MODEL-BASED DEVELOPMENT OF UBIQUITOUS SYNCHRONOUS COLLABORATIVE UIS

by

GEORGE VELLIS

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF INFORMATICS ENGINEERING

SCHOOL OF APPLIED TECHNOLOGIES

TECHNOLOGICAL EDUCATION INSTITUTION OF CRETE

2015

Approved by:

Major Professor
Demosthenes Akoumianakis

Copyright

GEORGE VELLIS

2015

Extended Abstract

A primary goal of CSCW systems is to allow people to work together productively and naturally, like they do while acting within a single-user context. These systems need to provide user interfaces that support group work appropriately. Traditionally, user interfaces to groupware applications are hard-coded using either low-level constructs or toolkit-based approaches (i.e.: DiscoTech[22], GT/SD[23], Tele-board[25], PowerMeeting[26], MAUI[24], etc.). The results today offer a rich insight to the desired groupware functionality and the features devised to facilitate such functionality (i.e., replication models, object sharing, floor control, etc.). Nevertheless, the advent of model-based user interface development [27] signifies a move towards transformation-based approaches to generating the user interface. Although, these have been proven to work for single-user applications[28][29], very little is known about their capability to facilitate generation of multi-user interfaces to groupware applications[22],[30],[31]. Indeed, model-based development as currently applied to user interfaces exploits device-independent markup languages[28][29] – dialects of XML – and some sort of abstraction-based mechanisms to facilitate transformation of abstract interaction components to concrete widgets as implemented by a target platform[5]. Typically, the user interfaces built are single-user and non-collaborative.

Thus far, the efforts devoted to using model-based engineering to build collaborative interfaces are very few and limited in the scope of collaborative behavior supported. Indicative examples include CIAF[32], Workspaces[33], AMENITIES[12][34], CIAM[13], TOUCHE[1][14], and FlowiXML[35]. These efforts concentrate primarily on devising notations and tools to model cooperative behavior and workflows. In effect, their primary contribution is that they make explicit different elements of collaboration (i.e., roles, responsibilities and tasks) using dedicated notations such CTT[10], CIAN[9], GTA[36] or the Co-interaction Diagram in [37]. Only some of the existing efforts make steps towards generating the user interface of collaborative applications. An example of this work is FlowiXML[35]. However, this later line of research seems to be at odds with multi-user toolkits such as MAUI[24] or other frameworks for rendering applications collaborative (i.e.[38], CORK[39]). It follows, therefore that at present

[1] http://www.isys.ucl.ac.be/bchi/publications/Ph.D.Theses/Penichet-PhD2007.pdf

generating collaborative user interfaces remains ad hoc and highly programming-intensive task. Moreover, existing efforts do not constitute a complete and coherent collaborative UI development lifecycle, thus leaving space for significant improvements towards this direction.

Scope of proposed research and expected solutions

The above motivate the current research, which aims to extend the capabilities of the established model-based user interface development methods so as to support generation of collaborative user interfaces. Our intention is to investigate this issue from two key perspectives. The first is to identify the type of models needed to capture collaborative aspects in synchronous settings, while the second relates to generating the collaborative user interface by making use of suitable platform-oriented architectural components. In particular, the specific goals of this work are summarized as follows:

- Consolidate shortcomings and impediments to generating user interfaces for ubiquitous, synchronous and collaborative applications.
- Devise a new technique for integrating concepts relevant to collaborative UIs by introducing suitable extensions to model-based UI engineering methods.
- Elaborate the proposed technique in terms of appropriate instruments / means that allow concurrently active views (of the same model), interaction objects' replication across distributed clients, abstract interaction object synchronization schemes (beyond state sharing), which are necessary for the development of collaborative user interfaces.
- Establish the tools needed and the architectural model for implementing the above by extending an existing model-based development framework, such as UsiXML[6].

Περιεχόμενα

Copyright ii
Extended Abstract iii
Περιεχόμενα v
Πίνακας Εικόνων vii
Πίνακας Πινάκων ix
Ευχαριστίες x
Σημείωση xi
Κεφάλαιο 1 – Εισαγωγή 1
Επισκόπηση προβλήματος 2
Εργαλειοθηκοκεντρικός προγραμματισμός (toolkit-based approach) 3
Μοντελοκεντρική μηχανική διεπαφών (Model-based UI Engineering) 4
Κίνητρα και Ερευνητικά Ερωτήματα 6
Δομή και οργάνωση εργασίας 7
Κεφάλαιο 2 – Επισκόπηση όρων, εννοιών και τεχνολογιών 8
Διαμοιρασμός δεδομένων (Data-Sharing) 8
Διαμοιρασμός Οθόνης (Screen sharing) 9
Διαμοιρασμός Παραθύρου (Window sharing) 11
Διαμοιρασμός σε επίπεδο εργαλειοθήκης (toolkit-layer/s sharing) 13
Αρχιτεκτονικές Διαμοιρασμού Δεδομένων (Data-Sharing Architectures) 15
Διαχείριση Συνόδων (Session Management) 20
Έλεγχος Ταυτοχρονισμού (Concurrency control) 21
Έλεγχος Πρόσβασης (Access control) 21
Κεφάλαιο 3 – Σχετική Βιβλιογραφία 23
Εργαλειοθηκο-κεντρικές προσεγγίσεις (toolkit-based approaches) 23
Εργαλειοθήκες Ανάπτυξης Προσαρμοστικών Διεπαφών (Adaptive UIs) 23
Εργαλειοθήκες Ανάπτυξης Σύγχρονων Πολυχρηστικών Διεπαφών (multi-user UIs) 24
Μοντελο-κεντρική Μηχανική διεπαφών (Model-based UI Engineering) 28
Κεφάλαιο 4 –Προσέγγιση 30

‘Σχήματα πολυμορφικής στιγμιοτυποποίησης’: Περίγραμμα της Μεθόδου 30
Οραματισμός αλληλεπιδραστικού σεναρίου χρήσης και προδιαγραφών 32
Προσέγγιση από πλευράς Μηχανικής (Engineering Approach) 33
Σύντομη περιγραφή της UsiXML 34
Επεκτάσεις στη UsiXML 36
Το Περιβάλλον Εκτέλεσης (The Runtime Environment) 48
Client-side Components 49
Server-Side Components 49
Κεφάλαιο 5 – Επίδειξη εφαρμογής & Σενάριο Χρήσης 51
Συνοπτική περιγραφή των αλληλεπιδραστικών αναπαραστάσεων 51
Ροή εργασίας υποστήριξης σεναρίου χρήσης 53
Πολυμορφική κωδικοποίηση των εναλλακτικών αλληλεπιδραστικών σχημάτων στιγμιοτυποποίησης 56
Ορισμός του διαμοιραζόμενου μοντέλου (Shared model) 61
Προσδιορισμός Feedthrough 63
Σύνοψη αποτελεσμάτων και σχολιασμός 64
Κεφάλαιο 6 – Επίλογος 66
Απαντώντας τα ερευνητικά ερωτήματα 66
Επεκτάσεις και Μελλοντική εργασία 68
Βιβλιογραφία - Αναφορές 69
Παράρτημα 77

Πίνακας Εικόνων

Εικόνα 1: Παράδειγμα μετασχηματισμών στη UsiXML 5
Εικόνα 2: Συνήθης (common) διαστρωμάτωση της δομής ενός single-user UI 8
Εικόνα 3: Στιγμιότυπο του 'CommunityBar' screen sharing widget 10
Εικόνα 4: Ανεξάρτητη ελαχιστοποίηση και τοποθέτηση διαμοιραζόμενων παραθύρων (Window-sharing) 12
Εικόνα 5: Ανεξάρτητη τοποθέτηση διαμοιραζόμενων παραθύρων (Window-sharing) 12
Εικόνα 6: Παράδειγμα επιπέδου σύζευξης-'What You See Is Not What I See' (WYSINWIS) .. 15
Εικόνα 7: Παράδειγμα Κεντρικοποιημένης αρχιτεκτονικής 16
Εικόνα 8: Παράδειγμα αρχιτεκτονικής Αντιγράφων 16
Εικόνα 9: Ακολουθία γεγονότων σε μια κεντρικοποιημένη (centralized) αρχιτεκτονική 17
Εικόνα 10: Ακολουθία γεγονότων στην περίπτωση της αρχιτεκτονικής αντιγράφων (replicated) 19
Εικόνα 11: Πολυμορφική στιγμιοτυποποίηση (Polymorphic instantiation) 31
Εικόνα 12: Platform-specific στιγμιότυπα νοτών 33
Εικόνα 13: Model-Driven-Engineering Cameleon reference framework 35
Εικόνα 14: Πιθανές διατάξεις ανάπτυξης πανταχού παρόντων διεπαφών στη UsiXML 36
Εικόνα 15: Δομή γλώσσας περιγραφής widgets (WSL) 37
Εικόνα 16: Επισκόπηση της ροής περιγραφής (specification) ενός 'αφηρημένου' widget 38
Εικόνα 17: Το αναθεωρημένο μοντέλο CUI 39
Εικόνα 18: Δομή του μοντέλου 'widgetResource' 41
Εικόνα 19: Παράδειγμα αλληλεπιδραστικής απόδοσης τιμών στα πλαίσια του πολυμορφικού στιγμιοτύπου μιας κυκλικής μουσικής νότας 42
Εικόνα 20: Δομή του μοντέλου 'Behaviour' 44
Εικόνα 21: Δομή του μοντέλου 'Abstraction' 46
Εικόνα 22: Δομή του μοντέλου 'Consistency' 47
Εικόνα 23: Δομή του μοντέλου 'session' 48
Εικόνα 24: Το runtime περιβάλλον 49
Εικόνα 25: Διαφορετικές ενσαρκώσεις της πολυμορφικής διεπαφής 52

Εικόνα 26: Ενδεδειγμένη ροή εργασίας για την υποστήριξη του σεναρίου χρήσης 54
Εικόνα 27: Διαδικασία εισαγωγής αντικειμένου νότας ('Note Artifact') στο εργαλείο 58
Εικόνα 28: Τμήμα της CUI περιγραφής της διεπαφή του σεναρίου χρήσης της μουσικής 59
Εικόνα 29: Σχεδιαστική σουίτα για την περιγραφή πολυμορφικής διεπαφής του σεναρίου της μουσικής..... 60
Εικόνα 30: Στιγμιότυπα παραμετροποίησης αφηρημένων και πολυμορφικών χαρακτηριστικών μιας νότας (αριστερά: η περίπτωση της κυκλικής νότας, δεξιά της νότας μιας ταμπλατούρας) 61
Εικόνα 31: Υποστήριξη Feedthrough 62
Εικόνα 32: Εν εκτελέσει ανάπτυξη του σεναρίου 64
Εικόνα 33: Πρόβλημα επικάλυψης παραθύρων στην περίπτωση του διαμοιρασμού παραθύρου (Window-sharing) 77
Εικόνα 34: Παράδειγμα περιγραφής (specification) διεπαφής στη UsiXML σε όλα τα υποστηριζόμενα επίπεδα αφαίρεσης..... 77
Εικόνα 35: Στιγμιότυπο του προσδιορισμού όλων των υποστηριζόμενων καταστάσεων (States) και transitions του σχήματος στιγμιοτυποποίησης της νότας πενταγράμμου του java/swing 79

Πίνακας Πινάκων

Πίνακας 1: Πολυμορφική περιγραφή εναλλακτικών σχημάτων στιγμιοτυποποίησης μιας μουσικής νότας 56
Πίνακας 2: Πολυμορφική περιγραφή ιδιοτήτων του ‘MusicScore’ 57
Πίνακας 3: Πολυμορφική περιγραφή ιδιοτήτων του ‘MusicMeasure’ 57
Πίνακας 4: Συγκριτικός πίνακας κύριων μεθόδων ανάπτυξης διεπαφών ως προς τις βασικές απαιτήσεις υποστήριξης synchronous distributed ubiquitous groupware 78

Ευχαριστίες

Θέλω να ευχαριστήσω όλους όσους βοήθησαν για την ολοκλήρωση αυτής της εργασίας, ειδικά, το καθηγητή μου κύριο Ακουμιανάκη για την εξαιρετική, μακρόχρονη, καθοριστική σε όλα τα επίπεδα και άκρως εποικοδομητική συνεργασία μας, τον καθηγητή μου κύριο Βιδάκη, το Μητσάρα, καθώς και όλα τα μέλη και φίλους του ερευνητικού εργαστηρίου "Τεχνολογίας & Συστημάτων Λογισμικού" (iSTLab). Θέλω επίσης να ευχαριστήσω τη Μαρία για την παροιμιώδη υπομονή που έδειξε όλο αυτό το διάστημα και την αμέριστη κατανόηση. Τέλος, θέλω να ευχαριστήσω την οικογένειά μου για όλα.

Η ολοκλήρωση της παρούσας διπλωματικής εργασίας συγχρηματοδοτήθηκε μέσω του Έργου "Υποτροφίες ΙΚΥ" από πόρους του ΕΠ "Εκπαίδευση και Δια Βίου Μάθηση", του Ευρωπαϊκού Κοινωνικού Ταμείου (ΕΚΤ) του ΕΣΠΑ, 2007-2013.

Σημείωση

Τμήμα της συνεισφοράς της παρούσας εργασίας αποτελεί μέρος μιας ευρύτερης ερευνητικής ατζέντας στην οποία συνεισέφερε ο συγγραφέας ως μέλος της ερευνητικής ομάδας του εργαστηρίου iSTLab[2] του Τμήματος Μηχανικών Πληροφορικής της Σχολής Τεχνολογικών Εφαρμογών του ΤΕΙ Κρήτης.

