει τα στοιχεία της αίτησης και της προσφοράς και αποτιμά τις μεταξύ τους συσχετίσεις. Στην περίπτωση της σύγκρισης για την εύρεση πηγών πλαισίου χρήσης, η αίτηση αφορά την περιγραφή της υπηρεσίας εφαρμογών, ενώ η προσφορά την αντίστοιχη περιγραφή της υπηρεσίας πλαισίου χρήσης. Η συνάρτηση αποτιμάται βάσει των περιπτώσεων προσαρμογής υπηρεσιών διαδικτύων που έχουν παρουσιαστεί. Ωστόσο, η τρίτη περίπτωση της τροποποίησης απάντησης δε συμπεριλαμβάνεται στις περιπτώσεις σύγκρισης, καθώς η διαθέσιμη πληροφορία στις περιγραφές των υπηρεσιών δεν επαρκεί για την εξαγωγή συμπερασμάτων που αφορούν τους πιθανούς τρόπους αλλαγής των μηνυμάτων απαντήσεων.

Συγκεκριμένα η σύγκριση έγκειται στα εξής:

- ***Περίπτωση 1 (αντικατάσταση παραμέτρου):*** στην περίπτωση αυτή οι παράμετροι των μηνυμάτων αιτήσεων των υπηρεσιών εφαρμογών θα πρέπει να ταιριάζουν με τις παραμέτρους που επιστρέφονται από τις μεθόδους των υπηρεσιών πλαισίου χρήσης, τόσο σε επίπεδο ονομάτων όσο και σε επίπεδο τύπων. Σε ορισμένες περιπτώσεις το όνομα της μεθόδου εκφράζει το είδος της πληροφορίας που επιστρέφει η μέθοδος (π.χ. `getCityInformation()`). Για το λόγο αυτό πραγματοποιείται και ο έλεγχος του ονόματος της μεθόδου πλαισίου χρήσης σε σχέση με τα ονόματα των παραμέτρων της μεθόδου της BWS. Το συγκεκριμένο είδος ελέγχου για την πρώτη περίπτωση εκφράζεται μέσω του παρακάτω αλγορίθμου που εκτελείται για όλες τις διαθέσιμες μεθόδους της BWS και για όλα

τα τερματικά σημεία διατηρώντας τη μεγαλύτερη τιμή ως τελική βαθμολογία για την υπηρεσία:

```
foreach (BWS_operation in BWS_descr)
 foreach (BWS_input_param in BWS_operation)
  foreach (CWS_descr in REPOSITORY)
   foreach (CWS_operation in CWS_descr)
    sc1 = sem_score(BWS_input_param.NAME, CWS_operation.NAME);
    foreach (CWS_output_param in CWS_operation)
sc2=sem_score(BWS_input_param.NAME+TYPE, CWS_output_param.NAME+TYPE);
    end
    sem_score = sc1/2 + sc2/2;
   end
  end
 end
end
```

- ***Περίπτωση 2 (επιλογή μεθόδου):*** αυτή η περίπτωση είναι εφαρμόσιμη σε υπηρεσίες που περιλαμβάνουν μεθόδους που υλοποιούν την ίδια λειτουργικότητα με διαφορετικούς τρόπους. Κατ' επέκταση πραγματοποιείται αρχικά ο υπολογισμός της αντίστοιχης πιθανότητας προσαρμογής βάσει επιλογής μεθόδου μέσω της σύγκρισης των προσφερόμενων μεθόδων σε επίπεδο ονομάτων και τύπων μηνύματων απαντήσεων. Αν προκύψει κάποια ομοιότητα μεταξύ των μεθόδων της υπηρεσίας, τα ονόματα των μεθόδων της BWS συγκρίνονται με τις μεθόδους της CWS (σε σχέση με τα ονόματα των μεθόδων και των παραμέτρων που επιστρέφονται). Για αυτή την περίπτωση προσαρμογής και για όλες τις υπηρεσίες που πληρούν τις παραπάνω προϋποθέσεις χρησιμοποιείται ο παρακάτω αλγόριθμος, ενώ ως τελική βαθμολογία διατηρείται η μεγαλύτερη τιμή από αυτές που προκύπτουν για κάθε μέθοδο της υπηρεσίας:

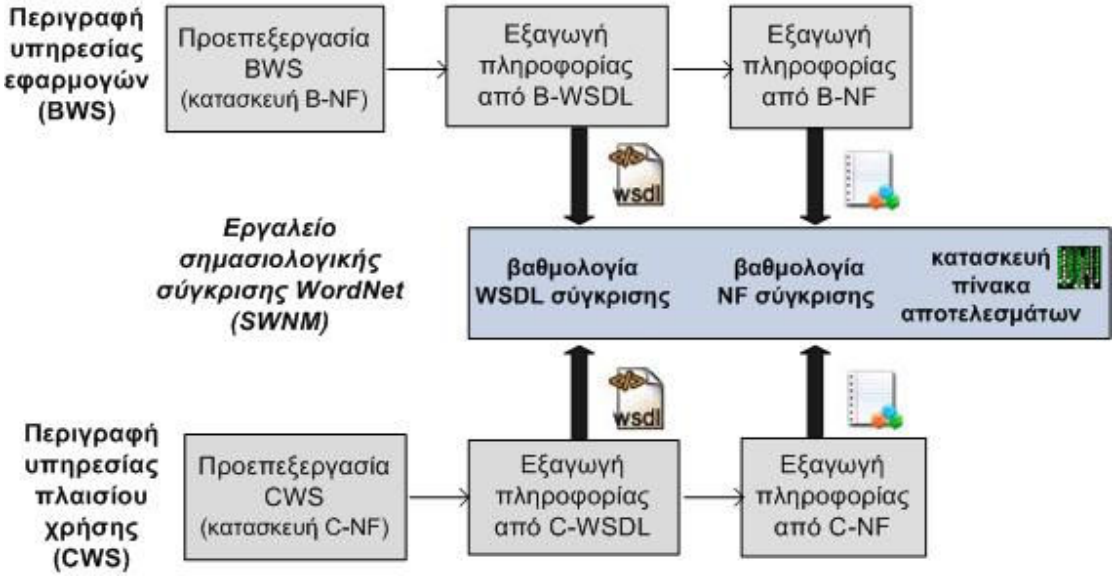
```
foreach (BWS_matching_operation in BWS_descr)
 foreach (CWS_descr in REPOSITORY)
  foreach (CWS_operation in CWS_descr)
  sc1 = sem_score(BWS_matching_operation.NAME, CWS_operation.NAME);
  foreach (CWS_output_param in CWS_operation)
  sc2=sem_score(BWS_matching_operation.NAME, CWS_output_param.NAME);
  end
  sem_score = sc1/2 + sc2/2;
  end
 end
end
```

8.3.2 Βήματα Διαδικασίας

Η διαδικασία εντοπισμού μπορεί να εφαρμοστεί είτε για μια συγκεκριμένη υπηρεσία εφαρμογών είτε για όλες τις διαθέσιμες περιγραφές της αποθήκης. Ολοκληρώνεται σε δύο στάδια: σύγκριση σε επίπεδο διεπαφής για τα WSDL αρχεία και σύγκριση σε επίπεδο NF αρχείων, τα οποία κατασκευάζονται κατά την εισαγωγή των υπηρεσιών στην αποθήκη από

τη διαθέσιμη για την υπηρεσία τεκμηρίωση. Τα βήματα παρουσιάζονται στην Εικόνα 59 και περιγράφονται αναλυτικά στη συνέχεια:

1. ***Κατασκευή NF αρχείου:*** το NF αρχείο με τις μη λειτουργικές ιδιότητες της υπηρεσίας κατασκευάζεται ακολουθώντας τις αρχές του προφίλ sWSP.
2. ***WSDL σύγκριση:*** πραγματοποιείται η σύγκριση των WSDL αρχείων βάσει των περιπτώσεων προσαρμογής. Αποτελεί το πιο σημαντικό βήμα της διαδικασίας, καθώς αφορά τη σύγκριση των διεπαφών των υπηρεσιών.
3. ***NF σύγκριση:*** αποτιμάται η συμβατότητα των μη λειτουργικών ιδιοτήτων των υπηρεσιών.



Εικόνα 59. Στάδια διαδικασίας εύρεσης πηγών πλαισίου χρήσης.

Οι βαθμολογίες που υπολογίζονται σε κάθε στάδιο εισάγονται σε έναν πίνακα αποτελεσμάτων. Υψηλότερες βαθμολογίες αποτελούν δείκτη μεγαλύτερης πιθανότητας συμβατότητας για τις υπηρεσίες, ενώ αναφέρεται και το αντίστοιχο είδος προσαρμογής στο πλαίσιο χρήσης. Για να διατηρηθεί κάποια υπηρεσία πλαισίου χρήσης ως συμβατή με την υπηρεσία εφαρμογών χρειάζεται να συγκεντρώσει συνολική βαθμολογία σύγκρισης πάνω από το 30% της μέγιστης δυνατής βαθμολογίας. Σε αντίθετη περίπτωση η υπηρεσία απορρίπτεται.

8.3.2.1 Κατασκευή NF αρχείου

Σε αυτό το στάδιο κατασκευάζεται το NF αρχείο που περιλαμβάνει τις μη λειτουργικές ιδιότητες της υπηρεσίας. Η δομή που προκύπτει για το αρχείο είναι προκαθορισμένη από το προφίλ sWSP και η διαδικασία κατασκευής πραγματοποιείται κατά την εισαγωγή μιας νέας υπηρεσίας στην αποθήκη.

Το αρχείο περιλαμβάνει πληροφορίες που έχουν εξαχθεί από το WSDL αρχείο και από τις οδηγίες χρήσης της υπηρεσίας και σχετίζονται με τις έννοιες στις οποίες εστιάζει η υπηρεσία (π.χ. αθλητικά προϊόντα, καιρός, κτλ.), τους περιορισμούς της χωρικής της διαθεσιμότητας (π.χ. παντού, Ηνωμένες Πολιτειες της Αμερικής / Η.Π.Α., Ρώμη, κτλ.) και οποιαδήποτε άλλη πληροφορία που είναι διαθέσιμη από τον πάροχο της υπηρεσίας. Σκοπός είναι να εξαχθούν πληροφορίες από την τεκμηρίωση και τα λειτουργικά χαρακτηριστικά της υπηρεσίας, αντί να γίνει χρήση διαθέσιμων σημασιολογικών πληροφοριών.

Τμήμα ενός NF αρχείου παρουσιάζεται στην Εικόνα 60 για την υπηρεσία διαδικτύου `MovieInformation`. Η τεκμηρίωση και η περιγραφή της διεπαφής της υπηρεσίας αναλύονται για την εξαγωγή πληροφοριών που εισάγονται ως ετικέτες στο αρχείο με κλειδί (key) “focus”. Παρόμοια διαδικασία ακολουθείται για την εισαγωγή πληροφοριών σχετικών με τους περιορισμούς διαθεσιμότητας της υπηρεσίας και τα πρωτόκολλα επικοινωνίας.

```
<wsnf:descriptor xmlns:wsnf="icbnet.contextws/WSNF">
 <wsnf:semantics>
. . .
    <wsnf:tag key="focus">zip code</wsnf:tag>
    <wsnf:tag key="focus">radius</wsnf:tag>
    <wsnf:tag key="focus">theaters</wsnf:tag>
    <wsnf:tag key="focus">area</wsnf:tag>
    <wsnf:tag key="focus">movie</wsnf:tag>
 </wsnf:semantics>
 <wsnf:configuration> . . .
  <wsnf:network>
    <wsnf:binding>SOAP</wsnf:binding>
    <wsnf:communicationProtocol>HttpGet</wsnf:communicationProtocol>
    <wsnf:communicationProtocol>HttpPost</wsnf:communicationProtocol>
     <wsnf:communicationProtocol>Soap12</wsnf:communicationProtocol>
     <wsnf:communicationProtocol>Soap</wsnf:communicationProtocol>
  </wsnf:network>
 </wsnf:configuration>
 <wsnf:locationAvailability>
   <wsnf:hasLocationRestriction>true</wsnf:hasLocationRestriction>
   <wsnf:country>USA</wsnf:country>
 </wsnf:locationAvailability>
. . .
</wsnf:descriptor>
```

Εικόνα 60. Ετικέτες και περιορισμοί διαθεσιμότητας για την υπηρεσία MovieInformation.

