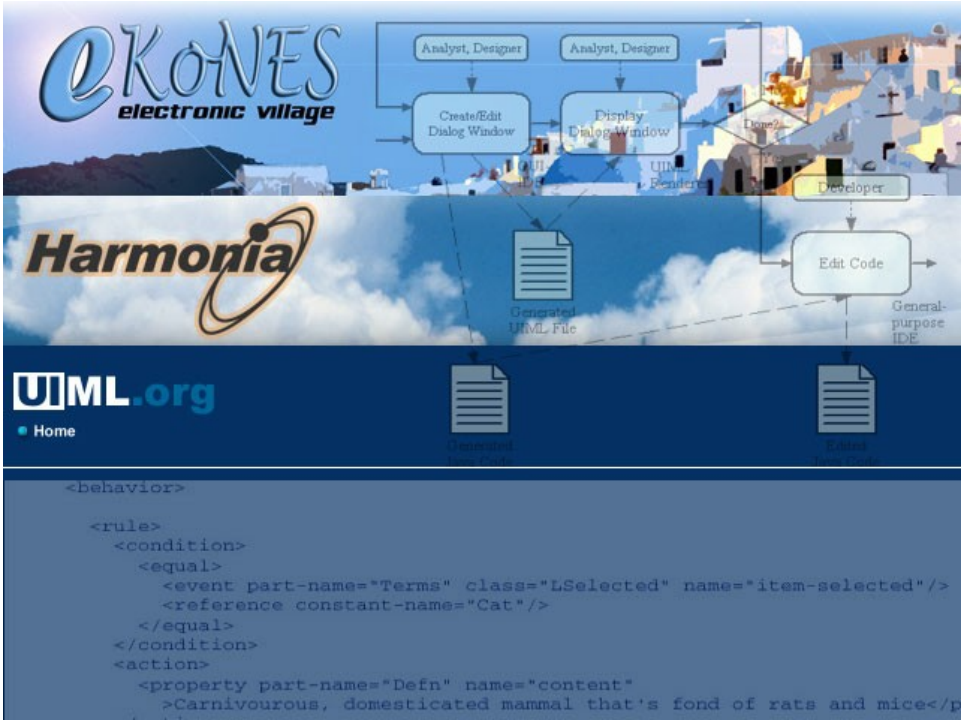
ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΠΟΛΥΜΕΣΩΝ

Πτυχιακή εργασία

Σπουδαστές: Δολαψάκης Παύλος , Κασωτάκης Στέφανος

Επιβλέπων Καθηγητής: Δημοσθένης Ακουμιανάκης



ΘΕΜΑ: Τεχνικές και εργαλεία ανάπτυξης διεπαφών βάση μοντέλων

Οκτώβριος 2007

Πίνακας περιεχομένων

1. Εισαγωγή....7

2 Γλώσσες ανάπτυξης διεπαφών....8

2.1 Συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου....8

2.1.1 Γλώσσες υψηλού επιπέδου....8

2.1.2 Γλώσσες σήμανσης εξαρτημένες από τη συσκευή....12

2.1.3 Γλώσσες σήμανσης διεπαφών ανεξάρτητες συσκευής....18

2.2 Τεχνικές ανάπτυξης διεπαφών με τη χρήση μοντέλων....20

2.2.1 UIDE....21

2.2.2 HUMANOID....21

2.2.3 MASTERMIND....22

2.2.4 Teresa....23

3 Η Προσέγγιση της UIML....25

3.1 Τι είναι η UIML....25

3.1.1 Κίνητρα για τη χρήση UIML....26

3.1.2 Γλώσσα σήμανσης διεπαφών ανεξάρτητη συσκευής....26

3.2 Το μοντέλο UIML....28

3.3 Παραδείγματα....30

3.3.1 Παράδειγμα 1: LayoutBorder.uiml....30

3.3.2 Παράδειγμα 2: Dictionary.uiml....33

4 Εφαρμογές στο έργο eKoNEΣ....37

4.1 Τι είναι το eKoNEΣ....37

4.2 Ανάπτυξη και επεξήγηση Container σε UIML....38

4.3 Κλήση εξωτερικών κλάσεων Java....45

4.3.1 Περιγραφή της ετικέτας <call>....45

4.3.2 Κλήση εξωτερικών κλάσεων Java στο έργο eKoNEΣ....52

4.4 Τα εργαλεία Liquid για την ανάπτυξη UIML εγγράφων....54

4.4.1 uDevelop....55

4.4.2 uBrowse....60

5 Συμπεράσματα....62

6 Επίλογος....63

Βιβλιογραφία....64

Σχήματα

Σχήμα 1: Πρωτότυπο διεπαφής υλοποιημένο σε Java.....11

Σχήμα 2: Παραδείγματα ιεραρχιών διαδραστικών αντικειμένων.....14

Σχήμα 3: Απεικόνιση ενός WAP δικτύου.....15

Σχήμα 4: Ροή κίνησης πληροφορίας.....16

Σχήμα 5: Ρόλος των ανεξάρτητων από τη συσκευή γλωσσών σήμανσης.....19

Σχήμα 6: Κατηγορίες μοντέλων που χρησιμοποιούνται για την ανάπτυξη διεπαφών.....20

Σχήμα 7: Αρχιτεκτονικές.....21

Σχήμα 8: Απεικόνιση συστήματος Teresa.....23

Σχήμα 9: Εργαλεία ανάπτυξης διεπαφών και φορητότητα παραγόμενης διεπαφής 25

Σχήμα 10: Ρόλος των ανεξάρτητων από τη συσκευή γλωσσών σήμανσης.....26

Σχήμα 11: Το μοντέλο της UIML.....29

Σχήμα 12: Στιγμιότυπο διαχειριστή διίαταξης – Αρχική μορφή πλαισίου.....30

Σχήμα 13: Στιγμιότυπο διαχειριστή διίαταξης – Μεγέθυνση πλαισίου.....31

Σχήμα 14: Στιγμιότυπο διαχειριστή διίαταξης – Σμίκρυνση πλαισίου.....31

Σχήμα 15: Στιγμιότυπο Dictionary – Πριν την επιλογή αντικειμένου.....33

Σχήμα 16: Στιγμιότυπο Dictionary – Μετά την επιλογή αντικειμένου.....33

Σχήμα 17: Λειτουργική όψη eKONEΣ.....37

Σχήμα 18: Πρωτότυπο διαπαφής με τη χρήση UIML.....38

Σχήμα 19: Στιγμιότυπο Counter1.....47

Σχήμα 20: Στιγμυότυπο Counter1 – Πάτημα "Press me" και αύξηση του μετρητή.....47

Σχήμα 21: Στιγμιότυπο Counter2.....49

Σχήμα 22: Στιγμιότυπο Counter2 – Πάτημα "Count" JButton και αύξηση μετρητή.....50

Σχήμα 23: Στιγμυότυπο Counter2 – Πάτημα "Reset" JButton και μηδενισμός μετρητή.....50

Σχήμα 24: Πρωτότυπο διαπαφής υλοποιμένο σε UIML.....52

Σχήμα 25: Πλαίσιο υλοποιημένο σε Java – Επιλογές αλληλεπίδραση με βάση δεδομένων 53

Σχήμα 26: Ενδιάμεσο πλαίσιο μετά το πάτημα του "Add" button.....53

Σχήμα 27: Αποτέλεσμα ενέργειας "Execute Update".....53

Σχήμα 28: Στιγμιότυπο uDevelop – Επιλογή Workspace.....55

Σχήμα 29: Στιγμιότυπο uDevelop – Δημιουργία νέου Project.....55

Σχήμα 30: Στιγμιότυπο uDevelop – Δημιουργία νέου Project.....56

Σχήμα 31: Στιγμιότυπο uDevelop – Δημιουργία νέου UIML file.....56

Σχήμα 32: Στιγμιότυπο uDevelop – Επιλογή Top Container.....57

Σχήμα 33: Στιγμιότυπο uDevelop – Επιφάνεια εργασίας.....57

Σχήμα 34: Στιγμιότυπο uDevelop – Swing Components.....58

Σχήμα 35: Η δυνατότητα rendering ενός UIML εγγράφου.....58

Σχήμα 36: Το παράθυρο που παράγεται μετά την αντιστοίχηση στο λεκτικό του Swing 59

Σχήμα 37: Στιγμιότυπο uDevelop – Μορφοποίηση Container.....59

Σχήμα 38: Στιγμιότυπο uDevelop – Properties Container.....59

Σχήμα 39: Στιγμιότυπο uDevelop – Properties Container υπό μορφή κώδικα.....60

Σχήμα 40 : Στιγμιότυπο uDevelop – Παρουσίαση κώδικα διεπαφής 60

Σχήμα 41: Στιγμιότυπο uBrowse – Επιλογή UIML file για προεπισκόπιση.....61

Σχήμα 42: Στιγμιότυπο uBrowse – Ενεργοποίηση Renderer.....61

Σχήμα 43: Στιγμιότυπο uBrowse – Αποτέλεσμα επιλογής ″Render to Java″ 61

Ευχαριστίες

Με την ολοκλήρωση της πτυχιακής μας εργασίας θα θέλαμε να ευχαριστήσουμε όλους όσους μας υποστήριξαν. Συγκεκριμένα, τον επόπτη-καθηγητή κύριο Δημοσθένη Ακουμιανάκη για τις πολύτιμες συμβουλές και την καθοδήγησή του κατά την εκπόνηση της εργασίας. Την κοινοπραξία Harmonia και ιδιαίτερα τον James Helms για την άψογη συνεργασία και τη διάθεση απαραίτητου υλικού. Το τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων του Τεχνολογικού Εκπαιδευτικού Ιδρύματος Κρήτης για τη φιλοξενία του στα χρόνια της ακαδημαϊκής μας πορείας, αλλά και για την εξασφάλιση των γνώσεων ώστε να ανταπεξέλθουμε στις δυσκολίες της μετέπειτα επαγγελματικής μας σταδιοδρομίας. Τέλος θα ήταν παράλειψη να μην ευχαριστήσουμε τις οικογένειές μας για την απεριόριστη ηθική υποστήριξη και κατανόηση.

Περίληψη

Η πτυχιακή εργασία αναφέρεται στην επισκόπηση τεχνικών μεθόδων και συστημάτων για την ανάπτυξη διεπαφών με τη χρήση μοντέλων (model-based user interface development) και ιδιαίτερα σε μία πρόσφατη κατηγορία τεχνικών η οποία βασίζεται στη χρήση μετασχηματισμών για τη σταδιακή μετατροπή ενός αφηρημένου (abstract) μοντέλου διαπαφής σε μοντέλο συγκεκριμένης (concrete) δομής και λειτουργίας. Οι μετασχηματισμοί εφαρμόζονται σε αρχεία μιας γλώσσας σήμανσης ανεξάρτητης συσκευής (π.χ XML, UIML) τα οποία χρησιμοποιούντα για να καταγράψουν τόσο το αφηρημένο όσο και το συγκεκριμένο μοντέλο μιας διεπαφής. Με το τρόπο αυτό οι μετασχηματισμοί γίνονται ο κύριος μηχανισμός ημι-αυτόματης παραγωγής πολλαπλών διεπαφών (διεπαφών που τρέχουν σε διαφορετικές πλατφόρμες).

Με βάση τα παραπάνω η συγκεκριμένη πτυχιακή έχει σκοπό να εμβαθύνει σε πρόσφατες τεχνικές και εργαλεία μετασχηματισμού διεπαφών και να παρουσιάσει τη χρησιμότητά τους σ’ένα ενδεικτικό πεδίο εφαρμογής.

1. Εισαγωγή

Η συγκεκριμένη πτυχιακή εργασία είναι δομημένη σε τρεις θεματικές ενότητες.Στην πρώτη θεματική ενότητα (ΚΕΦ.2) αναλύονται οι γλώσσες ανάπτυξης διεπαφών. Συγκεκριμένα γίνεται αναφορά στις συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου και στις γλώσσες προγραμματισμού υψηλού επιπέδου όπου παρατηρείται ανάπτυξη πρωτοτύπων διεπαφής. Ακόμη περιγράφονται, ο διαχωρισμός τους σε γλώσσες σήμανσης εξαρτημένες από τη συσκευή (HTML, WML) και γλώσσες σήμανσης ανεξάρτητες συσκευής (XML) καθώς και τα συστήματα που υιοθετούν τη χρήση μοντέλων. Στη δεύτερη θεματική ενότητα (ΚΕΦ.3) γίνεται προσέγγιση της UIML (User Interface Markup Language) μιας γλώσσας σήμανσης υψηλού επιπέδου, παραθέτοντας το μοντέλο (Meta-Interface Model) και τους μετασχηματισμους που χρησιμοποιεί, με σκοπό την κατανόηση του διαχωρισμού της διεπαφής από τη λογική εφαρμογή και τη συσκευή παρουσίασης. Στην τρίτη θεματική ενότητα (ΚΕΦ.4) εμβαθύνουμε στην ανάλυση της UIML μέσα από σενάρια χρήσης του έργου eKoNEΣ και περιγράφουμε το εργαλείο ανάπτυξης διεπαφών uDevelop.

2 Γλώσσες ανάπτυξης διεπαφών

2.1 Συμβολικές γλώσσες ή γλώσσες χαμηλού επιπέδου

Αρχικά για να μπορέσει ο υπολογιστής να εκτελέσει μία οποιαδήποτε λειτουργία, έπρεπε να δοθούν κατευθείαν οι κατάλληλες ακολουθίες από 0 και 1, δηλαδή εντολές σε μορφή κατανοητή από τον υπολογιστή αλλά πολύ δύσκολα κατανοητές από τον άνθρωπο. Ο τρόπος αυτός ήταν επίπονος και ελάχιστοι μπορούσαν να τον υλοποιήσουν, αφού απαιτούσε βαθιά γνώση του υλικού και της αρχιτεκτονικής του υπολογιστή.
Οι εντολές που δέχεται ένας υπολογιστής μετατρέπονται σε ακολουθίες που αποτελούνται από 0 και 1, γίνονται δηλαδή εντολές σε **γλώσσα μηχανής** όπως ονομάζονται, έτσι ώστε να γίνουν κατανοητές από τον υπολογιστή και στη συνέχεια να μπορέσει να τις εκτελέσει.

Από τα πρώτα χρόνια άρχισαν να γίνονται προσπάθειες για τη δημιουργία μιας συμβολικής γλώσσας, η οποία ενώ θα έχει έννοια για τον άνθρωπο, θα μετατρέπεται εσωτερικά από τους υπολογιστές στις αντίστοιχες ακολουθίες από 0 και 1. Για παράδειγμα η λέξη ADD (πρόσθεσε) ακολουθούμενη από δυο αριθμούς, είναι κατανοητή από τον άνθρωπο και απομνημονεύεται σχετικά εύκολα. Η εντολή αυτή θα μεταφραστεί από τον υπολογιστή σε μία ακολουθία δυαδικών ψηφίων και στη συνέχεια μπορεί να εκτελεστεί. Το έργο της μετάφρασης το αναλαμβάνει ένα ειδικό πρόγραμμα, ο συμβολομεταφραστής (assembler).

2.1.1 Γλώσσες υψηλού επιπέδου

Πρωτότυπα διεπαφής μπορούν να αναπτυχθούν και με τη χρήση γλωσσών προγραμματισμού υψηλού επιπέδου. Θα πρέπει να τονίσουμε ότι οι γλώσσες αυτές είναι γενικού ενδιαφέροντος και χρησιμοποιούνται ευρύτατα στην ανάπτυξη συστημάτων λογισμικού.

Μια ιστορική αναδρομή γλωσσών υψηλού επιπέδου παρουσιάζεται παρακάτω :

Το 1957 η IBM ανάπτυξε την πρώτη γλώσσα υψηλού επιπέδου τη **FORTRAN**. Το όνομα FORTRAN προέρχεται από τις λέξεις **FOR**mula **TRAN**slation που σημαίνουν μετάφραση τύπων. Η FORTRAN αναπτύχθηκε ως γλώσσα κατάλληλη για την επίλυση μαθηματικών και επιστημονικών προβλημάτων.

Το 1960 αναπτύχθηκε μία άλλη γλώσσα, σταθμός στον προγραμματισμό η γλώσσα **COBOL**. Η COBOL όπως δηλώνει και το όνομα της (**CO**mmon **B**usiness **O**riented **L**anguage-Κοινή γλώσσα προσανατολισμένη στις επιχειρήσεις) είναι κατάλληλη για ανάπτυξη εμπορικών εφαρμογών, και γενικότερα διαχειριστικών εφαρμογών, τομέας όπου η FORTRAN υστερούσε.

Μια από τις σημαντικότερες γλώσσα προγραμματισμού με ελάχιστη πρακτική εφαρμογή αλλά που επηρέασε ιδιαίτερα τον προγραμματισμό και τις επόμενες γλώσσες είναι η **ALGOL** (**Algo**rithmic **L**anguage-Αλγοριθμική γλώσσα).

Στο χώρο της Τεχνητής Νοημοσύνης αναπτύχθηκαν δυο γλώσσες διαφορετικές από τις άλλες. Στα μέσα του 60 αναπτύχθηκε η **LISP** (**LIS**t **P**rocessor- Επεξεργαστής λίστας), γλώσσα η οποία προσανατολίζεται σε χειρισμό λιστών από τα σύμβολα και η **PROLOG** (**PRO**gramming **LOG**ic- Λογικός Προγραμματισμός) στις αρχές του 70.

Δύο σημαντικότατες γλώσσες γενικού σκοπού, οι οποίες αναπτύχθηκαν τη δεκαετία του 60 αλλά χρησιμοποιούνται πάρα πολύ στις ημέρες μας, είναι η BASIC και η PASCAL.

Η γλώσσα προγραμματισμού **BASIC** (**B**eginner's **A**ll **P**urpose **S**ymbolic **C**ode-Συμβολικός Κώδικας Εντολών Γενικής Χρήσης για Αρχάριους) αρχικά αναπτύχθηκε, όπως δηλώνει και το όνομα της, ως γλώσσα για την εκπαίδευση αρχαρίων στον προγραμματισμό. Σχεδιάστηκε για να γράφονται σύντομα προγράμματα, τα οποία εκτελούνται με τη βοήθεια διερμηνευτή (interpreter).

Η γλώσσα **PASCAL** έφερε μεγάλες αλλαγές στον προγραμματισμό. Παρουσιάστηκε το 1970 και στηρίχθηκε πάνω στην ALGOL. Είναι μια γλώσσα γενικής χρήσης, η οποία είναι κατάλληλη τόσο για την εκπαίδευση όσο και τη δημιουργία ισχυρών προγραμμάτων κάθε τύπου. Χαρακτηριστικό της γλώσσας είναι η καταλληλότητα για τη δημιουργία δομημένων προγραμμάτων.

Μία ακόμη γλώσσα που γνώρισε μεγάλη διάδοση είναι η γλώσσα **C**. Η C χρησιμοποιήθηκε για την ανάπτυξη του λειτουργικού συστήματος Unix γλώσσα με ισχυρά χαρακτηριστικά, μερικά από αυτά κοινά με την Pascal κατάλληλη για την ανάπτυξη δομημένων εφαρμογών αλλά και με πολλές δυνατότητες γλώσσας χαμηλού επιπέδου. Η C εξελίχτηκε στη γλώσσα **C++,** που είναι αντικειμενοστραφής.

Τα τελευταία χρόνια χρησιμοποιείται ιδιαίτερα, ειδικά για προγραμματισμό στο Διαδίκτυο (Internet), η **JAVA**. Η JAVA είναι μια αντικειμενοστραφής γλώσσα που αναπτύχθηκε από την εταιρία SUN με σκοπό την ανάπτυξη εφαρμογών, που θα εκτελούνται σε κατανεμημένα περιβάλλοντα, δηλαδή σε διαφορετικούς υπολογιστές οι οποίοι είναι συνδεδεμένοι με το Διαδίκτυο.

Ένα από τα πιο ισχυρά συστήματα ανάπτυξης πρωτοτύπων διεπαφής χρήστη είναι το σύστημα της Smalltalk (Goldberg, 1984), αν και δεν υπήρξε ιδιαίτερα δημοφιλής. Η Smalltalk είναι μια αντικιμενοστραφής γλώσσα προγραμματισμού η οποία είναι «δεμένη» σφικτά με το περιβάλλον της , το οποίο περιλαμβάνει ένα γραφικό σύστημα διεπαφής χρήστη. Το μεγαλύτερο μέρος της αλληλεπίδρασης με το σύστημα γίνεται μέσω μενού, με επιλογές που γίνονται υποδεικνύοντας με το ποντίκι. Η SmallTalk είναι μια έξοχη γλώσσα ανάπτυξης πρωτοτύπων για δυο λόγους. Πρώτον, η αντικειμενοστραφής φύση της γλώσσας σημαίνει ότι τα συστήματα επιδέχονται αλλαγές. Η SmallTalk υποστηρίζει γρήγορες τροποποιήσεις ενός συστήματος χωρίς απρόβλεπτες συνέπειες στο υπόλοιπο σύστημα. Πράγματι η Smalltalk υποστηρίζει γρήγορες τροποποιήσεις ενός συστήματος χωρίς απρόβλεπτες συνέπειες στο υπόλοιπο σύστημα. Πράγματι, είναι κατάλληλη μόνο γι αυτό το είδος ανάπτυξης και δεν είναι καθόλου κατάλληλη για μια απλή, μονολιθική προσέγγιση ανάπτυξης

Δεύτερον, το σύστημα και το περιββάλον της Smalltalk είναι σύμφυτο τμήμα της γλώσσας. ‘Ολα τα αντικείμενα που ορίζονται είναι διαθέσιμα στον προγραμματιστή της SmallTalk. ‘Ετσι υπάρχει ένας μεγάλος αριθμός από επαναχρησιμοποιούμενα δομικά στοιχεία που είναι διαθέσιμα προς ενσωμάτωση στο υπό ανάπτυξη πρωτότυπο.