[2] http://www.istl.teiher.gr/

Κεφάλαιο 1 – Εισαγωγή

Η ραγδαία ανάπτυξη που παρατηρείται ιδιαιτέρως τα τελευταία χρόνια σε τομείς σχετικούς με την τεχνολογία της πληροφορικής αλλά και συναφών επιστημονικών πεδίων έχει οδηγήσει, μεταξύ άλλων, στην ανάπτυξη και ευρεία διαθεσιμότητα μιας εκτεταμένης και ταχύτατα διευρυνόμενης γκάμας καινοφανών υπολογιστικών πλατφορμών, με ολοένα αυξανόμενες δυνατότητες, τις οποίες οι χρήστες εκμεταλλεύονται σε καθημερινή βάση για την εκτέλεση σειράς καθηκόντων στα πλαίσια των οικείων τους πρακτικών. Σε αυτό το αναδυόμενο υπολογιστικό περιβάλλον (πανταχού παρών - ubiquitous), ιδιαίτερο ενδιαφέρον παρουσιάζει εκείνη η κατηγορία λογισμικού που επιτρέπει τη σύγχρονη κατανεμημένη συνεργατική εργασία (ubiquitous synchronous distributed collaborative work). Στόχος της συγκεκριμένης κατηγορίας λογισμικού, ασχέτως πεδίου εφαρμογής (π.χ. συστήματα σύγχρονης απομακρυσμένης διαβούλευσης, πολυχρηστικά παίγνια, κοκ.), είναι η υποστήριξη της διαφανούς και απρόσκοπτης συνεργασίας μεταξύ χωρικά διεσπαρμένων χρηστών ανεξαρτήτως των υπολογιστικών μέσων και πόρων που είναι διαθέσιμα και χρησιμοποιούνται ανά περίπτωση. Το τελευταίο ωστόσο συνεπάγεται ικανότητα των εν λόγω συστημάτων και κυρίως της διεπαφής τους, να μπορούν να προσαρμόζονται αποτελεσματικά (ούτως ώστε να εξασφαλίζουν την ομαλή, απρόσκοπτη και προβλεπόμενη λειτουργικότητά τους) με βάση τα ακόλουθα δυναμικά χαρακτηριστικά:

1) Τις δυνητικά ριζικά ετερογενείς και ασύμβατες μεταξύ τους πλατφόρμες εκτέλεσης εφαρμογών (π.χ. web, android/mobile, pc, κοκ.).
2) Τη διαθεσιμότητα των εναλλακτικών διαδραστικών συμβάσεων ή λεξικών (interaction vocabularies) και υποστηρικτικών εργαλείων (π.χ. περιβαλλόντων εκτέλεσης, βιβλιοθηκών τρίτων) σε κάθε πλατφόρμα.
3) Διαφορές στο περιβάλλον εκτέλεσης καθηκόντων που προκύπτουν από ιδιαιτερότητες και εν δυνάμει περιορισμούς όπως φωτεινότητα, θόρυβος, κίνηση, κοκ.
4) Ικανότητες και δεξιότητες των χρηστών (τυφλός, expert, αρχάριος, κοκ.) που συνιστούν διακριτά στερεότυπα.

Διεπαφές ικανές να αναδιαμορφώνονται βάσει των παραπάνω χαρακτηριστικών κατηγοριοποιούνται υπό τον όρο ‘πανταχού παρούσες’ (ubiquitous) ή εναλλακτικά ‘ενήμερες-

πλαισίου' (context-aware)[1][2]. Στην ειδική περίπτωση των πανταχού παρόντων σύγχρονων πολυχρηστικών (synchronous multi-user) εφαρμογών, οι διεπαφές τους πρέπει να επιδεικνύουν και τα ακόλουθα γνωρίσματα – χαρακτηριστικά – ιδιότητες[3]:

1) Να παραμένουν συγχρονισμένες (synchronized) σε κάθε περίπτωση παρά τις όποιες ασυνέπειες(δυνητικά) παρουσιάζουν μεταξύ τους, είτε σε λεκτικό (lexical/physical), συντακτικό (syntactic) ή ακόμη και σε σημασιολογικό (semantic) επίπεδο. Τέτοιου είδους ασυνέπειες μπορεί να οφείλονται σε εν εκτελέσει προσαρμογές (adaptations) που ενεργοποιούνται για την ικανοποίηση απαιτήσεων που αφορούν οποιοδήποτε συνδυασμό των τριών προαναφερθέντων διαστάσεων, δηλ. πλατφόρμας, στερεοτύπου χρήστη και περιβάλλοντος εκτέλεσης.
2) Πρέπει επίσης να δύνανται να συλλέγουν, ανα-μεταδίδουν, διερμηνεύουν κατά το δοκούν και να αποτυπώνουν στη συνέχεια σχετική πληροφορία η οποία είναι απαραίτητη σε συνθήκες κατανεμημένης συνεργασίας για την υποστήριξη της ενημερότητας (awareness) και της αποτελεσματικής συνεργασίας μεταξύ των συνεργαζόμενων χρηστών.

Η υποστήριξη των παραπάνω απαιτήσεων ωστόσο αποδεικνύεται αρκετά περίπλοκη διαδικασία για την τρέχουσα τεχνολογική στάθμιση στον κλάδο της μηχανικής διεπαφών χρήστη-υπολογιστή, γεννώντας ερωτήματα αναφορικά με την επάρκεια και ικανότητα των επικρατούντων μεθοδολογιών, εργαλείων και γλωσσών γενικότερα (προγραμματισμού, περιγραφής διεπαφών, κοκ.) να τις υποστηρίξουν στο σύνολό τους αποτελεσματικά.

Επισκόπηση προβλήματος

Κύρια πρόκληση στην ανάπτυξη σύγχρονων συνεργατικών κατανεμημένων πανταχού παρόντων αλληλεπιδραστικών συστημάτων είναι τόσο η σχεδίαση όσο και η μετέπειτα ανάπτυξη και υλοποίηση της διεπαφής τους. Αυτό οφείλεται σε εγγενείς περιορισμούς των πλέον διαδεδομένων και ευρύτατα χρησιμοποιούμενων μεθοδολογιών ανάπτυξης. Οι μεθοδολογίες αυτές διακρίνονται σε δύο βασικές κατηγορίες – τον εργαλειοθηκοκεντρικό προγραμματισμό και την μοντελοκετρική μηχανική διεπαφών – που κάθε μία σηματοδοτεί διαφορετικές προοπτικές στον κύκλο ανάπτυξης διεπαφών χρήστη-υπολογιστή.

Εργαλειοθηκοκεντρικός προγραμματισμός (toolkit-based approach)

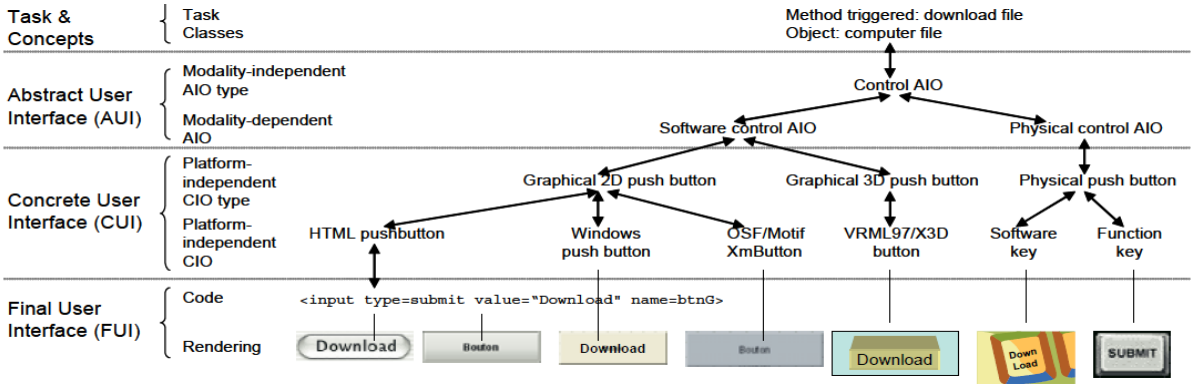
Ο εργαλειοθηκο-κεντρικός προγραμματισμός (toolkit-based programming) αποτελεί την κύρια μέθοδο ανάπτυξης λογισμικού διεπαφών εδώ και δεκαετίες. Είναι απόρροια της έρευνας σε συστήματα διαχείρισης διεπαφών χρήστη-υπολογιστή (user interface management systems) της δεκαετίας 1980-1990 που εδραίωσε το μοντέλο διαχείρισης γεγονότων (event-based model), τις επαναχρησιμοποιούμενες βιβλιοθήκες διαδραστικών αντικειμένων και την ενθυλάκωσή τους με γενικές γλώσσες προγραμματισμού όπως η C++ και η Java. Ωστόσο, λόγω δεσμεύσεων αναφορικά με τις κλάσεις αντικειμένων (object classes), το διάλογο (dialogue) και το μοντέλο εισόδου/εξόδου του εκάστοτε περιβάλλοντος, ο εργαλειοθηκο-κεντρικός προγραμματισμός αποτυγχάνει να υποστηρίξει ευέλικτα σχήματα στιγμιοτυποποίησης (flexible instantiation schemes) για την ικανοποίηση διαφορετικών αναγκών στα πλαίσια υποστήριξης πολλαπλών πλατφορμών ή/και πλαισίων χρήσης (contexts of use[2]) γενικότερα. Παραδείγματος χάριν, μια διεπαφή η οποία έχει αναπτυχθεί και τρέχει σε Android δεν μπορεί εκ φύσεως να εκτελεστεί σε οποιοδήποτε desktop περιβάλλον και αντίστροφα. Ως εκ τούτου, στην περίπτωση μιας αλληλεπιδραστικής εφαρμογής στα πλαίσια της οποίας πολλαπλές πλατφόρμες χρειάζεται να υποστηριχθούν, πρέπει: a) να ακολουθηθούν ξεχωριστές ροές ανάπτυξης (development workflows) – κατά κανόνα τόσες όσες και οι πλατφόρμες που είναι να υποστηριχθούν, b) να χρησιμοποιηθούν πιθανότατα διαφορετικές γλώσσες προγραμματισμού, c) να υιοθετηθούν διαφορετικά αλληλεπιδραστικά λεξικά - ανάλογα με τη διαθεσιμότητά τους σε κάθε πλατφόρμα, d) πρακτικές ανάπτυξης (π.χ. διαφορετική είναι η πρακτική επαύξησης ενός δια δραστικού λεξικού στο web απ’ ότι στο java/swing, κοκ.) όπως επίσης και e) ολοκληρωμένα εργαλεία ανάπτυξης (IDEs). Αναφανδόν, αυτού του είδους η προσέγγιση, που αποτελεί την πάγια τακτική προσέγγισης σε τέτοιου είδους περιπτώσεις αναγκών, συνεπάγεται σειρά συνεπειών που τις καθιστούν αν μη τι άλλο προβληματικές. Πιο συγκεκριμένα, i) προϋποθέτουν γνώση πολλών γλωσσών προγραμματισμού, εργαλείων, εργαλειοθηκών και των υιοθετημένων σε κάθε μια σχεδιαστικών προτύπων (design-patterns - γνώσεις που είναι δύσκολο να κατέχονται στο σύνολό τους από ένα και μόνο άτομο, σημειωτέο για την υλοποίηση της ίδιας αλληλεπιδραστικής λειτουργικότητας), ii) οδηγούν κατά κανόνα σε adhoc λύσεις μειώνοντας δραματικά τη δυνατότητα επαναχρησιμοποίησης τόσο αντικειμένων λογισμικού όσο και σχεδιαστικών προσεγγίσεων (σχήματα διαμοιρασμού, αρχιτεκτονικές, αντικείμενα, κοκ.), αυξάνοντας ως εκ τούτου iii) όχι μόνο το κόστος σε χρόνο ανάπτυξης, αλλά και iv) το κόστος όσον αφορά τη

συντήρηση (maintenance cost) και μελλοντική αναβάθμιση του λογισμικού ώστε να ενσωματώνονται αναδυόμενες απαιτήσεις (re-engineering cost). Επιπλέον, v) δεδομένων των παράλληλων ροών ανάπτυξης, που αναγκαία υιοθετούνται, ένα επιπλέον ζήτημα αφορά την ύπαρξη και διατήρηση συνοχής, όσον αφορά την υλοποίηση των (κοινών) απαιτήσεων, μεταξύ των παραγόμενων τμημάτων λογισμικού που αφορούν διαφορετικές πλατφόρμες.
Όλα τα προαναφερόμενα να σημειωθεί πως αποτελούν κοινό γνώρισμα τόσο των συμβατικών μονο-χρηστικών (signle-user toolkits) όσο και των σύγχρονων συνεργατικών εργαλειοθηκών (groupware toolkits).

Μοντελοκεντρική μηχανική διεπαφών (Model-based UI Engineering)

Η μοντελοκεντρική μηχανική διεπαφών (model-based UI engineering)[4], μια εναλλακτική του εργαλειοθηκο-κεντρικού προγραμματισμού που βασίζεται στη χρήση μοντέλων, προσφέρει εκ πρώτης όψης, ένα αδιαμφησβήτητα καλύτερο πλαίσιο αναφοράς για την προσέγγιση του προβλήματος. Πλεονεκτήματα απορρέουν από την προσήλωση στη χρήση αφηρημένης σημειογραφίας (abstract notations) και γλωσσών σήμανσης με στόχο την υποστήριξη του προσδιορισμού (specification) αφηρημένων αντικειμένων και της μετέπειτα διασύνδεσής τους (mapping) με ενδεδειγμένα λεξικά (platform-specific vocabularies) έτσι ώστε να εξασφαλίζεται συμβατότητα με την εκάστοτε πλατφόρμα στόχου. Τέτοιες διασυνδέσεις προϋποθέτουν τον ορισμό σχημάτων μετασχηματισμών (transformation-schemes)[8] που ως αποτέλεσμα έχουν την ανάθεση της παρουσίασης σε ενδεδειγμένους για κάθε πλατφόρμα ανάδοχους (renderers) [5].

Στην Εικόνα 1 συνοψίζεται μια ενδεικτική αποτύπωση των βημάτων που ακολουθούνται προκειμένου ένα μεμονωμένο αλληλεπιδραστικό καθήκον (‘κατέβασμα ενός αρχείου’), στα πλαίσια μιας τυχαίας μοντελο-κεντρικής αλληλεπιδραστικής διεπαφής (UsiXML[6] compatible), να μετασχηματιστεί από το πιο αφηρημένο επίπεδο ορισμού (‘Taks & Concepts’[6][7]), όπου προσδιορίζονται αποκλειστικά και μόνο χαρακτηριστικά ανεξαρτήτως πλατφόρμας και περιβάλλοντος εκτέλεσης, στη στιγμιοτυποποίηση (instantiation) του σε διαδραστικό συστατικό / στοιχείο (στην εν λόγω περίπτωση ένα button) της εκάστοτε πλατφόρμας στόχου (web, OSF/Motif, Windows, VRML, κοκ.).