8.3.2.2 WSDL σύγκριση

Η συμβατότητα σε επίπεδο WSDL αρχείων αποτιμάται μέσω της σύγκρισης των κατάλληλων ζευγών μέθοδος-μέθοδος, μέθοδος-παράμετρος, τύπος-τύπος, κτλ. από τις αντίστοιχες προδιαγραφές όπως έχει οριστεί για κάθε περίπτωση προσαρμογής στο

πλαίσιο χρήσης. Η σύγκριση παραμέτρων μπορεί να αναφέρεται είτε σε σύγκριση ονομάτων είτε σε σύγκριση τύπων. Η βαθμολογία που προκύπτει από τη σύγκριση των ονομάτων λαμβάνει μεγαλύτερο βάρος στον υπολογισμό της συνολικής βαθμολογίας, καθώς δύο σημασιολογικά όμοια ονόματα είναι πιο σημαντικά από τους αντίστοιχους όμοιους τύπους. Συγκεκριμένα, κατά τη σύγκριση ονομάτων χρησιμοποιείται βάρος 1,0 στον υπολογισμό της βαθμολογίας, ενώ για τους τύπους 0,5.

Η σύγκριση τύπων πραγματοποιείται βάσει ομάδων τύπων που χρησιμοποιούνται για τους βασικούς, αλλά και τους πολύπλοκους τύπους δεδομένων (π.χ. λίστες, πίνακες, κτλ.). Για ειδικούς τύπους που ορίζονται στο τμήμα του XML σχήματος των WSDL αρχείων (π.χ. `IPInformation`) ο τύπος διασπάται στα συστατικά του στοιχεία, τα οποία και συμμετέχουν στη διαδικασία της σύγκρισης, εκτός αν βρεθεί κάποια περίπτωση όπου τα ονόματα και τα namespaces των πολύπλοκων τύπων συμπίπτουν. Επιπλέον, υπάρχουν περιπτώσεις όπου δεν αναφέρεται κάποια συγκεκριμένη δομή για τους τύπους των παραμέτρων των μεθόδων. Σε αυτές τις περιπτώσεις χρησιμοποιούνται συνήθως λέξεις όπως “Any” ή “AnyType” στην αντίστοιχη WSDL προδιαγραφή. Όταν βρεθεί κάποια τέτοια λέξη εκχωρείται μια ακόμα χαμηλότερη βαθμολογία (1/4 της τιμής που δίνεται στους παρόμοιους τύπους δεδομένων).

Για τη σύγκριση των ονομάτων των μεθόδων και των παραμέτρων ο αλγόριθμος σύγκρισης εκτελείται για κάθε λέξη που έχει σημασιολογικό ενδιαφέρον. Λέξεις που χρησιμοποιούνται συχνά, αλλά δεν περιέχουν κάποια σημαντική πληροφορία για την υπηρεσία (π.χ. get, list, information, κτλ.) αγνοούνται και δε συμμετέχουν στη διαδικασία (αναφέρονται ως “stop words”). Για να καθοριστεί ποιες λέξεις έχει νόημα να συμμετάσχουν στη σύγκριση πραγματοποιήθηκε μια μελέτη σε μητρώα περιγραφών υπηρεσιών που είναι διαθέσιμα στο διαδίκτυο. Για το σκοπό αυτό χρησιμοποιήθηκε ένα δείγμα 110 WSDL περιγραφών υπηρεσιών που αποτελεί σημαντικό ποσοστό (~17%) επί του συνόλου των διαθέσιμων περιγραφών: 640 για το 2005 χωρίς διπλές εγγραφές ή μη έγκυρους υπερσυνδέσμους, όπως αναφέρεται στο [2]. Το δείγμα λήφθηκε από διάφορα μητρώα, όπως τα XMethods [22], WebServiceX [20] και Web Service List [19], καθώς και από τα αποτελέσματα τυχαίας αναζήτησης στο διαδίκτυο για περιγραφές υπηρεσιών διαδικτύου. Ωστόσο, δεν παρείχαν όλα τα μητρώα έγκυρους υπερσυνδέσμους σε περιγραφές υπηρεσιών διαδικτύου. Αντί αυτού οδηγούσαν στη σελίδα εκκίνησης του παρόχου της υπηρεσίας ή σε μη έγκυρους συνδέσμους. Αυτές οι υπηρεσίες απορρίφθηκαν κατά τη διαδικασία συλλογής περιγραφών.

Οι συχνότητες των βασικότερων λέξεων που εμφανίζονται στις περιγραφές συνοψίζονται στον Πίνακα 9. Παραδείγματα λιγότερα συχνά εμφανιζόμενων λέξεων που επίσης δε φαίνεται να παρουσιάζουν σημασιολογική αξία για την υπηρεσία είναι: and, for, set, in, data, add, details, do, return, search, object, κτλ. Σημαντική είναι η γλώσσα που εμφανίζεται σε κάθε περιγραφή. Στα πλαίσια της διατριβής μελετήθηκαν κυρίως υπηρεσίες διαδικτύου με λέξεις στην αγγλική γλώσσα. Επιπλέον, πληροφορίες που σχετίζονται με μηχανισμούς επικοινωνίας (π.χ. SoapIn, SoapOut, HttpGetIn, HttpGetOut, κτλ.) δε λήφθηκαν υπόψη στη διαδικασία. Παρατηρήθηκε, τέλος, πως η εμφάνιση συγκεκριμένων λέξεων σχετίζεται και με το περιβάλλον που χρησιμοποιείται για τη δημιουργία των υπηρεσιών διαδικτύου (π.χ. οι λέξεις response και result εμφανίζονται πάντα σε υπηρεσίες που έχουν υλοποιηθεί σε περιβάλλον MS.NET).

Λέξη	Συχνότητα Εμφάνισης
response	90
result	73
get	52
array	33
type	25
of	22
name	20
by	20
request	18
information / info	16
list / lists / listing	14
str / string	11
number	10
data	10
to	7
text	7
description	5
all	5
item	4
is	4
from	4

Πίνακας 9. Συχνότητες συχνά εμφανιζόμενων λέξεων σε WSDL αρχεία.

Κατά τη διαδικασία συλλογής περιγραφών υπηρεσιών εξάχθηκαν οι σημασιολογικά χρήσιμες – για λόγους προσαρμογής στο πλαίσιο χρήσης – λέξεις τόσο για τις υπηρεσίες εφαρμογών όσο και για τις υπηρεσίες πλαισίου χρήσης, ενώ οι μη σχετικές λέξεις απομακρύνθηκαν. Για τον υπολογισμό της σημασιολογικής βαθμολογίας των λέξεων ο αλγόριθμος σύγκρισης χρησιμοποιεί μια διεπαφή σε Java [5] για τη γλωσσική βάση δεδομένων της αγγλικής γλώσσας WordNet [21]. Οι σημασιολογικές λεξικές βάσεις δεδομένων αποτελούν ένα ενδιαφέρον πεδίο που έχει επηρεαστεί από το γενικό πεδίο της γλωσσολογίας. Το WordNet αποτελεί ένα ηλεκτρονικό λεξικό βασισμένο στις έννοιες των λέξεων και τις μεταξύ τους σημασιολογικές σχέσεις. Με τη βοήθεια των σημασιολογικών σχέσεων, οι έννοιες των λέξεων σε μια γλώσσα μπορούν να διασυνδεθούν, κατασκευάζοντας ένα δίκτυο εννοιών, δηλ. το WordNet. Στη βιβλιογραφία υπάρχουν διαθέσιμα λεξικά και για άλλες γλώσσες (π.χ. CORNETTO για την ολλανδική γλώσσα [7], Lexicographer για τη ρωσική [10]).

Η σημασιολογική αντιστοίχιση μέσω του WordNet περιλαμβάνει τη σύγκριση των λέξεων που εξάγονται από την παραπάνω διαδικασία (μέθοδοι, παράμετροι, κτλ.) βάσει των περιπτώσεων προσαρμογής στο πλαίσιο χρήσης. Αν δε βρεθούν όμοιες λέξεις σε αυτό το επίπεδο, η διαδικασία επαναλαμβάνεται για όλες τις πιθανές υπο-λέξεις που προκύπτουν μέσω της διαδικασίας κατάτμησης των λέξεων (tokenization). Οι λέξεις διασπώνται όχι μόνο στις υπο-λέξεις τελευταίου επιπέδου, αλλά και σε υπο-λέξεις που περιέχουν περισσότερες από μία λέξεις. Αυτές οι σύνθετες υπο-λέξεις συμμετέχουν νωρίτερα στη διαδικασία σύγκρισης. Είναι προτιμότερο να βρεθούν όμοιες λέξεις σε επίπεδο σύνθετων υπο-λέξεων παρά σε επίπεδο λέξεων του τελευταίου επιπέδου διάσπασης, καθώς κάτι τέτοιο φανερώνει στενότερη ομοιότητα μεταξύ των αρχικών λέξεων. Η βαθμολογία που εκχωρείται είναι υψηλότερη όταν ταιριάζουν σύνθετες υπο-λέξεις και μάλιστα ανάλογη του αριθμού των υπο-λέξεων που μπορούν να εξαχθούν από αυτό το επίπεδο. Αν θεωρηθεί για παράδειγμα η αντιστοίχιση των λέξεων `GetTheaters AndMovies` και `ResolveIP`, η σύγκριση θα εφαρμοσθεί αρχικά σε ολόκληρες τις λέξεις, στη συνέχεια στις υπο-φράσεις που περιέχουν παραπάνω από μία υπο-λέξεις και, τέλος, στις υπο-λέξεις του χαμηλότερου επιπέδου (π.χ. οι λέξεις GetTheatersAndMovies και ResolveIP θα συγκριθούν πρώτες, οι Theaters και ResolveIP αργότερα, και οι Theaters και IP ακόμα πιο μετά).

Η κατάτμηση των λέξεων βασίζεται στον εντοπισμό κεφαλαίων γραμμάτων (που εντοπίστηκαν στην πλειοψηφία των WSDL αρχείων των μητρώων στο διαδίκτυο) και σημείων κάτω παύλας _ (που χρησιμοποιείται στο WordNet) στη διαδοχή των χαρακτήρων

που συμμετέχουν στη λέξη. Άλλες περιπτώσεις λέξεων μόνο με κεφαλαίους χαρακτήρες ή γλωσσών εκτός της αγγλικής, δε μπορούν να εντοπιστούν από τη διαδικασία κατάτμησης. Το σημασιολογικό εργαλείο του WordNet που υλοποιήθηκε (Semantic WordNet Matcher / SWNM) χρησιμοποιεί επίσης διαφορετική βαθμολογία βάσει της εγγύτητας των εννοιών των λέξεων που συγκρίνονται, όπως ορίζονται στο WordNet: ακριβώς ίδιες λέξεις, συνώνυμες λέξεις (synonyms), λέξεις που συνδέονται μέσω σχέσης υπερωνύμων (hypernyms), όπου δηλ. η μία λέξη έχει ευρύτερη ή γενικότερη έννοια από την άλλη, και λέξεις που συνδέονται μέσω σχέσης μερωνύμων (meronyms), όπου η μία λέξη αποτελεί κομμάτι της έννοιας που περιγράφει η άλλη. Στις ακριβώς ίδιες και τις συνώνυμες λέξεις εκχωρείταιι υψηλότερη βαθμολογία (4 και 3 μονάδες αντίστοιχα) από αυτή που προκύπτει για τα υπερώνυμα ή τα μερώνυμα (1 μονάδα).

Ακολουθώντας τις παραπάνω αρχές υπολογίζονται οι βαθμολογίες για τις δύο πρώτες περιπτώσεις προσαρμογής στο πλαίσιο χρήσης για όλα τα τερματικά σημεία που περιγράφονται στην προδιαγραφή της υπηρεσίας. Η μεγαλύτερη βαθμολογία που προκύπτει διατηρείται ως η καλύτερη για τη συγκεκριμένη πηγή πλαισίου χρήσης και για κάθε περίπτωση προσαρμογής και εισάγεται στον πίνακα αποτελεσμάτων, στον οποίο συνοψίζονται όλες οι βαθμολογίες για τη συγκεκριμένη υπηρεσία εφαρμογών.