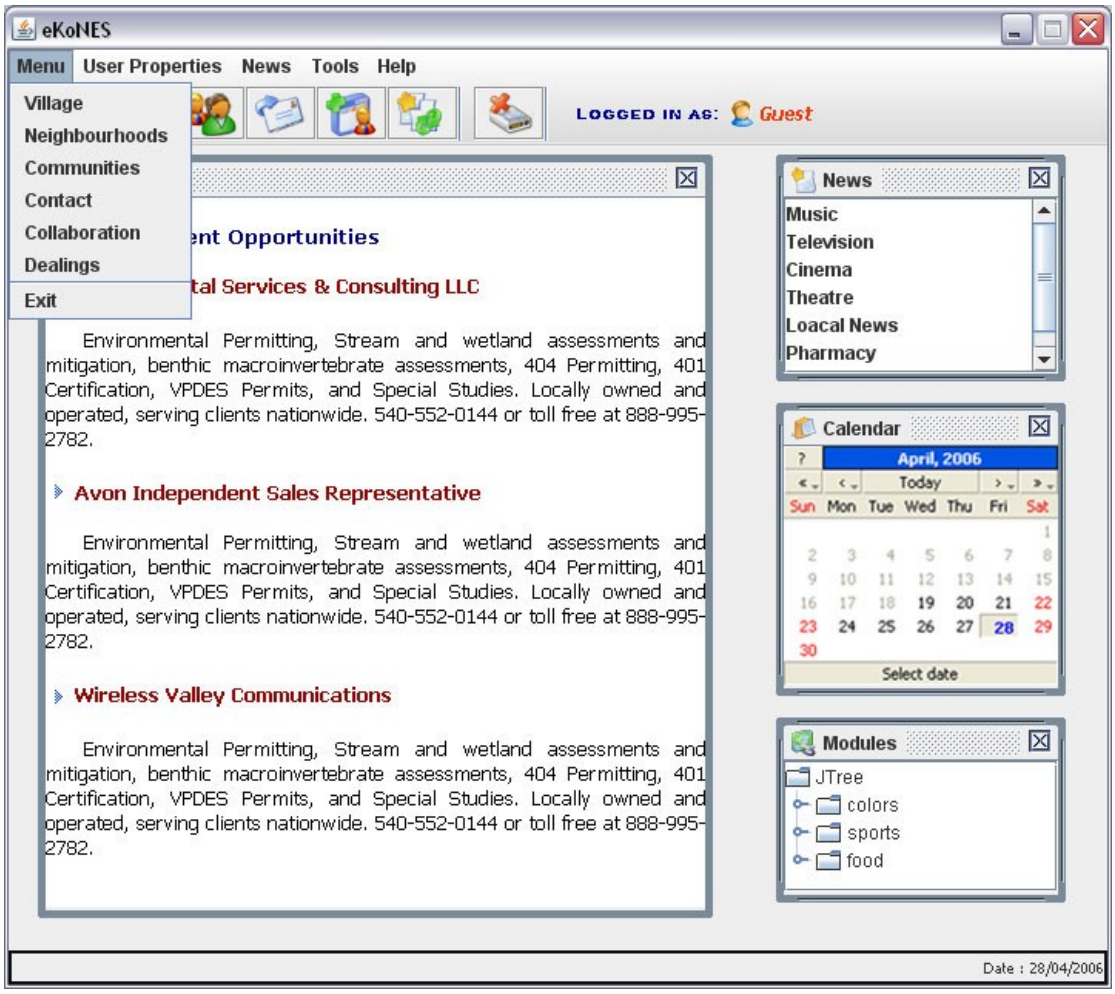
Η Smalltalk υπήρξε καινοτόμος στο βαθμό που επηρέασε καταλυτικά σύγχρονα ανντικειμενοστραφή εργαλεία και περιβάλλοντα προγραμματισμού όπως είναι η γλώσσα προγραμματισμού Java. Η Java σχεδιάστηκε για να δώσει λύσεις σε μια σειρά πρακτικών προβλημάτων που εφανίζονταν σε όλες τις ήδη υπάρχουσες γλώσσες προγραμματισμού όπως η C++. Ενδεικτικά αναφέρουμε ορισμένα χαρακτηριστικά της γλώσσας που την κάνουν σήμερα διιαιτερα δημοφιλή. Πρώτα από όλα η γλώσσα σχεδιάστηκε έτσι ώστε να είναι ανεξάρτητη αρχιτεκτονικής και να υποστηρίζει εφαρμογές που τρέχουν σε ένα μεγάλο σύνολο διαφορετικών μηχανών και λειτουργικών συστημάτων. Για να εξασφαλιστεί το ότι η ίδια εφαρμογή θα τρέχει ομοιογενώς σε αυτό το περιβάλλον, ο μεταγλωττιστής παράγει ένα ανεξάρτητο μηχανής κώδικα, ο οποίος με τη βοήθεια του συστήματος εκτέλεσης της Java μπορεί να τρέξει σε πολλούς διαφορετικούς επεξεργαστές. Το γεγονός αυτό δεν είναι μόνο χρήσιμο στη περίπτωση των δικτύων και των δικτυακών εφαρμογών αλλά και στις τοπικές εφαρμογές.

Μέχρι τώρα με την μεγάλη ποικιλία επεξεργαστών και μηχανημάτων που κυκλοφορούν στην αγορά, οι κατασκευαστές πακέτων λογισμικού είναι υποχρεωμένοι από τους νόμους της αγοράς να παράγουν το ίδιο προϊόν για πολλές διαφορετικές πλατφόρμες όπως για παράδειγμα UNIX, IBM PC, Apple, Macintosh. ‘Οσο περισσότερο προχωρά η αγορά επεξεργαστών, τόσο πιο πολύ εντείνεται το πρόβλημα. Με τη Java, η φιλοσοφία είναι ότι η εφαρμογή γράφεται μόνο μια φορά σε οποιαδήποτε πλατφόρμα και έπειτα θα μπορεί να τρέχει οπουδήποτε(WORA – Write Once, Run Anywhere)

‘Ενα δεύτερο χαρακτηριστικό της Java είναι η δυνατότητα παραγωγής μεταφέρσιμου (portable) κώδικα. Θα πρέπει να σημειώσουμε στο σημείο αυτό ότι το γεγονός ότι ο κώδικας που παράγει ένας μεταφραστής μιας γλώσσας προγραμματισμού είναι ανεξάρτητος αρχιτεκτονικής, δεν συνεπάγεται απαραίτητα πως θα είναι και μεταφέρσιμος. Για παράδειγμα, στη C και στη C++ υπάρχουν ζητήματα που εξαρτώνται από τη συγκεκριμένη αρχιτεκτονική στην οποία τρέχει ο μεταφραστής. Αντίθετα, στη Java, ακόμα και οι βιβλιοθήκες είναι μεταφέρσιμες σε όλες τις υποστηριζόμενες πλατφόρμες. ‘Ετσι για παράδειγμα, υπάρχει μια κλάση αντικειμένων για τα γραφικά παράθυρα που είναι ίδια για όλες τις UNIX,Windows 95/98/2000/NT, Mac πλατφόρμες.

Είναι προφανές ότι τα χαρακτηριστικά αυτά κάνουν τη Java ιδιαιτέρως δημοφιλή γλώσσα για την κατασκευή πρωτοτύπων διεπαφών. ‘Ενα παράδειγμα πρωτοτύπου διεπαφής υλοποιημένο σε Java παρουσιάζεται στο Σχήμα 1. Αξίζει ωστόσο να εξετάσουμε υπό ποιες προϋποθέσεις είναι χρήσιμο ή και απαραίτητο να χρησιμοποιηθεί μια γλώσσα υψηλού επιπέδου για την ανάπτυξη πρωτοτύπων διεπαφής. Συχνα, χρειάζεται να αναπτυχθούν μη συμβατικές διεπαφές που δεν υπακούουν σε ένα καθορισμένο σχεδιαστικό πρωτόκολλο. Οι διεπαφές αυτές απαιτούν συχνά εξειδικευμένα αντικείμενα αλληλεπίδρασης και διαδραστικές συμπεριφορές που δεν υπάρχουν υλοποιημένες σε υπάρχουσες και ευρέως διαθέσιμες

βιβλιοθήκες διαδραστικών αντικειμένων (π.χ. ACTIVE X, AWT, SWING). Στις περιπτώσεις αυτές απαιτείται η υλοποίηση αυτών των νέων τεχνικών και η ενσωμάτωση τους σε ένα πρωτότυπο διεπαφής, κάτι το οποίο μπορεί να γίνει χρησιμοποιώντας εκειδικευμένες τεχνικές προγραμματισμού και μια κατάλληλη γλώσσα υψηλού επιπέδου.



Σχήμα 1: Πρωτότυπο διεπαφής υλοποιημένο σε Java

Κλασσικό παράδειγμα ανάπτυξης τέτοιων διεπαφών που δεν μπορεί να αναπτυχθούν παρά μόνο με τη μέθοδο που μόλις περιγράψαμε είναι η περίπτωση της σχεδίασης μιας νέας μεταφοράς αλληλεπίδρασης, δηλαδή μιας διεπαφής που αντλεί τη σχεδιαστική λογική της από ένα πεδίο διαφορετικό της επιφάνειας του γραφείου.

Περιληπτικά τα πλεονεκτήματα των γλωσσών υψηλού επιπέδου σε σχέση με τις γλώσσες χαμηλού επιπέδου είναι τα ακόλουθα:

Ο φυσικότερος και πιο «ανθρώπινος» τρόπος έκφρασης των προβλημάτων. Τα προγράμματα σε γλώσσα υψηλού επιπέδου είναι πιο κοντά στα προβλήματα που επιλύουν.

Η ανεξαρτησία από τον τύπο του υπολογιστή. Προγράμματα σε μία γλώσσα υψηλού επιπέδου μπορούν να εκτελεστούν σε οποιονδήποτε υπολογιστή με

ελάχιστες ή καθόλου μετατροπές. Η δυνατότητα της **μεταφερσιμότητας** των προγραμμάτων είναι σημαντικό προσόν.
Η ευκολία της εκμάθησης και εκπαίδευσης ως απόρροια των προηγούμενων.
Η διόρθωση λαθών και η συντήρηση προγραμμάτων σε γλώσσα υψηλού επιπέδου είναι πολύ ευκολότερο έργο.

2.1.2 Γλώσσες σήμανσης εξαρτημένες από τη συσκευή

Η πρώτη κατηγορία γλωσσών σήμανσης διεπαφών του τύπου των εξαρτώμενων από τη συσκευή που χρησιμοποιείται για διεπαφές του Παγκόσμιου Ιστου είναι η Hyper Text Mark-up Language(Γλώσσα Σήμανσης Υπερκειμένου, Html). Η HTML είναι μια γλώσσα περιγραφής της μορφοποίησης μιας σελίδας του Παγκόσμιου Ιστού με τη χρήση ετικετών. Κάθε εντολή ή ετικέτα (tag) της HTML περικλείεται απο απο τα σύμβολα < και > ώστε να ξεχωρίζει από το υπόλοιπο κείμενο. Οι περισσότερες ετικέτες ομαδοποιούνται σε ζεύγη, με την πρώτη να δηλώνει την αρχή και τη δεύτερη (που ξεκινάει με </) το τέλος επίδρασης της συγκεκριμένης ετικέτας. Για να μπορούν οι browser να ερμηνεύουν σχεδόν απόλυτα σωστά την html έχουν θεσπιστεί κάποιοι κανόνες. Αυτοί οι κανόνες είναι γνωστοί ως προδιαγραφές. Επομένως σχεδόν κάθε είδος υπολογιστή μπορεί να δείξει το ίδιο καλά μια ιστοσελίδα. Οι πρώτες προδιαγραφές ήταν η html 2.0. Πρόβλημα προέκυψε όταν η Microsoft και η Netscape πρόσθεσαν στην html τέτοιες δυνατότητες που στην αρχή τουλάχιστον ήταν συμβατές μόνο με συγκεκρίμενους browser. Η γενική δομή ενός εγγράφου HTML είναι η ακόλουθη :

```
<HTML>
        <HEAD>
                <TITLE> Τίτλος Κειμένου </TITLE>
        </HEAD>
<BODY>
……….
Κυρίως Κείμενο
.............
</BODY>
</HTML>
```

Θα πρέπει να τονίσουμε ότι η HTML παράγει έγγραφα επί το πλείστον παθητικά απο την άποψη της αλληλεπίδρασης με το χρήστη. Για να μετατραπούν τα έγγραφα αυτά σε ενεργητικά, χρειάζεται η χρήση μιας γλώσσας προγραμματισμού όπως η JavaScript. Είναι χρήσιμο να αρχίσουμε με μια ιδέα από την ιστορία της γλώσσας. Η JavaScript δημιουργήθηκε το 1995 από τον Brendan Eich, έναν μηχανικό της Netscape, και εκδόθηκε με τον Netscape 2 στις αρχές του 1996. Το αρχικό της όνομα ήταν LiveScript, αλλά μια ατυχής απόφαση μάρκετινγκ την μετονόμασε σε JavaScript. Το σκεπτικό ήταν να κεφαλαιοποιηθεί η δημοτικότητα της γλώσσας Java της Sun, παρότι οι δύο γλώσσες έχουν ελάχιστα κοινά στοιχεία μεταξύ τους. Αυτό αποτελεί μόνιμη πηγή σύγχυσης από τότε μέχρι σήμερα.

Αντίθετα από τις περισσότερες γλώσσες προγραμματισμού η γλώσσα JavaScript δεν έρχεται εφοδιασμένη με δυνατότητες εισόδου/εξόδου. Είναι σχεδιασμένη ως γλώσσα scripting σε ένα περιβάλλον που τη φιλοξενεί, και η ύπαρξη μηχανισμών επικοινωνίας με τον έξω κόσμο είναι ευθύνη αυτού του περιβάλλοντος. Το πιο

συνηθισμένο περιβάλλον φιλοξενίας είναι ο browser, αλλά interpreters JavaScript μπορούν να βρεθούν στον Acrobat της Adobe, το Photoshop, τη μηχανή Widget της Yahoo! και αλλού. Η Javascript είναι client side.

Αυτό σημαίνει ότι η επεξεργασία του κώδικα Javascript και η παραγωγή του τελικού περιεχομένου HTML δεν πραγματοποιείται στον server, αλλά στο πρόγραμμα περιήγησης των επισκεπτών. Αυτή η διαφορά έχει και πλεονεκτήματα και μειονεκτήματα για καθεμιά από τις δύο γλώσσες. Συγκεκριμένα, η Javascript δεν έχει καμία απαίτηση από πλευράς δυνατοτήτων του server για να εκτελεστεί (επεξεργαστική ισχύ, συμβατό λογισμικό διακομιστή), αλλά βασίζεται στις δυνατότητες του browser των επισκεπτών. Επίσης μπορεί να ενσωματωθεί σε στατικές σελίδες HTML. Παρόλα αυτά, οι δυνατότητές της είναι σημαντικά μικρότερες από αυτές της PHP και δεν παρέχει συνδεσιμότητα με βάσεις δεδομένων.

Ο κώδικας της JavaScript ενσωματώνεται στην HTML χρησιμοποιώντας
την ετικέτα (tag) SCRIPT.

```
<SCRIPT language="JavaScript">
document.write("Καλημέρα!");
</SCRIPT>
```

Ο κώδικας JavaScript μπορεί να τοποθετηθεί σε οποιοδήποτε σημείο του αρχείου HTML. Οι συναρτήσεις συνηθίζεται να δηλώνονται μέσα στο τμήμα HEAD Της σελίδας HTML, επειδή αυτό το τμήμα διαβάζεται πρώτο και έτσι διασφαλίζεται ότι οι συναρτήσεις θα είναι έτοιμες να εκτελεστούν πριν ο χρήστης χρειαστεί να κάνει οτιδήποτε. Οι κλήσεις JavaScript που εκτελούν κάποια αλληλεπίδραση με το χρήστη τοποθετούνται στο τμλημα BODY της σελίδας HTML. Η παράμετρος language είναι προαιρέτικη αλλά χρήσιμη επειδή επιτρέπει να καθορίζεται ότι ένα τμλημα κώδικα θα εκτελείται μόνο από προγράμματα πλοήγησης που υποστηρίζουν μια συγκεκριμένη έκδοση της JavaScript. Ο κώδικας JavaScript μπορεί επίσης να αποθηκεύεται σε εξώτερικα αρχεία με κατάληξη

```
<script language="JavaScript"
src="somefunction.js"></script>
```

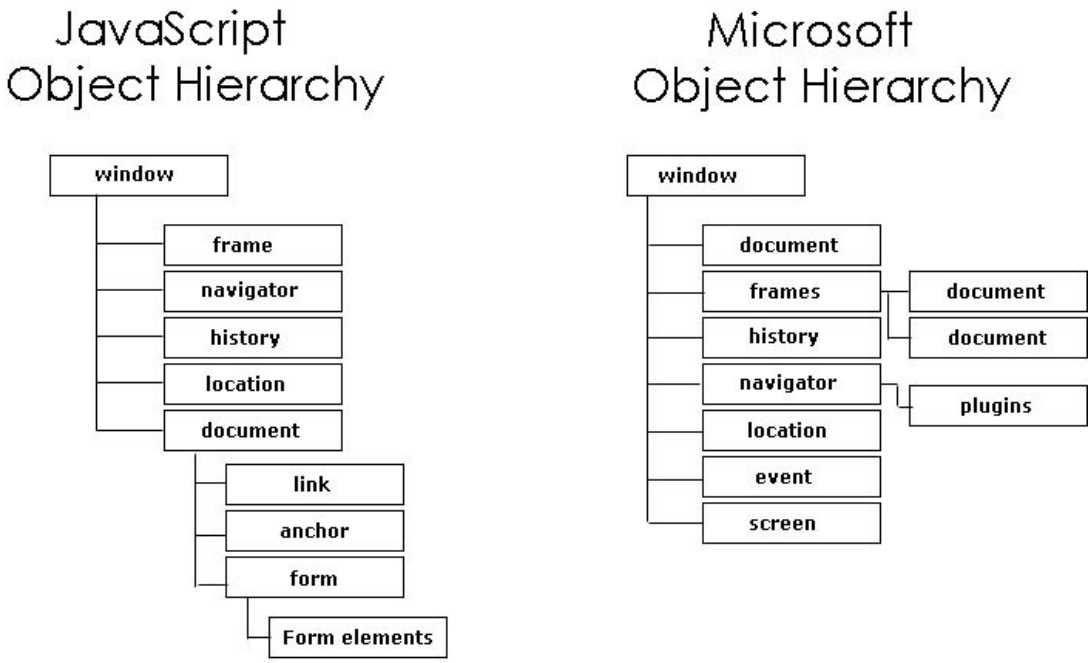
Τα αντικείμενα της HTML είναι μια συλλογή από δεδομένα μαζί με τις μεθόδους που επιδρούν στα δεδομένα του αντικειμένου. 'Οταν ανοίγει ένα έγγραφο στο πρόγραμμα πλοήγησης, δημιουργείται ένας αριθμός αντικειμένων JavaScript με τιμές που αντιστοιχούν στην HTML του εγγράφου. Με αυτόν τον τρόπο μπορεί ο προγραμματιστής να γνωρίζει πόσα αντικείμενα βρίσκονται σε μια φόρμα ή πόσα πλαίσια υπάρχουν στο παράθυρο του προγράμματος πλοήγησης. Αυτά τα αντικείμενα συνήθως ορίζονται με κάποια ιεραρχία (π.χ. βλέπε Σχήμα 2).

Τι Είναι το WAP

Η βιομηχανία των ασύρματων συσκευών ανέπτυξε την ιδέα του WAP. Ο στόχος αυτού του standard είναι να μπορούν να εμφανίζονται τα περιεχόμενα του Internet σε ασύρματες συσκευές, όπως είναι τα κινητά τηλέφωνα (mobile phones).

> Τα αρχικά **WAP** σημαίνουν Wireless Application Protocol, δηλ. Πρωτόκολλο Ασύρματων Εφαρμογών.

Το WAP είναι ένα πρωτόκολλο εφαρμογής επικοινωνιών (application communication protocol).
Το WAP χρησιμοποιείται για να έχουμε πρόσβαση σε υπηρεσίες και πληροφορίες.
Το WAP έχει κληρονομηθεί από τα standards του Internet.
Το WAP προορίζεται για φορητές συσκευές όπως είναι τα κινητά τηλέφωνα.
Το WAP είναι ένα πρωτόκολλο που είναι σχεδιασμένο για micro browsers.
Το WAP δίνει τη δυνατότητα δημιουργίας web εφαρμογών για κινητές συσκευές.
Το WAP χρησιμοποιεί τη γλώσσα σήμανσης (mark-up language) WML και όχι την HTML.
Η WML είναι ορισμένη σαν μια εφαρμογή της XML 1.0.

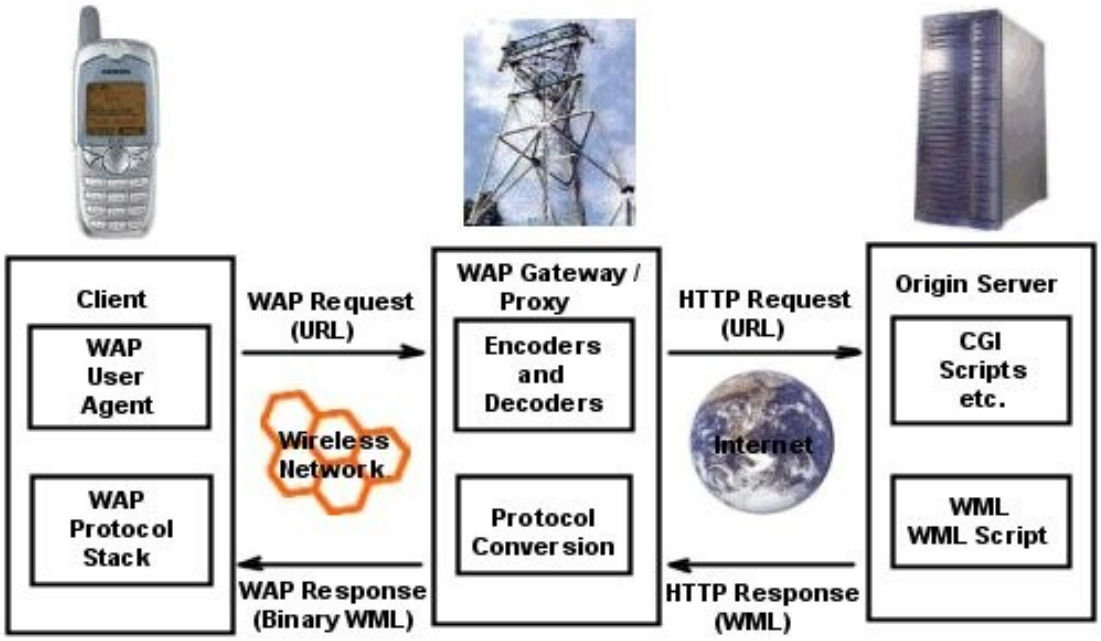


Σχήμα 2: Παραδείγματα ιεραρχιών διαδραστικών αντικειμένων

Το Πρωτόκολλο WAP

Το Πρωτόκολλο WAP αποτελεί το κορυφαίο στάνταρτ για υπηρεσίες πληροφοριών σε ασύρματες συσκευές όπως είναι τα ψηφιακά κινητά τηλέφωνα. Μια ροή κίνησης της πληροφορίας σε ένα WAP δίκτυο παρουσιάζεται στο Σχήμα 3. Το WAP standard βασίζεται στα Internet standards (HTML, XML και TCP/IP). Αποτελείται από μια προδιαγραφή (specification) της γλώσσας WML, από μια προδιαγραφή (specification) της γλώσσας WMLScript και από μια προδιαγραφή (specification) του Wireless Telephony Application Interface (WTAI).

Το WAP δημοσιεύθηκε από το **WAP Forum**, που ιδρύθηκε το 1997 από τις εταιρείες Ericsson, Motorola, Nokia και Unwired Planet. Τα μέλη του Forum αντιπροσωπεύουν τώρα πάνω από το 90% της παγκόσμιας αγοράς φορητών συσκευών καθώς επίσης και κορυφαίους παροχείς infrastructure, software developers και άλλους οργανισμούς.



Σχήμα 3: Απεικόνιση ενός WAP δικτύου

Οι Μικρο Φυλλομετρητές (Micro Browsers) του WAP

Για να χωρέσει σε μια μικρή ασύρματη συσκευή, το WAP χρησιμοποιεί έναν μικρο φυλλομετρητή (Micro Browser). Ένας μικρο-φυλλομετρητής είναι ένα μικρό κομμάτι λογισμικού (software) που έχει ελάχιστες απαιτήσεις σε υλικό (hardware), μνήμη και CPU. Μπορεί να εμφανίσει πληροφορίες που είναι γραμμένες σε μία αυστηρή γλώσσα σήμανσης (mark-up language), που αποκαλείται **WML** (Wireless Markup Language). Ο μικρο φυλλομετρητής μπορεί επίσης να διερμηνεύσει (interpret) μια μειωμένη έκδοση (version) της JavaScript, που αποκαλείται **WMLScript**.

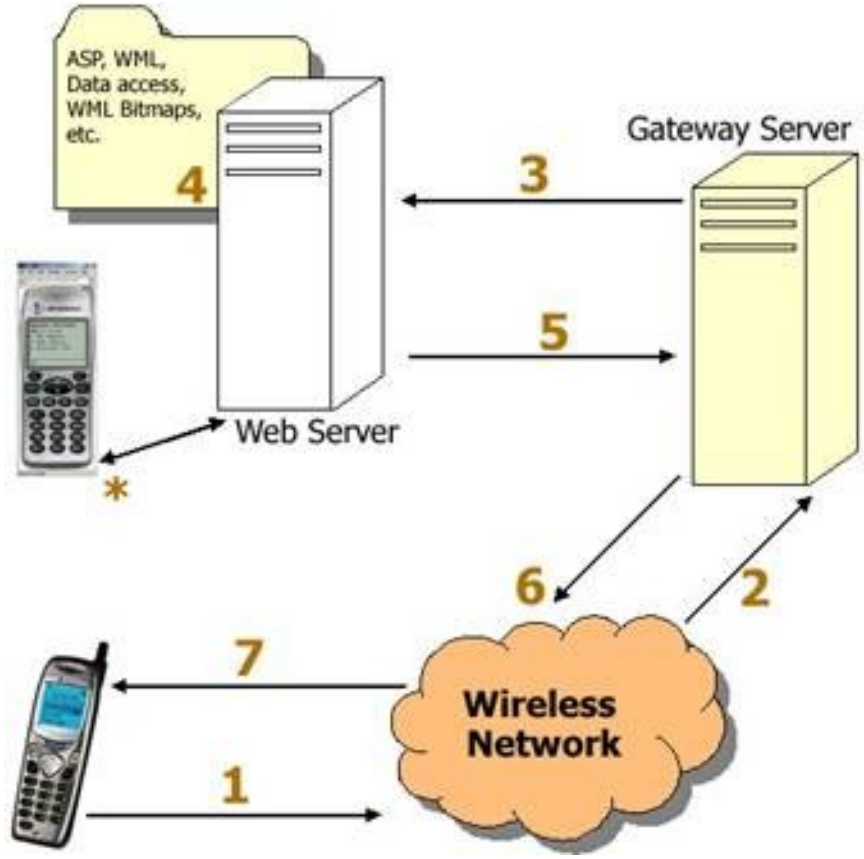
Τι Είναι η WML

Τα αρχικά **WML** σημαίνουν Wireless Markup Language, δηλ. Ασύρματη Γλώσσα Σήμανσης. Είναι μια γλώσσα σήμανσης που έχει κληρονομηθεί από την HTML, αλλά που βασίζεται στην XML και έτσι είναι πολύ περισσότερο αυστηρή από την HTML. Η WML χρησιμοποιείται για να δημιουργήσουμε σελίδες που να μπορούν να εμφανισθούν σ' έναν φυλλομετρητή WAP. Οι σελίδες της WML αποκαλούνται DECKS και τα Decks δημιουργούνται σαν ένα σύνολο από CARDS.