Εικόνα 1: Παράδειγμα μετασχηματισμών στη UsiXML[3]

Ωστόσο αυτού του είδους οι προσεγγίσεις βασίζονται στο σύνολό τους σε γενικεύσεις και όχι πραγματικά αφαιρετικές δομές, με αποτέλεσμα η εκφραστική τους δύναμη να περιορίζεται κατά κανόνα στην υποστήριξη αποκλειστικά και μόνο συγκεκριμένου τύπου διεπαφών (π.χ. φόρμες). Ας σημειωθεί ότι η στόχευση για γενίκευση αντί πραγματικής αφαίρεσης, συχνά, αποτελεί συνειδητή επιλογή αφού διασφαλίζεται ο μέγιστος βαθμός φορητότητας (portability) καθώς και η συνεκτικότητα των μοντέλων κατά μήκος όλων των πιθανών πλατφορμών-στόχου. Υπό αυτή το λογική, επικράτησε η τάση να υποστηρίζεται ένα περιορισμένο εύρος διαδραστικών αντικειμένων και συγκεκριμένα μόνο εκείνα τα οποία: α) συγκαταλέγονται στα πιο δημοφιλή (δηλ. βρίσκονται εντός της τομής των διαδραστικών στοιχείων των πιο συχνά χρησιμοποιούμενων διαδραστικών λεξικών), β) υποστηρίζονται γηγενώς σε όλες τις πλατφόρμες - διευκολύνοντας έτσι άρδην τη διαδικασία διαχείρισής τους γενικότερα, και είναι γ) απλά στη δομή και λειτουργικότητά τους (π.χ. δέντρα – 'tree components' αποκλείονται), ώστε να είναι εύκολο να μοντελοποιηθούν και να περιγράφουν. Για την απρόσκοπτη εκτέλεση των υποστηριζόμενων προδιαγραφών – μοντέλων, η λογική της στιγμιοτυποποίησης και της γενικότερης διαχείρισης σύνθετων (και μη γηγενώς υποστηριζόμενων) διαδραστικών στοιχείων είναι hardcoded, μιας και στο σύνολό τους είναι a priori γνωστά και αμετάβλητα, στον κώδικα του εκάστοτε platform-renderer.

Συνέπεια των ανωτέρω είναι ότι καμία μοντελοκεντρική γλώσσα δεν υποστηρίζει τη δυνατότητα επεκτασιμότητας (ως ποιοτικού χαρακτηριστικού της γλώσσας) για την ενσωμάτωση νέων

[3] Από: 'USIXML: a User Interface Description Language for Specifying Multimodal User Interfaces' by: J. Vanderdonckt, Q. Limbourg, B. Michotte, L. Bouillon, D. Trevisan, M. Florins W3C Workshop on Multimodal Interaction. Sophia Antipolis (2004), pp. 19-20

τύπων διαδραστικών στοιχείων και ως εκ τούτου δεν υπάρχει και η παραμικρή πρόνοια και αναφορά για οποιασδήποτε γλώσσα περιγραφής σύνθετων αντικειμένων (Widget Specification Language)[3]. Με αυτό τον τρόπο, οι ευρέως καταγεγραμμένες τεχνικές αδυνατούν να προβλέψουν για τις περιπτώσεις συστημάτων που απαιτούν προηγμένες και μη-εγγενώς υποστηριζόμενες δομές διάδρασης, που είναι απαραίτητες σε σύγχρονα πολυχρηστικά συστήματα. Ως αποτέλεσμα, η συνεισφορά τους στο πεδίο των σύγχρονων συνεργατικών εφαρμογών περιορίζεται αποκλειστικά και μόνο στο επίπεδο αφηρημένης σχεδίασης και περιγραφής διαφόρων στοιχείων που αφορούν τη συνεργασία (ρόλων, ροών εργασίας, κοκ.) ([9],[10],[11],[12],[13]) και όχι στη δημιουργία και διαχείριση πλήρως λειτουργικών διεπαφών. Μοναδική εξαίρεση αποτελεί το TOUCHE[14], το οποίο καταλήγει στη δημιουργία της εκτελέσιμης σύγχρονης συνεργατικής διεπαφής κάνοντας απευθείας adhoc κλήσεις σε μια προκαθορισμένη εργαλειοθήκη (groupware toolkit). Αυτό ωστόσο σε καμία περίπτωση δεν αποτελεί γενική λύση, ενώ στην περίπτωση παρούσης της ανάγκης διασύνδεσης και επικοινωνίας μεταξύ μη συμβατών groupware toolkits, τα προβλήματα θα ήταν ίδια με αυτά των εργαλειοθηκο-κεντρικών προσεγγίσεων.

Κίνητρα και Ερευνητικά Ερωτήματα

Η αναπόφευκτη ποικιλομορφία και ετερογένεια μεταξύ των εναλλακτικών υπολογιστικών περιβαλλόντων χρήσης εγείρει πλήθος τεχνολογικών προκλήσεων και ερωτημάτων αναφορικά με την ικανότητα και επάρκεια των σύγχρονων μεθόδων ανάπτυξης λογισμικού διεπαφών να τη διαχειριστούν αποτελεσματικά. Κίνητρο της παρούσας εργασίας αποτελεί το να συνεισφέρει στη βελτίωση της τρέχουσας τεχνολογικής στάθμισης, επιδιώκοντας να γεφυρώσει τα πλεονεκτήματα που απορρέουν από την χρήση του εργαλειοθηκο-κετρινού προγραμματισμού με αυτά της μοντελο-κεντρικής μηχανικής διεπαφών ως προς τη δημιουργία πανταχού παρόντων σύγχρονων πολυχρηστικών κατανεμημένων διεπαφών. Με τον τρόπο αυτό επιδιώκεται η σύζευξη των ωφελειών που απορρέουν από τον εκφραστικό πλουραλισμό και δυνατότητα πλήρους ελέγχου των διαδραστικών στοιχείων και των υποκείμενων λεξικών που προσφέρει ο εργαλειοθηκο-κετρινός προγραμματισμός και η ευελιξία, η επεκτασιμότητα και γενικότερα αφαιρετική θεώρηση που προάγει η μοντελο-κεντρική μηχανική διεπαφών για την εγγενή υποστήριξη ριζικά ετερογενών υπολογιστικών περιβαλλόντων χρήσης. Ένα τέτοιο κίνητρο δρομολογεί σειρά επιμέρους ερευνητικών ερωτημάτων που συνοψίζονται ως ακολούθως:

- Είναι δυνατή η υποστήριξη της περιγραφής πανταχού παρόντων διεπαφών μέσω μιας ενοποιημένης περιγραφής, στα πλαίσια μιας ενοποιημένης (unified) μεθοδολογίας, με κοινά εργαλεία?
- Τι δομές περιγραφής χρειάζονται για την ενοποιημένη περιγραφή πανταχού παρόντων διεπαφών?
- Είναι δυνατό να υποστηριχθεί σύγχρονος συγχρονισμός μεταξύ ριζικά ετερογενών διεπαφών με τρόπο γενικευμένο τόσο ώστε να μπορεί να καλύψει το σύνολο των πιθανών περιπτώσεων?
- Τι είδους περιβάλλοντα εκτέλεσης, υπολογιστικά εργαλεία και αρχιτεκτονικές χρειάζονται ως προς την εκπλήρωση των ζητούμενων στόχων?

Δομή και οργάνωση εργασίας

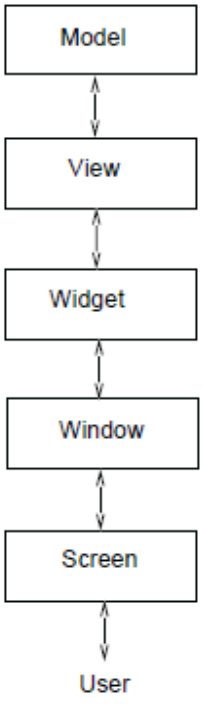
Το υπόλοιπο της εργασίας δομείται ως εξής: Στο **Κεφάλαιο 2**, αναλύονται βασικές έννοιες της συνεργασίας υποστηριζόμενης από υπολογιστή (όπως, σχήματα διαμοιρασμού δεδομένων, αρχιτεκτονικές συνεργατικών συστημάτων, έλεγχος ταυτοχρονισμού, κοκ), ούτως ώστε να εξοικειώσουν τον αναγνώστη με θεμελιώδης έννοιες του εν λόγω πεδίου. Στο **Κεφάλαιο 3**, γίνεται αναφορά στη σχετική βιβλιογραφία, εστιάζοντας τόσο την ανάπτυξη προσαρμοστικών διεπαφών, όσο και σύγχρονων συνεργατικών, υπό την οπτική των κυριάρχων προσεγγίσεων ανάπτυξης λογισμικού (εργαλειοθηκο-κεντρικός προγραμματισμός και μοντελο-κεντρική μηχανική). Εν συνεχεία, στο **Κεφάλαιο 4**, τεκμηριώνεται η προσέγγιση από πλευράς θεωρητικής θεμελίωσης αλλά και από πλευράς μηχανικής, ενώ στο **Κεφάλαιο 5**, ακολουθεί λεπτομερής παρουσίαση των βημάτων της προσέγγισης στα πλαίσια υποστήριξης ενός καινοφανούς σεναρίου χρήσης, στο πεδίο της κατανεμημένης μάθησης μουσικής. Το **Κεφάλαιο 6**, αποτελεί κατακλείδα της εν λόγω εργασίας, εξετάζοντας αρχικά το αν και κατά πόσο απαντήθηκαν τα ερευνητικά ερωτήματα που τέθηκαν εξ' αρχής, ολοκληρώνοντας με προτάσεις όσον αφορά μελλοντικές επεκτάσεις.

Κεφάλαιο 2 – Επισκόπηση όρων, εννοιών και τεχνολογιών

Σε αυτό το κεφάλαιο αναλύονται έννοιες και όροι της συνεργασίας υποστηριζόμενης από υπολογιστή ώστε να διευκολύνουν τον αναγνώστη στην περαιτέρω κατανόηση βασικών εννοιών του πλαισίου της εργασίας συνεπώς και των επόμενων κεφαλαίων.

Διαμοιρασμός δεδομένων (Data-Sharing)

Θεμελιώδες ρόλο στη σχεδίαση και υλοποίηση σύγχρονων κατανεμημένων πολυχρηστικών συστημάτων παίζει το είδος διαμοιρασμού ή αλλιώς ο βαθμός σύζευξης (coupling) μεταξύ των διαμοιραζόμενων αλληλεπιδραστικών συστημάτων. Ο Dewan ορίζει στο [62] τα πιθανά επίπεδα διαμοιρασμού, τα οποία αντιστοιχίζονται στην ουσία στα τυπικά επίπεδα τα οποία εκμεταλλεύεται ένα οποιοδήποτε αλληλεπιδραστικό σύστημα για την εμφάνιση και λειτουργία του.



Εικόνα 2: Συνήθης (common) διαστρωμάτωση της δομής ενός single-user UI.

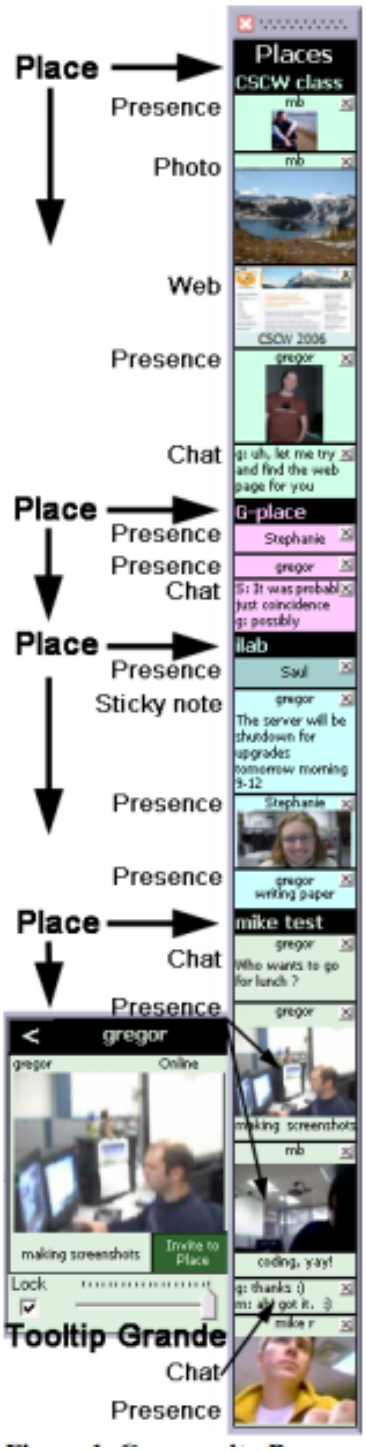
Όπως φαίνεται και στην Εικόνα 2, αυτά είναι: α) της οθόνης (Screen sharing) β) του παραθύρου (Window sharing) γ) ή κάποιου εκ των επιπέδων της αρχιτεκτονικής των χρησιμοποιούμενων widgets που υιοθετεί μια εργαλειοθήκη (MVC, PAC, κοκ.). Ανάλογα με το επίπεδο

διαμοιρασμού που επιλέγεται, επιλέγεται εμμέσως πλην σαφώς και ο τύπος εφαρμογών που μπορεί να υποστηριχθούν (screen sharing, single-user application sharing, workspace sharing, κοκ.), με κάθε τύπο διαμοιρασμού να συνεπάγεται εφαρμογές που κατά κανόνα αναπτύσσονται και χρησιμοποιούνται ανεξαρτήτως των άλλων, γι' αυτό και οι εναλλακτικοί αυτοί τύποι δεν επιζητείται ποτέ τυπικά να μπορούν να ενσωματωθούν (integrated). Κάθε επίπεδο διαμοιρασμού συνεπάγεται μοναδικά πλεονεκτήματα και μειονεκτήματα [62], τα οποία περιγράφονται στη συνέχεια.