8.3.2.3 NF σύγκριση

Στο δεύτερο στάδιο της διαδικασίας εύρεσης συγκρίνονται οι μη λειτουργικές ιδιότητες των υπηρεσιών. Με αυτόν τον τρόπο εντοπίζονται περιγραφές υπηρεσιών που παρουσιάζουν σημασιολογικές ομοιότητες, αλλά συγκρίνονται και πρόσθετες ιδιότητες, όπως η χωρική και χρονική διαθεσιμότητα των υπηρεσιών, ενδεχόμενοι περιορισμοί σε πρωτόκολλα επικοινωνίας, κτλ. Το στάδιο NF σύγκρισης παρουσιάζει μικρότερη πολυπλοκότητα από τη σύγκριση των WSDL προδιαγραφών. Ωστόσο, οι ιδιότητες που συμμετέχουν είναι ιδιαίτερα σημαντικές για τη διαδικασία αντιστοίχισης, καθώς για παράδειγμα μια πηγή πλαισίου χρήσης που είναι διαθέσιμη μόνο στη Γαλλία δε μπορεί να συνδυαστεί με μια υπηρεσία εφαρμογών για περιοχές των Η.Π.Α. Συγκεκριμένα, αν ο έλεγχος συμβατότητας των περιοχών στις οποίες διατίθεται η κάθε υπηρεσία δεν είναι επιτυχής, η βαθμολογία της σύγκρισης λαμβάνει αυτομάτως την τιμή μηδέν και δεν πραγματοποιείται κάποιο άλλο είδος σύγκρισης για τη συγκεκριμένη πηγή, η οποία και αφαιρείται από τη λίστα των συμβατών υπηρεσιών πλαισίου χρήσης.

Στην υλοποίηση που έχει πραγματοποιηθεί για τον αλγόριθμο σύγκρισης των NF αρχείων λαμβάνονται υπόψη οι περιορισμοί που αφορούν τη θέση και τα πρωτόκολλα επικοινωνίας μαζί με τις ετικέτες των υπηρεσιών, καθώς αυτές οι πληροφορίες μπορούν να

εξαχθούν από τις διαθέσιμες περιγραφές στα μητρώα του διαδικτύου. Η σύγκριση των ετικετών πραγματοποιείται και εδώ μέσω του WordNet, ενώ λέξεις που δεν παρουσιάζουν ενδιαφέρον δε λαμβάνονται υπόψη στις συγκρίσεις, όπως συμβαίνει και στη διαδικασία της WSDL σύγκρισης. Η σύγκριση για τη χωρική διαθεσιμότητα πραγματοποιείται συγκρίνοντας τόσο τη συντακτική δομή των στοιχείων που εμφανίζονται στο XML αρχείο των μη λειτουργικών απαιτήσεων, όσο και τις σχετιζόμενες χώρες, πόλεις, κτλ. μέσω αντίστοιχων σχέσεων μερωνυμίας.

8.4 Αξιολόγηση Μηχανισμού Εύρεσης

Για την αξιολόγηση και αποτίμηση της διαδικασίας εντοπισμού ένα σύνολο υπηρεσιών διαδικτύου εισήχθησαν σε μια αποθήκη υπηρεσιών συμβατή με το σημασιολογικό Προφίλ Υπηρεσιών Διαδικτύου. Επιλέχθηκαν κυρίως περιγραφές από το μητρώο XMethods, το οποία περιλαμβάνει πολλά έγκυρα URL, καθώς και περιγραφές από το WebServiceX και από κάποιες άλλες υπηρεσίες που αναπτύχθηκαν στα πλαίσια της αξιολόγησης της προτεινόμενης λύσης. Με αυτόν τον τρόπο προέκυψε μια αποθήκη με διάφορες περιγραφές υπηρεσιών, η οποία τοποθετήθηκε πάνω από έναν εξυπηρετητή εφαρμογών, ώστε να είναι εφικτή η πρόσβαση για πραγματοποίηση λειτουργιών δημιουργίας, ανάκτησης, ενημέρωσης και διαγραφής περιγραφών (Create, Retrieve, Update, Delete / CRUD).

Ένα τμήμα της αποθήκης για τις υπηρεσίες εφαρμογών παρουσιάζεται στον Πίνακα 10 μαζί με μια σύντομη περιγραφή για κάθε υπηρεσία, την περίπτωση προσαρμογής στο πλαίσιο χρήσης που απαιτείται (βάσει Τοποθεσίας, Χρόνου ή Καμία προσαρμογή) και τους περιορισμούς χωρικής διαθεσιμότητας που ενδεχομένως υπάρχουν. Παρόμοιες πληροφορίες παρουσιάζονται στον Πίνακα 11 για τις υπηρεσίες πλαισίου χρήσης που χωρίζονται σε τρεις κατηγορίες (Τοποθεσίας, Χρόνου και Καιρού). Σε κάθε περίπτωση φαίνεται και η πηγή από την οποία έχει προέλθει η αντίστοιχη περιγραφή.

Υπηρεσίες εφαρμογών / BWSs	**Σύντομη περιγραφή**	**Προσαρμογή**	**Χωρική διαθεσιμότητα**
AirportInformation (WebServiceX)	Επιστρέφει τον κωδικό του αεροδρομίου, το όνομα της πόλης ή του αεροδρομίου, τη χώρα, το διάδρομο ανύψωσης σε πόδια και άλλες πληροφορίες σχετικές με το αεροδρόμιο βάσει του ονόματος της χώρας, του κωδικού του αεροδρομίου, κτλ.	Τοποθεσία	Παντού

Driving (XMethods)	Παρέχει οδηγίες πλοήγησης από ένα σημείο σε ένα άλλο βάσει των διευθύνσεων που δίνονται ως είσοδος.	Τοποθεσία	Παντού
ElectronicProductsFinder (XMethods)	Δέχεται μια λέξη αναζήτησης για κάποιο ηλεκτρονικό προιόν και επιστρέφει ένα URL με το αποτέλεσμα.	Καμία	Παντού
ForeignExchangeRate (XMethods)	Παρέχει πληροφορίες για τρέχουσες και παλαιότερες συναλλαγματικές ισοτιμίες για 23 τρέχοντα και 9 παλαιότερα νομίσματα.	Χρόνος	Παντού
GreetingService (υλοποίηση στα πλάισια της αξιολόγησης)	Επιστρέφει ένα μήνυμα χαιρετισμού βάσει της ώρας της ημέρας και της τρέχουσας θέσης του χρήστη (πόλη).	Χρόνος και Τοποθεσία	Παντού
Historicoptiondata (XMethods)	Παρέχει στοιχεία ημέρας για δικαιώματα αγοράς μετοχών (stock options) στις Η.Π.Α.	Χρόνος	Η.Π.Α. (U.S.A.)
ISoapFindMP3Service (XMethods)	Βρίσκει URL για μουσικά κομμάτια MP3 βάσει των λέξεων αναζήτησης που δίνονται.	Καμία	Παντού
MidnightTraderFinancialNews (XMethods)	Παρέχει πρόσβαση σε τρέχουσες και παλαιότερες ειδήσεις μετοχών και άλλες ανακοινώσεις και πληροοφορίες για την αγορά μετοχών στις Η.Π.Α.	Χρόνος	Η.Π.Α.
MovieInformation (XMethods)	Δέχεται ένα ταχυδρομικό κωδικό και μια έγκυρη ακτίνα και επιστρέφει μια λίστα με τις κινηματογραφικές αίθουσες της περιοχής και τις ταινίες που προβάλλονται.	Τοποθεσία	Η.Π.Α.
Proximity (XMethods)	Επιστρέφει σημεία ενδιαφέροντος κοντά σε μια συγκεκριμένη τοποθεσία.	Τοποθεσία	Η.Π.Α.
SportingGoodsFinder (XMethods)	Δέχεται μια λέξη αναζήτησης για αθλητικά προιόντα και επιστρέφει ένα URL με το αποτέλεσμα.	Καμία	Παντού

Πίνακας 10. Περιγραφές υπηρεσιών εφαρμογών.

Υπηρεσίες πλαισίου χρήσης/ CWSs	Σύντομη περιγραφή	Κατηγορία	Χωρική διαθεσιμότητα
CDYNEWeather (XMethods)	Παρέχει πληροφορίες για καιρικές συνθήκες και προβλέψεις στις Η.Π.Α.	Καιρός	Η.Π.Α.
GlobalWeatherService (XMethods)	Παρέχει πληροφορίες για τον καιρό συμπεριλαμβανομένων προβλέψεων, συνθηκών και ειδοποιήσεων για πάνω από 5.000 διεθνείς τοποθεσίες.	Καιρός	Παντού

GlobalWeather (WebServiceX)	Παρέχει καιρικές αναφορές για όλες τις μεγάλες πόλεις του κόσμου.	Καιρός	Παντού
IP2Geo (XMethods)	Βάσει IP διευθύνσεων παρέχει πληροφορίες για το όνομα του ιδιοκτήτη του δικτύου, την πόλη, την πολιτεία ή το νομό και τη χώρα. Στις περισσότερες πόλεις στις Η.Π.Α. παρέχει πληροφορίες και για τον ταχυδρομικό κώδικα και τις συντεταμένες της τοποθεσίας.	Τοποθεσία	Η.Π.Α.
IP2Location (XMethods)	Επιτρέπει την άμεση αναζήτηση γεωγραφικών τοποθεσιών βάσει της IP διεύθυνσης. Επιστρέφει τη χώρα, την πολιτεία, την πόλη, τις συντεταγμένες, τον ταχυδρομικό κωδικό και το όνομα του παρόχου και του domain.	Τοποθεσία	Παντού
UKLocation (WebServiceX)	Ανακτά πληροφορίες για τοποθεσίες στο Ηνωμένο Βασίλειο (όνομα πόλης, ταχυδρομικό κωδικό, κτλ.).	Τοποθεσία	Ηνωμένο Βασίλειο (U.K.)
DateTime (υλοποίηση στα πλάισια της αξιολόγησης)	Ανακτά πληροφορίες για την τρέχουσα ημερομηνία, ώρα και ζώνη ώρας.	Χρόνος	Παντού
IPligenceGeoIPLocation (XMethods)	Παρέχει γεωγραφικές πληροφορίες, όπως εύρος IP διευθύνσεων, ιδιοκτήτης IP διεύθυνσης, συντεταγμένες, ζώνη ώρας και ένα χάρτη που αναπαριστά την τοποθεσία για μια συγκεκριμένη IP διεύθυνση.	Τοποθεσία	Παντού

Πίνακας 11. Περιγραφές υπηρεσιών πλαισίου χρήσης.

Οι WSDL προδιαγραφές και οι οδηγίες χρήσης που ήταν διαθέσιμες για τις υπηρεσίες στα μητρώα του διαδικτύου χρησιμοποιήθηκαν στο βήμα κατασκευής του NF αρχείου (π.χ. η πληροφορία τοπικής διαθεσιμότητας που φαίνεται στους παραπάνω πίνακες έχει προέλθει από αυτές τις περιγραφές κειμένου). Κάποιες από τις παραπάνω υπηρεσίες πληρούν τα κριτήρια αντιστοίχισης βάσει προσαρμογής στο πλαίσιο χρήσης (π.χ. η υπηρεσία εφαρμογών `MovieInformation` μπορεί να χρησιμοποιηθεί σε συνδυασμό με τις υπηρεσίες πλαισίου χρήσης `GeoIPService` ή `IPligenceGeoIPLocation`), άλλες σχετίζονται με παραπάνω από μία προσαρμογή (`GreetingService`), ενώ άλλες δεν παρουσιάζουν καμία ομοιότητα ή δεν απαιτούν κάποιο είδος προσαρμογής (π.χ. `ElectronicProductsFinder` και `ISoapFindMP3Service`).