Στο Σχήμα 4 παρουσιάζεται η ροή κίνησης της πληροφορίας σε ένα ασύρματο δίκτυο για πρόσβαση από τερματική συσκευή συμβατή με WML. Συγκεκριμένα:

Το κινητό τηλέφωνο στέλνει ένα αίτημα Ιστού
Ασύρματο δίκτυο διαβιβάζει το αίτημα στον κεντρικό υπολογιστή
Ο κεντρικός υπολογιστής πυλών στέλνει το αίτημα στον κεντρικό υπολογιστή δικτύου στα στοιχεία προσβάσεων κεντρικών υπολογιστών σχήματος

Ο Web Server εκχωρεί δεδομένα, τρέχει ASP/JSP/CGI Script/... και δημιουργεί WML.
Ο Web Server στέλνει WML στον κεντρικό υπολογιστή πυλών ως δυαδικές συμπιέσεις WML κεντρικών υπολογιστών απάντησης
server πυλών συμπιέζει σε δυαδική μορφή, WML και στο στελνει στο τηλεφωνο ως WAP.
Το τηλέφωνο παρουσιάζει το περιεχόμενο που του έχει σταλεί.



Σχήμα 4: Ροή κίνησης πληροφορίας

Τι Είναι η WMLScript

Η WML χρησιμοποιεί την WMLScript για να εκτελέσει απλό κώδικα στην πλευρά του χρήστη (on the client). Η WMLScript είναι μια μικρή παραλλαγή της γλώσσας JavaScript. Όμως, τα scripts της WML δεν ενσωματώνονται στις σελίδες της WML. Οι σελίδες της WML περιέχουν μόνο αναφορές σε URLs. Τα scripts της WML πρέπει να μεταγλωττιστούν σε κώδικα byte (byte code) σ' έναν server πριν μπορέσουν να εκτελεστούν (τρέξουν) σ' έναν φυλλομετρητή WAP.

Παραδείγματα χρήσης του WAP :

Πληροφορίες για τα δρομολόγια των τρένων.
Αγορά εισιτηρίων και check in σε πτήσεις.
Πληροφορίες κυκλοφορίας (traffic information).
Πληροφορίες για τις καιρικές συνθήκες.

Αναζήτηση τιμών στο Χρηματιστήριο.
Αναζήτηση αριθμών τηλεφώνου και διευθύνσεων.
Αναζήτηση αθλητικών αποτελεσμάτων.

Οι Αρχικές Σελίδες (Homepages) του WAP

Οι αρχικές σελίδες (homepages) του WAP δεν είναι πολύ διαφορετικές από τις αρχικές σελίδες (homepages) της HTML. Η γλώσσα σήμανσης που χρησιμοποιείται στο WAP είναι η WML (Wireless Markup Language), η οποία χρησιμοποιεί tags, όπως ακριβώς η HTML, αλλά η σύνταξή της είναι αυστηρότερη και υπάρχει συμμόρφωση με το XML 1.0 standard. Οι σελίδες της WML έχουν την επέκταση ***.WML**, όπως οι σελίδες της HTML έχουν την επέκταση *.HTML.

Τα Tags της WML

Η WML περιέχει κυρίως κείμενο. Τα tags που θα μπορούσαν να επιβραδύνουν την επικοινωνία με τις φορητές συσκευές δεν αποτελούν μέρος του WML standard και η χρήση πινάκων και εικόνων είναι πολύ περιορισμένη. Εφόσον η WML αποτελεί μια εφαρμογή της XML, όλα τα tags ξεχωρίζουν τα πεζά από τα κεφαλαία γράμματα (case sensitive). Δηλαδή, το <wml> δεν είναι το ίδιο με το <WML> και όλα τα tags πρέπει να κλείνουν σωστά.

Τα Decks και τα Cards της WML

Οι σελίδες της WML αποκαλούνται **DECKS** και δημιουργούνται σαν ένα σύνολο από **CARDS** που συσχετίζονται μεταξύ τους με συνδέσμους (links). Όταν γίνεται πρόσβαση σε μια σελίδα WML από ένα κινητό τηλέφωνο, όλα τα cards της σελίδας φορτώνονται (downloaded) από τον WAP server. Η πλοήγηση (navigation) ανάμεσα στα cards γίνεται από τον υπολογιστή του τηλεφώνου, μέσα στο τηλέφωνο, χωρίς να υπάρχει καμία επιπλέον πρόσβαση στον server. Ένα παράδειγμα ενός WML εγγράφου :

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="HTML" title="HTML School">
<p> Μαθήματα σε HTML </p>
</card>
<card id="XML" title="XML School">
<p> Μαθήματα σε XML </p>
</card>
</wml>
```

Οπως φαίνεται από το παραπάνω παράδειγμα, το έγγραφο της WML είναι ένα έγγραφο της XML. Το DOCTYPE ορίζεται ότι είναι ίσο με wml και η πρόσβαση στο DTD γίνεται στο www.wapforum.org/DTD/wml_1.1.xml. Το περιεχόμενο του εγγράφου βρίσκεται μέσα στα tags <wml> … </wml>. Το κάθε card του εγγράφου

βρίσκεται μέσα στα tags <card> … </card> και οι πραγματικές παράγραφοι βρίσκονται μέσα στα tags … . Το κάθε στοιχείο ενός card έχει ένα id και έναν τίτλο (title).

Η Μορφοποίηση (Formatting) της WML

Οι σελίδες της WML συχνά αποκαλούνται decks και ένα deck περιέχει ένα σύνολο από cards. Ένα στοιχείο card μπορεί να περιέχει κείμενο, σήμανση (markup), συνδέσμους (links), πεδία καταχώρησης (input-fields), εργασίες (tasks), εικόνες (images) κ.ά. Τα cards μπορούν να συσχετιστούν μεταξύ τους με συνδέσμους (links). Όταν γίνεται πρόσβαση σε μια WML σελίδα από ένα κινητό τηλέφωνο, όλα τα cards της σελίδας φορτώνονται (downloaded) από τον WAP server. Η πλοήγηση (navigation) ανάμεσα στα cards γίνεται από τον υπολογιστή του τηλεφώνου, μέσα στο τηλέφωνο, χωρίς να υπάρχει καμία επιπλέον πρόσβαση στον server :

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="no1" title="Card 1">
<p> Hello World! </p>
</card>
<card id="no2" title="Card 2">
<p> Welcome to WAP School! </p>
</card>
</wml>
```

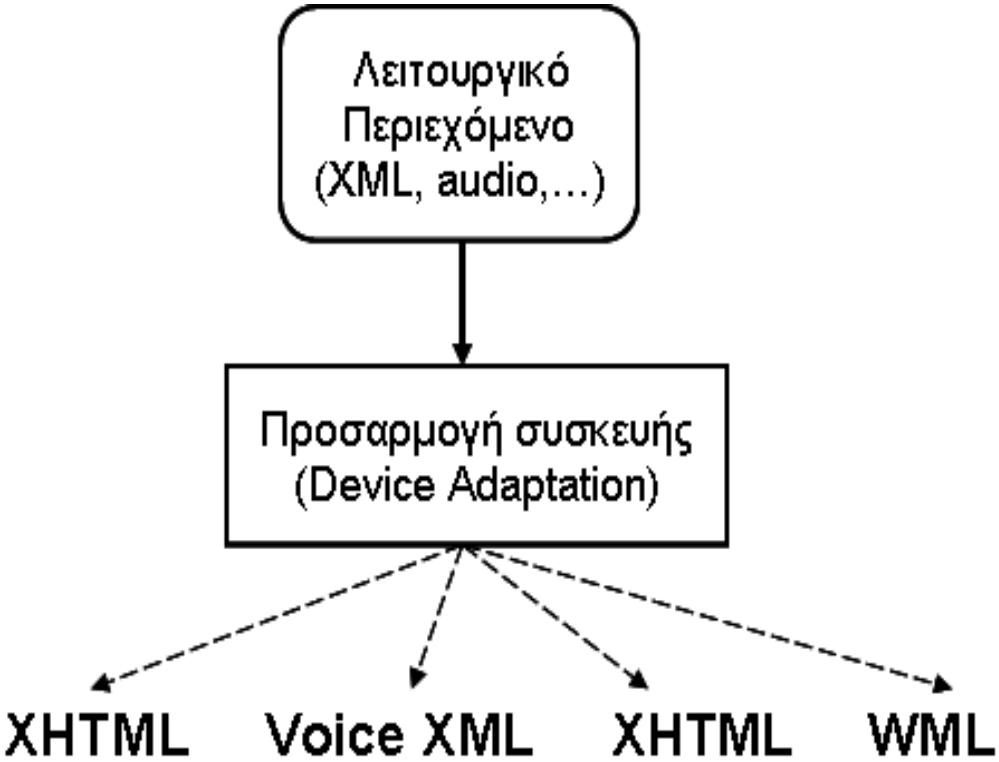
Το αποτέλεσμα μπορεί να φαίνεται κάπως έτσι στην οθόνη ενός κινητού τηλεφώνου και πρέπει να έχουμε υπόψη μας ότι μόνο ένα card εμφανίζεται κάθε φορά :

------ Card 1 ------

Hello World!

2.1.3 Γλώσσες σήμανσης διεπαφών ανεξάρτητες συσκευής

Οι γλώσσες της κατηγορίας αυτής (markup languages) επιδιώκουν να προωθήσουν περαιτέρω την ανεξαρτησία του περιεχομένου από τη συσκευή (βλέπε Σχήμα 5). Η φιλοσοφία τους βασίζεται στην πεποίθηση ότι πλησιάζει η εποχή που οι διεπαφές θα είναι τόσο διαχωρισμένα και διακριτά τμήματα των διαδραστικών συστημάτων όσο είναι διαχωρισμένο το λογισμικό από το υλικό. Για το σκοπό αυτό απαιτούνται πρότυπα εργαλεία περιγραφής διεπαφών που να ενσωματώνουν κατάλληλες έννοιες και επίπεδα αφαίρεσης. Η πλειοψηφία των γλωσσών αυτών είναι εφαρμογές της γλώσσας eXtensible Markup Language (XML, 1998). Η XML – που προτάθηκε και υποστηρίζεται από το W3C – προσφέρει ένα γενικό συντακτικό και κάθε επιμέρους γλώσσα ορισμού προσδιορίζει το λεξικό (vocabulary) και τη σημασιολογία (schematics).



Σχήμα 5: Ρόλος των ανεξάρτητων από τη συσκευή γλωσσών σήμανσης

Η XML δημιουργήθηκε έτσι ώστε να μπορούν να χρησιμοποιηθούν «πλουσιοπάροχα» δομημένα έγγραφα, εύκολα προσβάσιμα, μεταφέρσιμα και ευέλικτα, ανεξάρτητα οποιουδήποτε συστήματος και περιεχομένου,
για την ανάπτυξη του Παγκόσμιου Ιστού.Η XML δεν έχει προκαθορισμένες ετικέτες όπως για παράδειγμα η HTML.Αντίθετα, ο συγγραφέας πρέπει κάθε φορά να καθορίζει τις ετικέτες που θέλει. Η δυνατότητα αυτή έχει επιτρέψει την ανάπτυξη πληθώρας εξειδικευμένων γλωσσών σήμανσης οι οποίες βασίζονται στην XML για την προδιαγραφή διεπαφών.
Παραδείγματα τέτοιων γλωσσών, οι οποίες είναι ευρύτατα διαφημισμένες στο διαδίκτυο είναι οι:

Alternative Abstract Interface Markup Language (AAIML)
Abstract User Interface Markup Language (AUIML)
Extensible Interface Markup Language (XIML)
Extensible User Interface Language (XUL)
Microsoft Extensible Application Markup Language (XAML)
OASIS User Interface Markup Language (UIML) Specification Technical Committee
User Interface eXtensible Markup Language (UsiXML)
User Interface Markup Language (UIML)
W3C Device Independence Working Group (DIWG)
W3C XForms Working Group [XForms]

Θα πρέπει τέλος να τονίσουμε ότι η XML είναι μία περιγραφική, μη ολοκληρωμένη γλώσσα.Με άλλα λόγια, όμοια με της άλλες γλώσσες ορισμού ημιδομημένων

δεδομένων, η XML δεν θεωρείται – και δεν σχεδιάστηκε για να είναι – λειτουργικά πλήρες προγραμματιστικό εργαλείο.Συνήθως υπολείπονται ορισμένων σημαντικών προγραμματιστικών ιδιοτήτων και δομών όπως συνθήκες επιλογής,επαναλήψεις κ.τ.λ.

2.2 Τεχνικές ανάπτυξης διεπαφών με τη χρήση μοντέλων

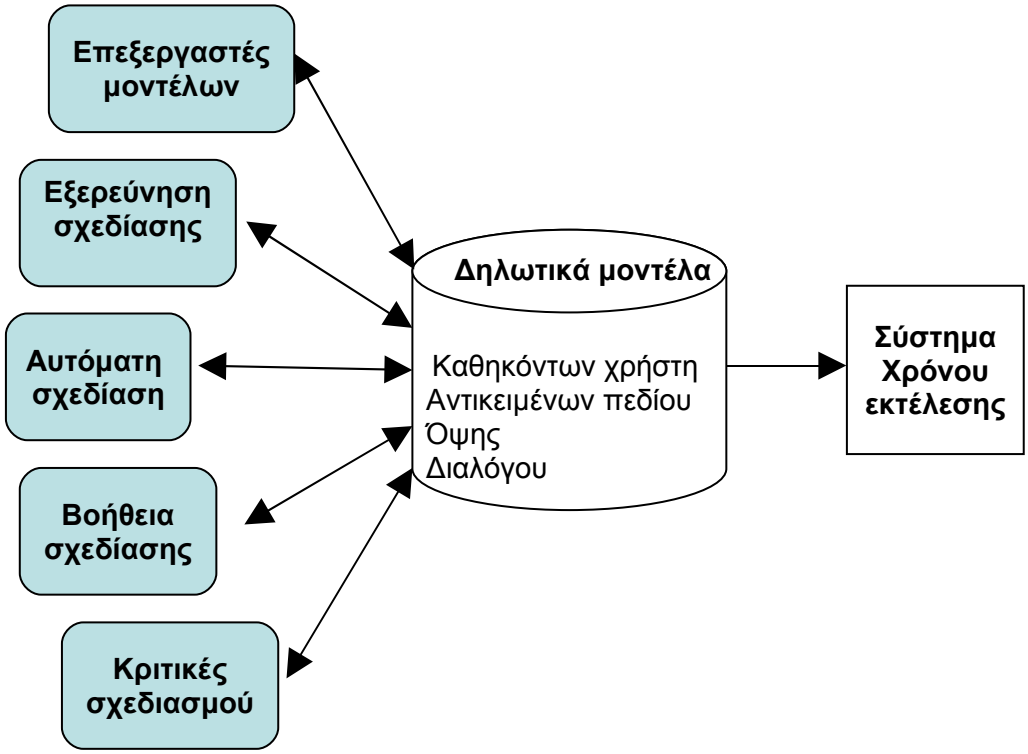
Στην κατηγορία αυτή περιλαμβάνονται τεχνικές οι οποίες αφορούν την αυτόματη ή ημιαυτόματη παραγωγή του κώδικα της διεπαφής με τη χρήση μοντέλων.Τα μοντέλα αυτά που χρησιμοποιούνται για την ανάπτυξη της διεπαφής είναι κυρίως δηλωτικά (declarative) μοντέλα, με ρίζες στην αναπαράσταση γνώσης με υπολογιστικές και στη μηχανική λογισμικου.Η βασική ιδέα πίσω από τις τεχνικές της κατηγορίας αυτής βρίσκεται στο τρίπτυχο:

Αναγνώριση χρήσιμων αφαιρετικών δομών (abstractions) που να χαρακτηρίζουν τα κύρια στοιχεία (τη γνώση) στη σχεδίαση μιας διεπαφής

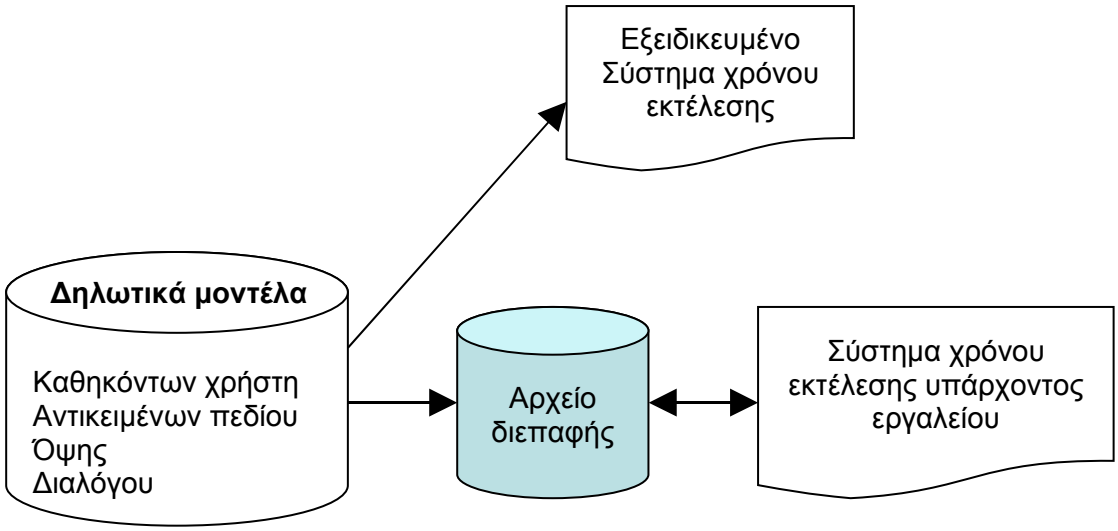
Αναπαράσταση των στοιχείων αυτών μέσω μοντέλων, κυρίως δηλωτικού (declarative) τύπου

Αντιστοίχηση μεταξύ στοιχείων των διαφόρων μοντέλων για την ανάπτυξη μιας διεπαφής.

Το Σχήμα 6 συνοψίζει τη βασική ιδέα στη διαχείριση μοντέλων για την ανάπτυξη διεπαφών, ενώ το Σχήμα 7 συνοψίζει εναλλακτικές αρχιτεκτονικές υλοποίησης του λογισμικού της διεπαφής με τη χρήση συστήματος χρόνου εκτέλεσης (run-time). Συγκεκριμένα, η φιλοσοφία της προσέγγισης βασίζεται στη χρήση εξειδικευμένων εργαλείων για την επεξεργασία σχεδιαστικής γνώσης και την προετοιμασία ενός μοντέλου της διεπαφής που είναι εκτελέσιμο από ένα σύστημα χρόνου εκτέλεσης (run-time system).



Σχήμα 6: Κατηγορίες μοντέλων που χρησιμοποιούνται για την ανάπτυξη διεπαφών



Σχήμα 7: Αρχιτεκτονικές

Σε αντίθεση με τα συστήματα διαχείρισης διεπαφών που στόχευαν στην παραγωγή κώδικα, στην παραγωγή εκτελέσιμων προδιαγραφών διεπαφής, ή στην παραγωγή και των δύο, οι τεχνικές που αξιοποιούν μοντέλα εστιάζουν περισσότερο στην ανάλυση, στην τεκμηρίωση του σχεδίου και στην επιχειρηματολογία σχεδιαστικών αποφάσεων. Υπάρχουν πολλές προσπάθειες συστημάτων που αξιοποιούν μοντέλα για την παραγωγή της υλοποίησης μιας διεπαφής.Από τα πλέον διαδεδομένα είναι τα συστήματα Mickey (Olsen, 1989), Jade (Judgement-based Automatic Dialog Editor) των Zanden και Myers (1990), DON των Kim και Foley (1990), UIDE (Foley et al., 1991 ; Sukaviriya et al., 1993), HUMANOID (Szekely et al., 1993), ITS (Wiecha et al., 1990), Javamatic (Phanouriou και M.Abrams 1997), MIKE (Olsen, 1986).

2.2.1 UIDE

Το User Interface Design Environment – UIDE (Foley et al., 1991 ; Sukaviriya et al., 1993) κατατάσσεται στην κατηγορία των τεχνικών ανάπτυξης διεπαφών που βασίζονται σε μοντέλα και μάλιστα, είναι ένα από τα πρώτα συστήματα που εμφανίστηκαν στην σχετική βιβλιογραφία ως ολοκληρωμένα περιβάλλοντα ανάπτυξης διεπαφών βάσει μοντέλων.Το χαρακτηριστικό γνώρισμα του UIDE, το οποίο σε ορισμένες μελέτες οδηγεί τους μελετητές να το χαρακτηρίσουν ως περιβάλλον ανάπτυξης διαδραστικών εφαρμογών, είναι η έμφαση των μοντέλοων του στην υποστήριξη της διαλειτουργικότητας της διεπαφής με την εφαρμογή και την περιγραφή των αποτελεσμάτων εντολών και όχι τοσο της διεπαφής.Η περιγραφή της διεπαφής περιλαμβάνει πρότερες και ύστερες συνθήκες των χειρισμών που εκτελεί το σύστημα για να παράγει αυτόματα τη διεπαφή.Μια άλλη χρήση των πρότερων και ύστερων συνθηκών στο UIDE είναι για την αυτόματησύνθεση βοήθειας προς το χρήστη (Sukaviriya et al., 1993).

2.2.2 HUMANOID

Το σύστημα Humanoid (High-level UIMS for Manufacturing Application Needing Organized Iterative Development) των Szekely et al. (1993) χρησιμοποιεί ένα

μοντέλο το οποίο αναπαριστά όψη και συμπεριφορά μιας διεπαφής, κωδικοποιώντας τη σημασιολογία της εφαρμογής (application semantics), περιγράμματα παρουσίασης (presentation templates ή styles), τη συμπεριφορά των αντικειμένων (behaviour), την εξέλιξη διαλόγου (dialog sequencing), και επιπτώσεων δράσεων (action side-effects). Η σημασιολογία της εφαρμογής αναφέρεται στις τάξεις των αντικειμένων πεδίου, των γνωρισμάτων, και των μεθόδων του πεδίου εφαρμογής. Τα περιγράμματα παρουσίασης προσδιορίζουν την όψη της διεπαφής όπως αυτή ορίζεται από συγκεκριμένα διαδραστικά αντικείμενα. Η συμπεριφορά προσδιορίζει τον τρόπο αλληλεπίδρασης του χρήστη με τα διαδραστικά αντικείμενα. Η ακολουθία διαλόγου καθορίζει την οργάνωση των εντολών και συνήθως εκφράζεται με περιορισμούς ακολουθίας (π.χ. τι πρέπει να υφίσταται πριν ενεργοποιηθεί μια εντολή). Τέλος, οι επιπτώσεις δράσεων αφορούν τις δράσεις που εκτελούνται αυτόματα μετά την ενεργοποίηση μιας εντολής (δηλαδή τις επιπτώσεις εκτέλεσης της εντολής).
Θα πρέπει να τονίσουμε ότι το Humanoid κατασκευάστηκε με στόχο να εξισορροπήσει μεταξύ της παροχής υψηλού βαθμού ελέγχου στο σχεδιαστή και του υψηλού βαθμού αυτοματοποίησης του σχεδίου.Για το σκοπό αυτό, το σύστημα εισάγει ένα συνεργατικό (ή μικτής πρωτοβουλίας) περιβάλλον ανάπτυξης διεπαφών από μοντέλα, όπου οι σχεδιαστές είναι αρμόδιοι για τη λήψη των αποφάσεων σχεδίασης και το σύστημα αυτοματοποιεί την εφαρμογή τους. Το τμήμα αυτόματης παραγωγής της διεπαφής χρησιμοποιεί πληροφορίες από τα πρότυπα παρουσίασης (presentation templates), και προκαθορισμένες επιλογές που ελέγχουν την τελική διαδικασία παραγωγής της διεπαφής.Εντούτοις, το σύστημα επιτρέπει στους σχεδιαστές να τροποποιήσουν τη σχεδίαση σύ,φωνα με την προτίμησή τους.