Διαμοιρασμός Οθόνης (Screen sharing)

Ο διαμοιρασμός οθόνης εκμεταλλεύεται το διαμοιρασμό του framebuffer, διαχειριζόμενος την οθόνη σαν ένα απλό δισδιάστατο πίνακα από pixels. Πλεονεκτήματα που απορρέουν από αυτή την προσέγγιση είναι: 1) ο 'out-of-the-box' απρόσκοπτος διαμοιρασμός οποιασδήποτε μονοχρηστικής (single-user) εφαρμογής, 2) χωρίς δηλαδή ανάγκη για γνώση του, πρόσβαση ή τροποποιήσεις στον κώδικά τους, και 3) χωρίς να απαιτείται το installation και συνεπώς η ύπαρξη του προς διαμοιρασμό λογισμικού ή συμβατής έκδοσης αυτών σε όλους τους κατανεμημένους κόμβους. Παραδείγματα τέτοιων λογισμικών είναι το NLS[64] που ήταν και το πρώτο στο είδος, το VNC[66], το DisplayCast[65], κοκ. Ο διαμοιρασμός οθόνης αποτελεί αντικείμενο κλειδί (key-component) σε συστήματα τηλεδιάσκεψης όπου ροές ήχου και βίντεο συνδυάζονται με το διαμοιρασμό οθόνης-εφαρμογών επιτρέποντας σε χρήστες να συνεργαστούν εξ' αποστάσεως. Στα μειονεκτήματα αυτής της κατηγορίας διαμοιρασμού συγκαταλέγονται τα εξής: α) έλλειψη ενημερότητας (ενσωματωμένης στο διαμοιραζόμενο χώρο ή στους διαλόγους των αντικειμένων) για τον καλύτερο συντονισμό και εποπτεία της κατάστασης των συμμετεχόντων στα πλαίσια από κοινού εκτελούμενων ή εξαρτώμενων καθηκόντων, και β) έλλειψη ευελιξίας ως προς το βαθμό σύζευξης (tight coupling), π.χ. όλοι οι χρήστες βλέπουν-διαμοιράζονται αποκλειστικά και μόνο το ίδιο viewport στα πλαίσια ενός shared editor (window και scroll wars), κοκ.. Πιο πρόσφατα παραδείγματα όπως το Notification Collage [68] και ο διάδοχός του το CommunityBar [67] (βλ. Εικόνα 8), συνδυάζοντας διαμοιρασμό οθόνης και broadcasting των ροών ήχου και βίντεο, εστιάζουν επιπλέον α) στην κατά το δυνατόν υποστήριξη ομαδικής ενημερότητας (group-awareness) μέσω ενδείξεων παρουσίας (presence indicator) κοκ. και β) επικοινωνίας μέσω ενός ενδεδειγμένου απλού chat-widget. Τέλος να

σημειωθεί ότι υπάρχουν προσεγγίσεις που για την προστασία της ιδιωτικότητας (privacy) προνοούν για το διαμοιρασμό μόνο (ορθογώνιας μορφής κατά κανόνα) μέρους της οθόνης.



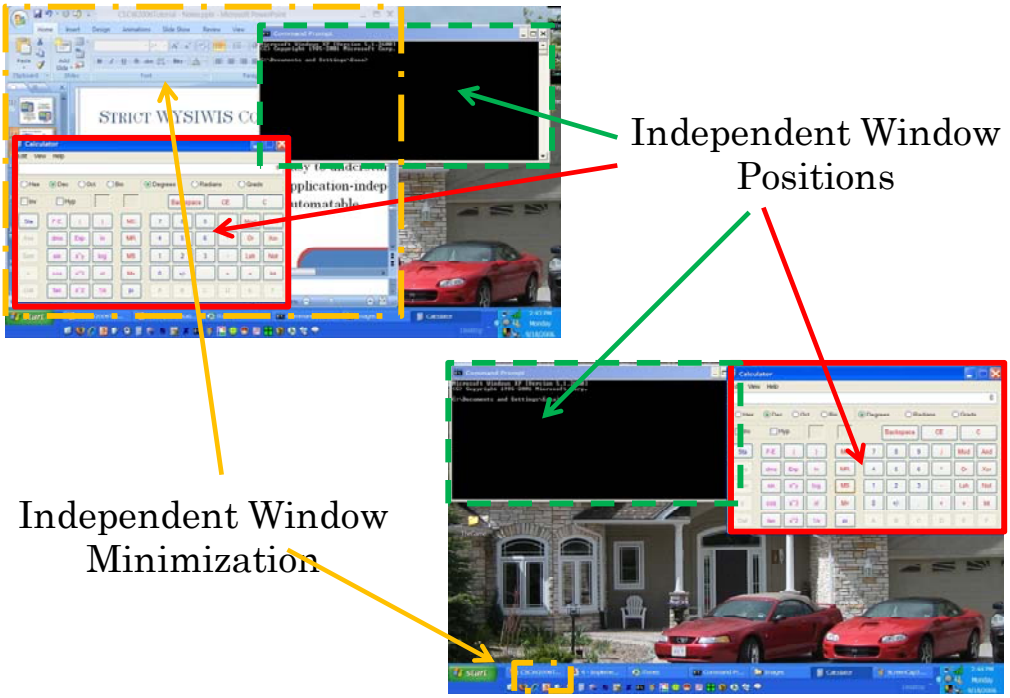
Εικόνα 3: Στιγμιότυπο του 'CommunityBar' screen sharing widget

Διαμοιρασμός Παραθύρου (Window sharing)

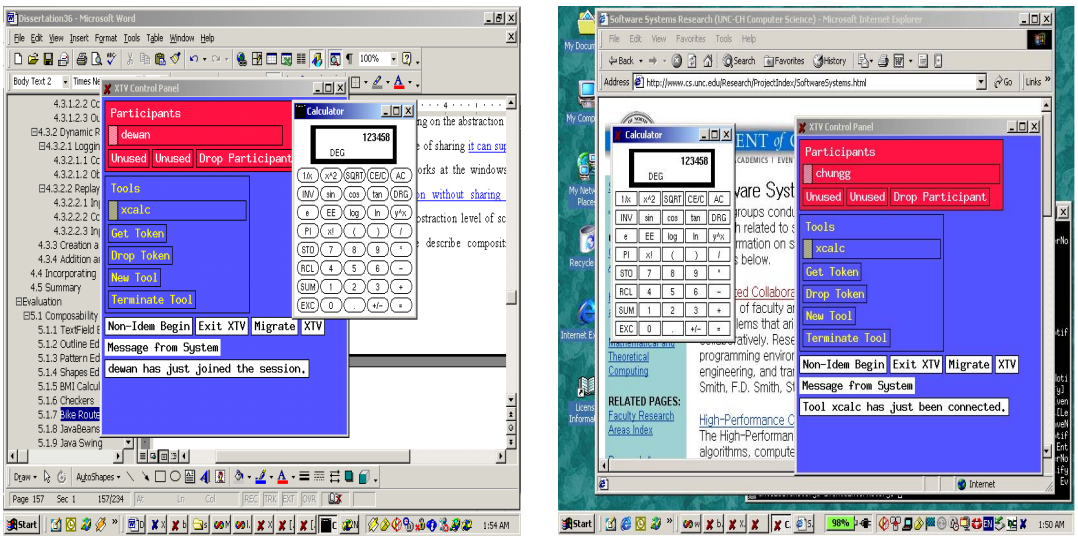
Ένα παραθυρικό σύστημα (Window system) διαιρεί την οθόνη σε μικρότερες υπο-περιοχές, καλούμενες παράθυρα (Windows), τις οποίες διαχειρίζεται ειδικό λογισμικό – ο window manager – του λειτουργικού συστήματος. Τα 'παράθυρα' αποτελούν αντικείμενα τύπου υποδοχέα που μπορούν να φιλοξενούν εφαρμογές και έτσι να επιτρέπεται η εκτέλεση πολλαπλών καθηκόντων. Ο διαμοιρασμός σε αυτό το επίπεδο στηρίζεται α) στην αναχαίτηση (interception) των γεγονότων τα οποία ο window manager τροφοδοτεί στο εκάστοτε ενεργό παράθυρο β) τη σύλληψη (capture) της κατάστασης των γραφικών που παρουσιάζονται στο παράθυρο και γ) στην προώθηση αυτών (relay) στους υπόλοιπους εμπλεκόμενους χρήστες (στα αντίστοιχα windows). Οι διαδικασίες αναχαίτησης (interception) και λήψης (reception) από τα παράθυρα των συνεργαζόμενων χρηστών, γίνεται διαμέσου ενός αντικειμένου που ονομάζεται proxy και είναι ενήμερο της συνεργασίας (collaboration-aware). Αυτού του είδους ο διαμοιρασμός προέκυψε από την ανάγκη για μεγαλύτερη ευελιξία όσον αφορά το επίπεδο σύζευξης (coupling) από αυτή του screen sharing και του αυστηρού WYSIWIS (strict What You See Is What I See)[58] σχήματος σύζευξης. Χαλαρότερο επίπεδο σύζευξης (near-What You See Is What I See [59][63]), υιοθετείται σε πλήθος περιπτώσεων όπως και στην περίπτωση του Cognoter[60] της Xerox. Στα πλαίσια της εν λόγω προσέγγισης, επιτρέπεται η ανεξάρτητη τοποθέτηση / μετατόπιση των διαμοιραζόμενων παραθύρων σε διαφορετικά σημεία της οθόνης ενώ επίσης επιτρέπεται και η απομονωμένη επεξεργασία των περιεχομένων ενός παραθύρου (shared window) και ο a posteriori συγχρονισμός τους μετά της βούλησης του χρήστη. Σε παρόμοιους κανόνες κινούνται και άλλες 'Window-Sharing' προσπάθειες (βλ. MMConf[71], XTV[69], VConf[70], κ.α.), όπως η [55], όπου υποστηρίζονται πιο εξεζητημένες δυνατότητες παραμετροποίησης του επιπέδου σύζευξης μεταξύ των διαμοιραζόμενων παραθύρων (με χρήση δικαιωμάτων - permissions).

Πλεονεκτήματα αυτού του είδους διαμοιρασμού είναι: α) η ανεξάρτητή τοποθέτηση (βλ. Εικόνα 4, και β) ελαχιστοποίηση (βλ. Εικόνα 5), διαμοιραζόμενων πόρων (παραθύρων) στην οθόνη κάθε χρήστη, γ) η δυνατότητα των χρηστών να τρέχουν πολλές παράλληλες εφαρμογές σε μη διαμοιραζόμενα παράθυρα (privacy) δ) προωθεί μια μη επεμβατική μέθοδο διαμοιρασμού – non-invasive approach που σημαίνει ότι δεν απαιτούνται επεμβάσεις σε επίπεδο κώδικα των έμμεσα διαμοιραζόμενων εφαρμογών. Από την άλλη πλευρά στα μειονεκτήματα συγκαταλέγεται ότι, δεν επιτρέπει ουδεμία πρόσβαση στα διαδραστικά αντικείμενα (στην αφαίρεση επιπέδου widget)

καθ' εαυτό με αποτέλεσμα να μην μπορούν να κλειδώσουν (lock) μεμονωμένα. Αυτό συνεπάγεται πως σε περίπτωση ταυτόχρονης αλλαγής μιας παραμέτρου της (collaboration-unaware) εφαρμογής από δύο χρήστες, ο κώδικάς της μπορεί κάλλιστα να κάνει break.



Εικόνα 4: Ανεξάρτητη ελαχιστοποίηση και τοποθέτηση διαμοιραζόμενων παραθύρων (Window-sharing)



Εικόνα 5: Ανεξάρτητη τοποθέτηση διαμοιραζόμενων παραθύρων (Window-sharing)

Η λύση σε αυτό το πρόβλημα είναι η πρόνοια ενός άμεσου κλειδώματος (explicit-lock) σε επίπεδο παραθύρου το οποίο μπορεί να έχει ένας χρήστης τη φορά εξασφαλίζοντας ότι μόνο ένας χρήστης τη φορά επεξεργάζεται τους διαμοιραζόμενους πόρους. Η εν λόγω λύση δεν είναι πολύ ευέλικτη ωστόσο μιας και αποτρέπει χρήστες που μπορεί να εστίαζαν σε 'αποκλειστικού ενδιαφέροντος' σημείο της εφαρμογής. Ένας άλλο περιορισμός είναι ότι δεν υπάρχει το επονομαζόμενο 'referential transparency', π.χ. ένας χρήστης δεν μπορεί να καταδείξει τη θέση ενός παραθύρου σε σχέση με ένα άλλο μιας και τα υπόλοιπα παράθυρα μπορεί να είναι μη-κοινόχρηστα. Τέλος, επιβάλλεται τα διαμοιραζόμενα παράθυρα να έχουν το ίδιο ακριβώς περιεχόμενο το οποίο δεν είναι αρκετά ευέλικτο για πλήθος λόγων σε πλήθος περιπτώσεων.

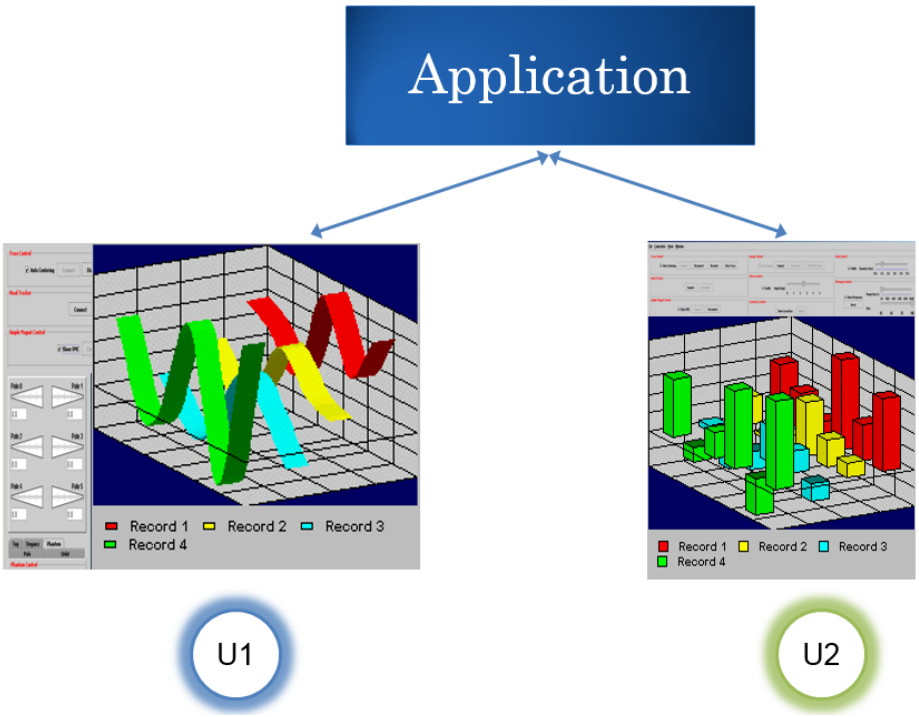
Διαμοιρασμός σε επίπεδο εργαλειοθήκης (toolkit-layer/s sharing)

Εκτός του επιπέδου διαμοιρασμού οθόνης (screen sharing) και παραθύρου (Window sharing) υπάρχει τέλος και η δυνατότητα διαμοιρασμού σε επίπεδο εργαλειοθήκης (toolkit-level sharing). Η εν λόγω προσέγγιση αποτελεί τον πιο ευέλικτο τρόπο υπό το πρίσμα της διαχείρισης του επιθυμητού επιπέδου σύζευξης (coupling-strength), επιτρέποντας πλουραλισμό επιλογών. Στην ουσία τους οι εργαλειοθήκες προωθούν την περαιτέρω διαίρεση των παραθύρων (όπως τα windows την οθόνη) σε μικρότερου επιπέδου συμπαγείς αφαιρετικές δομές (finer grained abstractions), τα λεγόμενα διαδραστικά στοιχεία (interaction elements), συμπεριλαμβανομένων text-boxes, sliders, μενού, κουμπιών, πινάκων, δέντρων, κοκ.. Κάθε αλληλεπιδραστική εργαλειοθήκη υιοθετεί κατά κανόνα ένα καθολικό, στα πλαίσιά της, αρχιτεκτονικό πρότυπο (architectural-pattern) με βάσει το οποίο προτυποποιεί τη δομή, τις συσχετίσεις και τον τρόπο επικοινωνίας των προβλεπόμενων ρόλων των κλάσεων που υλοποιούν κάθε διαδραστικό αντικείμενο. Τα συνήθη επίπεδα είναι αυτά της όψης (View), του μοντέλου (Model) και του ελεγκτή (controller) ο οποίος ανά περίπτωση μπορεί να ενσωματωθεί ως λογική και σε αυτό της όψης (βλ. Model-UI delegate design pattern[4]). Να σημειωθεί πως, αν και έχουν προταθεί εναλλακτικά αρχιτεκτονικά πρότυπα του MVC (βλ. PAC [72]), η γενικότερη ιδέα της διαστρωμάτωσης μιας εργαλειοθήκης και του διαμοιρασμού επιπέδων αυτής παραμένει ίδια, οπότε ως σημείο αναφορά θα έχουμε εφεξής το MVC. Με βάση την ορολογία του MVC, η όψη (View) ενός διαδραστικού αντικειμένου είναι το επίπεδο που υλοποιεί τη διαδραστική αναπαράσταση που βλέπει και επεξεργάζεται ο χρήστης η οποία αντανακλά σε κάθε περίπτωση

[4] Υιοθετείται από το δημοφιλές GUI toolkit της Java, το Swing.