Επιπροσθέτως, διάφοροι χωρικοί περιορισμοί είναι ορατοί σε κάποιες υπηρεσίες, ενώ δεν έχει παρατηρηθεί και δεν έχει συμπεριληφθεί κάποιος περιορισμός που σχετίζεται με πρωτόκολλα επικοινωνίας ή κάποιο άλλο είδος περιορισμού. Η αξιολόγηση της διαδικασίας έγκειται στον εντοπισμό των συμβατών υπηρεσιών βάσει της προσαρμογής

στο πλαίσιο χρήσης, αλλά και της πληροφορίας που εξάγεται από τις μη λειτουργικές ιδιότητες των υπηρεσιών (πληροφορίες χωρικής διαθεσιμότητας και πληροφορίες ετικετών).

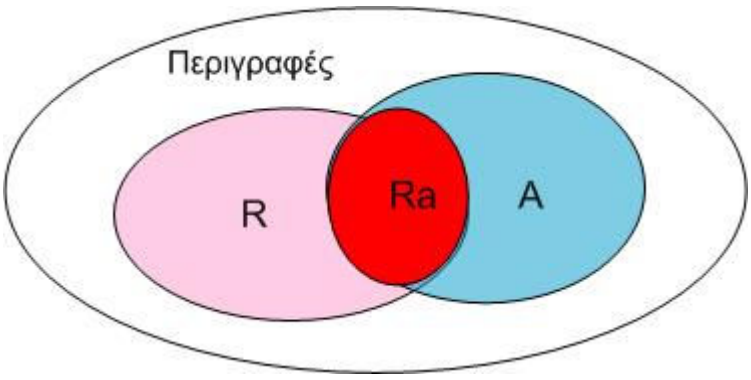
Η διαδικασία αντιστοίχισης πραγματοποιήθηκε για κάθε υπηρεσία εφαρμογών και για κάθε υπηρεσία πλαισίου χρήσης που εμφανίζεται στους παραπάνω πίνακες. Για λόγους αποτίμησης εισάγονται δύο ευρέως αποδεκτές μετρικές από το πεδίο της ανάκτησης πληροφοριών: η ακρίβεια (precision) και η ανάκληση (recall). Συγκεκριμένα *"Ακρίβεια είναι το ποσοστό των ανακτημένων αρχείων που θεωρούνται σχετικά και ανάκληση είναι το ποσοστό των σχετικών αρχείων που έχουν ανακτηθεί"* [15] (Εικόνα 61). Τιμές κοντά στη μονάδα αποτελούν ένδειξη αποδοτικών μεθόδων ανάκτησης πληροφορίας:

$$\text{Precision} = \frac{|\text{σχετικές ανακτημένες υπηρεσίες}|}{|\text{σύνολο ανακτημένων υπηρεσιών}|} \quad \text{ή}$$

$$\text{Precision} = \frac{|\text{Relevant answers}|}{|\text{Total answers}|} = \frac{|Ra|}{|A|}$$

$$\text{Recall} = \frac{|\text{σχετικές ανακτημένες υπηρεσίες}|}{|\text{πραγματικό σύνολο σχετικών υπηρεσιών}|} \quad \text{ή}$$

$$\text{Recall} = \frac{|\text{Relevant answers}|}{|\text{Relevant documents}|} = \frac{|Ra|}{|R|}$$



Εικόνα 61. Διάγραμμα Venn για ανάκτηση πληροφορίας.

Οι τιμές των μετρικών που υπολογίστηκαν για κάθε υπηρεσία εφαρμογών μέσω της διαδικασίας SWNM παρουσιάζονται στους Πίνακες 12 και 13, όπου με έντονους χαρακτήρες φαίνονται οι υπηρεσίες πλαισίου χρήσης που πράγματι ταιριάζουν με την αντίστοιχη υπηρεσία εφαρμογών. Ο Πίνακας 12 περιλαμβάνει τις μετρικές, όπως προκύπτουν όταν πραγματοποιείται μόνο το στάδιο αντιστοίχισης βάσει των WSDL

προδιαγραφών, ενώ ο Πίνακας 13 αφορά την περίπτωση εκτέλεσης όλων των φάσεων (αντιστοίχιση WSDL αρχείων και στη συνέχεια αντιστοίχιση NF αρχείων).

Υπηρεσία εφαρμογών αίτησης	Ανακτημένες υπηρεσίες πλαισίου χρήσης	Ακρίβεια	Ανάκτηση
AirportInformation	CDYNEWeather IP2Geo **IP2Location** UKLocation **IPligenceGeoIPLocation**	0,4	0,5
driving	CDYNEWeather GlobalWeatherService GlobalWeather IP2Geo **IP2Location** UKLocation **IPligenceGeoIPLocation**	0,286	1
ElectronicProductsFinder	<None>	0/0	1
ForeignExchangeRate	**DateTime** IPligenceGeoIPLocation	0,5	1
GreetingService	CDYNEWeather GlobalWeather IP2Geo **IP2Location** **DateTime** UKLocation **IPligenceGeoIPLocation**	0,429	1
historicoptiondata	**DateTime**	1	1
ISoapFindMP3Service	<None>	0/0	1
MidnightTraderFinancialNews	CDYNEWeather GlobalWeatherService **DateTime**	0,333	1
MovieInformation	CDYNEWeather GlobalWeatherService **IP2Geo** **IP2Location** UKLocation **IPligenceGeoIPLocation**	0,6	1
Proximity	CDYNEWeather GlobalWeatherService GlobalWeather **IP2Geo** **IP2Location** UKLocation **IPligenceGeoIPLocation**	0,429	1
SportingGoodsFinder	<None>	0/0	1

Πίνακας 12. Τιμές για μετρικές ακρίβειας και ανάκλησης για κάθε υπηρεσία εφαρμογών για τη φάση WSDL σύγκρισης της διαδικασίας SWNM.

Υπηρεσία εφαρμογών αίτησης	Ανακτημένες υπηρεσίες πλαισίου χρήσης	Ακρίβεια	Ανάκτηση
AirportInformation	**IP2Location**	1	0,333
Driving	GlobalWeather **IP2Location** **IPligenceGeoIPLocation**	0,667	1
ElectronicProductsFinder	<None>	0/0	1
ForeignExchangeRate	**DateTime** IPligenceGeoIPLocation	0,5	1
GreetingService	GlobalWeather **IP2Location** **DateTime**	0,667	0,667
Historicoptiondata	**DateTime**	1	1
ISoapFindMP3Service	<None>	0/0	1
MidnightTraderFinancialNews	**DateTime**	1	1
MovieInformation	**IP2Geo** **IP2Location** **IPligenceGeoIPLocation**	1	1
Proximity	**IP2Geo** **IP2Location** **IPligenceGeoIPLocation**	1	1
SportingGoodsFinder	<None>	0/0	1

Πίνακας 13. Τιμές για μετρικές ακρίβειας και ανάκλησης για κάθε υπηρεσία εφαρμογών για όλες τις φάσεις της διαδικασίας SWNM.

Για την περαιτέρω αποτίμηση της προτεινόμενης διαδικασίας υλοποιήθηκε επίσης στα πλαίσια της διατριβής ένας αλγόριθμος αντιστοίχισης που βασίζεται στη συντακτική ομοιότητα των λέξεων που συμμετέχουν στις συγκρίσεις, όπως εξάγονται από την περιγραφή της υπηρεσίας διαδικτύου (αντί του WordNet). Τα αντίστοιχα αποτελέσματα των μετρικών ακρίβειας και ανάκλησης για την εφαρμογή αυτής της περίπτωσης σε WSDL αρχεία παρουσιάζονται στον Πίνακα 14.

Υπηρεσία εφαρμογών αίτησης	Ανακτημένες υπηρεσίες πλαισίου χρήσης	Ακρίβεια	Ανάκτηση
AirportInformation	CDYNEWeather IP2Geo **IP2Location** UKLocation **IPligenceGeoIPLocation**	0,4	1
Driving	GlobalWeatherService UKLocation **IPligenceGeoIPLocation**	0,333	0,5

ElectronicProductsFinder	<None>	0/0	1
ForeignExchangeRate	***DateTime***	1	1
GreetingService	CDYNEWeather IP2Geo **IP2Location** **DateTime** **IPligenceGeoIPLocation**	0,6	1
Historicoptiondata	**DateTime**	1	1
ISoapFindMP3Service	<None>	0/0	1
MidnightTraderFinancialNews	GlobalWeatherService	0	0
MovieInformation	CDYNEWeather **IP2Geo** **IP2Location** UKLocation **IPligenceGeoIPLocation**	0,6	1
Proximity	GlobalWeatherService	0	0
SportingGoodsFinder	<None>	0/0	1

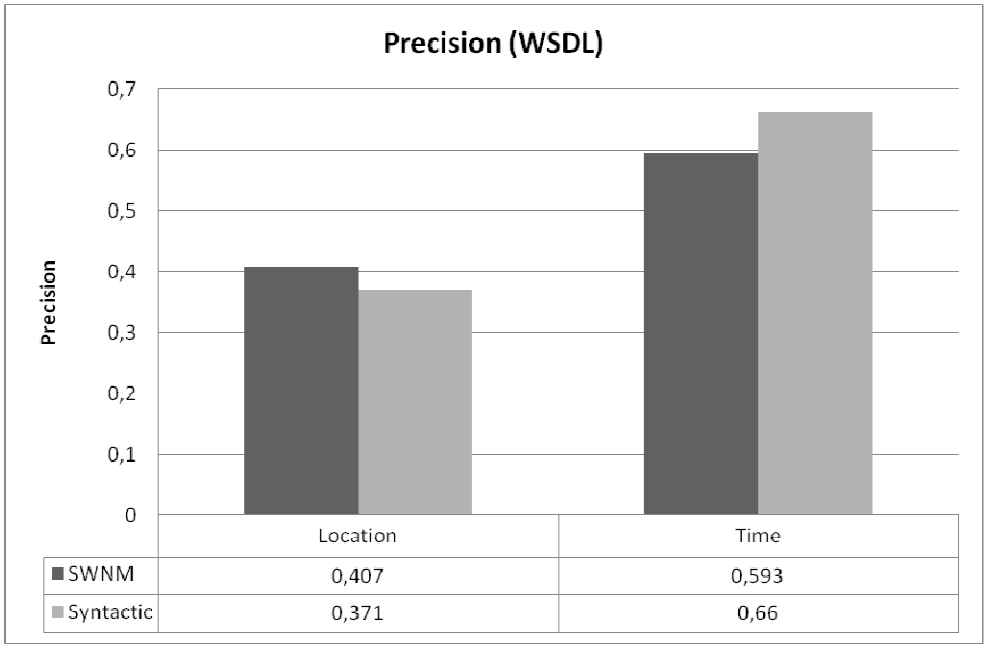
Πίνακας 14. Τιμές για μετρικές ακρίβειας και ανάκλησης για κάθε υπηρεσία εφαρμογών για τη συντακτική σύγκριση.

Παρατηρώντας τους παραπάνω πίνακες συμπεραίνεται πως ο συνδυασμός της σύγκρισης των WSDL και των NF αρχείων υπερτερεί έναντι της αντιστοίχισης μόνο σε επίπεδο WSDL προδιαγραφών. Το κύριο όφελος προκύπτει από τη χρήση του NF αρχείου που περιλαμβάνει πληροφορίες για τους περιορισμούς διαθεσιμότητας της υπηρεσίας που δεν περιέχονται στις WSDL περιγραφές.