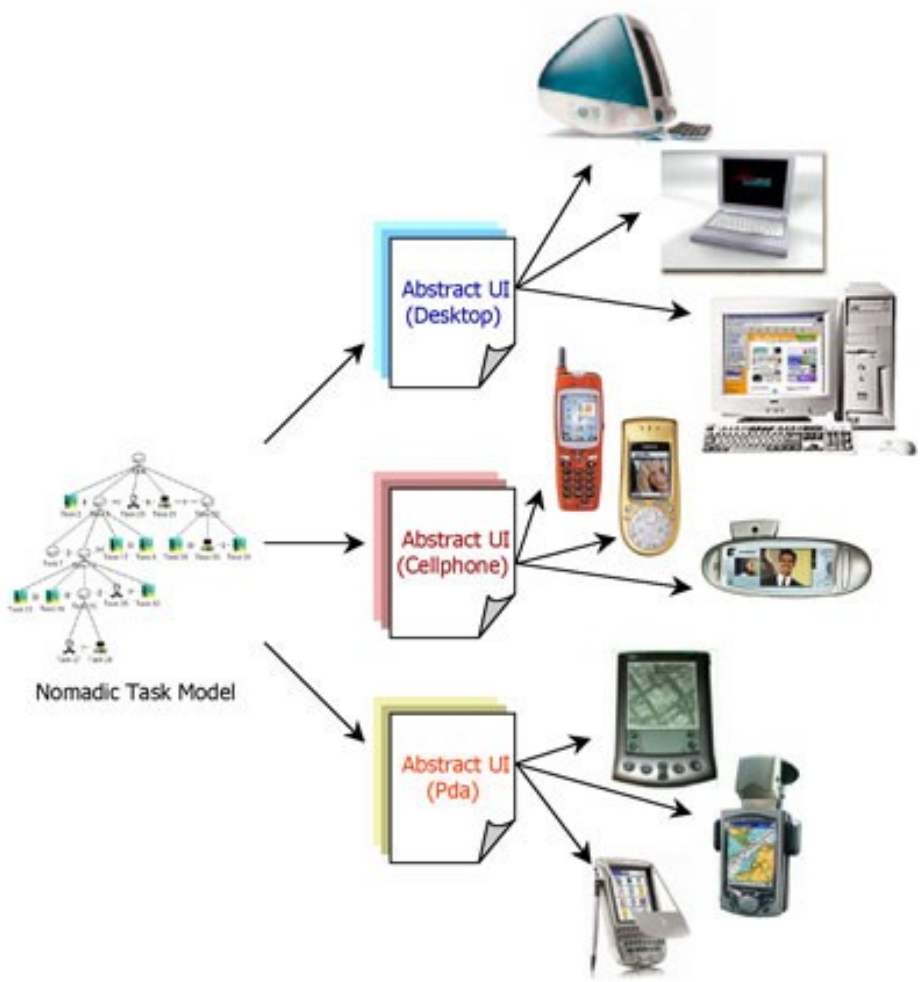
2.2.3 MASTERMIND

Το MASTERMIND (Models Allowing Shared Tools and Explicit Representation to Make Interfaces Natural to Develop) υπήρξε ένα ερευνητικό έργο που απασχολήθηκε με την αυτόματη παραγωγή κώδικα διεπαφής από δηλωτικά μοντέλα. Σε μεγάλο βαθμό το MASTERMIND ήταν μια προσπάθεια συνδυασμού των ερευνητικών εργαλείων UIDE και Humanoid. Το MASTERMIND παρήγαγε μια εννοιολογική αρχιτεκτονική που βασιζόταν σε τρεις τύπους μοντέλων – το μοντέλο παρουσίασης, το μοντέλο διαλόγου, και το μοντέλο αλληλεπίδρασης. Το μοντέλο παρουσίασης περιγράφει τα διαδραστικά αντικείμενα που εμφανίζονται στο τερματικό του χρήστη και τις μεταξύ τους εξαρτήσεις. Το μοντέλο διαλόγου περιγράφει διαφορές χαμηλού επιπέδου δράσης εισόδου που εκτελούνται από τον χρήστη κατά τη διάρκεια αλληλεπίδρασης με μια διεπαφή του MASTERMIND. Το μοντέλο περιλαμβάνει επίσης τις σχετικές ακολουθίες δραστηριοτήτων και τις επιπτώσεις που μπορούν να έχουν στη διεπαφή και την εφαρμογή. Το μοντέλο αλληλεπίδρασης προδιαγράφει τη διαδραστική συμπεριφορά μεταξύ του χρήστη και του περιβάλλοντος εκτέλεσης του MASTERMIND. Το περιεχόμενο του μοντέλου καθορίζεται από τη βιβλιοθήκη διαδραστικών αντικειμένων για την οποία το MASTERMIND παράγει τον κώδικα της διεπαφής.
Για να παραχθεί εκτελέσιμος κώδικας μιας διεπαφής από τα παραπάνω μοντέλα, το MASTERMIND χρησιμοποιεί εξειδικευμένα εργαλεία παραγωγής κώδικα από τα μοντέλα (model-specific code generators). Καθε τέτοιο εργαλείο δημιουργεί στοιχεία C++ τα οποία στη συνέχεια μεταγλωτίζονται και συνδέονται με διάφορες βιβλιοθήκες χρόνου εκτέλεσης (run-time libraries). Η σύνθεση του κώδικα που παράγεται από τα διάφορα μοντέλα είναι αρκετά περίπλοκη διαδικασία, η οποία απαιτεί από τον

αναγνώστη εις βάθος μελέτη της αντίστοιχης βιβλιογραφίας και των πληροφοριών που υπάρχουν στον ιστότοπο του MASTERMIND (http://isi.edu/isd/Mastermind/mastermind.html.old).

2.2.4 Teresa

Το σύστημα Teresa (Transformation Environment for inteRactivE Systems represantAtion) είναι ένα πρόσφατο εργαλείο, το οποίο υποστηρίζει την ανάπτυξη διεπαφών πολλαπλής πλατφόρμας με τη χρήση μοντέλων (Mori et al., 2004). Το εργαλείο βασίζεται στη φιλοσοφία των μετασχηματισμών (βλέπε Σχήμα 8). Βάσει της φιλοσοφίας αυτής, παράγεται από ένα αφηρημένο μοντέλο καθηκόντων ένα αφηρημένο μοντέλο παρουσίασης (διεπαφής χρήστη), το οποίο στη συνέχεια μετασχηματίζεται σε συγκεκριμένο μοντέλο παρουσίασης (διεπαφής χρήστη) σε μία επιλεγμένη πλατφόρμα. Με το τρόπο αυτό, το Teresa επιτρέπει στους σχεδιαστές να έχουν μία ενιαία άποψη για τη σχεδίαση μίας δεδομένης νομαδικής εφαρμογής. Για να επιτύχει το στόχο της, το Teresa βασίζεται σε σειρά μοντέλων, με κυριότερο το μοντέλο καθηκόντων που βασίζεται με τη σειρά του στην τεχνική ConcurTaskTrees (CTT), το αφηρημένο μοντέλο παρουσίασης και το συγκεκριμένο μοντέλο παρουσίασης.



Σχήμα 8: Απεικόνιση συστήματος Teresa

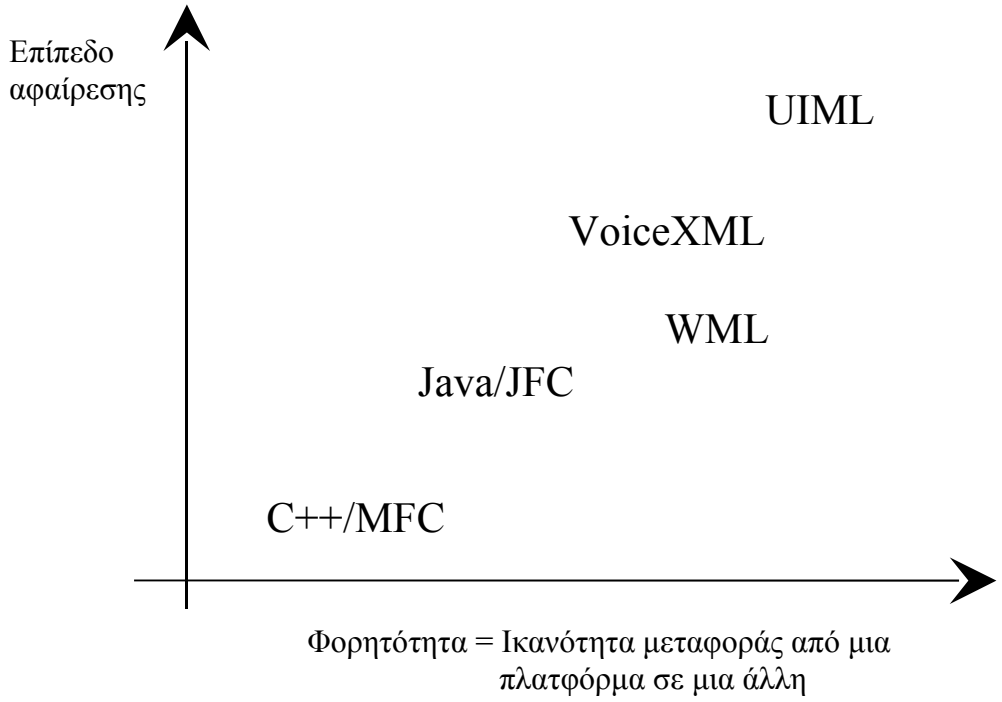
Συνοψίζοντας οι τεχνικές που βασίζονται σε μοντέλα αντιπροσωπεύουν μία πιο πρόσφατη εξέλιξη, που θεμελιώνεται στην ιδέα της αυτόματης ή ημιαυτόματης κατασκευής (ή παραγωγής) διαδραστικής συμπεριφοράς από κατάλληλα μοντέλα του συστήματος, τα οποία μπορεί να αφορούν στο πεδίο της εφαρμογής, στους χρήστες και στα καθήκοντα που καλούνται να εκτελέσουν, στην πλατφόρμα εκτέλεσης, στο περιβάλλον χρήσης και άλλα. Τα τελευταία χρόνια έχουν γίνει σημαντικά βήματα προς την κατεύθυνση ανάπτυξης συστημάτων που να αξιοποιούν μοντέλα και να παράγουν διεπαφές είτε για συγκεκριμένες πλατφόρμες και συσκευές – βλέπε Humanoid των Szekely, Luo και Neches (1992) και MASTERMIND των Szekely et al. (1995) – είτε για πολλές και διαφορετικές συσκευές, που αποτελεί και σύγχρονο πρόβλημα εστίασης της όλης προσέγγισης – βλέπε ενδεικτικά το σύστημα Teresa των Mori et al.(2004).

3 Η Προσέγγιση της UIML

3.1 Τι είναι η UIML

Η UIML (User Interface Markup Language) είναι μία γλώσσα σήμανσης υψηλού επιπέδου η οποία αξιοποιεί την XML για να εξυπηρετήσει την ανάπτυξη διεπαφών πολλαπλής πλατφόρμας για ποικιλία συσκευών πρόσβασης αξιοποιώντας κατάλληλες δομές αφαίρεσης (βλέπε Σχήμα 9). Η XML δημιουργήθηκε με στόχο να επιτρέψει στο σχεδιαστή / συγγραφέα να ορίσει / προσαρμόσει / προδιαγράψει πληροφορία χρησιμοποιώντας ετικέτες της αρεσκείας του. Με τον τρόπο αυτό υποστηρίζεται η δημιουργία δομημένων εγγράφων και η διακίνησή τους μέσω του παγκόσμιου Ιστού. Η XML είναι μία περιγραφική γλώσσα, και όχι ένα λειτουργικά πλήρες προγραμματιστικό εργαλείο.

Η UIML ενσωματώνει ένα ριζικά νέο τρόπο να χτιστούν τα συστατικά μιας διεπαφής χρήστη-υπολογιστή, αξιοποιώντας κανόνες αντιστοίχησης (mapping rules) αφηρημένων δομών αλληλεπίδρασης (abstract interaction elements) σε γραφικά συστατικά (widgets) μιας πλατφόρμας / εργαλειοθήκης. Οι αφηρημένες δομές αλληλεπίδρασης αποτελούν εξειδικευμένες ετικέτες σε ένα XML έγγραφο το οποίο μετασχηματίζεται / αντιστοιχίζεται σε κλήσεις κλάσεων υλοποιημένων σε μια εργαλειοθήκη (vocabulary) κατά το χρόνο εκτέλεσης του λογισμικού μιας διεπαφής (run-time).



Σχήμα 9: Εργαλεία ανάπτυξης διεπαφών και φορητότητα παραγόμενης διεπαφής

Η φιλοσοφία που χρησιμοποιείται για την ανάπτυξη προδιαγραφών διεπαφής σε UIML, βασίζεται στην ανάγκη αυτόματης ή ημιαυτόματης προσαρμογής της διεπαφής σε ποικίλες συσκευές (τερματικά πρόσβασης) καθώς και στην πεποίθηση ότι πλησιάζει η εποχή που οι διεπαφές θα είναι τόσο διαχωρισμένα και διακριτά τμήματα των διαδραστικών συστημάτων όσο είναι διαχωρισμένο το λογισμικό από το υλικό. Για το σκοπό αυτό απαιτούνται πρότυπα εργαλεία περιγραφής διεπαφών που να ενσωματώνουν κατάλληλες έννοιες και επίπεδα αφαίρεσης.

3.1.1 Κίνητρα για τη χρήση UIML

Επιτρέπει στο χρήστη την ανάπτυξη διεπαφών χρήστη-υπολογιστή πολλαπλές συσκευές / πλατφόρμες χωρίς να απαιτείται η εκμάθηση εξειδικευμένων γλωσσών προγραμματισμού και των αντίστοιχων προγραμματιστικών μοντέλων επικοινωνίας (API's) με λειτουργικά τμήματα των εφαρμογών

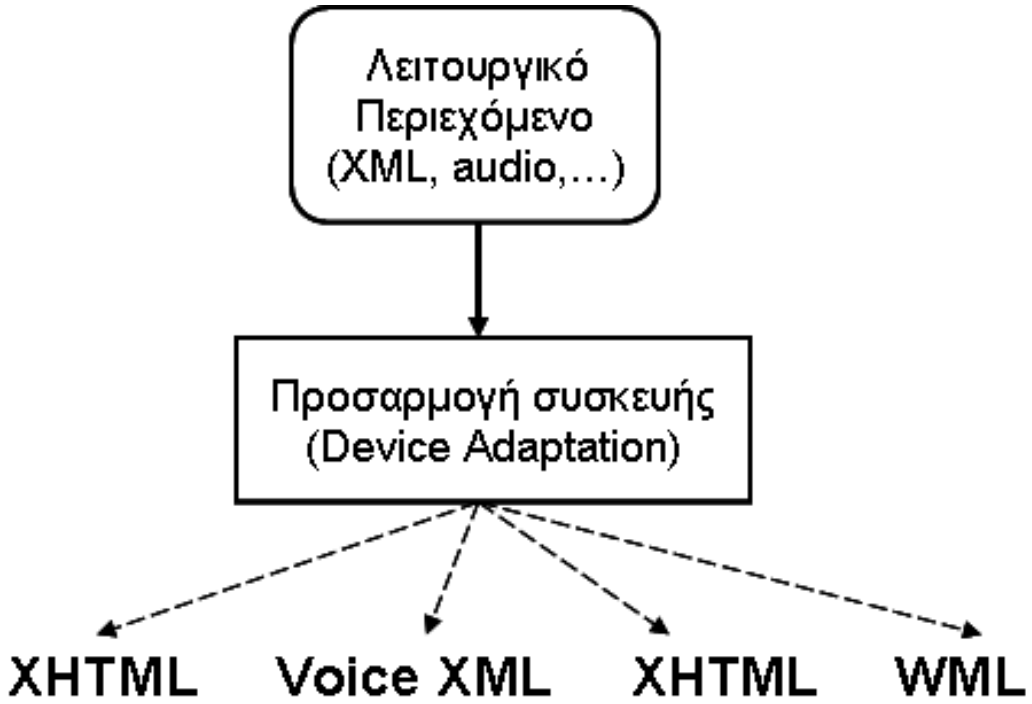
Μειώνει το χρόνο ολοκλήρωσης διεπαφών για μία οικογένεια συσκευών

Παρέχει ένα φυσικό διαχωρισμό μεταξύ του κώδικα της διεπαφής και του λειτουργικού τμήματος της εφαρμογής (κώδικα λογικής εφαρμογής)

Επιτρέπει την επέκταση για την υποστήριξη αναδυόμενων τεχνολογιών αλληλεπίδρασης που αναμένεται να εφευρεθούν στο μέλλον.

3.1.2 Γλώσσα σήμανσης διεπαφών ανεξάρτητη συσκευής

Οι γλώσσες της κατηγορίας σήμανσης διεπαφών ανεξαρτήτου συσκευής – με χαρακτηριστικό αλλά όχι μοναδικό παράδειγμα την UIML – επιδιώκουν να προωθήσουν περαιτέρω την ανεξαρτησία του περιεχομένου από τη συσκευή πρόσβασης. Στόχος δηλαδή της UIML είναι η παροχή μιας 'τεχνολογικά ουδέτερης' αναπαράστασης της διεπαφής που να μπορεί να αντιστοιχηθεί / μετασχηματιστεί σε υλοποιημένες κλάσεις διαδραστικών αντικειμένων μιας εργαλειοθήκης (MFC, Java's Swing) ή μιας συμβατικής (device-specific) γλώσσας σήμανσης όπως οι HTML, WML, κλπ. Η φιλοσοφία αυτή συνοψίζεται σχηματικά στο Σχήμα 10.



Σχήμα 10: Ρόλος των ανεξάρτητων από τη συσκευή γλωσσών σήμανσης

Για τη κατασκευή μίας διεπαφής απαιτείται η σύνταξη ενός εγγράφου UIML το οποίο περιλαμβάνει δομές παρουσίασης κατάλληλες για τις συσκευές για τις οποίες προορίζεται η διεπαφή. Στη συνέχεια, το έγγραφο της UIML αντιστοιχίζεται αυτόματα στη γλώσσα που χρησιμοποιεί η συγκεκριμένη συσκευή – για παράδειγμα στην XHTML, τη WML, τη VoiceXML, τη C++ (με τη βοήθεια μιας βιβλιοθήκης όπως η MFC), την Java(με μία βιβλιοθήκη όπως η JFC) κ.ο.κ.

Για τη δημιουργία μιας διεπαφής ο μηχανικός αναπτύσσει ένα έγγραφο UIML του οποίου ο βασικός σκελετός συνοψίζεται ως εξής:

```
<?xml version="1.0"  encoding="ISO-8859-1" ?>
<!DOCTYPE uiml PUBLIC "- //Harmonia//DTD UIML 2.0 Draft//EN"
"UIML2_0e.dtd">
<uiml xmlns='http: //uiml.org/dtds/UIML2_0e.dtd'>
        <head>……………….</head>
        <interface>……….</interface>
        <peers>……………….</peers>
        <template>……….</template>
</uiml>
```

Τα τέσσερα τμήματα του εγγράφου με τις ετικέτες <head> ,<interface>, <peers> και <template> μπορούν να εμφανίζονται με οποιαδήποτε ακολουθία, ενώ το έγγραφο UIML θα πρέπει να περιλαμβάνει τουλάχιστον ένα στοιχείο που να παρουσιάζεται σε μια συσκευή και να γίνεται αντιληπτό από το χρήστη.Το τμήμα του εγγράφου <head>. . . </head> είναι προαιρετικό και περιλαμβάνει μεταδεδομένα σχετικά με το έγγραφο.Το τμήμα <interface> . . . </interface> είναι και αυτό προαιρετικό και περιλαμβάνει μία περιγραφη των συστατικών της διεπαφής, της δομής τους, του περιεχομένου τους, της όψης τους και της συμπεριφοράς τους. Το επόμενο προαιρετικό τμήμα του εγγράφου <peers>. . . </peers> περιγράφει την αντιστοίχιση μεταξύ των ιδιοτήτων και των συμβάντων που αναφέρονται στο UIML έγγραφο με τη βιβλιοθήκη διαδραστικών αντικειμένων και τη λογική της εφαρμογής. Το τέταρτο τμήμα του εγγράφου, <template>. . . </template>, είναι και αυτό προαιρετικό και επιτρέπει την επαναχρησιμοποίηση επιλεγμένων τμημάτων UIML.

Παρακάτω συνοψίζεται ένα παράδειγμα δημιουργίας μιας απλής διεπαφής που παρουσιάζει το μήνυμα «Hello World».

```
<?xml version="1.0"  encoding="ISO-8859-1" ?>
<!DOCTYPE uiml PUBLIC "- //Harmonia//DTD UIML 2.0 Draft//EN"
"UIML2_0e.dtd">
<uiml>
   <interface>
        <structure>
                <part name=" TopHello"  class="Container">
                    <part name="HelloStr"  class="String">
```

```
                </part>
        </structure>
        <style>
            <property part-name="TopHello"  name="content"
              > Hello  </property>
             <property part-name="HelloStr"  name="content">
 Hello World  </property>
          </style>
    </interface>
</uiml>
```

Για την ολοκλήρωση της διεπαφής θα πρέπει να προσδιοριστεί το τμήμα του εγγράφου <peers>. . . </peers> που εξειδικεύει την επικοινωνία με τον έξω κόσμο, δηλαδή τη συσκευή πρόσβασης του χρήστη. 'Ετσι ο παραπάνω κώδικας σε συνδυασμό με το ακόλουθο τμήμα

```
<peers>
    <presentation name="VoiceXML">
          <component name="Container"  maps-to="vxml : form" />
          <component name="String"  maps-to="vxml : block" />
          <attribute name= "content"  maps-to="PCDATA"/>
          </component>
    </presentation>
</peers>
```

Θα παράγει τον ακόλουθο κώδικα VoiceXML:

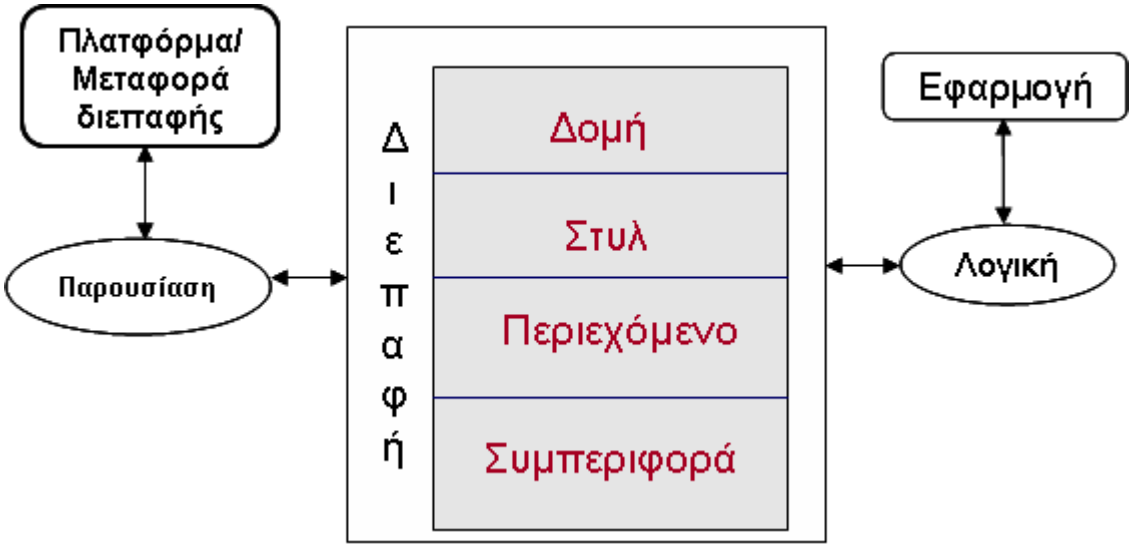
```
<?xml version="1.0" ?>
<vxml>
        <form>
            <block> Hello World </block>
        </form>
</vxml>
```

Θα πρέπει να τονίσουμε ότι το τμήμα κώδικα που προσδιορίζει την επικοινωνία μιας διεπαφής με τον εξωτερικό κόσμο αναπτύσσεται μία και μόνο φορά.