την κατάσταση (state) και τα semantics του έτερου επιπέδου που ονομάζεται μοντέλο (Model). Συναθροιζόμενες όψεις συνιστούν-συναρμολογούν τη διεπαφή μιας εφαρμογής.

Τα λεγόμενα ‘συνεργατικά εργαλεία’ (collaboration tools)[42] επιτρέπουν το διαμοιρασμό ενός η περισσοτέρων επιπέδων της δομής μιας εργαλειοθήκης. Χωρίζονται σε δύο κατηγορίες, τα ‘layer-independent collaboration tools’ (ανεξάρτητο διαστρωμάτωσης) και τα ‘layer-dependent collaboration tools’ (εξαρτώμενο διαστρωμάτωσης). Τα πρώτα στην ουσία υλοποιούν λειτουργίες γενικού σκοπού, όπως απομακρυσμένη κλήση συνάρτησης (RPC) και διαμοιραζόμενα αντικείμενα (shared objects)[20][31], τα οποία μπορεί να χρησιμοποιηθούν από τους προγραμματιστές μιας εφαρμογής ώστε χειροκίνητα να αναχαιτίσουν και διαμοιράσουν (share) γεγονότα και κλήσεις (events/calls) που αφορούν ένα ή περισσότερα επίπεδα διαστρωμάτωσης μεταξύ διαφορετικών χρηστών. Από την άλλη πλευρά τα λεγόμενα ‘εξαρτώμενα-διαστρωμάτωσης εργαλεία’, έχουν κατασκευαστεί ώστε να κατανοούν τα γεγονότα και τις κλήσεις του επιπέδου διαστρωμάτωσης για το οποίο ενδείκνυνται (π.χ. μοντέλο, ελεγκτή, κοκ.), παρέχοντας αυτόματο διαμοιρασμό (sharing) των αυτομάτως, επίσης, ανιχνευόμενων γεγονότων και κλήσεων σε κάθε ένα από τα επίπεδα αυτά (πιθανόν χρησιμοποιώντας αφαιρέσεις ενός ‘layer-independent tool’). Όλες οι εργαλειοθήκες ανάπτυξης σύγχρονων groupware υιοθετούν αυτή τη φιλοσοφία διαμοιράζοντας ένα ή περισσότερα επίπεδα των υποστηριζόμενων διαστρωματώσεων (αναλόγως της αρχιτεκτονικής τους) κατά το δοκούν (βλ. GroupKit[20], Suite[55], COAST[56], Rendezvous[21], Dream Objects[61], MAUI[24], κοκ.), υλοποιώντας κατ’ αυτό τον τρόπο το μεγαλύτερο του εύρους των διαθέσιμων σύγχρονων συνεργατικών εφαρμογών. Στα μειονεκτήματα αυτών των προσεγγίσεων καταμετράται το γεγονός ότι απαιτούν τη συγγραφή κώδικα γεγονός που αυξάνει σε μεγαλύτερο ή μικρότερο βαθμό, ανάλογα με την περίπτωση, την πολυπλοκότητα της υλοποίησης του επιδιωκόμενου αποτελέσματος. Επίσης λόγω της μεγάλης ευελιξίας που υποστηρίζουν σχετικά με τις εναλλακτικές όψεις που μπορεί να διαμοιράζονται, μεταξύ των χρηστών, επί των κοινόχρηστων δεδομένων (βλ. π.χ. Εικόνα 6), απαιτείται ειδική πρόνοια για την υποστήριξη ενημερότητας ούτως ώστε, παρά τις ασυνέπειες σε επίπεδο διεπαφής, οι χρήστες να μπορούν να συντονιστούν και συνεργαστούν αποτελεσματικά. Κάτι τέτοιο δεν είναι εύκολο διότι ούτε οι μηχανισμοί ενημερότητας είναι a priori γνωστοί και προτυποποιημένοι αλλά και γιατί η υπολογιστική τους υλοποίηση απαιτεί χαμηλού επιπέδου εντατικό προγραμματισμό ούτως ώστε οι διάλογοι των αντικειμένων και ο κοινόχρηστος χώρος να επαυξηθεί αναλόγως ώστε να άγει τις σχετικές πληροφορίες.

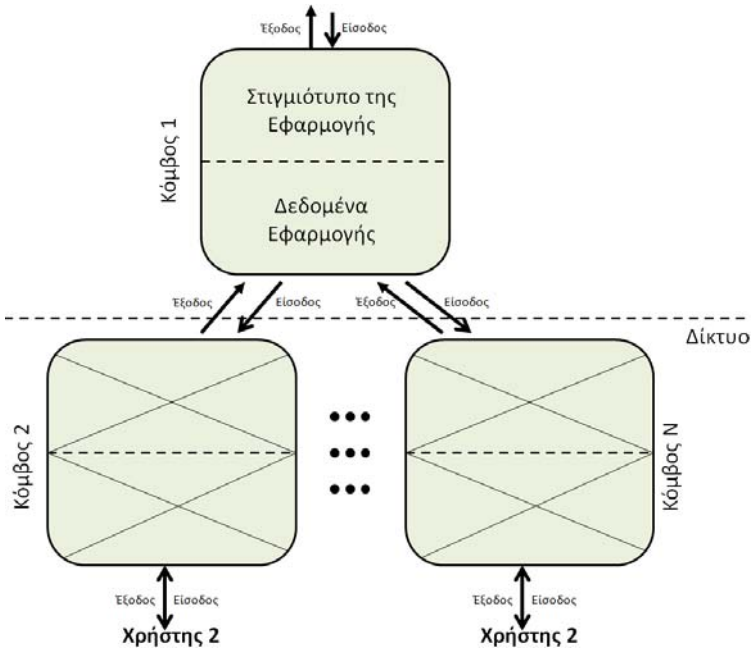


Εικόνα 6: Παράδειγμα επιπέδου σύζευξης-'What You See Is Not What I See' (WYSINWIS)

Αρχιτεκτονικές Διαμοιρασμού Δεδομένων (Data-Sharing Architectures)

Βασική απόφαση στη σχεδίαση ενός σύγχρονου συνεργατικού συστήματος αφορά την τοπολογία (topology) εναπόθεσης (reside) των διαμοιραζόμενων δεδομένων των διεργασιών (processes) που εκτελούνται στα πλαίσια της εκάστοτε groupware εφαρμογής[62][73]. Κατά κανόνα μπορεί να ανήκει σε μια εκ των τριών ακόλουθων κατηγοριών: α) αντιγράφων (replicated), β) κεντρικοποιημένη (centralized) και γ) υβριδική (hybrid), με κάθε μια να έχει τα δικά της πλεονεκτήματα και μειονεκτήματα με τα οποία έχει ασχοληθεί η διεθνής βιβλιογραφία εκτενώς [21][62][74][75][76][77][77][79].

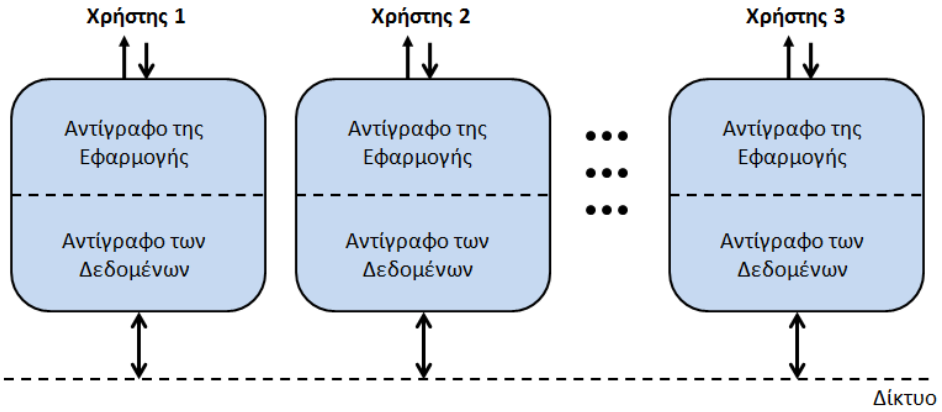
- Οι *Centralized* αρχιτεκτονικές, χρησιμοποιούν ένα και μόνο στιγμιότυπο (single instance) της εφαρμογής η οποία είναι εγκατεστημένη και εκτελείται αποκλειστικά και μόνο στα πλαίσια ενός κεντρικού υπολογιστικού κόμβου, ο οποίος αναλαμβάνει να χειρίζεται τις ροές εισόδου (input) και εξόδου (output) προς όλους τους κατανεμημένους συμμετέχοντες (participants). Οι υπολογιστικές διεργασίες (client processes) που εκτελούνται (reside) στην πλευρά κάθε χρήστη είναι αποκλειστικά και μόνο υπεύθυνες για την προώθηση αιτημάτων στο κεντρικά εκτελούμενο πρόγραμμα και στην εμφάνιση οποιασδήποτε εξόδου (output) λαμβάνεται από αυτό (βλ. Εικόνα 7).



Εικόνα 7: Παράδειγμα Κεντρικοποιημένης αρχιτεκτονικής

Το πλεονέκτημα αυτή της τοπολογίας είναι πως η ύπαρξη συγχρονισμού (synchronization) είναι εύκολη – η πληροφορία που αφορά την κατάσταση της εφαρμογής είναι συνεπής (consistent) μιας και είναι αποθηκευμένη αποκλειστικά και μόνο σε ένα (κεντρικό) σημείο ενώ τα γεγονότα τα οποία αποστέλλονται στις διεργασίες κάθε κόμβου-πελάτη (client-site) χειρίζονται πάντα με την ίδια ακριβώς σειρά με την οποία εκπέμφθηκαν μιας και σειριοποιούνται (serialized) από το server.

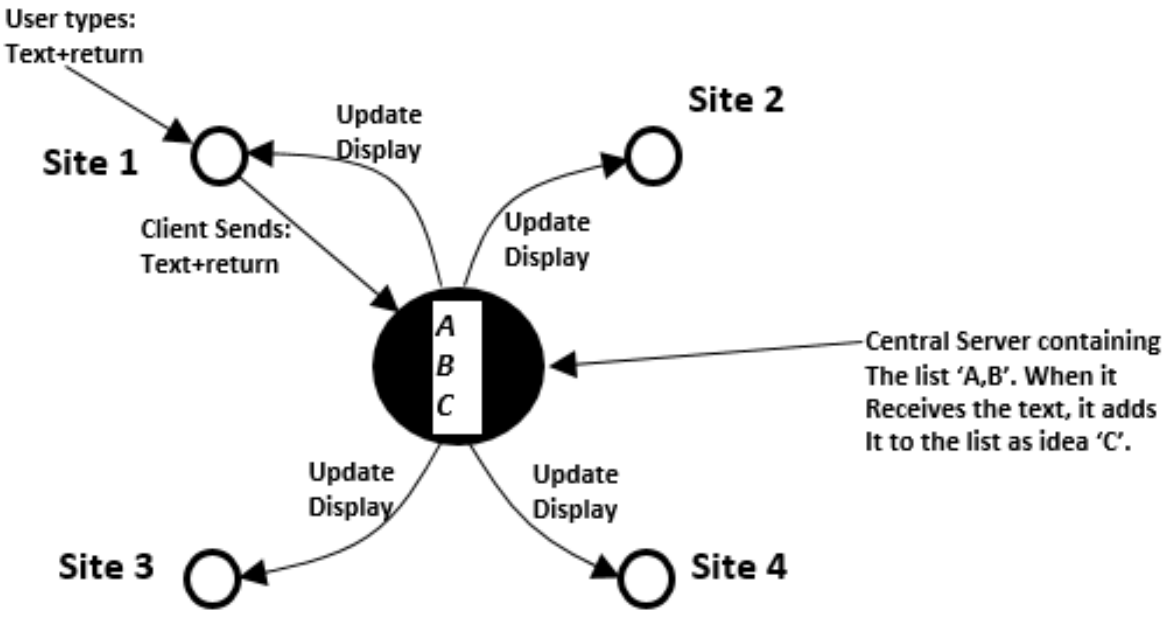
- Οι *Replicated* αρχιτεκτονικές, από την άλλη πλευρά, εκτελούν ένα αντίγραφο της εφαρμογής σε κάθε client.



Εικόνα 8: Παράδειγμα αρχιτεκτονικής Αντιγράφων

Γι' αυτό το λόγο κάθε αντίγραφο πρέπει να συντονίσει (coordinate) ρητώς τόσο τις τοπικές (local) όσο και τις απομακρυσμένες (remote) δράσεις (actions), και να προνοεί ώστε όλα τα αντίγραφα να είναι συγχρονισμένα ώστε να μην παρουσιάζονται ασυνέπειες (inconsistencies).