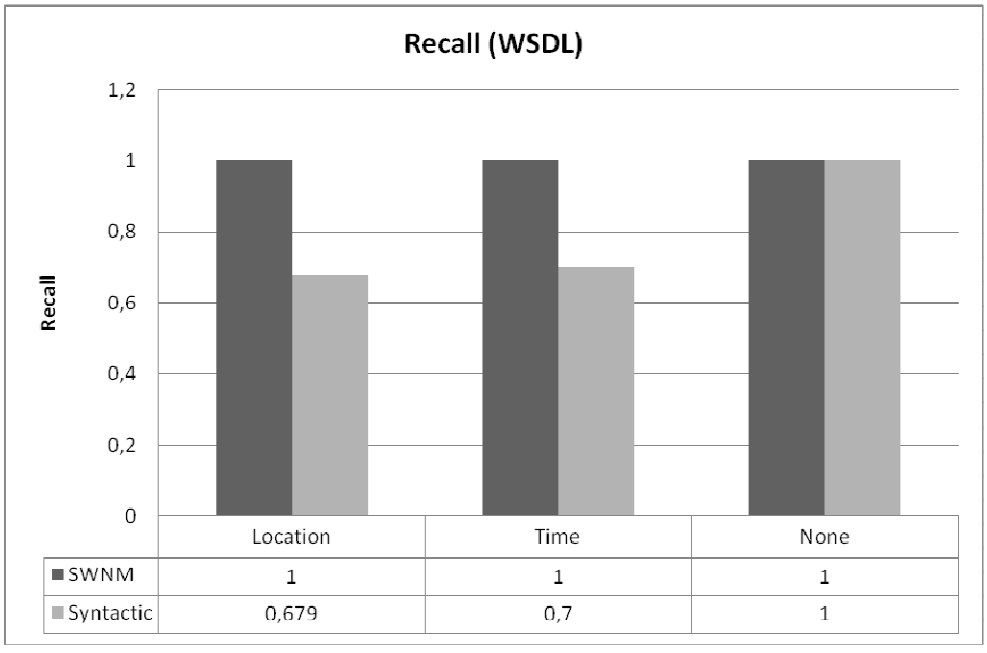
Προκειμένου να απεικονιστούν διαγραμματικά τα αποτελέσματα, πραγματοποιείται στο σημείο αυτό μια σύγκριση μεταξύ του SWNM και της συντακτικής αντιστοίχισης σε ένα σύνολο διαγραμμάτων. Οι τιμές της ακρίβειας και της ανάκλησης προκύπτουν από το μέσο όρο των τιμών κάθε κατηγορίας προσαρμογής από τους παραπάνω πίνακες (δηλ. Τοποθεσία, Χρόνος και Καμία προσαρμογή). Δύο περιπτώσεις σύγκρισης χρησιμοποιούνται και στα διαγράμματα: WSDL αντιστοίχιση (Εικόνες 62 και 63) και συνδυασμένη WSDL και NF αντιστοίχιση (Εικόνες 64 και 65).

Ο υπολογισμός της τιμής της μετρικής της ακρίβειας για την περίπτωση χωρίς προσαρμογή δεν έχει νόημα (0/0) και γι' αυτό δεν έχει συμπεριληφθεί στα αντίστοιχα διαγράμματα. Επίσης αξίζει να σημειωθεί πως για τον υπολογισμό του μέσου όρου για την υπηρεσία `GreetingService` χρησιμοποιήθηκαν κατάλληλα βάρη, καθώς συμμετέχει τόσο στην προσαρμογή βάσει τοποθεσίας όσο και στην προσαρμογή βάσει χρόνου, με δύο και μία συμβατή υπηρεσία πλαισίου χρήσης αντίστοιχα (οι τιμές των μετρικών

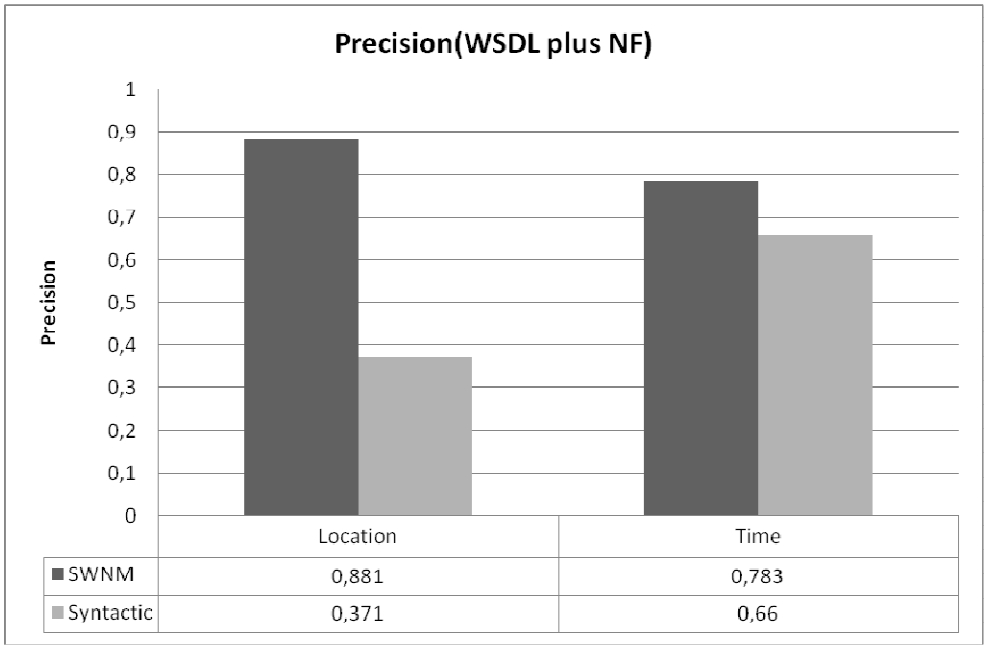
συμμετέχουν συνεπώς με βάρη 2/3 και 1/3 στον υπολογισμό του μέσου όρου για την τοποθεσία και το χρόνο αντίστοιχα).



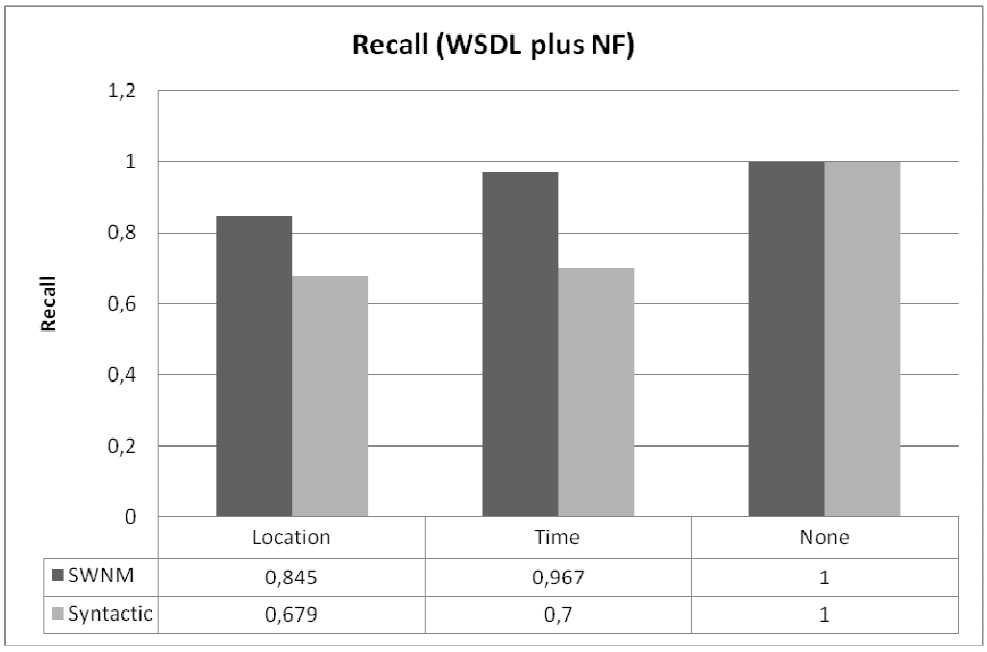
Εικόνα 62. Σύγκριση μέσων όρων ακρίβειας για περιπτώσεις προσαρμογής βάσει WSDL σύγκρισης



Εικόνα 63. Σύγκριση μέσων όρων ανάκλησης για περιπτώσεις προσαρμογής βάσει WSDL σύγκρισης.



Εικόνα 64. Σύγκριση μέσων όρων ακρίβειας για περιπτώσεις προσαρμογής βάσει συνδυασμένης WSDL και NF σύγκρισης.



Εικόνα 65. Σύγκριση μέσων όρων ανάκλησης για περιπτώσεις προσαρμογής βάσει συνδυασμένης WSDL και NF σύγκρισης.

Από τα παραπάνω διαγράμματα παρατηρείται πως αν και ο συνδυασμός της αντιστοίχισης σε επίπεδο WSDL και NF αρχείων αποδεικνύεται καλύτερος από τη συντακτική αντιστοίχιση για όλες τις περιπτώσεις και για τις δύο μετρικές, δε συμβαίνει το ίδιο όταν λαμβάνονται υπόψη στη διαδικασία μόνο οι WSDL προδιαγραφές (μετρική

precision). Αυτό παρατηρείται κυρίως λόγω της ύπαρξης λέξεων στα WSDL αρχεία που δε συνδέονται άμεσα με τη λειτουργικότητα της υπηρεσίας. Ωστόσο, αυτές οι λέξεις αποκτούν μια αποδεκτή βαθμολογία από τη σύγκριση του WordNet, καθώς παρουσιάζουν σημασιολογικές ομοιότητες. Κάτι τέτοιο παρουσιάζεται για παράδειγμα όταν κάποιες υπηρεσίες που σχετίζονται με τον καιρό επιστρέφονται για την περίπτωση προσαρμογής βάσει τοποθεσίας, καθώς παρέχουν προγνώσεις καιρού για συγκεκριμένες πόλεις ή χώρες, που είναι ως έννοιες συνδεδεμένες με την τοποθεσία, ενώ το φαινόμενο είναι πιο έντονο στην περίπτωση της προσαρμογής στο χρόνο, όπου ο συντακτικός αλγόριθμος παρουσιάζει καλύτερη απόδοση από το SWNM.

Ένα γενικό μειονέκτημα της διαδικασίας που χρησιμοποιεί το WordNet είναι ότι δεν είναι εφικτό να καθοριστεί αν μια συγκεκριμένη λέξη αποτελεί ρήμα, επίρρημα ή κάποιο άλλο μέρος του λόγου οδηγώντας σε λάθος αποτελέσματα (για παράδειγμα η αγγλική λέξη “address” μπορεί να αναφέρεται είτε στο ουσιαστικό που εκφράζει τη διεύθυνση κάποιου ατόμου είτε στο ρήμα που εκφράζει έναν τρόπο επικοινωνίας). Αυτό το μειονέκτημα έχει παρατηρηθεί και σε άλλες ερευνητικές εργασίες για προηγούμενες εκδόσεις του WordNet [8]. Ένας πιθανός τρόπος επίλυσης μπορεί να προκύψει μέσω της ανάλυσης των ονομάτων των μεθόδων και των παραμέτρων που μπορούν να αποτελούνται από ουσιαστικά και ρήματα, όπως καθορίζεται σε μελέτες γλωσσολογίας.

Παρά τα παραπάνω μειονεκτήματα η διαδικασία είναι ιδιαίτερα χρήσιμη για τους μηχανικούς λογισμικού που επιθυμούν να εντοπίσουν πιθανές πηγές πλαισίου χρήσης για επαναχρησιμοποίηση στις δικές τους εφαρμογές διαδικτύου. Η διαδικασία εντοπισμού επιτυγχάνει σημαντικά αποτελέσματα μέσω της ανάλυσης, κανονικοποίησης και σύγκρισης των περιγραφών των υπηρεσιών διαδικτύου, ενώ καλύτερα αποτελέσματα επιτυγχάνονται όταν χρησιμοποιείται ο συνδυασμός των WSDL και NF σταδίων.

Συνοπτικά, τα βασικά οφέλη που προκύπτουν είναι:

- Πρώτη προσπάθεια αντιστοίχισης υπηρεσιών διαδικτύου για λόγους προσαρμογής στο πλαίσιο χρήσης.
- Δυνατότητα εντοπισμού και επαναχρησιμοποίησης υπηρεσιών πλαισίου χρήσης που ανήκουν ακόμα και σε εξωτερικούς παρόχους.
- Η προτεινόμενη περιγραφή υπηρεσιών δε μεταβάλλει τη δομή των WSDL αρχείων, αλλά εισάγει πρόσθετες περιγραφές σε νέα τεχνήματα.

8.5 Βιβλιογραφία

[1] Bansal A., Kona S., Simon L., Mallya A., Gupta G., Hite T.D., “A universal service-semantics description language”, Proc. 3rd IEEE European Conference on Web Services (ECOWS ‘05), 2005, pp. 1-12.

[2] Fan, J., Kambhampati, S., “A Snapshot of Public Web Services”, ACM SIGMOD Record archive, vol. 34(1), 2005, pp. 24-32.

[3] Gannod G. C., Bhatia S., “Facilitating Automated Search for Web Services”, Proc. IEEE International Conference on Web Services, 2004, pp. 761-764.