3.2 Το μοντέλο UIML

Το μοντέλο που υιοθετεί η UIML (Meta-Interface Model ή MIM) διαιρεί τη διεπαφή σε τρία βασικά μέρη : παρουσίαση, διεπαφή και λογική (βλέπε Σχήμα 11). Με τον

τρόπο αυτό επιτυγχάνεται ο διαχωρισμός μεταξύ του διαδραστικού τμήματος και του λειτουργικού μέρους ενός συστήματος, επιτρέποντας μεταξύ άλλων τη γρήγορη ανάπτυξη διεπαφών πολλαπλής συσκευής.



Σχήμα 11: Το μοντέλο της UIML

Το τμήμα της *λογικής* περιγράφει τη λειτουργία της εφαρμογής, ενώ την ίδια στιγμή αποκρύπτει στοιχεία, όπως μεταφράσεις δεδομένων (data translation), ονόματα μεθόδων (method names) και άλλα, βασικά για την επικοινωνία με την ίδια την εφαρμογή. Η UIML παρέχει την ετικέτα <logic> ώστε να περιγράψει τα συστατικα *λογικής*.

Ακολουθεί ένα ενδεικτικό παράδειγμα για την περιγραφή της ετικέτας <logic>:

```
<peers>
  <logic>
    <d-component id="Main" maps-to="com.harmonia.test.Main">
      <d-method id="startx" maps-to="startx" return-type="void"/>
    </d-component>
  </logic>
  <presentation id="base" base="Java_1.4_Harmonia_1.3"/>
 </peers>
```

Ένα έγγραφο UIML μπορεί να περιέχει πολλαπλά στοιχεία *λογικής*.

Το τμήμα της *διεπαφής* περιγράφει το διάλογο μεταξύ του χρήστη και της εφαρμογής χρησιμοποιώντας ένα σύνολο αφηρημένων μερών (abstract part), γεγονότων (events) και κλήσεων μεθόδων
(method *calls*).
Το Meta-Interface Model πηγαίνει ένα βήμα μπροστά από τα προηγούμενα μοντέλα και υποδιαιρεί το τμήμα της διεπαφής σε τέσσερα πρόσθετα μέρη:

1. Δομή (structure) :Περιγράφει τα δομικά στοιχεία συστατικών της διεπαφής.
2. Στυλ (style) :Περιγράφει τις ιδιότητες παρουσίασης για κάθε μέρος της διεπαφής.
3. Περιεχόμενο (content) :Περιγράφει την πληροφορία που ένα συστατικό παρουσιάζει στο χρήστη.
4. Συμπεριφορά (behavior) :Περιγράφει συγκεκριμένες συμπεριφορές και ενέργειες των συστατικών της διεπαφής.

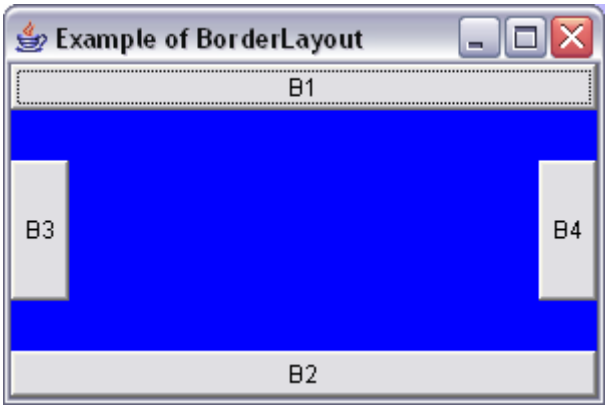
Το τμήμα *παρουσίασης* ενσωματώνει αντιστοιχίσεις κλάσεων γεγονότων, ιδιοτήτων συστατικών γραφικού περιβάλλοντος σε μία αφηρημένη εργαλειοθήκη (abstract toolkit) η οποία χρησιμοποιείται για το καθορισμό της συσκευής πρόσβασης του χρήστη.

3.3 Παραδείγματα

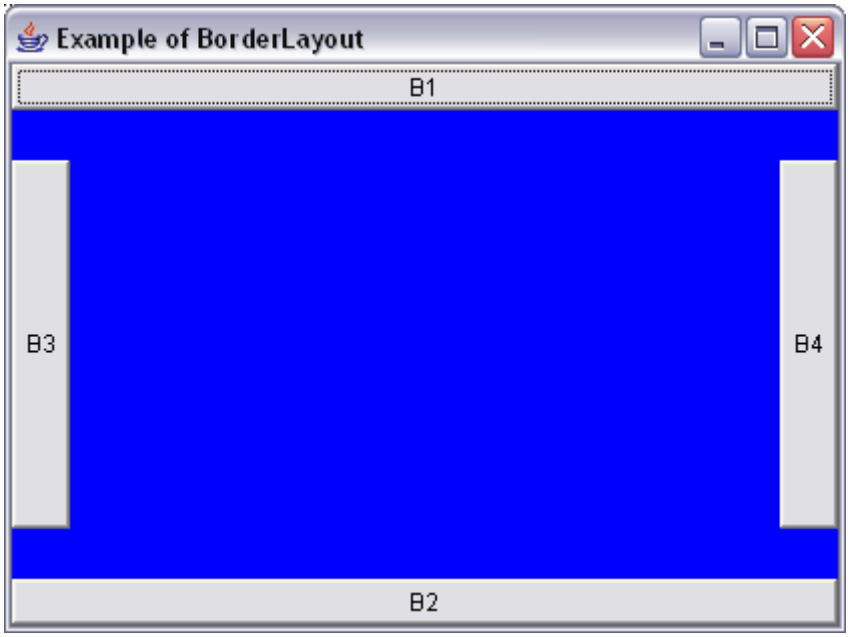
Στο τμήμα αυτό επιχειρείται η παρουσίαση και η λεπτομερής ανάλυση / επεξήγηση δύο παραδειγμάτων υλοποιημένων σε UIML. Συγκεκριμένα στο πρώτο παράδειγμα περιγράφεται ο διαχειριστής διάταξης ″BorderLayout″, ενώ στο δεύτερο ο χειρισμός γεγονότων σε ένα UIML έγγραφο.

3.3.1 Παράδειγμα 1: LayoutBorder.uiml

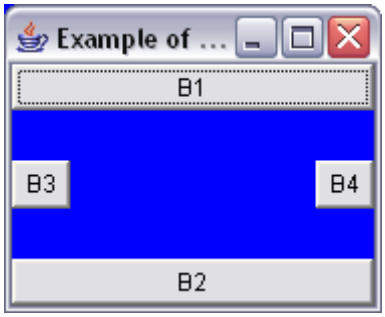
Αυτό το παράδειγμα χρησιμοποιεί "BorderLayout" ως διαχειριστή διάταξης του πλαισίου. Η διεπαφή αποτελείται από ένα πλαίσιο που περιέχει τέσσερα κουμπιά. Σε κάθε κουμπί ορίζεται μια διεύθυνση (Βορράς, Νότος, Ανατολή, Δύση). Ουσιαστικά αυτό που επιτυγχάνουμε με το συγκεκριμένο παράδειγμα είναι ότι με οποιοδηποτε ″resize″ του πλαισίου της διεπαφής τα buttons παραμένουν προσκολημμένα εκεί που αρχικά βρισκόταν, ενώ αυξομειώνεται και το μεγεθός τους ανάλογα με το μέγεθός του παραθύρου. Παρακάτω παραθέτονται κάποια στιγμιότυπα της διεπαφής.



Σχήμα 12: Στιγμιότυπο διαχειριστή διίαταξης – Αρχική μορφή πλαισίου



Σχήμα 13: Στιγμιότυπο διαχειριστή διίαταξης – Μεγέθυνση πλαισίου



Σχήμα 14: Στιγμιότυπο διαχειριστή διίαταξης – Σμίκρυνση πλαισίου

Παρακάτω εμφανίζεται ο κώδικας που χρησιμοποιήθηκε για την υλοποίηση της διεπαφής:

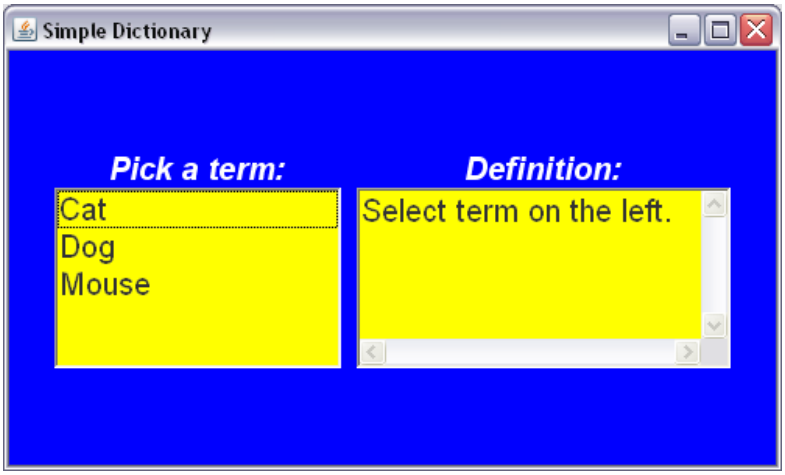
```
1.<uiml>
2.  <head>
3.    <meta name="Purpose" content="Example of BorderLayout
LayoutManager"/>
5.  </head>
6.  <peers>
7.    <presentation base="Java_1.4_Harmonia_1.0"/>
8.  </peers>
9.  <interface id="myinterface">
10.    <structure>
11.      <part id="TopHello" class="Frame">
12.        <part id="B1" class="Button">
13.          <style>
14.            <property name="label">B1</property>
15.            <property name="borderAlignment">North</property>
16.          </style>
```

```
17.       </part>
18.       <part id="B2" class="Button">
19.         <style>
20.           <property name="label">B2</property>
21.           <property name="borderAlignment">South</property>
22.         </style>
23.       </part>
24.       <part id="B3" class="Button">
25.         <style>
26.           <property name="label">B3</property>
27.           <property name="borderAlignment">West</property>
28.         </style>
29.       </part>
30.       <part id="B4" class="Button">
31.         <style>
32.           <property name="label">B4</property>
33.           <property name="borderAlignment">East</property>
34.         </style>
35.       </part>
36.       <style>
37.         <property name="rendering">Frame</property>
38.         <property name="title">Example of 39.BorderLayout</property>
40.         <property name="layout">java.awt.BorderLayout</property>
41.         <property name="layout_hgap">10</property>
42.         <property name="layout_vgap">25</property>
43.         <property name="background">blue</property>
44.         <property name="bounds">100,100,300,200</property>
45.       </style>
46.     </part>
47.   </structure>
48. </interface>
49.</uiml>
```

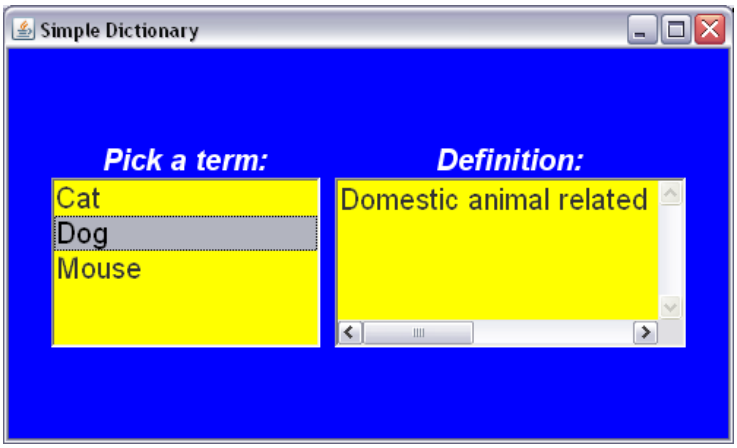
Στη γραμμή 1 ανοίγει το uiml έγγραφο το tag <uiml> όπου αναμένεται να κλείσει στη τελευταία γραμμή του κώδικα όπως και γίνεται. Στη γραμμή 3 έως 5 χρησιμοποείται το tag <head></head> όπου δηλώνονται τα meta-name. Στη συνέχεια και συγκεκριμένα στη γραμμή 6 έως 8 χρησιμοποιείται το tag <peers></peers> όπου δηλώνεται το vocabulary βάσει του οποίου θα ανπτυχθεί η διεπαφή. 'Επειτα στη γραμμή 9 δηλώνεται το Interface με συγκεκριμένο id όπου αναμένεται να ολοκληρωθεί πριν το </uiml> όπως και γινεται. Στη γραμμή 10 ξεκινάει η δομή της διεπαφής με το tag <structure> και αναμένεται να ολοκληρωθεί πριν το </structure>. Στις γραμμές 11 έως και 46 γίνεται η δήλωση των συστατικών που απαρτίζουν τη διεπαφη ενώ παράλληλα δηλώνονται και οι ιδιότητες κάθε συστατικού ξεχωριστά. Η διεπαφή όπως φαίνεται ξεκάθαρα από τον κώδικα, απαρτίζεται από τέσσερα buttons και το Frame. Αξιοσημείωτο σε αυτή τη διεπαφή είναι πως κάλείται η java.awt.BorderLayout στη γραμμή 40 ώστε να επιτευχθέι το επιθυμήτο αποτέλεσμα. Επίσης για κάθε button αντίστοιχα δηλώνεται <property name="borderAlignment">North</property>, έτσι ώστε να τοποθετηθεί βόρεια, ανατολικά, δυτικά ή νότια στο frame.

3.3.2 Παράδειγμα 2: Dictionary.uiml

Αυτό το τμήμα περιέχει ένα απλό παράδειγμα ενός εγγράφου UIML, το οποίο περιγράφει το χειρισμό γεγονότος. Στο παρακάτω στιγμιότυπο (βλέπε Σχήμα 15) παρουσιάζεται ένα πλαίσιο που αντιπροσωπεύει ένα λεξικό. Το λεξικό περιέχει ένα παράθυρο καταλόγων στο οποίο ένας χρήστης μπορεί να επιλέξει έναν όρο όπως γάτα, σκύλο, ποντίκι. Το λεξικό περιέχει επίσης μια περιοχή στην οποία επιδεικνύεται ο καθορισμός του επιλεγμένου όρου. Για παράδειγμα εάν επιλεγεί ο όρος "Dog" (βλέπε Σχήμα 16), μία περιγραφή του επιλεγμένου όρου εμφανίζεται στη συγκεκριμένη περιοχή (Definition area). Στο UIML έγγραφο χρησιμοποιούνται συστατικά των βιβλιοθηκών AWT και Swing της Java.



Σχήμα 15: Στιγμιότυπο Dictionary – Πριν την επιλογή αντικειμένου



Σχήμα 16: Στιγμιότυπο Dictionary – Μετά την επιλογή αντικειμένου

Παρακάτω παρατίθεται ο κώδικας με τον οποίο αναπτύχθηκε η συγκεκριμένη διεπαφή.

```
1.<uiml>
2.  <peers>
3.    <presentation id="java-jfc" base="Java_1.3_Harmonia_1.0"/>
4.  </peers>
```

```
5.  <interface>
6.    <structure>
7.      <part id="JFrame" class="JFrame">
8.        <part id="TermLabel" class="JLabel">
9.          <style/>
10.        </part>
11.        <part id="TermList" class="List">
12.          <style/>
13.        </part>
14.        <part id="DefnLabel" class="JLabel">
15.          <style/>
16.        </part>
17.        <part id="DefnArea" class="TextArea">
18.          <style/>
19.        </part>
20.        <style/>
21.      </part>
22.    </structure>
23.    <style>
24.      <property part-name="JFrame"
25.name="layout">java.awt.GridBagLayout</property>
26.    <property part-name="JFrame" name="background">blue</property>
27.   <property part-name="JFrame" name="location">100,100</property>
28.   <property part-name="JFrame" name="size">500,300</property>
29.   <property part-name="JFrame" name="title">Simple
30.Dictionary</property>
31.
32.      <property part-class="JLabel" 33.name="foreground">white</property>
34.      <property part-class="JLabel" 35.name="gridx">RELATIVE</property>
36.   <property part-class="JLabel" name="gridy">RELATIVE</property>
37.  <property part-class="JLabel" name="font">Helvetica-bolditalic-38.
20</property>
39.
40.    <property part-name="TermLabel" name="text">Pick a
41.    term:</property>
42.
43.<property part-name="DefnLabel"
44.name="text">Definition:</property>
45.
46.      <property part-name="DefnLabel" name="gridx">1</property>
47.      <property part-name="DefnLabel" name="gridy">0</property>
48.      <property part-name="DefnLabel" 49.name="insets">0,10,0,0</property>
50.
51.      <property part-name="TermList"
52.name="background">yellow</property>
53.      <property part-name="TermList" name="gridx">0</property>
54.      <property part-name="TermList" 55.name="gridy">RELATIVE</property>
56.      <property part-name="TermList" name="fill">BOTH</property>
57.      <property part-name="TermList" name="font">Helvetica-58.20</property>
59.      <property part-name="TermList" name="content">
```

```
60.       <constant model="list">
61.       <constant id="Cat" value="Cat"/>
62.       <constant id="Dog" value="Dog"/>
63.       <constant id="Mouse" value="Mouse"/>
64.     </constant>
65.   </property>
66.
67.     <property part-name="DefnArea"
68.name="background">yellow</property>
69.     <property part-name="DefnArea" name="gridx">1</property>
70.     <property part-name="DefnArea" 71.name="gridy">RELATIVE</property>
72.     <property part-name="DefnArea" name="text">Select term on 73.the
left.</property>
74.     <property part-name="DefnArea" name="columns">20</property>
75.     <property part-name="DefnArea" name="rows">4</property>
76.     <property part-name="DefnArea" 77.name="editable">false</property>
78.     <property part-name="DefnArea" 79.name="insets">0,10,0,0</property>
80.     <property part-name="DefnArea" name="font">Helvetica-81.20</property>
82.
83.     <property part-name="TermLabel" 84.name="location">0,0</property>
85.     <property part-name="TermLabel" 86.name="size">100,50</property>
87.
88.     <property part-name="DefnLabel" 89.name="location">0,0</property>
90.     <property part-name="DefnLabel" 91.name="size">100,50</property>
92.   </style>
93.   <behavior>
94.     <rule>
95.       <condition>
96.         <equal>
97.           <event name="item" class="itemStateChanged" part-
98.name="TermList"/>
99.           <constant value="0"/>
100.         </equal>
101.       </condition>
102.       <action>
103.         <property part-name="DefnArea" 104.name="text">Carnivourous,
domesticated mammal that's fond of rats 105.and mice</property>
106.       </action>
107.     </rule>
108.     <rule>
109.       <condition>
110.         <equal>
111.           <event name="item" class="itemStateChanged" part-
112.name="TermList"/>
113.           <constant value="1"/>
114.         </equal>
115.       </condition>
116.       <action>
117.         <property part-name="DefnArea" name="text">Domestic 118.animal
related to a wolf that's fond of chasing cats</property>
```

```
119.      </action>
120.     </rule>
121.     <rule>
122.       <condition>
123.         <equal>
124.           <event name="item" class="itemStateChanged" part-
125.name="TermList"/>
126.           <constant value="2"/>
127.         </equal>
128.       </condition>
129.       <action>
130.         <property part-name="DefnArea" name="text">Small rodent 131.often
seen running away from a cat</property>
132.       </action>
133.     </rule>
134.   </behavior>
135.  </interface>
136.</uiml>
```

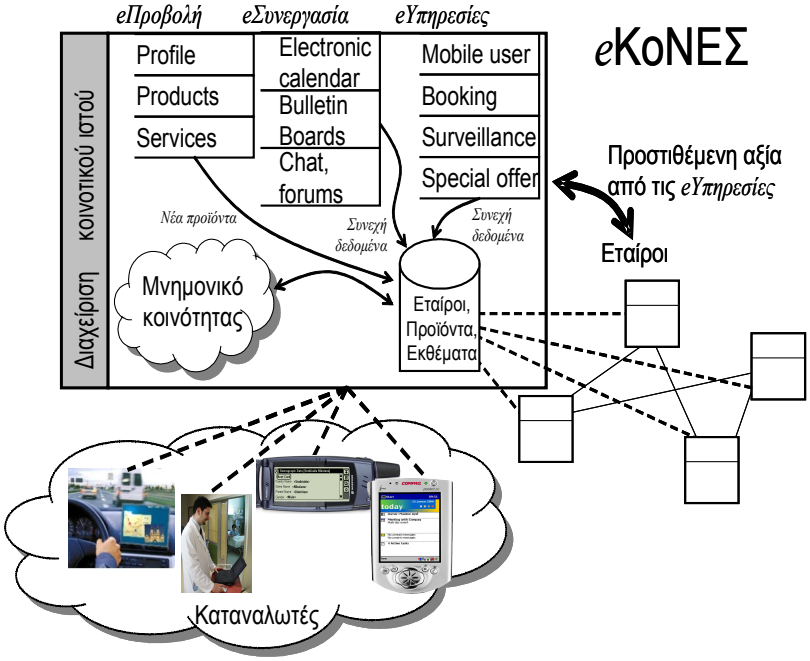
Στη γραμμή 1 του κώδικα ανοίγουμε ενα uiml έγγραφο με το <uiml>, όπου θα κλείσει όταν τελειώσει η ανάπτυξη όλης της διεπαφής με </uiml>. Στη γραμμή 2-4 έχουμε τα peers όπου με το presentation base μας κάνουν γνωστό σε ποιο vocabulary θα βασιστεί η ανάπτυξη της διεπαφής. Η γραμμή 5 αναφέρεται στο interface όπου περικλείει όλη τη δομή της διεπαφης, και κλεινει πριν το </uiml>. Από τη γραμμή 6 έως 22 έχουμε το structure element όπου περικλύει όλα τα συστατικά από τα οποία αποτελείται η διεπαφή. Σε κάθε περίπτωση με το part class προσδιορίζεται το είδος του συστατικού (JFrame,TextArea κτλ.) ενώ απαιτείται και η δήλωση ενός id για κάθε συστατικό. Από τη γραμμή 23 έως και τη γραμμή 92 του κώδικα αναπτύσσεται το style της διεπαφής. Σε όλο αυτό το μέρος του κώδικα κάθε συστατικό που έχει δηλωθεί στο structure αναπτύσει ξεχωριστα το style του. Με style αναφερόμαστε σε όλα τα properties που ο προγραμματιστής θέλει να αποδώσει σε κάθε συστατικό. Μερικές απο τις ιδιότητες που μπορούν να προσδιοριστούν ειναι το χρώμα του background, το μέγεθος, ο τίτλος, η γραμματοσειρά, κ.α. Αξιοσημείωτο είναι πως από τη γραμμή 60 εως 64 δηλώνονται τα περιεχόμενα της λίστας με τη χρήση του <constant>. Από τη γραμμή 93 έως 134 αναπτύσσονται τα “behaviors” κάθε συστατικού και ουσιαστικά εδώ βρίσκεται και όλη η λειτουργικότητα της διεπαφής.
Το <behavior> αποτελείται από μια ακολουθία κανόνων (rules), κάθε μία που περιέχει έναν όρο(condition) και μια δράση(action). Κάθε όρος (condition) ισχύει όταν εμφανίζεται κάποιο γεγονός. Το στοιχείο σε κάθε όρο ονομάζει ένα γεγονός (event) της Java μέσω των ιδιοτήτων κατηγορίας του που καθορίζει το λεξιλόγιο. Συγκεκριμένα όταν επιλεγούν στοιχεία της βιβλιοθήκης AWT αντιμετωπίζονται από java.awt.event.ItemListener. Ως εκ τούτου το vocabulary Java_1.3_Harmonia_1.0 καθορίζει τις μεθόδους ItemListener's, όπως και itemStateChanged, ως γεγονότα UIML εγγρράφου. Στη γραμμή 135 κλείνει το </interface> και στη τελευταια γραμμή ολοκληρώνεται η αναπτυξη του κώδικα με το </uiml.>

4 Εφαρμογές στο έργο eKoNΕΣ

4.1 Τι είναι το eKoNΕΣ

Το έργο eKoNΕΣ στοχεύει στη διερεύνηση, σχεδιασμό, κατασκευή και πιλοτική επίδειξη τεχνολογικών εργαλείων που αποσκοπούν στην προαγωγή της καινοτομικής δραστηριότητας επιχειρηματικών φορέων της περιφέρειας της Κρήτης μέσω προηγμένων μορφών δικτύωσης και συνεργασίας. Συγκεκριμένα, το έργο εστιάζει στην ανάπτυξη ή σύσφιξη κοινοτικών ιστών (community webs) μεταξύ εταίρων (επιχειρηματικών, ερευνητικών, ΟΤΑ ή φορέων που εποπτεύονται από ΟΤΑ) που αναπτύσσονται με βάση τον παγκόσμιο ιστό (WWW) για να προάγουν την σύμπραξη και την συνεργασία και την βελτίωση των τεχνικών ικανοτήτων των μελών και κατ΄ επέκταση την αύξηση της παραγωγικής δύναμης του κοινοτικού ιστού ή της ηλεκτρονικής κοινότητας.