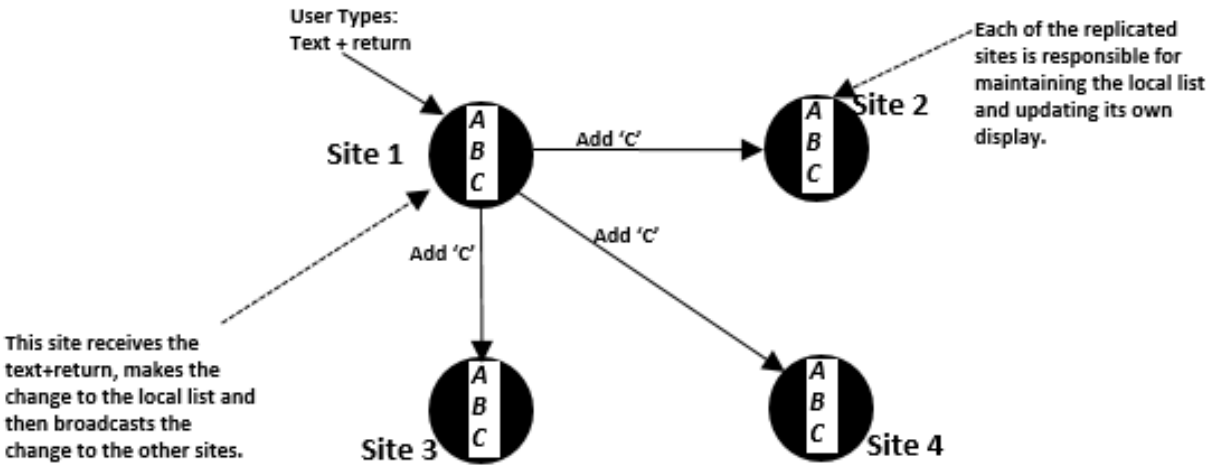
Λόγω της απλότητας στο χειρισμού του ταυτοχρονισμού (concurrency) και της διατήρησης ενός μοναδικού στιγμιοτύπου στο οποίο διατηρείται η κατάσταση (μοντέλο) της εφαρμογής, οι centralized αρχιτεκτονικές έχουν αρκετούς υποστηρικτές [21][74][76][78], δημιουργώντας εύλογα το ερώτημα του γιατί να υπάρχουν οι υπόλοιπες. Τα κύρια μειονεκτήματά της συμπεριλαμβάνουν α) το πιθανόν υψηλό latency (χρονικό διάστημα που μεσολαβεί μεταξύ action και feedback), β) πιθανές συμφορήσεις (bottlenecks), και η γ) αδυναμία υποστήριξης ετερογενών περιβαλλόντων. Πριν επεξηγηθούν να τονιστεί πάλι πως ένα centralized σχήμα συνεπάγεται (implies) σειριακή επεξεργασία, όπου η είσοδος ενός χρήστη προωθείται (transmitted) από το απομακρυσμένο μηχάνημα στην κεντρική εφαρμογή, η οποία με τη σειρά της πρέπει να το χειριστεί και να ενημερώσει τις οθόνες όλων των χρηστών (εφόσον κριθεί απαραίτητο) πριν επεξεργαστεί το επόμενο αίτημα.



Εικόνα 9: Ακολουθία γεγονότων σε μια κεντρικοποιημένη (centralized) αρχιτεκτονική

Αν το latency είναι χαμηλό δεν υφίσταται κάποιο πρόβλημα, στην αντίθετη περίπτωση ωστόσο, ολόκληρο το σύστημα θα καταστεί βραδυκίνητο. Ενώ η αργοπορία στην απόκρισή ενός συστήματος ως αποτέλεσμα της εφαρμογής γεγονότων που έχουν δημιουργηθεί διαμέσου της αλληλεπίδρασης ενός άλλου χρήστη με την εφαρμογή είναι απλά 'ενοχλητική', η καθυστέρηση της απόκρισης της εφαρμογής στην αλληλεπίδραση του ίδιου του χρήστη με το σύστημα, ειδικά σε υψηλού επιπέδου διαδραστικές εφαρμογές, όπου το σύστημα καθίσταται unresponsive είναι καταστροφική. Επίσης, δεδομένου ότι υψηλού επιπέδου διαδραστικές εφαρμογές (highly interactive applications) μπορεί να ωθήσουν ακόμα και τους ισχυρότερους επεξεργαστές στα όριά τους, δεδομένης της ανάγκης ενημέρωσης πλήθους οθονών ομοίως η αναμετάδοση (relay) όλων των δραστηριοτήτων σε – από μια μόνο διεργασία μπορεί να προκαλέσει συμφόρηση της κίνησης σε ορισμένα περιβάλλοντα. Τέλος, οι centralized εφαρμογές έχουν προβλήματα με τη διαχείριση ετερογενών περιβαλλόντων, καθώς είναι δύσκολο μια διεργασία να ενημερώνει απομακρυσμένους clients που ενεργούν διαμέσου διαφορετικών λειτουργικών συστημάτων με διαφορετικό look& feel (π.χ. Windows και Mac).

Από την άλλη πλευρά το σχήμα αντιγράφων (replication scheme), υποστηρίζει - συνεπάγεται παράλληλη επεξεργασία (parallel processing), με το χειρισμό των αλληλεπιδράσεων (interactions) και της ενημέρωσης της οθόνης (screen updates) να συμβαίνει παράλληλα σε όλα τα αντίγραφα. Όταν αυτό συμβαίνει με τον προβλεπόμενο τρόπο, η επικοινωνία καθίσταται αρκετά αποδοτική μιας και τα αντίγραφα πρέπει να ανταλλάσσουν μεταξύ τους μόνο εκείνη την πληροφορία (τμήμα) που χρειάζεται ώστε να διατηρούν τα μοντέλα τους ενήμερα (up-to-date). Η Εικόνα 10 δείχνει μια υψηλού επιπέδου όψη της ροής γεγονότων σε μια replicated αρχιτεκτονική. Παρά το γεγονός πως οι ειδοποιήσεις απομακρυσμένων δραστηριοτήτων μπορεί και σε αυτή την περίπτωση να εμφανίζουν καθυστερήσεις στην άφιξή τους, δεν επηρεάζουν στο παραμικρό την αποκρισιμότητα (responsiveness) του συστήματος σε επίπεδο τοπικών αλληλεπιδράσεων. Επίσης, είναι μάλλον δύσκολο να υπάρξει συμφόρηση των διεργασιών, μιας και κάθε αντίγραφο είναι υπεύθυνο για την ενημέρωση μόνο του τοπικού αντιγράφου, σε αντίθεση με τη centralized όπου πρέπει να ανανεώνεται η όψη (graphics) όλων των οθονών των χρηστών. Σε συνέχεια του προηγούμενο είναι εμφανές ότι η περίπτωση της υποστήριξης ετερογενών περιβαλλόντων μπορεί να διαχειριστούν σχετικά εύκολα, μιας και η επικοινωνία των πληροφοριών μπορεί αν γίνει με τρόπο που δεν εμπλέκει φυσικά χαρακτηριστικά των όψεων των χρηστών (Look&Feel, κοκ.) παρά μόνο τμήματα του κοινού μοντέλου.



Εικόνα 10: Ακολουθία γεγονότων στην περίπτωση της αρχιτεκτονικής αντιγράφων (replicated)

Το κόστος υιοθέτησης της συγκεκριμένου σχήματος διαμοιρασμού είναι η αυξημένη πολυπλοκότητα, μιας και ανακύπτουν θέματα λόγω της φύσης της αρχιτεκτονικής, που αφορούν τον έλεγχο ταυτοχρονισμού (concurrency control), της σειριακή επεξεργασία των γεγονότων (απομακρυσμένων και μη), που απαιτούν ειδική πρόνοια και εξιδεικευμένους αλγορίθμους καθιστώντας την αποτελεσματική διαχείρισή τους μακράν δυσκολότερη των centralized. Στη βιβλιογραφία το θέμα του concurrency control έχει προσεγγιστεί με διάφορους τρόπους, το Share-Kit[80] π.χ. δεν το υποστηρίζει απευθείας, αφήνοντάς το ως καθήκον των προγραμματιστών να το υλοποιήσουν σε περίπτωση που αυτό απαιτείται. Άλλες εργαλειοθήκες το υποστηρίζουν εγγενώς. Το DistEdit[81] π.χ. χρησιμοποιεί ατομικές εκπομπές (atomic broadcasts), ενώ τα διαμοιραζόμενα αντικείμενα (shared objects) που υποστηρίζει το ObjectWorld [82] έχουν την εγγενή δυνατότητα να ανιχνεύουν και να διαχειρίζονται μηνύματα που αφίχθηκαν εκτός σειράς (out of order) ενώ επιτρέπει επίσης στους προγραμματιστές το API ώστε να εφαρμόσουν κατά το δοκούν το μη-οπτιμιστικό κλείδωμα (non-optmistic locking). Το Groupkit[20] από την άλλη υλοποιεί ένα μηχανισμό (διαμέσου μιας μεμονωμένης διεργασίας) [54] που επιτρέπει τη σειριακή επεξεργασία των γεγονότων.

Μεταξύ της αρχιτεκτονικής αντιγράφων και της κεντρικοποιημένης βρίσκονται οι λεγόμενες υβριδικές (hybrid), οι οποίες περιέχουν τόσο centralized όσο και replicated αντικείμενα (components). Παραδείγματος χάριν, ο Patterson [83] υποστηρίζει ένα centralized notification server, του οποίου καθήκον είναι α) να διατηρεί το κοινό state (maintain common state), β) αποκρίνεται σε αιτήματα των clients για αλλαγή της κοινής κατάστασης, και γ) να ενημερώνει

τους υπόλοιπους clients όποτε συμβαίνει κάποια αλλαγή του κοινού μοντέλου. Τα υπόλοιπα εναπόκεινται σε κάθε αντίγραφο (replica), ώστε να αποφασιστεί πώς η όψη πρέπει να μοιάζει ώστε να εμφανίζει κάθε φορά το κοινόχρηστο μοντέλο με τον επιθυμητό τρόπο.

Διαχείριση Συνόδων (Session Management)

Βασική έννοια στη σύγχρονη συνεργασία υποστηριζόμενη από υπολογιστή είναι αυτή της συνόδου (session). Στόχος των συνόδων (sessions) είναι η ομαδοποίηση τόσο των στιγμιότυπων των διαμοιραζόμενων, μεταξύ των εμπλεκόμενων χρηστών, αντικειμένων (shared objects), όσο και η οργάνωση των χρηστών που τα προσπελαύνουν. Δεδομένου ότι η ύπαρξη συνεργασίας προϋποθέτει την ύπαρξη κοινόχρηστων αντικειμένων οργανωμένων και προσβάσιμων μεταξύ των μελών μιας συγκεκριμένης ομάδας χρηστών, ουσιαστικά μια σύνοδος λειτουργεί και μπορεί εναλλακτικά να οριστεί σαν ένα εικονικό αντικείμενο συμπερίληψης (κοινόχρηστος εικονικός υποδοχέας: shared virtual container) του οποίου στόχος είναι η ομαδοποίηση τόσο των στιγμιότυπων των διαμοιραζόμενων μεταξύ των εμπλεκόμενων χρηστών αντικειμένων (shared objects), αλλά και η οργάνωση των χρηστών στους οποίους εξουσιοδοτείται η πρόσβαση σε αυτά.

Στα πλαίσια ενός σύγχρονου συνεργατικού συστήματος μπορεί να υπάρχουν κατά κανόνα ταυτόχρονα καμία, μια ή και περισσότερες σύνοδοι, καθόλου, μερικώς ή πλήρως χρονικά επικαλυπτόμενες. Η υπολογιστική εξασφάλιση της ύπαρξης και γενικότερης λειτουργικότητας των συνόδων εξασφαλίζεται από την υλοποίηση και ύπαρξη ενός αντικειμένου το οποίο ονομάζεται διαχειριστής συνόδων(session manager).

Στις τυπικές δυνατότητες – αρμοδιότητές του διαχειριστή συνόδων περιλαμβάνονται:

1) Η διαχείριση του κύκλου ζωής της συνόδου (δηλ. Δημιουργία: εν τη ζητήσει ή προγραμματισμένη, Καταστροφή: εν τη ζητήσει ή με το πέρας της προκαθορισμένης διάρκειας),
2) Η διαχείριση της εισόδου (εξουσιοδοτημένη ή μη) και εξόδου χρηστών από αυτή,
3) Προώθηση - διάχυση της διαμοιραζόμενης πληροφορίας διαμέσου των πλαισίων της σωστής συνόδου και των χρηστών της κάθε φορά, και
4) Η δυνατότητα συσχετισμού σε επίπεδο συνόδου των διαμοιραζόμενων κάθε φορά αντικειμένων-δεδομένων.

Μια σύνοδος, ανάλογα με την αρχιτεκτονική που υιοθετείται στα πλαίσια ενός groupware συστήματος μπορεί να βρίσκεται (residing) και να εκτελείται είτε στο server–, είτε στο client–side.

Έλεγχος Ταυτοχρονισμού (Concurrency control)

Όταν πολλοί χρήστες ενεργούν στα πλαίσια διαμοιραζόμενων πόρων υπάρχει ο κίνδυνος της ταυτόχρονης προσπάθειας αλλαγής (τη τιμής) του ίδιου διαμοιραζόμενου πόρου έχοντας ως αποτέλεσμα τη δημιουργία μη επιθυμητών ασυνεπειών (inconsistencies) και απρόβλεπτων δυσλειτουργιών για την εφαρμογή. Για το λόγο αυτό τα groupware toolkits, δηλαδή οι εργαλειοθήκες ανάπτυξης σύγχρονων συνεργατικών εφαρμογών, παρέχουν τουλάχιστον ένα (βλ. [55],[56]) ή περισσότερους (βλ. [54]) αλγορίθμους για τη διαχείριση τέτοιων περιπτώσεων. Οι κατηγορίες αλγορίθμων αυτού του είδους υποπίπτουν σε δύο γενικές κατηγορίες, αυτές των λεγόμενων οπτιμιστικών (optimistic)[56] και των πεσιμιστικών (pessimistic)[55]. Η διαφορά τους έγκειται στο γεγονός πως οι μεν (οπτιμιστικοί) θεωρούν την περίπτωση των συγκρούσεων (conflicts) όχι τόσο πιθανή (όχι όμως αδύνατη), οπότε αποφεύγουν τη χρήση κλειδώματος (locks) κατά την αλλαγή των κοινόχρηστων δεδομένων κερδίζοντας κατ' αυτό τον τρόπο σε ταχύτητα επεξεργασίας, ενώ οι δε (πεσιμιστικοί) απαιτώντας το a priori κλείδωμα της κοινόχρηστης προς αλλαγή μεταβλητής αποφεύγουν κάθε πιθανή σύγκρουση- ασυνέπεια (inconsistency), έχοντας ωστόσο ως αποτέλεσμα την αύξηση του χρόνου επεξεργασίας τους. Αξίζει να σημειωθεί ότι τα σύγχρονα συνεργατικά συστήματα δανείζονται τη φιλοσοφία τους σε αυτό το επίπεδο από το πεδίο των συστημάτων διαχείρισης βάσεων δεδομένων, ενώ δεδομένου ότι κανένα concurrency-control σχήμα δεν ικανοποιεί το σύνολο των πιθανών απαιτήσεων[57] σε σχέση με το πεδίο εφαρμογής, υπάρχουν περιπτώσεις groupware εργαλειοθηκών που υποστηρίζουν τόσο τη χρήση γκάμας οπτιμιστικών όσο και πεσιμιστικών αλγορίθμων [54].