[4] Gerard R., Downs R. R., Marshall J. J., Wolfe R. E., “The Software Reuse Working Group: A Case Study in Fostering Reuse”, Proc. IEEE International Conference on Information Reuse and Integration (IRI'07), 2007, pp. 24-29.

[5] JWI 2.1.5, MIT Java Wordnet Interface, http://projects.csail.mit.edu/jwi/.

[6] Kapitsaki, G. M., Kateros, D. A., Venieris, I. S., “Extending Reusable Asset Specification to Describe Simple Mobile Services”, Proc. 17th IST Mobile and Wireless Communications Summit, 2008, pp. 1-8.

[7] Maks, I. Vossen, P., Segers, P., van der Vliet, H., “Adjectives in the Dutch semantic lexical database CORNETTO”, Proc. 6th Int. Language Resources and Evaluation (LREC'08), 2008, pp. 715-719.

[8] Mandala, R., Takenobu, T., T. Hozumi, “The Use of WordNet in Information Retrieval”, Proc. COLING/ACL Workshop on Usage of WordNet in Natural Language Processing Systems, 1998, pp. 31-37.

[9] OMG, ''Reusable Asset Specification'', version 2.2, November 2005, http://www.omg.org/docs/formal/05-11-02.pdf.

[10] Paducheva, E. V., Rakhilina, E. V., Filipenko, M. V., “Semantic dictionary viewed as a lexical database”, Proc. 14th conference on Computational linguistics, 4, 2002, pp. 1295-1299.

[11] Park S., Park S., Sugumaran V., “Extending Reusable Asset Specification to Improve Software Reuse”, Proc. 2007 ACM symposium on Applied computing (SAC’07), pp. 1473-1478.

[12] Platzer C., Dustdar S., “A vector space search engine for Web services”, Proc. 3rd IEEE European Conference on Web Services (ECOWS’05), 2005.

[13] Sabou M., Pan J., “Towards semantically enhanced Web service repositories”, Web Semantics: Science, Services and Agents on the World Wide Web 2007, vol. 5(2), pp. 142-150.

[14] Sajjanhar A., Hou J., Zhang Y., “Algorithm for Web Services Matching”, Proc. 6th Asia-Pacific Web Conference on Advanced Web Technologies and Applications (APWeb’04), 2004, pp. 665-670.

[15] Voorhees, E., “Using WordNet for Text Retrieval”, in WordNet: An Electronic Lexical Database (The MIT Press 1998), pp.285-303.

[16] W3C, “OWL-S: Semantic markup for web services”, 2004, http://www.w3.org/ Submission/OWL-S/.

[17] W3C, “Semantic Annotations for Web Services Description Language”, http://www.w3.org/ 2002/ws/sawsdl/

[18] Wang Y., Stroulia E., “Flexible interface matching for Web-service discovery”, Proc. 4th International Conference on Web Information Systems Engineering (WISE 2003), 2003, pp. 147-156.

[19] Web Service List, http://www.webservicelist.com/.

[20] WebServiceX, http://www.webservicex.net/.

[21] WordNet, http://wordnet.princeton.edu/.

[22] XMethods , http://www.xmethods.com.

[23] Yang S., Zhang J., Chen I., “A JESS-enabled context elicitation system for providing context-aware Web services”, Expert Systems with Applications: An International Journal 2008, vol. 34(4), pp. 2254-2266.

9ο Κεφάλαιο

Επίλογος και Συμπεράσματα

Στο παρόν Κεφάλαιο που αποτελεί τον επίλογο της διατριβής συνοψίζονται τα βασικά συμπεράσματα και δίνονται κατευθύνσεις μελλοντικής έρευνας και επέκτασης έχοντας ως βάση τη λύση που προτείνεται στη διατριβή.

9.1 Σύνοψη και Συμπεράσματα

Βασικό έναυσμα για την εξέλιξη της διατριβής αποτέλεσε η ευρεία διάδοση εφαρμογών με επίγνωση του πλαισίου χρήσης και η διάδοση μεθοδολογιών ανάπτυξης που διευκολύνουν τη διαδικασία κατασκευής υπηρεσιών και επιτρέπουν την επαναχρησιμοποίηση δομικών συστατικών για τη δημιουργία καινούριων εφαρμογών. Στα πλαισια της διατριβής δόθηκε μια ολοκληρωμένη λύση για τη διαχείριση του πλαισίου χρήσης για την περίπτωση των διαδεδομένων υπηρεσιών διαδικτύου και την ανάπτυξη εφαρμογών που λαμβάνουν υπόψη στη λειτουργία τους το περιβάλλον και τα χαρακτηριστικά του υπό το πρίσμα της επαναχρησιμοποίησης υπηρεσιών. Συγκεκριμένα, τα βασικά μέρη που απαρτίζουν τη διατριβή είναι τα εξής:

- Ένας μηχανισμός για την προσαρμογή εφαρμογών διαδικτύου στο πλαίσιο χρήσης. Η προσαρμογή αφορά εφαρμογές που αποτελούνται από υπηρεσίες διαδικτύου, οι οποίες παρουσιάζουν εξαρτήσεις με διάφορα χαρακτηριστικά του χρήστη και του περιβάλλοντος εκτέλεσης. Η προσαρμογή πραγματοποιείται μέσω “σύλληψης” μηνυμάτων χωρίς να συνενώνεται με τη βασική υλοποίηση της εκάστοτε υπηρεσίας και είναι εφαρμόσιμη σε διάφορα πλαίσια εργασίας για υπηρεσίες διαδικτύου αρκεί να υποστηρίζεται η εισαγωγή χειριστών μηνυμάτων. Στα πλαίσια της διατριβής η υλοποίηση του μηχανισμού έχει πραγματοποιηθεί στο πλαίσιο Apache Axis2.

- Μια μοντελοκεντρική μεθοδολογία ανάπτυξης για την κατασκευή εφαρμογών διαδικτύου. Ξεκινώντας από το μοντέλο της εφαρμογής σε UML, το οποίο είναι συμβατό με διάφορα UML προφίλ που έχουν οριστεί, μπορεί να δημιουργηθεί μια σύνθετη εφαρμογή για διάφορες τεχνολογίες υλοποίησης. Η σχεδίαση της εφαρμογής στο επίπεδο μοντελοποίησης διατηρείται – σε πολύ μεγάλο βαθμό – ανεξάρτητη από τις λεπτομέρειες υλοποίησης που σχετίζονται με συγκεκριμένες πλατφόρμες και επαρκώς ευέλικτη, ώστε να επιτρέπει την εισαγωγή διαφόρων πληροφοριών στο μοντέλο. Ωστόσο, η παρουσίαση και εξέλιξη της διατριβής επικεντρώθηκε στην παραγωγή εφαρμογών για ευρέως διαδεδομένες πλατφόρμες που ακολουθούν την υλοποίηση της προτεινόμενης αρχιτεκτονικής προσαρμογής. Τόσο κατά την ανάπτυξη της εφαρμογής όσο και κατά την εκτέλεσή της η προσαρμογή στο πλαίσιο χρήσης παραμένει ανεξάρτητη από την κύρια εφαρμογή παρέχοντας ευελιξία και διευκολύνοντας με αυτόν τον τρόπο τη συντήρηση του συστήματος.
- Μια διαδικασία εύρεσης κατάλληλων υπηρεσιών πλαισίου χρήσης που μπορούν να συνδυαστούν με υπάρχουσες υπηρεσίες εφαρμογών. Η αντιστοίχιση των υπηρεσιών μπορεί να πραγματοποιηθεί σε επίπεδο WSDL προδιαγραφών, αλλά εμφανίζεται πιο αποτελεσματική όταν η WSDL σύγκριση συνδυάζεται με συγκρίσεις περιγραφών που ορίζονται στο σημασιολογικό Προφίλ Υπηρεσιών Διαδικτύου που περιλαμβάνει και τις μη λειτουργικές ιδιοτήτες της υπηρεσίας. Με αυτόν τον τρόπο είναι εφικτή η επαναχρησιμοποίηση υπηρεσιών διαδικτύου για τα διάφορα τμήματα της υπό κατασκευή εφαρμογής (και ως υπηρεσίες εφαρμογών και ως υπηρεσίες πλαισίου χρήσης) και διευκολύνονται σημαντικά τα πρώτα στάδια ανάπτυξης υπηρεσιών με επίγνωση του πλαισίου χρήσης.

Όλα τα παραπάνω τμήματα που απαρτίζουν την προσέγγιση που επιχειρείται στη διατριβή παρουσιάστηκαν αναλυτικά σε διάφορα Κεφάλαια μαζί με την υπάρχουσα βιβλιογραφία κάθε ερευνητικής περιοχής. Η προτεινόμενη λύση επεκτείνει και συνδυάζει την τρέχουσα επιστημονική δραστηριότητα σε ένα κοινό πλαίσιο. Οι βασικές συνεισφορές της προτεινόμενης λύσης και συνεπώς της διατριβής έγκεινται στα εξής:

- Η προσαρμογή στο πλαίσιο χρήσης πραγματοποιείται στο επίπεδο διεπαφής των υπηρεσιών διαδικτύου (χωρίς να επηρεάζει την υλοποίηση τους) και διατηρείται επιπλέον ανεξάρτητη από τον πελάτη χωρίς να απαιτεί κάποιο συγκεκριμένο υλικό ή λογισμικό για τη συσκευή του χρήστη.

- Η μοντελοποίηση της εφαρμογής κατά την ανάπτυξη πραγματοποιείται στη γλώσσα UML που προτιμάται έναντι κάποιας γλώσσας ειδικού πεδίου, καθώς με αυτόν τον τρόπο εξασφαλίζεται η συμβατότητα της μεθοδολογίας ανάπτυξης με ευρέως διαδεδομένα περιβάλλοντα σχεδίασης και ενισχύεται η επαναχρησιμοποίηση των μοντέλων μέσω της εισαγωγής διαθέσιμων μοντέλων ή της εξαγωγής των νέων που σχεδιάζονται για την περαιτέρω επαναχρησιμοποίησή τους.
- Η προσαρμογή στο πλαίσιο χρήσης και η μοντελοποίηση της εφαρμογής αντιμετωπίζονται παράλληλα οδηγώντας στην παραγωγή εφαρμογών με επίγνωση του πλαισίου χρήσης χωρίς να εισάγεται ανεπιθύμητη σύνδεση μεταξύ της λειτουργίας της εφαρμογής και της προσαρμογής στο πλαίσιο χρήσης.
- Παρέχεται η δυνατότητα χρήσης υπηρεσιών από εξωτερικούς παρόχους μειώνοντας με αυτόν τον τρόπο το χρόνο ανάπτυξης και δίνοντας τη δυνατότητα ενσωμάτωσης των υπηρεσιών σε νέες εφαρμογές.
- Η προσαρμογή στο πλαίσιο χρήσης πραγματοποιείται μέσω ειδικών συστατικών και μπορεί να τροποποιηθεί χωρίς να είναι απαραίτητο να επέμβει ο μηχανικός λογισμικού στην υλοποίησή τους.
- Στα πλαίσια της επαναχρησιμοποίησης υπηρεσιών υποστηρίζεται ο εντοπισμός πιθανών υπηρεσιών-πηγών πλαισίου χρήσης μέσω της ανάλυσης των αντίστοιχων περιγραφών των υπηρεσιών που υποστηρίζονται ακόμα και από εξωτερικούς παρόχους.

9.2 Πεδία Μελλοντικής Έρευνας

Η τρέχουσα και μελλοντική ερευνητική εργασία για την επέκταση της διατριβής κινείται στους άξονες που παρουσιάζονται στην παρούσα ενότητα.