Η προσέγγιση που προτείνεται βασίζεται στην ανάπτυξη προηγμένων εργαλείων που θα συνθέτουν ένα τεχνολογικό περιβάλλον βασισμένο στον παγκόσμιο ιστό, μέσω του οποίου θα μπορούν να γίνονται τα ακόλουθα (βλέπε Σχήμα 17):



Σχήμα 17: Λειτουργική όψη eKONΕΣ

e-Παρουσίαση: Επιχειρηματικοί φορείς θα μπορούν να εγγράφονται στο eKoNΕΣ και χρησιμοποιώντας τα εργαλεία του να αποκτούν ατομική ηλεκτρονική παρουσίαση διαφημίζοντας υπηρεσίες / προϊόντα, παρουσιάζοντας συγκριτικά πλεονεκτήματα, κλπ. Η εγγραφή στο eKoNΕΣ θα αποφέρει συγκριτικά πλεονεκτήματα στα μέλη και πρόσβαση σε υπηρεσίες προστιθέμενης αξίας (βλέπε παρακάτω)

e-Συνεργασία: Μέλη του eKoNΕΣ θα έχουν πρόσβαση σε κλασικές μορφές ασύγχρονης και συγχρονισμένης επικοινωνίας και συνεργασίας όπως email, συζήτηση, συναντήσεις, πίνακα ανακοινώσεων, κλπ που θα καταγράφουν δυναμικά την αλληλεπίδραση των μελών της κοινότητας

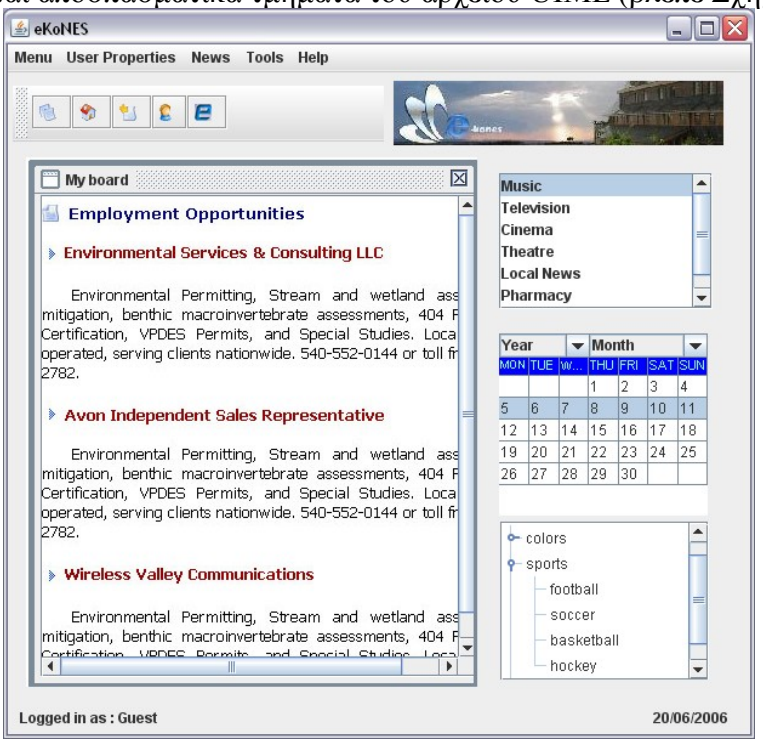
e-Services: Στην κατηγορία αυτή προβλέπεται να υποστηρίζονται εξειδικευμένες υπηρεσίες προστιθέμενης αξίας για τα μέλη της κοινότητας που θα αφορούν διαφόρους τύπους συναλλαγών μεταξύ των εταίρων.

Τα εργαλεία του eKoNES θα σχεδιαστούν έτσι ώστε να είναι γενικά (generic), ενώ η ενοποιημένη πλατφόρμα θα υποστηρίζει κλιμάκωση τόσο του περιεχομένου όσο και των υπηρεσιών της. Με τον τρόπο αυτό το έργο επιχειρεί να συστήσει μια τεχνολογική υποδομή μακράς πνοής που σταδιακά θα προσφέρει προστιθέμενη αξία σε φορείς της Περιφέρειας της Κρήτης. Στο πιλοτικό μέρος του έργου, το eKoNES θα επικυρώσει την τεχνολογική του προσέγγιση και τα εργαλεία του στον κλάδο του τουρισμού, αναπτύσσοντας το eKoNES-Τουρισμός – *το πρώτο ηλεκτρονικό χωριό / κοινότητα* στην Περιφέρεια της Κρήτης με σαφή προσανατολισμό την διεπιχειρησιακή συνεργασία. Το eKoNES-Τουρισμός θα αποτελέσει τη βάση για την μελέτη των δραστηριοτήτων που αναπτύσσουν οι εμπλεκόμενοι (μέλη της κοινότητας), τόσο σε επίπεδο συμπεριφοράς (δηλαδή ποιος επιλέγει να συνεργάζεται με ποιόν, για ποιο σκοπό και με ποιο τρόπο) όσο και σε επίπεδο αύξησης παραγωγικών ικανοτήτων και καινοτομικής δραστηριότητας (ποιοι επωφελούνται, με ποιο τρόπο και γιατί).

Ο όρος «ηλεκτρονικό χωριό» ή «Electronic Village» χρησιμοποιείται για να περιγράψει την χρήση τεχνολογικών εργαλείων επικοινωνίας και συνεργασίας για την υποστήριξη της πληροφόρησης και του διαλόγου μεταξύ μελών μιας κοινότητας σχετικά με θέματα τοπικού ενδιαφέροντος (π.χ. δραστηριότητες, επιχειρηματικότητα, διακυβέρνηση, κλπ)

4.2 Ανάπτυξη και επεξήγηση Container σε UIML

Ακολουθεί μία πρωτότυπη διεπαφή για το έργο eKoNES, για την οποία παρουσιάζονται αποσπασματικά τμήματα του αρχείου UIML (βλέπε Σχήμα 18).



Σχήμα 18: Πρωτότυπο διαπαφής με τη χρήση UIML

Η συγκεκριμένη διεπαφή παράγεται αντιστοιχίζοντας δομές της UIML σε συνιστώσες Java/Swing. Συγκεκριμένα η προδιαγραφή της UIML για το παράθυρο συμπερίληψης είναι:

```
<?xml version="1.0"  encoding="ISO-8859-1" ?>
<uiml>
  <peers>
    <presentation id="base" base="Java_1.4_Harmonia_1.3"/>
  </peers>
<interface>
    <structure>
      <part id="JFrame" class="JFrame" where="last" export="optional"
how="replace">
        <part id="JPanel" class="JPanel" where="last" export="optional"
how="replace">
          <part id="JPanel4" class="JPanel" where="last" export="optional"
how="replace">
            <part id="JPanel6" class="JPanel" where="last" export="optional"
how="replace">
              <part id="JScrollPane1" class="JScrollPane" where="last"
export="optional" how="replace">
                <part id="JTree" class="JTree" where="last" export="optional"
how="replace">
                  <part id="JTree_Root" class="HarmoniaTreeNode" where="last"
export="optional" how="replace">
                    <part id="JTree_Root_Node1" class="HarmoniaTreeNode"
where="last" export="optional" how="cascade">
                      <part id="JTree_Root_Node1_Node1" class="HarmoniaTreeNode"
where="last" export="optional" how="cascade">
…

    </structure>

<style>
      <property part-name="JFrame" name="size" export="optional"
how="replace">620,602</property>
      <property part-name="JFrame" name="title" export="optional"
how="replace"><![CDATA[eKoNES]]></property>
      <property part-name="JFrame" name="layout" export="optional"
how="replace">null</property>
      <property part-name="JFrame" name="location">69,1</property>
      <property part-name="MultiLineLabel" name="text" export="optional"
how="replace"><![CDATA[Logged in as : Guest]]></property>
      <property part-name="MultiLineLabel" name="size" export="optional"
how="replace">125,27</property>
      <property part-name="MultiLineLabel" name="location">10,518</property>
…
</style>
```

Η αντίστοιχη προδιαγραφή για το τη λίστα στο εσωτερικό παράθυρο είναι:

```
<part id="JList1" class="JList" where="last" export="optional" how="replace">
 <style>
  <property part-name="JList1" name="location">-2,5</property>
  <property part-name="JList1" name="size">177,128</property>
  <property part-name="JList1" name="content">
   <constant model="list">
    <constant value="Music"/>
    <constant value="Television"/>
    <constant value="Cinema"/>
    <constant value="Theatre"/>
    <constant value="Local News"/>
    <constant value="Pharmacy"/>
    <constant value="Religion"/>
   </constant>
  </property>
  <property part-name="JList1"
name="debugGraphicsOptions">NONE_OPTION</property>
  <property part-name="JList1" name="layout">null</property>
  <property part-name="JList1" name="doubleBuffered">true</property>
  <property part-name="JList1" name="background">255,255,255</property>
  <property part-name="JList1"
name="selectionMode">SINGLE_SELECTION</property>
 </style>
</part>
```

Το ημερολόγιο είναι υλοποιημένο σε UIML με τη βοήθεια συστατικών Swing που υποστηρίζει το vocabulary Java_1.4_Harmonia_1.3. Συγκεκριμένα απαρτίζεται από δύο JComboBox και ένα JTable. Αναλυτικά η συγκεκριμένη προδιαγραφή υπό μορφή κώδικα είναι η εξής:

Για το πρώτο JComboBox με τίτλο «Year»:

```
<part id="JComboBox" class="JComboBox" where="last" export="optional"
how="replace">
 <style>
  <property part-name="JComboBox" name="location">12,12</property>
  <property part-name="JComboBox" name="size">75,20</property>
  <property part-name="JComboBox" name="content">
   <constant model="list">
    <constant value="Year"/>
    <constant value="1995"/>
    <constant value="1996"/>
    <constant value="1997"/>
    <constant value="1998"/>
    <constant value="1999"/>
    <constant value="2000"/>
    <constant value="2001"/>
```

```
      <constant value="2002"/>
      <constant value="2003"/>
      <constant value="2004"/>
      <constant value="2005"/>
      <constant value="2006"/>
      <constant value="2007"/>
      <constant value="2008"/>
      <constant value="2009"/>
      <constant value="2010"/>
      <constant value="2011"/>
      <constant value="2012"/>
    </constant>
  </property>
 </style>
</part>
```

Για το δεύτερο JComboBox με τίτλο «Month» :

```
<part id="JComboBox1" class="JComboBox" where="last" export="optional"
how="replace">
 <style>
  <property part-name="JComboBox1" name="location">86,12</property>
  <property part-name="JComboBox1" name="size">98,20</property>
  <property part-name="JComboBox1" name="content">
    <constant model="list">
      <constant value="Month"/>
      <constant value="January"/>
      <constant value="February"/>
      <constant value="March"/>
      <constant value="Aprill"/>
      <constant value="May"/>
      <constant value="Jyne"/>
      <constant value="July"/>
      <constant value="August"/>
      <constant value="September"/>
      <constant value="October"/>
      <constant value="November"/>
      <constant value="December"/>
    </constant>
  </property>
 </style>
</part>
```

Για το JTable η αντίστοιχη προδιαγραφή έχει ως εξής:

```
<part id="JTable" class="JTable" where="last" export="optional" how="replace">
 <style>
  <property part-name="JTable" name="location">12,31</property>
  <property part-name="JTable" name="size">172,133</property>
```

```
    <property part-name="JTable" name="editable">false</property>
    <property part-name="JTable" name="background">255,255,255</property>
    <property part-name="JTable"
name="autoCreateColumnsFromModel">false</property>
    <property part-name="JTable" name="content">
      <constant model="table.rowMajor">
        <constant>
          <constant value="MON">
            <constant id="editable" value="false"/>
            <constant id="font" value="Arial-Plain-9"/>
            <constant id="background" value="0,0,255"/>
            <constant id="foreground" value="128,255,255"/>
          </constant>
          <constant value="TUE">
            <constant id="editable" value="false"/>
            <constant id="font" value="Arial-Plain-11"/>
            <constant id="background" value="0,0,255"/>
            <constant id="foreground" value="128,255,255"/>
          </constant>
          <constant value="WED">
            <constant id="editable" value="false"/>
            <constant id="font" value="Arial-Plain-10"/>
            <constant id="background" value="0,0,255"/>
            <constant id="foreground" value="128,255,255"/>
          </constant>
          <constant value="THU">
            <constant id="editable" value="false"/>
            <constant id="font" value="Arial-Plain-11"/>
            <constant id="background" value="0,0,255"/>
            <constant id="foreground" value="128,255,255"/>
          </constant>
          <constant value="FRI">
            <constant id="editable" value="false"/>
            <constant id="font" value="Arial-Plain-11"/>
            <constant id="background" value="0,0,255"/>
            <constant id="foreground" value="128,255,255"/>
          </constant>
          <constant value="SAT">
            <constant id="editable" value="false"/>
            <constant id="font" value="Arial-Plain-11"/>
            <constant id="background" value="0,0,255"/>
            <constant id="foreground" value="128,255,255"/>
          </constant>
          <constant value="SUN">
            <constant id="editable" value="false"/>
            <constant id="font" value="Arial-Plain-11"/>
            <constant id="background" value="0,0,255"/>
            <constant id="foreground" value="128,255,255"/>
          </constant>
        </constant>
```

```
<constant>
  <constant value=" ">
    <constant id="editable" value="false"/>
  </constant>
  <constant value=" "/>
  <constant value=" ">
    <constant id="font" value="Academy Engraved LET-Plain-12"/>
  </constant>
  <constant value="1"/>
  <constant value="2">
    <constant id="editable" value="false"/>
  </constant>
  <constant value="3">
    <constant id="editable" value="false"/>
  </constant>
  <constant value="4">
    <constant id="editable" value="false"/>
  </constant>
</constant>
<constant>
  <constant value="5">
    <constant id="editable" value="false"/>
  </constant>
  <constant value="6">
    <constant id="editable" value="false"/>
  </constant>
  <constant value="7"/>
  <constant value="8"/>
  <constant value="9">
    <constant id="editable" value="false"/>
  </constant>
  <constant value="10">
    <constant id="editable" value="false"/>
  </constant>
  <constant value="11">
    <constant id="editable" value="false"/>
  </constant>
</constant>
<constant>
  <constant value="12">
    <constant id="editable" value="false"/>
  </constant>
  <constant value="13"/>
  <constant value="14"/>
  <constant value="15"/>
  <constant value="16">
    <constant id="editable" value="false"/>
  </constant>
  <constant value="17">
    <constant id="editable" value="false"/>
```

```
        </constant>
        <constant value="18">
          <constant id="editable" value="false"/>
        </constant>
      </constant>
      <constant>
        <constant value="19">
          <constant id="editable" value="false"/>
        </constant>
        <constant value="20"/>
        <constant value="21"/>
        <constant value="22"/>
        <constant value="23">
          <constant id="editable" value="false"/>
        </constant>
        <constant value="24"/>
        <constant value="25">
          <constant id="editable" value="false"/>
        </constant>
      </constant>
      <constant>
        <constant value="26">
          <constant id="editable" value="false"/>
        </constant>
        <constant value="27"/>
        <constant value="28"/>
        <constant value="29"/>
        <constant value="30">
          <constant id="editable" value="false"/>
        </constant>
        <constant value=" "/>
        <constant value=" "/>
      </constant>
    </constant>
  </property>
 </style>
</part>
```

Η αντίστοιχη προδιαγραφή για το JInternalFrame στο εσωτερικό παράθυρο με τίτλο «MyBoard» είναι:

```
<part id="JInternalFrame2" class="JInternalFrame" where="last" export="optional"
how="replace">
 <style>
  <property part-name="JInternalFrame2" name="size" export="optional"
how="replace">372,436</property>
  <property part-name="JInternalFrame2" name="title" export="optional"
how="replace"><![CDATA[My board]]></property>
```

```
   <property part-name="JInternalFrame2" name="layout" export="optional"
how="replace">null</property>
   <property part-name="JInternalFrame2" name="location">2,2</property>
   <property part-name="JInternalFrame2" name="resizable">false</property>
   <property part-name="JInternalFrame2"
name="debugGraphicsOptions">NONE_OPTION</property>
   <property part-name="JInternalFrame2" name="enabled">true</property>
   <property part-name="JInternalFrame2" name="icon">true</property>
   <property part-name="JInternalFrame2" name="iconifiable">false</property>
   <property part-name="JInternalFrame2" name="closable">true</property>
   <property part-name="JInternalFrame2" name="visible">true</property>
   <property part-name="JInternalFrame2" name="closed">true</property>
   <property part-name="JInternalFrame2" name="font">Academy Engraved LET-
Plain-12</property>
  </style>
</part>
```

Στην περιγραφή των παραπάνω προδιαγραφών παρατηρείται ο σαφής διαχωρισμός τους από τις υλοποιημένες κάσεις μιας πλατφόρμας. Αυτό επιτρέπει στη UIML την αυτόματη μετατροπή / αντιστοίχιση προδιαγραφών σε συνιστώσες διαφορετικής πλατφόρμας γεγονός που εξυπηρετεί την ανάπτυξη πολλαπλών διεπαφών.

4.3 Κλήση εξωτερικών κλάσεων Java

Αυτό το τμήμα εξηγεί πώς να καλέσουμε ή να αποσύρουμε μεθόδους από ένα UIML έγγραφο. Η έκδοση 3.0 υποστηρίζει κλήσεις αντικειμένων υλοποιημένων σε Java με τη βοήθεια της ετικέτας <call>,σε συνεργασία με τις ετικέτες <peers> και <logic>.

4.3.1 Περιγραφή της ετικέτας <call>

Η φιλοσοφία της UIML όσο αφορά των τομέα των επικλήσεων είναι η δυνατότητα που προσφέρει στον προγραμματιστή της UIML να επιλέγει ελεύθερα ένα σετ (μια ομάδα) ονομάτων για widget, γεγονότων και λειτουργίων που αναφέρονται στο τμήμα του <interface>. Κάθε ένα από τα ονόματα είναι χαρτογραφημένα στο τμήμα του <peers> για την εφαρμογή οντοτήτων (π.χ Swing user-interface components κτλ.). Για τα συστατικά του Swing, η σκληρή δουλειά της δημιουργίας της χαρτογράφησης έχει ήδη υλοποιηθεί. Οι προγραμματιστές της UIML χρειάζονται μόνο να εισάγουν ένα ήδη υπάρχον σώμα από προκαθορισμένες χαρτογραφήσεις, αντί να γράφουν ένα κατάλληλο τμήμα <peers>. Για τις μεθόδους , ο προγραμματιστής της UIML πρέπει να γράψει ένα <d-component> στοιχείο για κάθε διαδικασία της οποίας η μέθοδος ή οι μέθοδοι μπορούν να κληθούν για την ολοκλήρωση του εγγράφου (σημειώνουμε ότι το <d-component> είναι ένα παιδί του <logic>, το οποίο με τη σειρά του είναι ένα παιδί του <peers>).

Ας υποθέσουμε ότι το έγγραφο UIML περιέχει ένα στοιχείο <action> κατά το οποίο το αντικείμενο back1 επικαλείται τη μέθοδο m1 με παραμέτρους 5 και 10 (΄΄back1.m1(5,10)΄΄). Το τμήμα <action> περιγράφεται ως εξής:

```
<behavior>
    <rule>
    ...
        <action>
            <call name=”back1.m1”>
                <param>5</param>
                <param>10</param>
            </call>
        </action>
    </rule>
</behavior>
```

Το στοιχείο <d-component> προσδιορίζοντας το αντικείμενο back1 θα έχει την παρακάτω μορφή:

```
<peer>
    <logic>
        <d-component id="back1"
              maps-to="org.uiml.example.myClass">
            <d-method id="m1" maps-to="myFunction">
              <d-param id="p1"/>
              <d-param id="p2"/>
            </d-method>
        </d-component>
    </logic>
</peer>
```

Το <d-component> χαρτογραφεί το όνομα back1 στην κλάση org.uiml.example.myClass που περιλαμβάνει μία μέθοδο με το όνομα myFunction, στην οποία χαρτογραφείται με τη σειρά της η μέθοδος m1. Για την επίκληση της myClass.myFunction(p1,p2) επικαλούμε το στοιχείο <call> που θα αναφέρεται στις αντίστοιχες αξίες των ιδιοτήτων id των στοιχείων <d-component> και <d-method>, με αποτέλεσμα η έκφραση back1.m1(5,10) να αποδοθεί ως μία ‘εκτίμηση’ της myClass.myFunction(5,10). Σημειώνουμε ότι αποδόσεις των id σε στοιχεία <d-param> όπως για παράδειγμα <d-param id = “p1”> είναι προαιρειτικές, απλώς εξυπηρετούν στην παρουσίαση των παραμέτρων της myFunction. Ο μηχανισμός με τον οποίο ο Java Renderer καλεί μία μέθοδο περιγράφεται ως εξής:

1. Ο Renderer χρησιμοποιεί την είκονα της Java για να αναγνωρίσει τη μέθοδο η οποία περιγράφεται από το στοιχείο <d-method>.
2. Εάν η μέθοδος δεν είναι στατική και δεν υπάρχουν instance της κλάσης ,ο Java Renderer δημιουργεί ένα στιγμιότυπο της κλάσης. Η κλάση πρέπει να ειναι no-argument constructor.

3. Μερικά ορίσματα που περιγράφονται στο στοιχείο <d-method> μετατρέπονται απο java.lang.string στο τύπο που απαιτείται.
4. Η κλήση εκτελείται
5. Εάν δεν υπάρχει μια τιμή επιστροφής, αυτή μετατρέπεται σε java.lang.string.To String χρησιμοποιείται σε οποιοδήποτε στοιχείο εμπεριέχεται στο τμήμα <call>, ώς η τιμή (value) του ίδιου του στοιχείου.

Πρώτο παράδειγμα χρήσης της <call> (object instance method)

Ας υποθέσουμε για παράδειγμα, μία διεπαφή η οποία περιέχει ένα JLabel που αναγράφει τη φράση "Please press the button" και ένα JButton "Press me" (βλέπε Σχήμα 19). Με το πάτημα του κουμπιού θα εμφανίζεται ένας ακέραιος (βλέπε Σχήμα 20) που θα περιγράφει πόσες φορές επαναλήφθηκε η παραπάνω ενέργεια.

Σχήμα 19: Στιγμιότυπο Counter1

Σχήμα 20: Στιγμυότυπο Counter1 – Πάτημα "Press me" και αύξηση του μετρητή

Στόχος μας είναι να κάλεσουμε μέσω του εγγράφου UIML τη μέθοδο Count της παρακάτω κλάσης:

```
Public class Counter1 {
        private int _currCount = 0;
        public int count() {
                return ++ _currCount;
        }
}
```

Παρατηρούμε ότι στο παρακάτω UIML έγγραφο και συγκεκριμένα στο τμήμα <action>… </action> η τιμή του <property> είναι :<call name = "Counter.count" />. Η ονομασία Counter.count θα χρησιμοποιηθεί στο τμήμα <logic>… </logic> ώστε να χαρτογραφηθούν τα συστατικά της παραπάνω ονομασίας. Συγκεκριμένα το τμήμα Counter χαρτογραφειται σε μια κλάση της Java χρησιμοποιώντας τα χαρακτηριστικά που έχουν αποδοθεί στο συστατικο Counter1.