Έλεγχος Πρόσβασης (Access control)

Σε αντίθεση με τον έλεγχο ταυτοχρονισμένης πρόσβασης, ο οποίος καταπιάνεται με το πώς επιλύεται η περίπτωση που περισσότεροι του ενός χρήστες προσπαθούν να αλλάξουν τον ίδιο διαμοιραζόμενο πόρο, ο έλεγχος πρόσβασης εστιάζει στον αλγόριθμο – στη λογική με βάση την οποία παραχωρείται το δικαίωμα επεξεργασίας ενός κοινόχρηστου πόρου σε ένα ή περισσότερους χρήστες ταυτόχρονα. Συνήθως ο έλεγχος πρόσβασης αφορά σε επίπεδο πολυχρηστικών διαδραστικών στοιχείων στα πλαίσια ενός κοινόχρηστου συνεργατικού χώρου

(shared workspace) και όχι σε επίπεδο δεδομένων-μεταβλητών. Σε περιπτώσεις όπου ζητείται η ύπαρξη μηχανισμού ελέγχου πρόσβασης μπορεί να υπακούει σε διαφόρους αλγορίθμους, όπως:

- First In First Served: ο έλεγχος δίνεται με βάσει τη σειρά που ζητήθηκε (δημοκρατική λειτουργία)
- Role Based: Αυτός που έχει 'ισχυρότερο' ρόλο(π.χ. Administrator vs Simple User) εκτοπίζει βίαια αυτούς με το λιγότερο ισχυρό ρόλο
- Mixed: Π.χ. μεταξύ χρηστών που έχουν τον ίδιο ρόλο ισχύει η πρώτη μέθοδος, ενώ σε αντίθετη η δεύτερη

Αξίζει να σημειωθεί πως μπορεί να υπάρχει και δίδεται αρκετές φορές η δυνατότητα εν εξελίξει (runtime) αλλαγής του καθεστώτος (Floor policy) βάσει του οποίου διαμοιράζεται ο έλεγχος.

Κεφάλαιο 3 – Σχετική Βιβλιογραφία

Εργαλειοθηκο-κεντρικές προσεγγίσεις (toolkit-based approaches)

Εργαλειοθήκες Ανάπτυξης Προσαρμοστικών Διεπαφών (Adaptive UIs)

Η υπολογιστική υποστήριξη της εν εκτελέσει δυνατότητας διαχείρισης ποικιλίας τύπων εναλλακτικών αλληλεπιδραστικών αναπαραστάσεων(γραφικών και μη, εγγενώς και μη υποστηριζόμενων, κοκ.), για την ικανοποίηση απαιτήσεων σχετικά με το εκάστοτε πλαίσιο χρήσης (context of use) έχει προσεγγιστεί κατά το παρελθόν με διαφόρους τρόπους. Ο πολυμορφισμός (polymorphism) σε επίπεδο διεπαφής αποτελεί την ενδεικτικότερη, αναφερόμενος στη δυνατότητα μιας διεπαφής να διαχειρίζεται εν εκτελέσει κατά το δοκούν εναλλακτικές αλληλεπιδραστικές ενσαρκώσεις του ίδιου αφηρημένου διαδραστικού στοιχείου ώστε να καλύψει απαιτήσεις του πλαισίου χρήσης. Η μέθοδος (polymorphic instantiation) προτάθηκε και υλοποιήθηκε αρχικά στα πλαίσια του 'Platform Integration Module'[15] και εν συνεχεία από το HOMER UIMS[16]. Σε αμφότερες τις προσεγγίσεις ο πολυμορφισμός υλοποιήθηκε ως χαρακτηριστικό της οικείας γλώσσας προγραμματισμού ($4^{ης}$ γενιάς), ενσωματωμένος στον κώδικα των εν λόγω εργαλειοθηκών, με κύριο πεδίο εφαρμογής και στόχο τη διαχείριση προκλήσεων που αφορούν θέματα προσβασιμότητας (accessibility) (πρόσβαση σε γραφικές και μη-γραφικές διεπαφές). Στο ίδιο μήκος κύματος καταγράφονται και άλλες προσεγγίσεις που καταπιάστηκαν με το πρόβλημα χρησιμοποιώντας ωστόσο διαφορετική ορολογία. Τα 'Meta-widgets'[17] παραδείγματος χάριν, προτάθηκαν ως αφηρημένα αντικείμενα ('Meta-widgets'), διαστρωματωμένα στην κορυφή συμβατικών αντικειμένων εργαλειοθηκών (implementation-specific), με στόχο τη διευκόλυνση της κατασκευής πολυκαναλικών διεπαφών (multi-modal UIs). Η βασική ιδέα που διατρέχει τη φιλοσοφία τους εστιάζεται στην υιοθέτηση ενός αρχιτεκτονικού στυλ που στόχο έχει την ενθυλάκωση εναλλακτικών κλάσεων αντικειμένων σε αφηρημένα υπερ-αντικείμενα συμπερίληψης (meta-widgets), τα οποία χρησιμοποιούνται κατ' αποκλειστικότητα για τον απευθείας προγραμματισμό της διεπαφής. Υλοποιήσεις ωστόσο των meta-widgets υποθέτουν μη-επεκτάσιμα σχήματα στιγμιοτυποποίησης (non-extensible instantiation schemes), αγνοώντας παράλληλα πλήρως την όποια πρόνοια για υποστήριξη απαιτήσεων σχετικών με σύγχρονες πολυχρηστικές διεπαφές, όπως της ενημερότητας (awareness), τη κοινωνικής ημι-διαφάνειας (social-translucence), αλλά και άλλων αναδυόμενων

καινοφανών ποιοτικών χαρακτηριστικών όπως της πλαστικότητας (plasticity), κοκ.. Σε μεταγενέστερες περιπτώσεις όπως σε αυτές των [18] και [19], η εν εκτελέσει διαχείριση των υποστηριζόμενων εναλλακτικών διαδραστικών αναπαραστάσεων, προσεγγίστηκε μέσω του διαχωρισμού των μηχανισμών εισόδου και εξόδου από την 'κοινή' συμπεριφορά (common states) που υποστηρίζεται ανεξαρτήτως σχήματος στιγμιοτυποποίησης (instantiation-scheme). Και σε αυτή την περίπτωση ωστόσο, όπως και στην περίπτωση των Meta-widgets, απαιτήσεις και χαρακτηριστικά που αφορούν την υποστήριξη σύγχρονων συνεργατικών κατανεμημένων διατάξεων αγνοούνται στο σύνολό τους.

Να προστεθεί σε όλα τα παραπάνω πως μιας και η επεκτασιμότητα των υποστηριζόμενων λεξικών δεν ήταν το κύριο ζητούμενο στο σύνολο των περιπτώσεων, τυχόν ανάγκη για υποστήριξη καινοφανών σεναρίων (π.χ. [3],[40]) που επισύρουν τη χρήση εξειδικευμένων διαδραστικών αντικειμένων ή/και πολύπλοκων αλληλεπιδραστικών μεταφορών θα απαιτούσε ριζικές αλλαγές και επαναπρογραμματισμό του πυρήνα των εργαλειοθηκών με ό,τι αυτό αναπόφευκτα θα συνεπάγετο.

Εργαλειοθήκες Ανάπτυξης Σύγχρονων Πολυχρηστικών Διεπαφών (multi-user UIs)

Η διαχείριση απαιτήσεων εγειρόμενων στα πλαίσια της κατασκευής κατανεμημένων σύγχρονων πολυχρηστικών διεπαφών (synchronous distributed multi-user UIs), έχει απασχολήσει και αποτελεί επίκεντρο ενδιαφέροντος των groupware-εργαλειοθηκών (groupware-toolkits). Η συνεισφορά τους εστιάζεται στην παροχή έτοιμων προς χρήση υπολογιστικών αφαιρέσεων (μέσω ενδεδειγμένων προγραμματιστικών διεπαφών - APIs), των οποίων στόχος είναι η υλοποίηση γενικού σκοπού ('general purpose', άρα και reusable), αυξημένης πολυπλοκότητας και χαμηλού επιπέδου κατά κανόνα λειτουργικών συνιστωσών, επιδιώκοντας τη σημαντική εξομάλυνση της υποκείμενης πολυπλοκότητας του εκ του μηδενός προγραμματισμού κατανεμημένων πολυχρηστικών συστημάτων. Τέτοιου είδους αφαιρετικές δομές και (groupware) APIs αφορούν τη διαχείριση συνόδων (session management), διαχείριση αντιγράφων αντικειμένων (object replication), την κατανεμημένη επικοινωνία και συγχρονισμό (synchronization), διαχείριση ταυτοχρονισμού (concurrency control), διαχείριση ελέγχου πρόσβασης (access control), κοκ.. Στη διεθνή βιβλιογραφία υπάρχει πλήθος καταγεγραμμένων groupware toolkits (από το [20], [21], [39] και [41] έως τα πιο πρόσφατα [22],[23],[24] και[50]), έχοντας ως κύριο άξονα επιχειρηματολογίας το συγκριτικό βαθμό υποστήριξης της εξομάλυνσης

της πολυπλοκότητας (διαμέσου των προτεινόμενων - παρεχόμενων υπολογιστικών αφαιρέσεων) που εγγενώς συνεπάγεται η ανάπτυξη σύγχρονων πολυχρηστικών συστημάτων. Ορισμένες προσεγγίσεις όπως το MAUI [24], εκμεταλλευόμενες το διαμοιρασμό σε επίπεδο εργαλειοθήκης (toolkit-sharing), παρέχουν επιπρόσθετα σε επίπεδο διαλόγου μια σειρά από διαδραστικά αντικείμενα, εν προκειμένω επαυξημένα ώστε να άγουν ομαδική ενημερότητα (group-awareness), διευκολύνοντας άρδην την ανάπτυξη βασικών (plain: form-based) πολυχρηστικών διεπαφών. Ωστόσο ακόμα και στην περίπτωση εργαλειοθηκών, όπως το MAUI, που προνοούν ως προς τη διευκόλυνση της σύνταξης της πολυχρηστικής διεπαφής, η υποστήριξη σεναρίων που επισύρουν τη χρήση μη εγγενώς (native) υποστηριζόμενων αντικειμένων αποτελεί πρόβλημα. Αυτό διότι πέραν της αναπόφευκτης τυπικής δυσκολίας ανάπτυξης του νέου διαδραστικού στοιχείου (συνεπαγόμενου χαμηλού επιπέδου εκτεταμένο προγραμματισμό), ο προγραμματιστής πρέπει να είναι οικείος με το γενικότερο αρχιτεκτονικό μοντέλο, τις σχεδιαστικές αποφάσεις και τα χαμηλότερου επιπέδου APIs της εκάστοτε groupware βιβλιοθήκης, για τον προγραμματισμό όλων των groupware συνιστωσών στα πλαίσια του νέου στοιχείου (awareness, concurrency control, floor control, διαμοιρασμό, feedthrough, κοκ.). Τα groupware toolkits αποτελούν έως σήμερα την κύρια μέθοδο ανάπτυξης πολυχρηστικών συστημάτων, ωστόσο η υποστήριξη πανταχού παρόντων σύγχρονων συνεργατικών συστημάτων τις καθιστά άκυρες. Αυτό διότι:

1) Ως toolkits (αλληλεπιδραστικών αντικειμένων), λόγω υποθέσεων που καλούνται να κάνουν σχετικά με τις κλάσεις αντικειμένων και το διάλογο περιορίζονται στην υποστήριξη αποκλειστικά μιας ή εύρους ομογενών ωστόσο μεταξύ τους πλατφορμών.
2) Στο σύνολό τους δεν υποστηρίζουν προσαρμοστικούς διαλόγους.
3) Αδύνατο για την υποστήριξη σεναρίων που εμπλέκουν ριζικά ετερογενείς πλατφόρμες να επιτύχεις, της απαιτούμενή διαλειτουργικότητα μεταξύ διαφορετικών groupware toolkits, λόγω πιθανότατα: διαφορετικών υποθέσεων α) σε σχέση με την αρχιτεκτονική, β) τις τεχνολογίες κατανεμημένης επικοινωνίας, γ) τη διαχείριση συνόδων και χρηστών, δ) το διαμοιρασμό δεδομένων γενικότερα, ε) τους διαφορετικούς και ασύμβατους μηχανισμούς ενημερότητας που κάθε ένα υλοποιεί, κοκ. [42].