9.2.1 Προστασία Ιδιωτικότητας

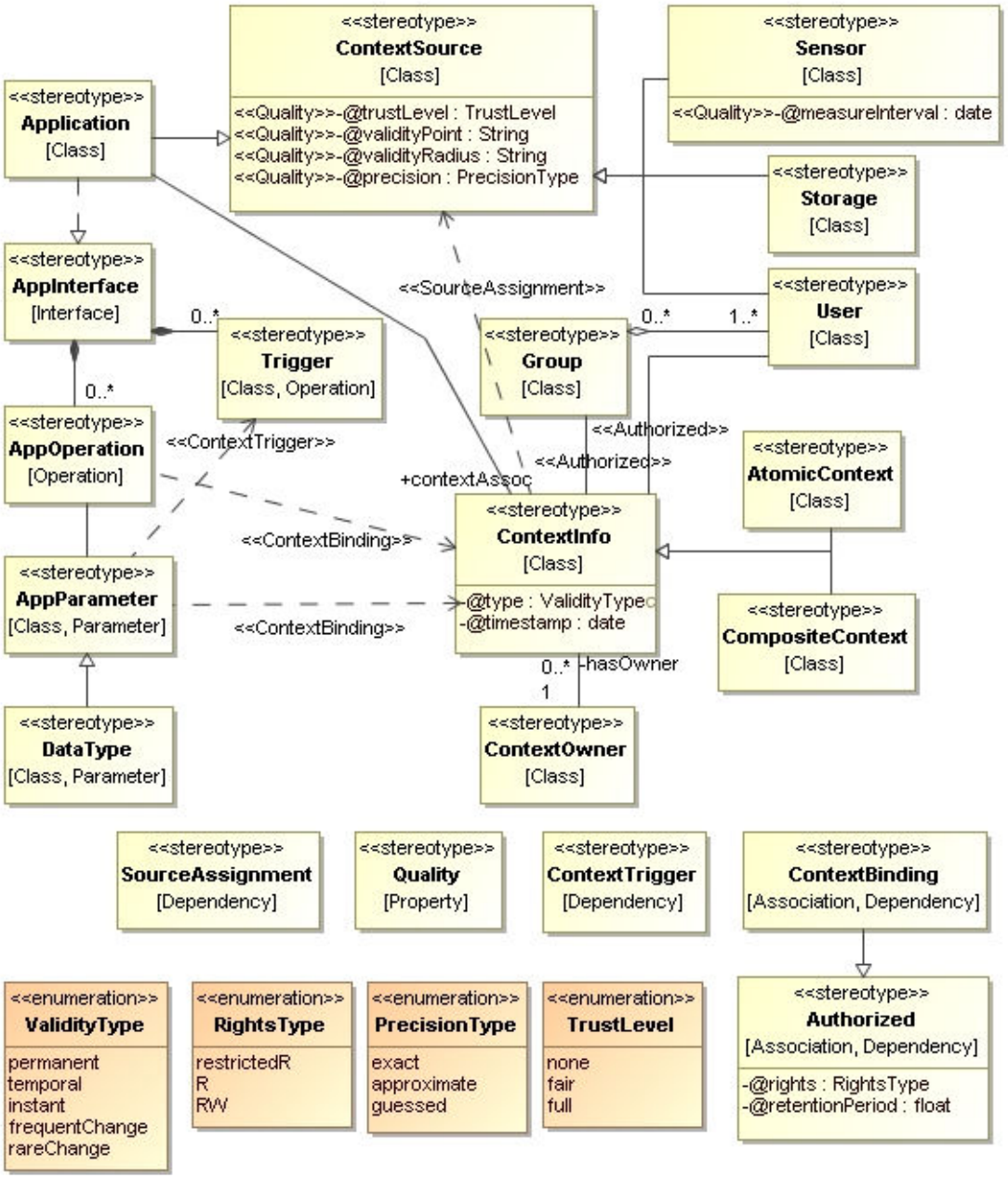
Η προστασία της ιδιωτικότητας, δηλ. η προστασία της διάδοσης και χρήσης των ευαίσθητων δεδομένων του χρήστη, αποτελεί ένα σημαντικό ζήτημα για τις εφαρμογές με επίγνωση του πλαισίου χρήσης, οι οποίες σχετίζονται τόσο με προσωπικά στοιχεία του χρήστη όσο και με πληροφορίες για την τρέχουσα κατάσταση και τη δραστηριότητα του. Σε αυτό το πλαίσιο μια ενδιαφέρουσα επέκταση θα ήταν η προσθήκη στοιχείων διασφάλισης ιδιωτικότητας σε όλα τα στάδια ανάπτυξης τόσο σε επίπεδο σχεδίασης όσο και σε επίπεδο υλοποίησης (μέσω κατάλληλης επέκτασης του μηχανισμού προσαρμογής). Προσεγγίσεις μοντελοποίησης του πλαισίου χρήσης που λαμβάνουν υπόψη θέματα ιδιωτικότητας είναι

διαθέσιμες στη διεθνή βιβλιογραφία. Ένα παράδειγμα συναντάται στο UML Προφίλ Μοντελοποίησης Πληροφορίας Πλαισίου Χρήσης (Context Modeling Profile / CMP) [6]. Στο CMP υπάρχουν εκφράσεις που καθορίζουν ποιος έχει πρόσβαση σε συγκεκριμένες τιμές πλαισίου χρήσης (π.χ. όλοι, συγκεκριμένες ομάδες, κτλ.), ενώ εισάγονται και περιορισμοί σε OCL εκφράσεις. Το θέμα της ιδιωτικότητας μελετάται και στο [2], όπου προτείνεται ένα μοντέλο βασισμένο όχι σε UML, αλλά στη μέθοδο Object-Role Modeling (ORM) [5]. Στο μοντέλο εισάγονται οι έννοιες της ιδιοκτησίας πλαισίου χρήσης και οι προτιμήσεις του χρήστη για την αποκάλυψη ή μη των στοιχείων πλαισίου χρήσης που τον αφορούν.

Στην κατεύθυνση αυτής της επέκτασης έχει προταθεί ένα προφίλ σε UML που λαμβάνει υπόψη – μεταξύ άλλων – την προστασία της ιδιωτικότητας: το μοντέλο πλαισίου χρήσης με επίγνωση ιδιωτικότητας (Privacy-Aware Context Profile / PCP) [4] που αποτελεί μια γενίκευση και επέκταση του προφίλ ExtContextUML που παρουσιάστηκε στη διατριβή. Η αρχική έκδοση του PCP που περιλαμβάνει κάποια στοιχεία προστασίας ιδιωτικότητας παρουσιάζεται στο διάγραμμα της Εικόνας 66.

9.2.2 Υποστήριξη Ιστορικών Τιμών Πλαισίου Χρήσης

Μια ακόμα ενδιαφέρουσα επέκταση της προτεινόμενης λύσης είναι η χρήση ιστορικών τιμών για το πλαίσιο χρήσης κατά την προσαρμογή της υπηρεσίας. Μέσω της επέκτασης θα δίνεται η δυνατότητα αποθήκευσης των πληροφοριών πλαισίου χρήσης επιτρέποντας την προσθήκη και ανάκτηση παλαιότερων τιμών, καθώς και τον ευφυή συνδυασμό των παλαιότερων τιμών για τη δημιουργία σύνθετων πληροφοριών context. Με αυτόν τον τρόπο διάφορες οντότητες θα είναι σε θέση να εκμεταλλευτούν τη διαθέσιμη (αποθηκευμένη) πληροφορία για την προσαρμογή της συμπεριφοράς τους σε προηγούμενες τιμές πλαισίου χρήσης, καθώς και σε ομάδες γεγονότων που υποδεικνύουν την κατάσταση του χρήστη (π.χ. η παρουσία πληροφοριών για διαφορετικές τοποθεσίες του χρήστη υποδεικνύει ότι ο χρήστης πραγματοποιεί κάποιο ταξίδι). Η συγκεκριμένη προσέγγιση θα μπορούσε να συνδυαστεί με διαθέσιμες ερευνητικές εργασίες που υποστηρίζουν την εξάρτηση από ιστορικές τιμές πλαισίου χρήσης προτείνοντας επεκτάσεις γλωσσών προγραμματισμού (π.χ. [3]). Η προσθήκη μηχανισμού αποθήκευσης αναιρεί μεν εν μέρει το διαχωρισμό της διαδικασίας προσαρμογής από τη λογική της εφαρμογής, αποτελεί δε σημαντική λειτουργία (με την ευρύτερη έννοια του πλαισίου χρήσης) που θα μπορούσε να αποδώσει σημαντικά αποτελέσματα.



Εικόνα 66. Προφίλ Πλαισίου Χρήσης με στοιχεία προστασίας Ιδιωτικότητας.

9.2.3 Μεταφορά Μηχανισμού στο Σύστημα του Έργου SMS

Το έργο Simple Mobile Services (SMS) [7] είναι ένα ευρωπαϊκό ερευνητικό έργο (IST 2006 034620) που έχει ως θέμα την παροχή απλών υπηρεσιών σε κινητές συσκευές, προσφέροντας προηγμένα εργαλεία ανάπτυξης και σχεδιασμού για χρήστες μη εξοικειωμένους με τις προγραμματιστικές αρχές. Στα πλαίσια του συγκεκριμένου έργου έχει αναπτυχθεί το Ανεξάρτητο Στρώμα Απλού Μεσισμικού (Simple Middleware Independent LayEr / SMILE) [1] που αναφέρεται στην ανταλλαγή SMILE μηνυμάτων μεταξύ οντοτήτων (που αναφέρονται ως smile peers) για τη διενέργεια διαφόρων λειτουργιών για

κατανεμημένες κινητές εφαρμογές. Μια ενδιαφέρουσα επέκταση του συγκεκριμένου συστήματος θα ήταν η ενσωμάτωση του μηχανισμού προσαρμογής στο πλαίσιο χρήσης που αναπτύχθηκε στα πλαίσια της διατριβής για τις υπηρεσίες διαδικτύου στο στρώμα μεσισμικού SMILE. Συγκεκριμένα, με τον ίδιο τρόπο που πραγματοποιείται η "σύλληψη" SOAP μηνυμάτων, θα γίνεται αντίστοιχα η σύλληψη και κατάλληλη τροποποίηση των SMILE μηνυμάτων που ανταλάσσονται μεταξύ των οντοτήτων. Στα μηνύματα περιλαμβάνονται πληροφορίες παραμέτρων για τις μεθόδους που καλούνται, κάτι που καθιστά εφικτή την κατάλληλη τροποποίηση αυτής της πληροφορίας βάσει της τρέχουσας κατάστασης πλαισίου χρήσης.

9.3 Βιβλιογραφία

[1] Bartolomeo, G., Salsano, S., Melazzi, N. B., Trubiani, C., "SAA 01-3 - SMILE- Simple Middleware Independent LayEr for Distributed Mobile Applications", Proc. IEEE Wireless Communications and Networking Conference (WCNC 2008), 2008, pp. 3039-3044.

[2] Henricksen, K., Wishart, R., McFadden, T., Indulska, J., "Extending context models for privacy in pervasive computing environments", Proc. 3rd International Conference on Pervasive Computing and Communication Workshops (PerCom'05 Workshops), 2005, pp. 20-24.

[3] Herzeel, C., Gybels, K., Costanza, P., Roover, C. D., D'Hondt, T., "Forward Chaining in HALO: An Implementation Strategy for History-based Logic Pointcuts", Proc. 2007 international conference on Dynamic languages: in conjunction with the 15th International Smalltalk Joint Conference 2007, pp. 157-182.

[4] Georgia M. Kapitsaki, Iakovos S. Venieris, "PCP: Privacy-aware Context Profile towards Context-aware Application Development", Proc. 10th International Conference on Information Integration and Web-based Applications and Services (iiWAS'08), 2008, pp. 104-110.

[5] "Object Role Modeling: An Overview", November 2001, http://msdn.microsoft.com/en-us/library/aa290383(VS.71).aspx.

[6] Simons, C., “CMP: A UML Context Modeling Profile for Mobile Distributed Systems”, Proc. 40th Annual Hawaii International Conference on System Sciences (HICSS’07), 2007, pp. 289b-289b.

[7] Simple Mobile Services (SMS) project, http://www.ist-sms.org/.