<d-component id = "Counter" maps-to = "Counter1" />

Ομοίως και με τη μέθοδο count, η οποία 'αντιστοιχίζεται' με τη μέθοδο count της κλάσης Counter1 με τη βοήθεια του στοιχείου <d-method>

<d-method id = "count" return-type = "int" maps-to = "count" />

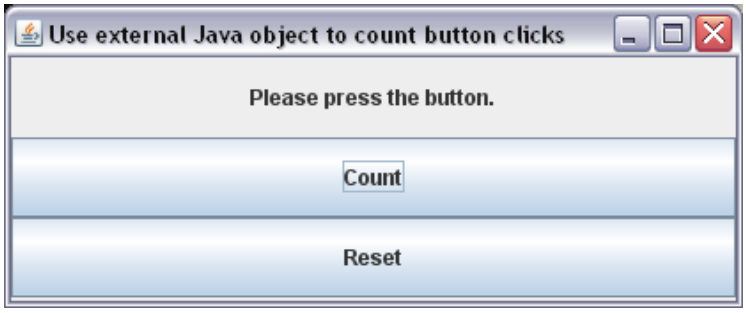
Αναλυτικά η διεπαφή υπό μορφή κώδικα:

```
<?xml version="1.0"?>
<!DOCTYPE uiml PUBLIC "-//Harmonia//DTD UIML 2.0 Draft//EN"
"UIML2_0g.dtd">
<uiml>
    <interface id="myinterface">
        <structure>
            <part id="TopHello" class="JFrame">
                <style>
                    <property name="title">Use external Java object to
                        count button clicks</property>
                    <property name="layout">java.awt.GridLayout</property>
                    <property name="layout_rows">2</property>
                    <property name="bounds">100,100,375,120</property>
                </style>

            <part id="Label" class="JLabel">
                <style>
                     <property name="text">Please press the button.</property>
                      <property name="horizontalAlignment">CENTER</property>
                 </style>
            </part>

            <part id="Button" class="JButton">
                <style>
                      <property name="text">Press me!</property>
                 </style>
            </part>
        </part>
    </structure>
    <behavior>
         <rule>
            <condition>
              <event class="actionPerformed" part-name="Button"/>
```

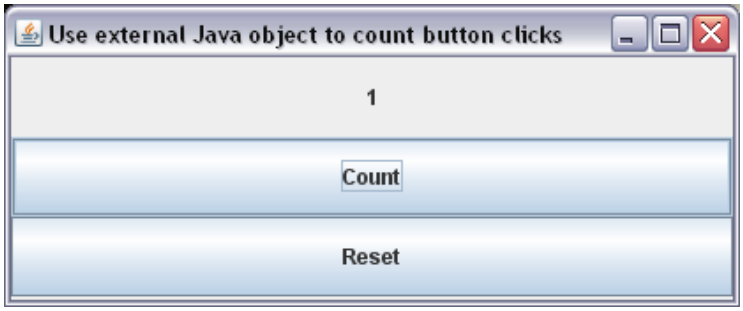
```
            </condition>
            <action>
               <property part-name="Label"
                 name="text"><call name="Counter.count"/></property>
            </action>
          </rule>
      </behavior>
    </interface>

     <peers>
        <presentation base="Java_1.3_Harmonia_1.0"/>
        <logic>
            <d-component id="Counter"maps-to="Counter1">
            <d-method id="count" return-type="int" maps-to="count"/>
            </d-component>
        </logic>
     </peers>
</uiml>
```

Δεύτερο παράδειγμα χρήσης της <call> (Static Method Class)

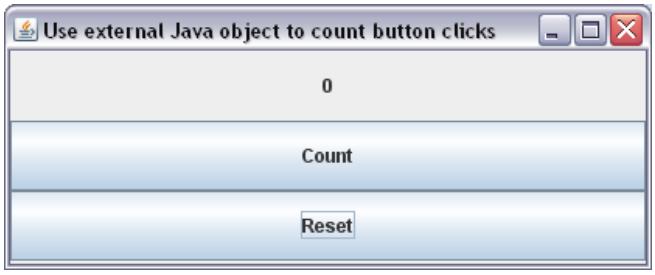
Αυτό το παράδειγμα διαφέρει από το προηγούμενο σε πολλά σημεία. Υπάρχει ένα δεύτερο κουμπί με την ονομασία «Reset» (βλέπε Σχήμα 21) το οποίο μηδενίζει το μετρητή και αναγράφει στο JLabel την τιμή μηδέν (βλέπε Σχήμα 23). Το στοιχείο <d-componetnts> του εγγράφου UIML περιέχει δυο συστατικά <d-method>, όπου το δεύτερο αναφέρεται για το Reset button και προστίθεται και μία παράμετρο τύπου Int. Η παράμετρος περνάει στο UIML έγγραφο μέσω του στοιχείου <param>, το οποίο περικλύεται στο τμήμα <action>... </action>. Σημειώνουμε ότι οι μέθοδοι στο συγκεκριμένο παράδειγμα είναι στατικές.



Σχήμα 21: Στιγμιότυπο Counter2



Σχήμα 22: Στιγμιότυπο Counter2 – Πάτημα ″Count″ JButton και αύξηση μετρητή



Σχήμα 23: Στιγμυότυπο Counter2 – Πάτημα ″Reset″ JButton και μηδενισμός μετρητή

Η κλάση Counter2:

```
public class Counter2 {
        private static int _count = 0;

        public static int count() {
        return ++_count;
        }

        public static int setCount(int newVal) {
        _count = newVal;
        return _count;
        }
}
```

Αναλυτικά ο κώδικας του UIML εγγράφου το οποίο καλεί την παραπάνω κλάση:

```
<?xml version="1.0"?>
<!DOCTYPE uiml PUBLIC "-//Harmonia//DTD UIML 2.0 Draft//EN"
"UIML2_0g.dtd">

<uiml>
    <interface id="myinterface">

        <structure>
            <part id="TopHello" class="JFrame">
                <style>
                    <property name="title">Use external Java object to
                     count button clicks</property>
                    <property name="layout">java.awt.GridLayout</property>
                    <property name="layout_rows">3</property>
                    <property name="bounds">100,100,375,150</property>
                </style>

                <part id="Label" class="JLabel">
                    <style>
                        <property name="text">Please press the button.</property>
```

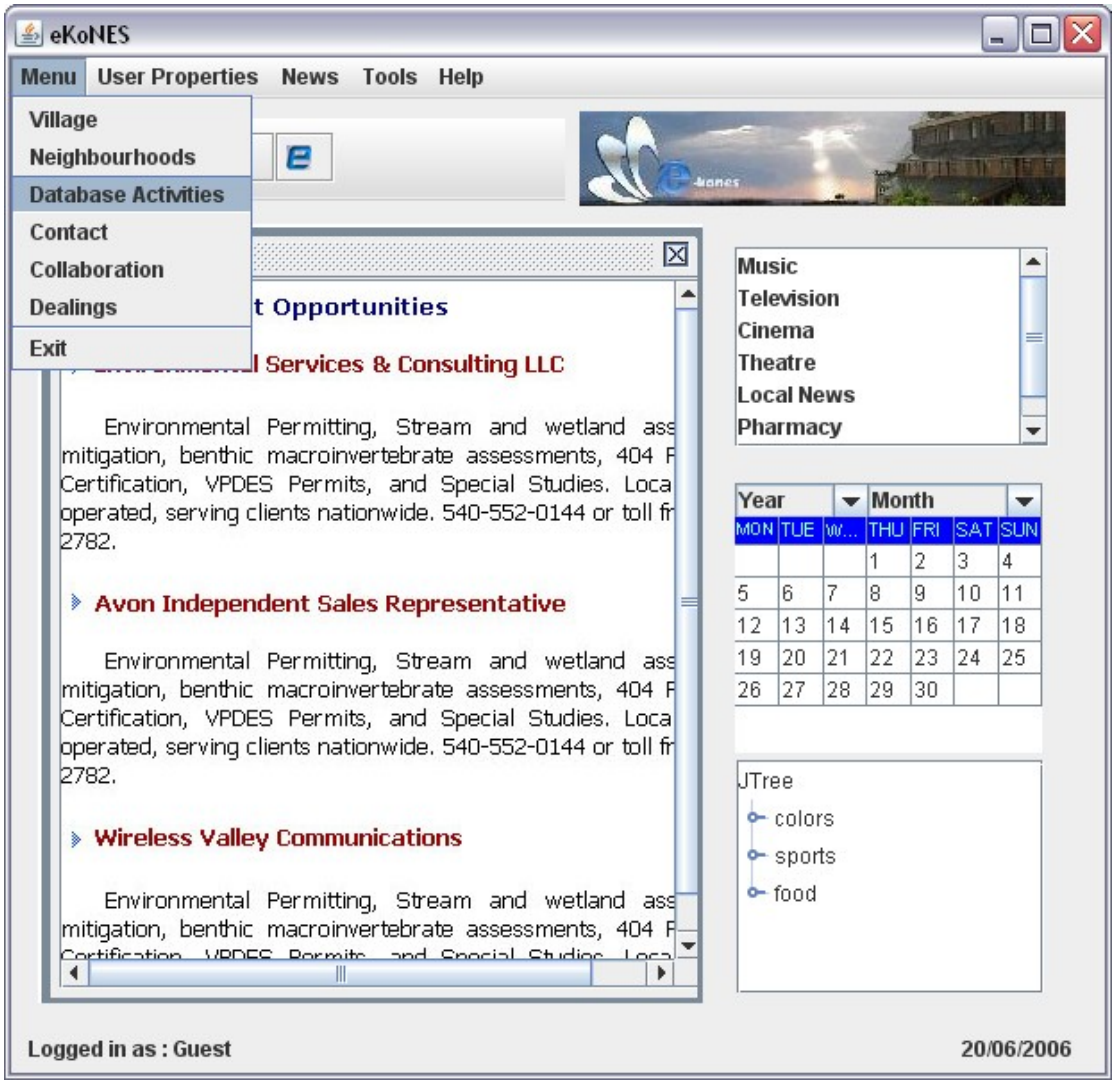
```
                    <property name="horizontalAlignment">CENTER</property>
                </style>
            </part>

        <part id="Button" class="JButton">
            <style>
                <property name="text">Count</property>
            </style>
        </part>
        <part id="ResetButton" class="JButton">
            <style>
                <property name="text">Reset</property>
            </style>
        </part>

    </part>
  </structure>

  <behavior>
      <rule>
          <condition>
              <event class="actionPerformed" part-name="Button"/>
          </condition>
          <action>
          <property part-name="Label"
           name="text"><call name="Counter.count"/></property>
          </action>
      </rule>

      <rule>
          <condition>
              <event class="actionPerformed" part-name="ResetButton"/>
          </condition>
          <action>
          <property part-name="Label" name="text">
              <call name="Counter.reset">
                  <param name="newVal">0</param>
              </call>
          </property>
          </action>
      </rule>
  </behavior>
</interface>

 <peers>
     <presentation base="Java_1.3_Harmonia_1.0"/>
     <logic>
         <d-component id="Counter" maps-to="Counter2">
         <d-method id="count" return-type="int" maps-to="count"/>
         <d-method id="reset" return-type="int" maps-to="setCount">
```

```
        <d-param id="newVal" type="int"/>
        </d-method>
        </d-component>
      </logic>
  </peers>
</uiml>
```

4.3.2 Κλήση εξωτερικών κλάσεων Java στο έργο eKoNES

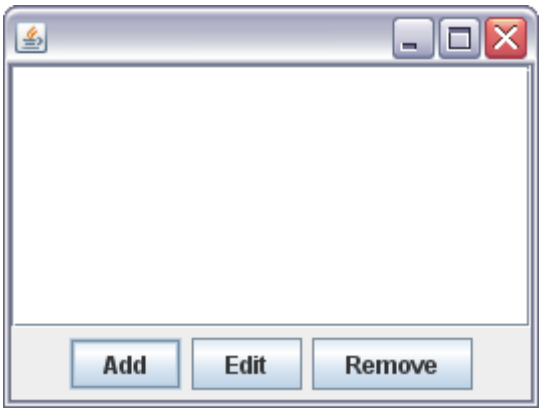
Στόχος μας είναι η δημιουργία μιας εφαρμογής κατα την οποία από το έγγραφο UIML θα καλέσουμε μία κλάση υλοποιημένη σε Java, η οποία μας δίνει τη δυνατότητα εισαγωγής και αφαίρεσης δεδομένων (όπως για παράδειγμα πακέτα πληροφοριών) από μία βάση δεδομένων. Η παραπάνω διαδικασία περιγράφεται σχηματικά ώς εξής:

Επιλογή Menu → Database Activities



Σχήμα 24: Πρωτότυπο διαπαφής υλοποιμένο σε UIML

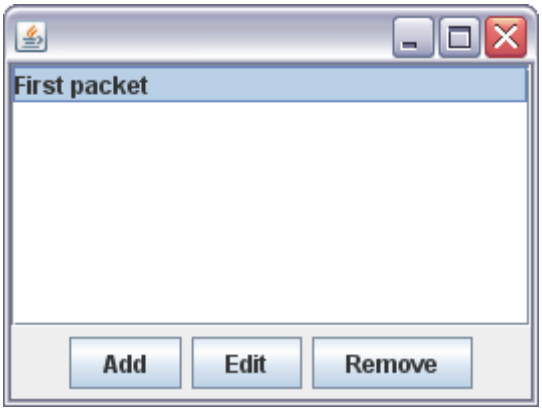
Εμφανίζεται το παρακάτω πλαίσιο με δυνατότητα επιλογής Add , Edit και Remove



Σχήμα 25: Πλαίσιο υλοποιημένο σε Java – Επιλογές αλληλεπίδραση με βάση δεδομένων

Επιλέγοντας το κουμπί "Add" εμφανίζουμε το πλαίσιο για την εισαγωγή δεδομένων με προορισμό τη βάση του eKoNES και στη συνέχεια "Execute Update".Το αποτέλεσμα διακρίνεται στο στιγμιότυπο Σχήμα 4-11.

Σχήμα 26: Ενδιάμεσο πλαίσιο μετά το πάτημα του "Add" button



Σχήμα 27: Αποτέλεσμα ενέργειας "Execute Update"

Για την υλοποίηση της παραπάνω διαδικασίας θα πρέπει αρχικά να οριστεί το τμήμα <rule>...</rule>, το οποίο συμπεριλαμβάνεται στο στοιχείο <behavior>, και καθορίζει τους όρους και τις δράσεις (actions) για κάποιο γεγονός.

```
<rule id="JMenuItem22" export="optional" how="replace">
    <condition>
```

```
      <event class="mouseReleased" part-name="JMenuItem22"/>
    </condition>
    <action>
      <property part-name="JMenuItem22" name="first_action">
        <call name="Main.startx" class="JMenuItem"/>
      </property>
    </action>
  </rule>
```

Το ″JMenuItem22″ αναφέρεται στο ″Database Activities″ όπου με την ενέργεια του χρήστη ″mouseReleased″ θα πραγματοποιθεί η κλήση της έκφρασης Main.startx. ″Main″ είναι η κλάση που έχει υλοποιθεί σε Java και ″startx″ η μέθοδος που περιέχεται σ'αυτή.Με τη χρήση της μεθόδου γίνεται η ανάκτηση του πλαισίου (π,χ new MyJFrame();). Το στοιχείο <peers> του UIML εγγράφου περιγράφει τις χαρτογραφήσεις των κλάσεων ,διαδικασία όμοια με αυτή που περιγράψαμε στα παραδείγματα της ενότητας 4.3, με τη μόνη διαφορά ότι το ″com.Harmonia.test″ αναφέρεται στο πακέτο της Java (Java package), στο οποίο περιέχονται τα συστατικά για την ολοκλήρωση της εφαρμογής.

```
<peers>
  <logic>
    <d-component id="Main" maps-to="com.Harmonia.test.Main">
      <d-method id="startx" maps-to="startx" return-type="void"/>
    </d-component>
  </logic>
  <presentation id="base" base="Java_1.4_Harmonia_1.3"/>
 </peers>
```

Τέλος πρέπει να τονίσουμε ότι το αποτέλεσμα της εφαρμογής δεν ήταν το αναμενόμενο. Υπάρχει ένας σαφής διαχωρισμός μεταξύ του UIML εγγράφου και της κλήσης συστατικών υλοποιημένων σε Java. Συγκεκριμένα δεν μπορούν να ενσωματωθούν οι υλοποιμένες κλάσεις της Java ώς συστατικά του UIML Container, αλλά ″τρέχουν″ σαν ξεχωριστές εφαρμογές κάτι το οποίο επιβεβαίωσε και η κοινοπραξία Harmonia (www.harmonia.com) με την ελπίδα ότι σε επόμενες εκδόσεις, όπως ακριβώς ανέφερε, θα βρεθεί λύση για το συγκεκριμένο πρόβλημα.

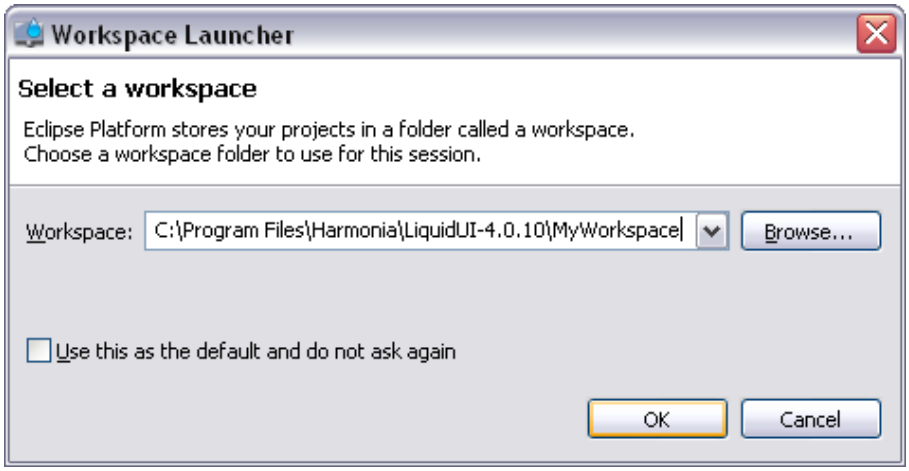
4.4 Τα εργαλεία Liquid για την ανάπτυξη UIML εγγράφων

Ένα από τα εργαλεία που διατίθεται μέσω της κοινοπραξίας Harmonia (http://www.harmonia.com) είναι το LiquidUI. Το όραμα του LiquidUI είναι η παροχή μιας ακολουθίας εργαλίων (uDevelop & uDraw,uBrowse) για τη δημιουργία και παρουσίαση multi-platform διεπαφών με τη χρήση της UIML.Η προσπάθεια αυτή βασίζεται στις ανάγκες για υγιή εφαρμογή μιας γλώσσας σε ποικίλες συσκευές.

4.4.1 uDevelop

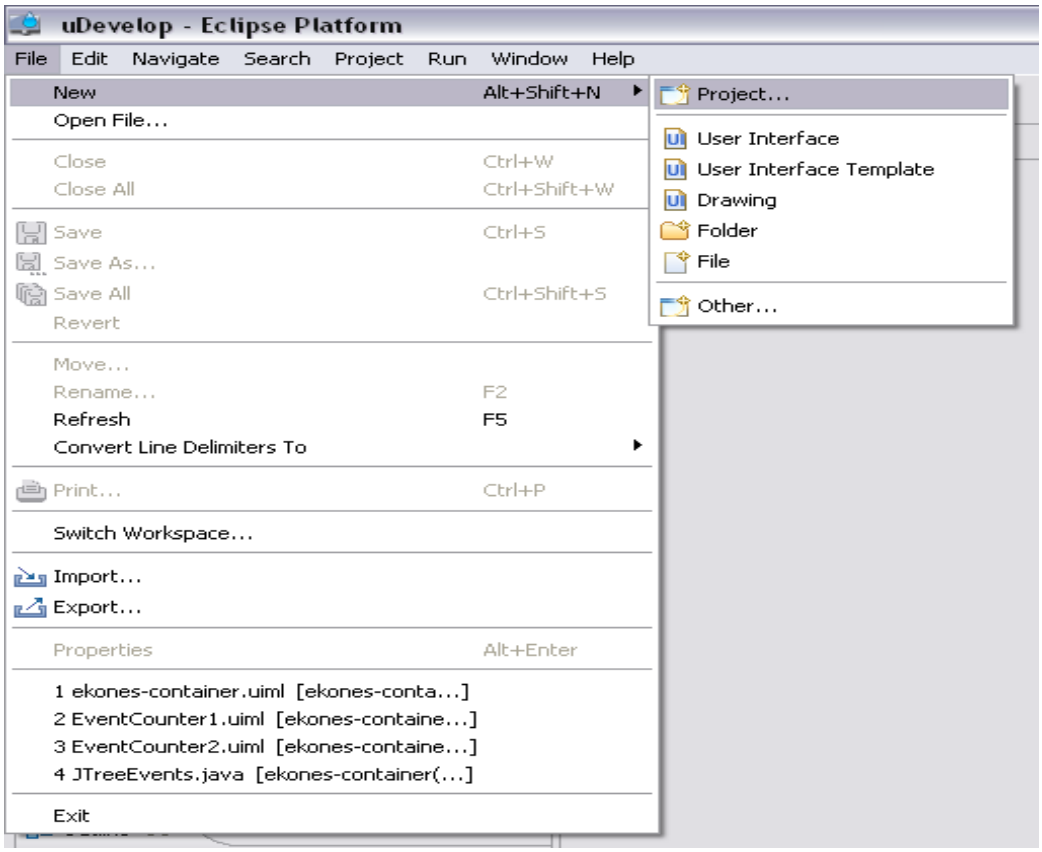
Η έναρξη του εργαλείου γίνεται μέσω της ακολουθίας All programs → LiquidUI-4.0 → uDevelop & uDraw και στη συνέχεια:

Επιλογή “workspace”

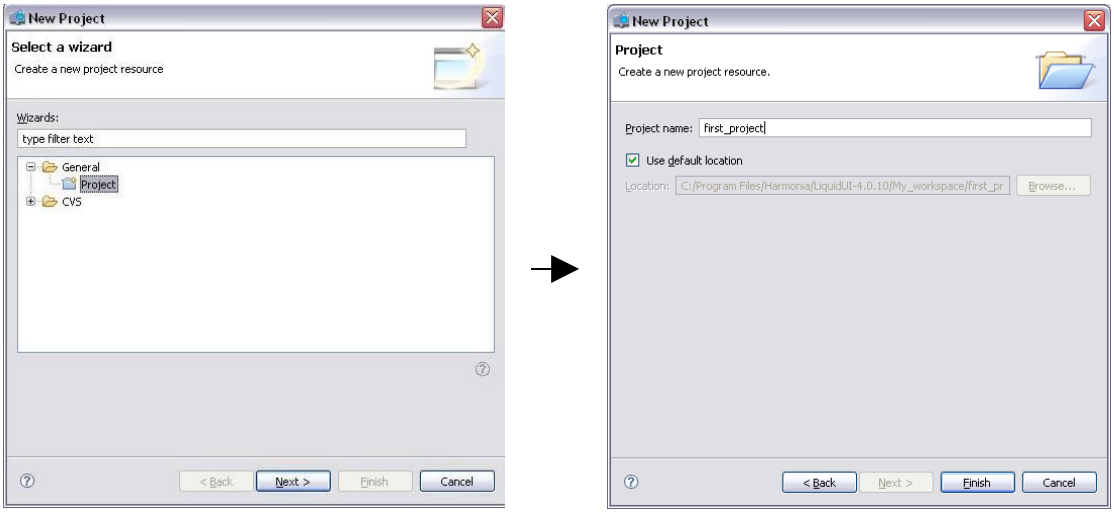


Σχήμα 28: Στιγμιότυπο uDevelop – Επιλογή Workspace

Δημιουργία νέου project:

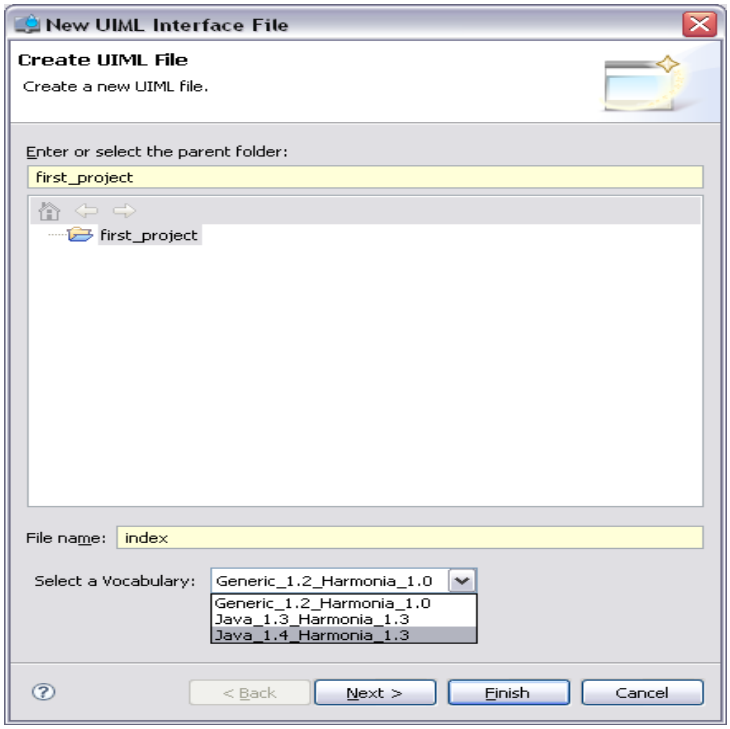


Σχήμα 29: Στιγμιότυπο uDevelop – Δημιουργία νέου Project



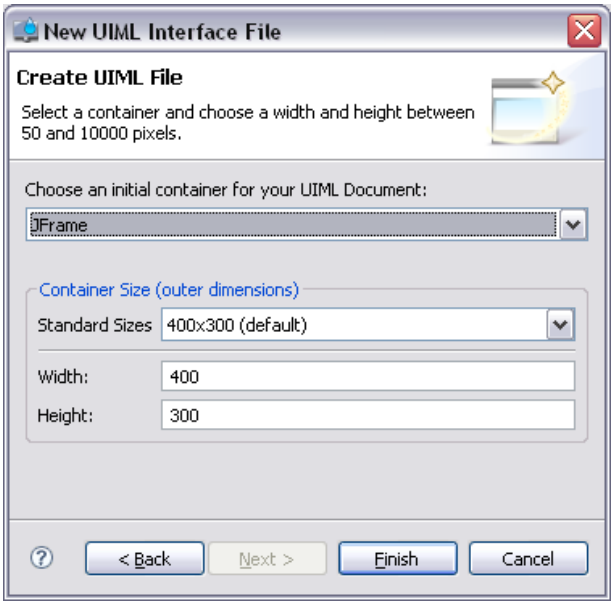
Σχήμα 30: Στιγμιότυπο uDevelop – Δημιουργία νέου Project

Επιλογή ονόματος του UIML File (π.χ index) καθώς και ενός από τα διαθέσιμα vocabularies



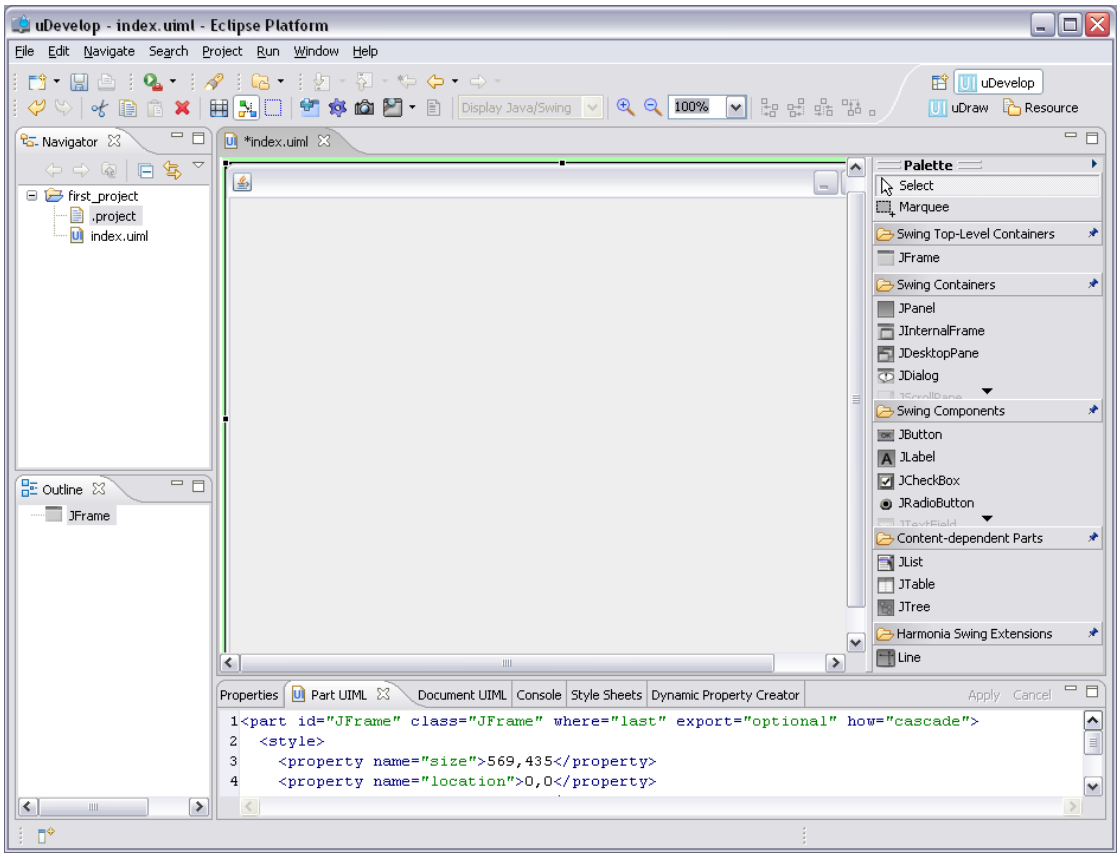
Σχήμα 31: Στιγμιότυπο uDevelop – Δημιουργία νέου UIML file

Δυνατότητα επιλογής Top Container κατά τη δημιουργία του New UIML Document π.χ JFrame, καθώς και κάποια από τα χαρακτηριστικά του (Width,Height)



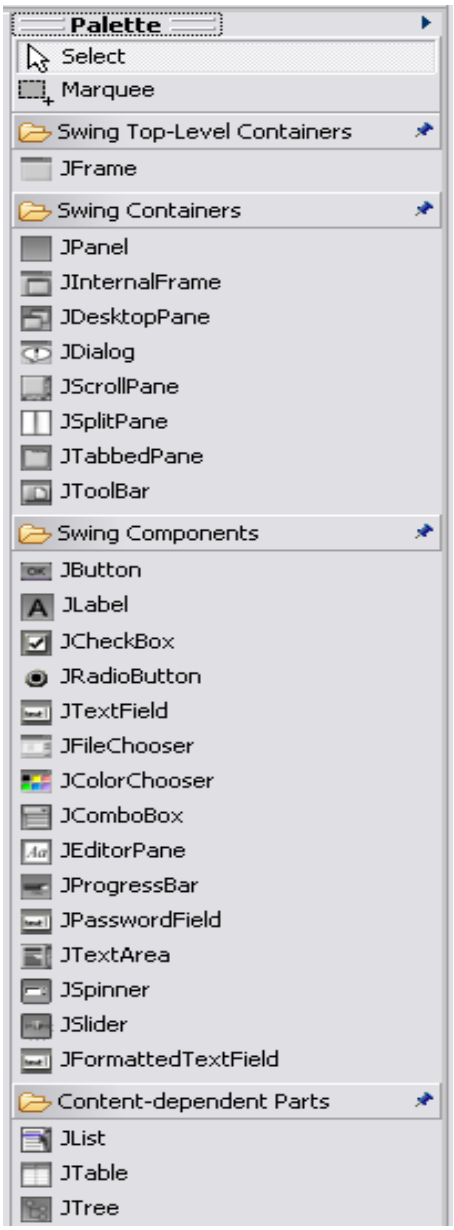
Σχήμα 32: Στιγμιότυπο uDevelop – Επιλογή Top Container

Το αποτέλεσμα της παραπάνω διαδικασίας οδηγεί στην παρουσίαση της διεπιφάνειας χρήσης που εμφανίζεται στο Σχήμα 33.



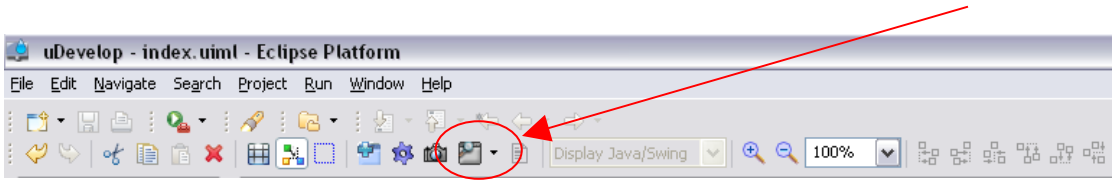
Σχήμα 33: Στιγμιότυπο uDevelop – Επιφάνεια εργασίας

Παρατηρούμε ότι το εργαλείο LiquidUI μας δίνει τη δυνατότητα χρήσης Swing components. Για τη χρήση συστατικών Swing επιλογή vocabulary: Java_1.4_Harmonia_1.3

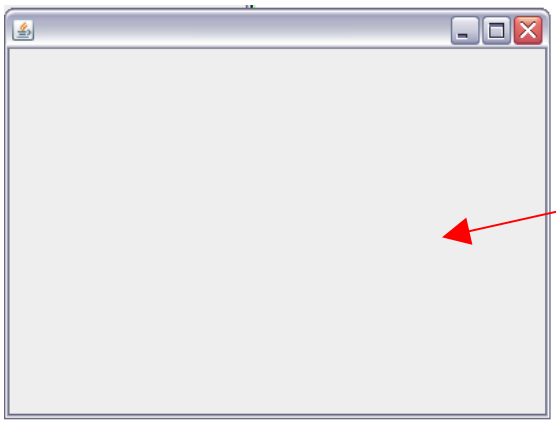


Σχήμα 34: Στιγμιότυπο uDevelop – Swing Components

Για την προεπισκόπηση της διεπαφής κανουμε κλικ (Preview User Interface for Currently Opened Editor using Java).



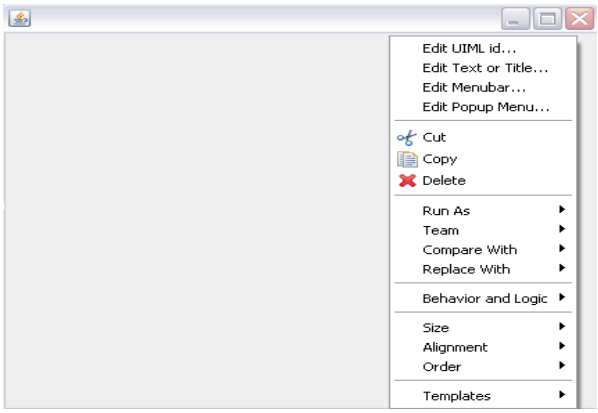
Σχήμα 35: Η δυνατότητα rendering ενός UIML εγγράφου



Σχήμα 36: Το παράθυρο που παράγεται μετά την αντιστοίχηση στο λεκτικό του Swing

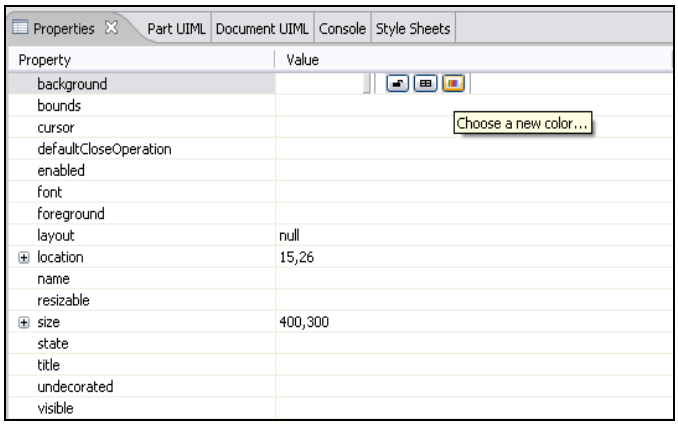
Το LiquidUI μας δίνει τη δυνατότητα να τροποποιήσουμε, να εισάγουμε καθώς επίσης και να αφαιρέσουμε χαρακτηριστικά του επιλεγμένου αντικειμένου με διάφορους τρόπους:

Δεξί κλικ εντός των ορίων του αντικειμένου:



Σχήμα 37: Στιγμιότυπο uDevelop – Μορφοποίηση Container

Από τη κατηγορία Properties του επιλεγμένου αντικειμένου



Σχήμα 38: Στιγμιότυπο uDevelop – Properties Container

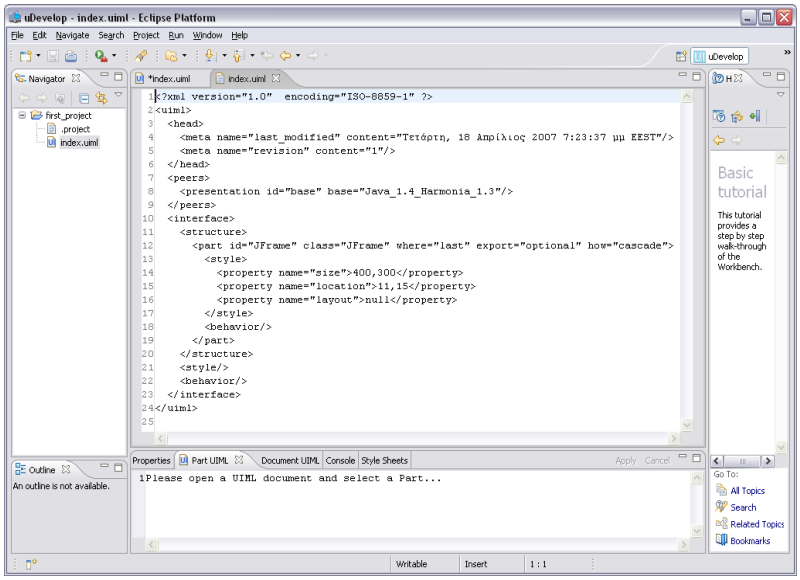
Από τη κατηγορία Part UIML

```
Properties | Part UIML | Document UIML | Console | Style Sheets        Apply  Cancel
1 <part id="JFrame" class="JFrame" where="last" export="optional" how="cascade">
2   <style>
3     <property name="size">400,300</property>
4     <property name="location">15,26</property>
5     <property name="layout">null</property>
6   </style>
7 </part>
8
```

Σχήμα 39: Στιγμιότυπο uDevelop – Properties Container υπό μορφή κώδικα

Σημείωση: Οι τροποποιήσεις ισχύουν μόνο για το επιλεγμένο αντικείμενο(στη συγκεκριμένη περίπτωση JFrame) και όχι για το σύνολο της διεπαφής μας, που θα μπορούσε να περιέχει JButtons JTrees, JPanels κτλ.

Για τη παρουσίαση της συνολικής διεπαφής υπό μορφή κώδικα Δεξί κλικ στο index.uiml → Open with → Text Editor

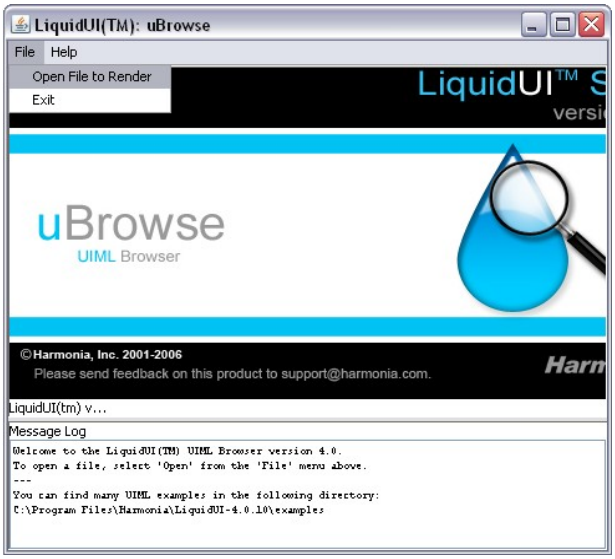


Σχήμα 40 : Στιγμιότυπο uDevelop – Παρουσίαση κώδικα διεπαφής

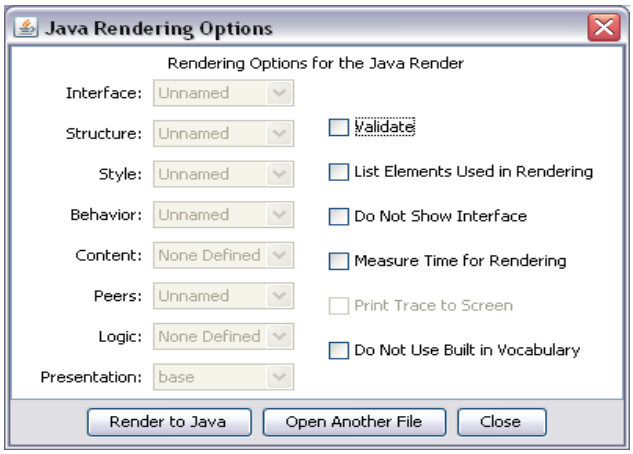
4.4.2 uBrowse

Η κοινοπραξία Harmonia εμπεριέχει στο πακέτο LiquidUI το εργαλείο uBrowse. Πρόκειται για ένα εργαλείο παρουσίασης εφαρμογών υλοποιημένων σε UIML χωρίς τη χρήση uDevelop & uDraw.

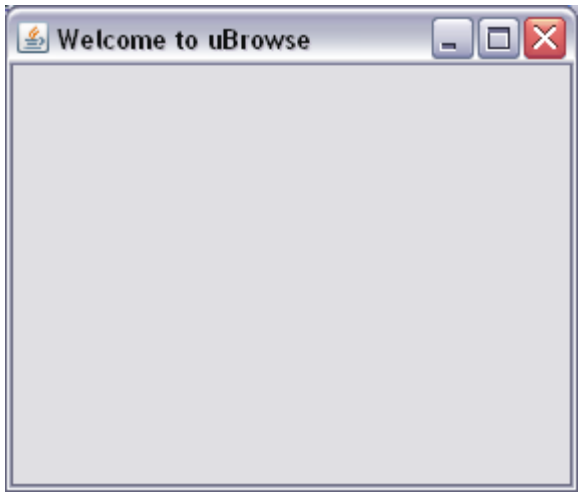
Σχηματικά:



Σχήμα 41: Στιγμιότυπο uBrowse – Επιλογή UIML file για προεπισκόπιση



Σχήμα 42: Στιγμιότυπο uBrowse – Ενεργοποίηση Renderer



Σχήμα 43: Στιγμιότυπο uBrowse – Αποτέλεσμα επιλογής "Render to Java"

Αν και το LiquidUI δεν είναι ευρέως διαθέσιμο, αλλά ούτε και ολοκληρωμένο, συνεισφέρει ουσιαστικά στη γρήγορη ανάπτυξη διεπαφών.

5 Συμπεράσματα

Η προσέγγιση της UIML μας εισάγει σε νέες τεχνολογίες ανάπτυξης διεπαφών, πλούσιες σε ερευνητική δραστηριότητα και συνεχώς εξελισσόμενες. Η παραπάνω γλώσσα είναι εύκολα κατανοητή και αρκεί λίγη εμπειρία στο αντικείμενο των tags ώστε να υλοποιηθεί μια διεπαφή μεσω της χρήσης UIML. Τα πλεονεκτήματα της, όπως έχουν προαναφερθεί, είναι πολλά, σημαντικά και ελπιδοφόρα για το μέλλον. Η UIML εκτός από το πόσο φιλική είναι με τον προγραμματιστή, προσφέρει τεράστιο κέρδος χρόνου αφού μειώνει το χρόνο ολοκλήρωσης διεπαφών για μία οικογένεια συσκευών. Το βασικότερο πλεονέκτημα όλων είναι πως επιτρέπει την επέκταση για την υποστήριξη αναδυόμενων τεχνολογιών αλληλεπίδρασης που αναμένεται να εφευρεθούν στο μέλλον.Κατά την εκπόνηση της πτυχιακής εργασίας παρατηρήθηκαν αρκετά προβλήματα. Συγκεκριμένα οι περιορισμένες πηγές εγγράφων πάνω στο αντικείμενο της UIML ήταν σοβαρό εμπόδιο για την έναρξη της διεπαφής του ekones και γενικότερα μιας εμπεριστατωμένης προσέγισσης της γλώσσας. Στον Παγκόσμιο Ιστο, εκτός τους ιστοτόπους www.uiml.org και www.harmonia.com οι αναφορές σε UIML ήταν ελάχιστες. ‘Οσο αναφορά το εργαλείο για ανάπτυξη UIML βρίσκόταν και βρίσκεται σε δοκιμαστική (beta) έκδοση με συνεχόμενες αναβαθμίσεις, πράγμα το οποίο δυσχέραινε την κατανόηση ενός συγκεκριμένου εργαλείου σε βάθος. Η συμβατότητα με τις γλώσσες που αναφέρονται ότι μπορούν να ενωματωθούν σε ένα έγγραφο και κατ’ επέκταση μια διεπαφή της UIML δεν ήταν η αναμενόμενη. Συγκεκριμένα παρατηρήθηκαν εμπόδια στην ενσωμάτωση κλάσεων Java στον Container της UIML Επίσης το UBrowse που παρέχεται μέσα στο πακέτο του LiquidUI εμφανίζει μόνο διεπαφές που έχουν υλοποιηθεί με με vocabulary Generic και Java. Δηλαδή αναπτυγμένες διεπαφές σε wml και html δεν ήταν αναγνώσιμες από το εργαλείο.

Συμπερασματικά, απο την ολοκλήρωση της πτυχιακής εργασίας, ο στόχος που είχε τεθεί πραγματοποιήθηκε κατα τη γενικότερη έννοια, ενώ έθεσε τις βάσεις για τις μελλοντικές αναβαθμίσεις της γλώσσας που θα καλύπτουν πλήρως όλες τις απαιτήσεις.

6 Επίλογος

Οι δυνατοτητές της συγκεκριμένης γλώσσας σήμανσης, σήμερα είναι περιορισμένες, όμως είναι μια παρακαταθήκη για το μέλλον, αφού συνεχώς αναβαθμίζεται και ενημερώνεται. Όραμα της UIML και όραμα όσων ασχολούνται με αυτήν, είναι η ανάπτυξη διεπαφών σε σύντομο χρονικό, που θα καλύπτουν τις ανάγκες πολλών πλατφορμων. Το έργο ekones με τη σειρά του εισάγει μια καινοτομία στο τρόπο αντίληψης διεπαφών διαδικτύου. 'Ολος του ο σχεδιασμός και ανάπτυξη του έχει να προσφέρει πολλά τόσο στην εκπαιδευτική κοινότητα με τις τεχνολογίες που χρησιμοποιεί όσο και στον τομέα που αναλώνεται, δηλαδή τον τουρισμό. Ευελπιστούμε στα επόμενα χρόνια να επιτευχθούν πλήρως οι στόχοι της UIML, η απλοποίηση δηλαδή της διαδικασίας που απαιτείται από τους συντάκτες των ενδιάμεσων με το χρήστη, ανεξάρτητα από τη συσκευή στην οποία στοχεύουν και να λάβει μέρος στην ανάπτυξη εφαρμογών μεγάλης κλίμακας και υψηλής φορητότητας. Είναι όμως αδύνατο να αντικαταστήσει τις ανθρώπινες αποφάσεις που απαιτούνται για τη δημιουργία user interfaces. Εν κατακλείδι, το έργο ekones και η ανάπτυξη της διεπαφής του με χρήση της UIML έκαναν τους συντάκτες της πτυχιακης εργασίας να δούν γενικότερα την ανάπτυξη διεπαφών από διαφορετική οπτική γωνία, περισσότερο εμπεριστατωμένη αλλά και ευέλικτη.

Βιβλιογραφία

Δημοσθένη Ακουμιανάκη (2006): Διεπαφή Χρήστη – Υπολογιστή, Μια συγχρονη Προσεγγιση» Εκδόσεις Κλειδάριθμος, Αθήνα.

Marc Abrams & James Helms (2004) *Wd-UIML UIMLspecification-3.1*

Harmonia (2001) *Java Renderer User Manual*

Constantinos Phanouriou (2000) UIML: A Device-Independent User Interface Markup Language