Παράρτημα Α΄

"Οδηγός" για την παραγωγή κλάσεων μηνυμάτων

```
#if (${util.getStereotype($class, null)} == "ExtContextUML__ComplexType")
   package gr.ntua.icbnet.${class.getModel().getName()}.ws;
#else
   package wsgenerated.${class.getPackage().getName()};
#end

import java.util.*;
import java.io.Serializable;
#foreach($attr in ${class.getOwnedAttributes()}) ##A
#if ($attr.getType().getName())
import gr.ntua.icbnet.${class.getModel().getName()}.ws.*;

#end
#end ##A

**
 *
 * Generated Class $class.getName()
 *
 */

 public class $class.getName() #if (${util.getStereotype($class, null)} ==
"ExtContextUML__ComplexType") implements Serializable #end {
    #foreach($attr in ${class.getOwnedAttributes()})
    #if (${util.getPropertyDatatype($attr).indexOf("[]")}>-1) ##2
    protected List<$datautil.removeStringBrackets(${util.getPropertyDatatype($attr)})>
${attr.getName()};
    #else
    protected ${util.getPropertyDatatype($attr)} ${attr.getName()};
    #end ##2
    #end

    public $class.getName()() {
    }

#foreach($op in $class.getOwnedOperations())
    public ${op.getType().getName()} $op.getName() ( .. ) {
    //Developer TODO: implement operation logic
        return null;
    }
#end

    //Getters and setters for attributes
```

```
#foreach($attr in $class.getOwnedAttributes())
  #if (${util.getPropertyDatatype($attr).indexOf("[]")}>-1) ##2
    public List<$datautil.removeStringBrackets(${util.getPropertyDatatype($attr)})>
get${attr.getName().substring(0,1).toUpperCase()}${attr.getName().substring(1,$attr.get
Name().length())}() {
      return this.$attr.getName();
    }

    public void
set${attr.getName().substring(0,1).toUpperCase()}${attr.getName().substring(1,$attr.get
Name().length())}(List<$datautil.removeStringBrackets(${util.getPropertyDatatype($attr)})
> val) {
      this.$attr.getName() = val;
    }
    #else
    public ${util.getPropertyDatatype($attr)}
get${attr.getName().substring(0,1).toUpperCase()}${attr.getName().substring(1,$attr.get
Name().length())}() {
      return this.$attr.getName();
    }

    public void
set${attr.getName().substring(0,1).toUpperCase()}${attr.getName().substring(1,$attr.get
Name().length())}(${util.getPropertyDatatype($attr)} val) {
      this.$attr.getName() = val;
    }
    #end
#end
}
```

Παράρτημα Β'

XML Σχήμα μη λειτουργικών ιδιοτήτων

```
<?xml version="1.0" encoding="UTF-8"?>
...
<xs:element name="descriptor" type="ws:WSDescriptor"/>
...
 <xs:complexType name="semantics">
  <xs:sequence>
   <xs:element name="version" minOccurs="0" maxOccurs="1"/>
   <xs:element name="tag" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <xs:simpleContent>
       <xs:extension base="xs:string">
        <xs:attribute name="key" type="xs:string" use="optional"/>
       </xs:extension>
     </xs:simpleContent>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="security">
  <xs:sequence>
   <xs:element name="policy" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="restriction" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
    <xs:element name="privacySupport" type="xs:boolean" minOccurs="0"
maxOccurs="1"/>
 ...
  </xs:sequence>
 </xs:complexType>
  <xs:complexType name="configuration">
   <xs:sequence>
    <xs:element name="software" type="ws:software" minOccurs="0" maxOccurs="1" />
    <xs:element name="hardware" type="ws:hardware" minOccurs="0" maxOccurs="1"
/>
    <xs:element name="network" type="ws:network" minOccurs="0" maxOccurs="1" />
   </xs:sequence>
 </xs:complexType>
 <xs:complexType name="software">
  <xs:sequence>
   <xs:element name="operatingSystem"> … </xs:element>
   <xs:element name="browserName">
    <xs:complexType>
      <xs:attribute name="name" type="xs:string"/>
      <xs:attribute name="version" type="xs:string"/>
    </xs:complexType>
    </xs:element>
    <xs:element name="htmlVersion" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

```
      </xs:sequence>
      </xs:complexType>
      <xs:complexType name="hardware"> ... </xs:complexType>
      <xs:complexType name="network">
       <xs:sequence>
        <xs:element name="binding" type="xs:string" minOccurs="0" maxOccurs="1"/>
         <xs:element name="communicationProtocol" type="xs:string" minOccurs="0"/>
       </xs:sequence>
      </xs:complexType>
    <xs:complexType name="locationAvailability">
    <xs:sequence>
     <xs:element maxOccurs="1" minOccurs="0" name="hasLocationRestriction"
type="xs:boolean"/>
     <xs:element maxOccurs="unbounded" minOccurs="0" name="country"
type="xs:string"/>
     <xs:element maxOccurs="unbounded" minOccurs="0" name="province"
type="xs:string"/>
     <xs:element maxOccurs="unbounded" minOccurs="0" name="city"
type="xs:string"/>
     <xs:element maxOccurs="unbounded" minOccurs="0" name="area"
type="xs:string"/>
     <xs:element maxOccurs="unbounded" minOccurs="0" name="address"
type="xs:string"/>
     <xs:element maxOccurs="unbounded" minOccurs="0" name="geocodes"
type="ws:geocodesType"/>
    </xs:sequence>
    </xs:complexType>
    <xs:complexType name="timeAvailability">
   <xs:sequence>
    <xs:element maxOccurs="1" name="isContinuouslyAvailable" type="xs:boolean"/>
    <xs:element name="availableAtOrDuring" type="ws:temporalType"/>
   </xs:sequence>
   </xs:complexType>
   ...
   <xs:complexType name="TemporalType">
    <xs:choice>
     <xs:element minOccurs="0" name="WeeklyAvailable">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="startWeekDay" type="xs:string"/>
        <xs:element minOccurs="0" name="endWeekDay" type="xs:string"/>
        <xs:element name="time" type="ws:TimeInterval"/>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
     <xs:element name="PeriodAvailable">
      <xs:complexType>
       <xs:sequence>
        <xs:element name="startDate" type="xs:date"/>
        <xs:element minOccurs="0" name="endDate" type="xs:date"/>
        <xs:element name="time" type="eus:TimeInterval"/>
       </xs:sequence>
      </xs:complexType>
     </xs:element>
   ...
    </xs:choice>
   </xs:complexType>
   ...
   </xs:schema>
```

Ακρωνύμια

BWS Business Web Service

CASE Computer-Aided Software Engineering

CORBA Common Object Request Broker Architecture

CWS Context Web Service

EAI Enterprise Application Integration

EJB Enterprise JavaBeans

EMF Eclpse Modeling Framework

ERP Enterprise resource planning

GPS Global Positioning System

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

JMS Java Message Service

JSP Java Server Pages

JSON JavaScript Object Notation

IP Internet Protocol

MDA Model Driven Architecture

MDE Model Driven Engineering

MOF MetaObject Facility

OCL Object Constraint Language

OMG Object Management Group

ORM Object-Role Modeling

OWL Web Ontology Language

PIM Platform Independent Model

PSM Platform Specific Model

QVT Query/View/Transformation

RAS Reusable Asset Specification

REST Representational State Transfer

SOAP Simple Object Access Protocol

UML Unified Modeling Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

WS Web Services

W3C World Wide Web Consortium

WSDL Web Services Description Language

XMI XML Metadata Interchange

XML Extensible Markup Language

Λίστα Δημοσιεύσεων

Διεθνή Περιοδικά Πλήρους Κειμένου, με Κρίση

[1] Kapitsaki, G. M., Prezerakos, G. N., Tselikas, N. D., Venieris, I. S., “Context-aware Service Engineering: A Survey”, Elsevier Journal of Software and Systems, 2009, in press, DOI: 10.1016/j.jss.2009.02.026.

[2] Kapitsaki, G. M., Kateros, D. A., Prezerakos, G. N., Venieris, I. S., “Model-driven Development of Composite Context-aware Web Applications”, Elsevier Journal of Information and Software Technology Volume 51, Issue 8**,** pp. 1244-1260, 2009, DOI: 10.1016/j.infsof.2009.03.002.

Κεφάλαια Βιβλίων

[1] Tselikas, N. D., Kapitsaki, G. M., Dimitrios Makris and Venieris, I. S., "Open APIs and Protocols for Services and Applications in Telecoms", Book chapter in Handbook of Research on Telecommunications Planning and Management, Information Science Reference, February 2009.

Πρακτικά Διεθνών Συνεδρίων, με Κρίση

[1] Broll, G., Hußmann, H., Prezerakos, G. N., Kapitsaki, G., Salsano, S., “Modeling Context Information for Realizing Simple Mobile Services”, Proc. 16th IST Mobile and Wireless Communications Summit, Budapest, Hungary, July, 2007, pp. 1-5.

[2] Kapitsaki, G., Kateros, D. A., Foukarakis, I. E., Prezerakos, G. N., Kaklamani, D. I., Venieris, I. S., “Service Composition: State of the art and future challenges”, Proc. 16th IST Mobile and Wireless Communications Summit, Budapest, Hungary, July, 2007, pp. 1-5.

[3] Kapitsaki, G. M., Kateros, D. A., Lioudakis, G. V., Venieris, I. S., “Extending Reusable Asset Specification to Describe Simple Mobile Services”, Proc. 17th IST Mobile and Wireless Communications Summit, Stockholm, Sweden, June, 2008, pp. 1-8.

[4] Kateros, D. A., Kapitsaki, G. M., Tselikas, N. D., Venieris, I. S., "A Methodology for Model-Driven Web Application Composition", Proc. 2008 IEEE International Conference on Services Computing (SCC'08), Honolulu, Hawaii, USA, July, 2008, vol. 2, pp. 489-492.

[5] Kapitsaki, G. M., Kateros, D. A., Venieris, I. S., "Architecture for Provision of Context-aware Web Applications based on Web Services", Proc. IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'08), Cannes, France, September 14-18, 2008, pp. 1-5.

[6] Kapitsaki, G. M., Venieris, I. S., "Model-Driven development of Context-aware Web Applications based on a Web service Context Management Architecture", Proc. ACM/IEEE 11th International Conference on Model Driven Engineering Languages and Systems (MODELS'08), Toulouse, France, September 29, 2008, *Best Paper Award.*

[7] Lioudakis, G. V., Dellas, N. L., Koutsoloukas, E. A., Kapitsaki, G. M., Kaklamani, D. I., Venieris, I. S., "A Semantic Framework for Privacy-Aware Access Control", Proc. 3rd International Workshop on Secure Information Systems (SIS'08), International Multiconference on Computer Science and Information technology, Wisla, Poland, October 20-22, 2008, pp. 813-820.

[8] Kapitsaki, G. M., Venieris, I. S., "PCP: Privacy-aware Context Profile towards Context-aware Application Development", Proc. 10th International Conference on Information Integration and Web-based Applications and Services (iiWAS'08), Linz, Austria, 24-26 November, 2008, pp. 104-110.

[9] Kapitsaki, G. M., Kateros, D. A., Pappas, C., Tselikas, N. D., Venieris, I. S., "Model-Driven Development of Composite Web Applications", Proc. 10th International Conference on Information Integration and Web-based Applications and Services (iiWAS'08), Linz, Austria, 24-26 November, 2008, pp. 399-402.

Λίστα Αναφορών

Kapitsaki, G., Kateros, D. A., Foukarakis, I. E., Prezerakos, G. N., Kaklamani, D. I., Venieris, I. S., "Service Composition: State of the art and future challenges", 16th IST Mobile and Wireless Communications Summit, Budapest, Hungary, July, 2007, pp. 1-5.

[1] Shin, Y., Yu, C., Chung, S., Kim, S., “End-user driven Service Creation for Converged Service of Telecom and Internet” , Proc. Fourth Advanced International Conference on Telecommunications, 2008, pp. 71-76.

[2] Zhang, X., “Personalized ESG for converged digital broadcast and 3G mobile services”, Master thesis, CICT·Technical University of Denmark, January 2008.

[3] Xiaoning, M., Baotian, D., Ming, H., “AND/OR tree search algorithm in web service composition”, Proc. 2008 Pacific-Asia Workshop on Computational Intelligence and Industrial Application (PACIIA’08), Volume 2, 2008, pp. 23-27.

Foukarakis, I., Kapitsaki, G.., Kateros, D. et al., “Sms project deliverable 4.3.1-design of service authoring wizard”, Technical report, 2007.

[1] Balagtas-Fernandez, F. T., Hussmann, H., “Model-Driven Development of Mobile Applications”, Proc. 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE’08), September 2008, pp. 509-512.