Μοναδική εξαίρεση προς την κατεύθυνση της υποστήριξης πολλαπλών και ετερογενών πλατφορμών αποτελεί το ‘Multi-domain toolkits’[42]. Η εν λόγω προσέγγιση υιοθετεί και προτείνει ένα ‘factory-based’ προγραμματιστικό μοντέλο στα πλαίσια του οποίου υλοποιεί και

παρέχει ένα API το οποίο αποκρύπτει λεπτομέρειες της υλοποίησης των διαδραστικών στοιχείων των κείμενων υποκάτω υποστηριζόμενων αλληλεπιδραστικών εργαλειοθηκών. Πιο συγκεκριμένα παρέχει ένα ενοποιημένο API το οποίο αποκρύπτει τη διαχείριση των κλήσεων στα πλαίσια του προκαθορισμένου εύρους interaction toolkits τα οποία υποστηρίζονται (GWT[5], Swing[6], AWT[7] και SWT[8]). Στην ουσία καλύπτει τα ακόλουθα όπως ονομάζει 'domains': αυτόνομης εφαρμογής (βλ. Swing και AWT), eclipse (βλ. SWT) και web (βλ. GWT). Η συγκεκριμένη προσέγγιση ωστόσο βρίσκεται ακόμα σε αρκετά πρώιμο στάδιο, υιοθετεί ένα μάλλον ιδιότυπο και δύσχρηστο προγραμματιστικό στυλ με το οποίο οι προγραμματιστές διεπαφών δεν είναι εξοικειωμένοι, ενώ οι παραγόμενες διεπαφές δεν επιτρέπουν εν εκτελέσει προσαρμογές. Επιπλέον η επεκτασιμότητα του υποστηριζόμενου διαδραστικού λογισμικού είναι κάτι που αγνοείται εντελώς και για να υποστηριχθεί προς το παρόν συνεπάγεται υπερβολικά αυξημένη πολυπλοκότητα, εκτεταμένη γνώση της δομής των υποστηριζόμενων εργαλειοθηκών και υπερβολικά αυξημένο χρονικά κόστος. Επίσης πλήρως αγνοούνται μηχανισμοί υποστήριξης ενημερότητας που είναι θεμέλιας σημασίας σε τέτοιου είδους συστήματα. Επιπρόσθετα, έχει επαληθευτεί η εγκυρότητα της προσέγγισης σε ένα υπερβολικά περιορισμένο εύρος πλαισίων (domains, java-technology based) το οποίο σε καμία περίπτωση δεν καθιστά την εν λόγω προσέγγιση επαρκή για την κάλυψη αναγκών των πανταχού παρόντων περιβαλλόντων χρήσης. Τέλος, υπάρχει μηδενική υποστήριξη όσον αφορά το design phase (IDE) ενώ δεν παρέχεται – προσδιορίζεται-προτείνεται καμία σχετική ενοποιημένη μεθοδολογία ανάπτυξης που να καθοδηγεί την όλη διαδικασία σχεδίασης και ανάπτυξης.

Προσανατολισμένες στο Web εργαλειοθήκες – (Web-oriented toolkits)

Η υποστήριξη σύγχρονης συνεργασίας στο web δεν αποτελεί νέα ιδέα, έχοντας δοκιμαστεί κατά το σχετικά πρόσφατο παρελθόν με επιτυχία για την υποστήριξη πλήθους περιπτώσεων (βλ. π.χ. CORK [39], Promondia [52], JETS[53], DISCIPLE[51]). Ωστόσο, λόγω ανεπάρκειας των πρότερων τεχνολογικών μέσων να υποστηρίξουν σειρά απαιτήσεων (π.χ. push-based κατανεμημένη επικοινωνία), όλες οι προσεγγίσεις καθίσταντο αναγκαστικά εξαρτώμενες από

[5] **G**oogle**W**eb**T**oolkit: http://www.gwtproject.org/

[6] Java Swing: http://openjdk.java.net/groups/swing/

[7] **A**bstract**W**indow**T**oolkit: http://docs.oracle.com/javase/7/docs/technotes/guides/awt/

[8] **S**tandard**W**idget**T**oolkit: https://www.eclipse.org/swt/

την χρήση εξιδεικευμένων plugins με τους όποιους περιορισμούς αυτά εξ ορισμού επέβαλαν [48]. Π.χ. στην περίπτωση του CORK, το οποίο στηρίζεται κατ' αποκλειστικότητα στην τεχνολογία των java/Applets ως σύλληψη, μεταξύ άλλων (π.χ. περιορισμένη πρόσβαση σε υπολογιστικούς πόρους), δεν θα μπορούσε να λειτουργήσει ούτε καν εκτελεστεί σε πληθώρα κινητών περιβαλλόντων χρήσης (mobile contexts of use), μιας και η εν λόγω τεχνολογία υποστηρίζεται στα πλαίσια ελάχιστων λειτουργικών συστημάτων κινητών συσκευών.

Με την πρόοδο ωστόσο του web, τη θέσπιση νέων τεχνολογικών προτύπων (HTML5, css3, κ.α.) και υπολογιστικών frameworks γενικότερα (web-sockets, Angular JS, jQuery, κοκ.), σειρά δυνατοτήτων ανέκυψαν, διανοίγοντας την προοπτική της ανάπτυξης σύγχρονων συνεργατικών αλληλεπιδραστικών συστημάτων χωρίς χρήση plugins άρα και των όποιων συνεπαγόμενων περιορισμών (π.χ. φορητότητας). Πρόσφατες προσπάθειες [30][48], επαλήθευσαν την εν λόγω δυνατότητα μέσω σειράς δοκιμών, ελέγχοντας κατά κύριο λόγο το εύρος των χρόνων απόκρισης το οποίο βρέθηκε εντός των ανεκτών ορίων από πλευράς διατήρησης της ευχρηστίας του συστήματος οπότε και πρακτικά ικανό να υποστηρίξει απρόσκοπτα πλήθος συνεργατικών περιπτώσεων.

Ωστόσο η ανάπτυξη πανταχού παρόντων σύγχρονων πολυχρηστικών διεπαφών (με βάση το στερεότυπο των χρηστών π.χ.), συνεπάγεται πρόσθετες απαιτήσεις ως προς τις οποίες οι υπάρχουσες τεχνολογικές υποδομές λογισμικού αδυνατούν να ανταποκριθούν.

Αυτό διότι πρώτον, υπάρχει παντελής ανυπαρξία προσαρμοστικών frameworks ή interaction toolkits ικανών να υποστηρίξουν την ανάπτυξη προσαρμοστικών, ως προς το στερεότυπο χρήστη ή/και το περιβάλλον εκτέλεσης (π.χ. σκοτεινό, ηχηρό, κοκ.), διεπαφών. Π.χ. υπάρχει ανυπαρξία ενοποιημένων εργαλειοθηκών και frameworks για την ανάπτυξη μη οπτικών ή και πολυκαναλικών web-based αλληλεπιδραστικών συστημάτων. Επίσης, υπάρχει παντελής έλλειψη προηγμένων ολοκληρωμένων εργαλείων (περιβαλλόντων ανάπτυξης) και ενοποιημένων μεθοδολογιών, ώστε να διευκολύνεται η φάση σχεδίασης και ανάπτυξης προνοώντας παράλληλα και διατάξεις τύπου 'affordance-based UI design' όπως των [3] και [40]. Επίσης, μιας και δεν υπάρχει προτυποποιημένος τρόπος επικοινωνίας του browser με εξωτερικές συσκευές (APIs) και αισθητήρια όργανα, για την επικοινωνία πληροφοριών με το εξωτερικό περιβάλλον – πλαίσιο εκτέλεσης, καθίσταται δύσκολη η ανάπτυξη διεπαφών που να μπορούν με γενικευμένο τρόπο να τις εκμεταλλεύονται ώστε να είναι σε θέση να πυροδοτήσουν κατά το δοκούν τις όποιες εν εκτελέσει προσαρμογές. Τέλος, μιας και το εύρος των διαθέσιμων διαδραστικών λεξικών

υπολείπεται κατά πολύ της εκφραστικότητας και ποικιλίας των συμβατικών εργαλειοθηκών (PC-oriented), για την υποστήριξη καινοφανών σεναρίων χρήσης απαιτείται προγραμματισμός τους από βάσης.

Μοντελο-κεντρική Μηχανική διεπαφών (Model-based UI Engineering)

Με την πρόοδο των μοντελο-κετρικών μεθόδων ανάπτυξης πολλοί από τους περιορισμούς που εγγενώς επιβάλλονται στα πλαίσια των εργαλειοθηκο-κεντρικών μπορεί να εξομαλυνθούν ή και αρθούν. Πολλά νέα frameworks έχουν προταθεί ισχυριζόμενα πλεονεκτημάτων επί της υιοθέτησής τους έναντι του toolkit-based παραδείγματος ανάπτυξης (development paradigm). Στο [6], οι συγγραφείς περιγράφουν τον ορισμό (specification) μιας διεπαφής σε πολλαπλά επίπεδα αφαίρεσης διαμέσου της χρήσης συγκεκριμένων τύπων μοντέλων. Προσαρμογές (adaptations) υποστηρίζονται διαμέσου ενδεδειγμένων μετασχηματισμών (transformations) εφαρμοζόμενων σε πηγαία-μοντέλα (source-models) ούτως ώστε να παραχθεί το επιθυμητό μοντέλο-στόχος (target-model). Οι όποιοι μετασχηματισμοί μπορεί να εφαρμοστούν σε μοντέλα του ιδίου επιπέδου (συγκροτώντας την λεγόμενη 'διαδικασία μετάφρασης' – 'translation-process'), με φορά από ένα αφηρημένο προς ένα λιγότερο αφηρημένο (concrete) επίπεδο (δηλ. διαδικασία συγκεκριμενοποίησης – reification process), ή το αντίθετο, δηλ. από ένα πιο εξειδικευμένο (concrete) επίπεδο σε ένα πιο αφηρημένο ορισμό (συνιστώντας αφαίρεση - abstraction). Σε αυτά τα πλαίσια, η ικανότητα μιας διεπαφής να προσαρμόζεται καταλλήλως στο εκάστοτε πλαίσιο χρήσης (context of use) καθορίζεται από την ικανότητά της να μηχανευτεί κάθε φορά το κατάλληλο σύνολο κανόνων μετασχηματισμού (transformation rules)[8]. Το COMMETS Framework [44] αντιπροσωπεύει ένα περαιτέρω βήμα προς την ίδια κατεύθυνση. Το εν λόγω Framework προσεγγίζει το καίριο θέμα των μετασχηματισμών[45] ('The mapping problem'), με χρήση σημασιολογικών δικτύων (semantic-networks), επιτρέποντας κατ' αυτό τον τρόπο τη δυνατότητα εξειδικευμένων εν εκτελέσει προσαρμογών (runtime adaptations) σε οποιοδήποτε από τα υποστηριζόμενα επίπεδα αφαίρεσης (tasks, AUI και CUI) , λαμβάνοντας υπόψιν ενδεδειγμένες μετρικές για την παράλληλη εξασφάλιση της ευχρηστίας και της συνέχειας στη χρήση (δηλ. πλαστικότητας - plasticity). Ωστόσο η εν λόγω προσέγγιση δεν υποστηρίζει ρόλους χρηστών (user roles), διαχείριση συνόδων (session management), αντιγραφή αντικειμένων (object replication) και ενημερότητα (awareness). Επιπλέον, προσφέρει μηδενική υποστήριξη εργαλείων όσον αφορά την πρόνοια για χαμηλού επιπέδου θέματα, διαχείριση

πολλαπλών συλλογών αντικειμένων (managing diverse collections of objects), κατανεμημένη φόρτωση κλάσεων (distributed class loading) και εξαρτώμενων βιβλιοθηκών (dependency libraries), κωδικοποίηση της λογικής στιγμιοτυποποίησης αναλόγως των διαθέσιμων εργαλειοθηκών στόχο, καθορισμό της προ-στιγμιοτυποποιημένης κατάστασης των widgets (pre-instantiation configuration), κοκ. Από την άλλη πλευρά, εισάγει μια νέα ροή ανάπτυξης (development workflow) η οποία είναι υπερβολικά απαιτητική ως προς την κατανόηση και αποτελεσματική-επιτυχή διαχείρισή της.

Ένα εξειδικευμένο ερευνητικό ρεύμα στα πλαίσια του ευρύτερου χώρου της μοντελοκεντρικής μηχανικής έχει ρητώς εστιάσει στη μελέτη της συνεργατική αλληλεπίδρασης (collaborative interaction). Ενδεικτικά παραδείγματα συμπεριλαμβάνουν το FlowiXML[35], το AMENITIES [12] [34], το UML-G[46][47], το CIAM[13], το TOUCHE[14], κ.α.. Όλες αυτές οι προσπάθειες εστιάζουν κατά κύριο λόγο στην επινόηση σημειογραφικών γλωσσών (notations)[11][37][9] και εργαλείων[43] για τη μοντελοποίηση της συνεργατικής συμπεριφοράς και των ροών εργασίας. Ως εκ τούτου η κύρια συνεισφορά τους εδράζεται στο ότι κάνουν διακριτά ποικίλα στοιχεία της συνεργασίας (δηλ. ρόλους, καθήκοντα και ενέργειες) χρησιμοποιώντας ενδεδειγμένη σημειογραφία. Ωστόσο, μόνο κάποιες από αυτές (ελάχιστες) κάνουν προσπάθειες ως προς τη γέννηση της (λειτουργικής) συνεργατικής διεπαφής (collaborative UI). Παραδείγματα προς αυτή την κατεύθυνση αποτελούν το TOUCHE[14] και το Fiia[31], παρέχοντας πολυχρηστική λειτουργικότητα χρησιμοποιώντας adhoc διασυνδέσεις (mappings) σε ένα προκαθορισμένο (fixed), custom, υποκείμενο groupware toolkit. Αυτό έχει ως αποτέλεσμα, η υποστήριξη πολλαπλών πλατφορμών να περιορίζεται από τα όρια της διαθεσιμότητας της υποκείμενης εργαλειοθήκης (groupware toolkit) στο εκάστοτε πλαίσιο χρήσης.

Κεφάλαιο 4 –Προσέγγιση

Κύριος άξονας της επιχειρηματολογίας την εν λόγω προσέγγισης αφορά το γεγονός ότι για τις κατηγορίες διεπαφών που αποτελούν αντικείμενο μελέτης στην παρούσα εργασία (βλέπε Κεφάλαιο 2), η τρέχουσα τεχνολογική στάθμιση (state of the art) και οι διαθέσιμες προσεγγίσεις (βλέπε Κεφάλαιο 3), αποτυγχάνουν να διαχειριστούν τις απαιτήσεις τους στο σύνολο, εξαιτίας της υιοθέτησης μηχανισμών γενίκευσης (generalizations) αντί της υποστήριξης πραγματικής αφαίρεσης (true abstraction). Συγκεκριμένα, οι εργαλειοθήκες (toolkits), λόγω γενικεύσεων σχετικά με τις κλάσεις αντικειμένων και το διάλογο περιορίζονται στην υποστήριξη μόνο ομογενών πλατφορμών, ενώ οι μοντελο-κεντρικές τεχνικές λόγω γενικεύσεων με μοναδικό γνώμονα την απρόσκοπτη υποστήριξη φορητότητας (mere-portability) αναγκάζονται να περιορίσουν το εύρος των υποστηριζόμενων διαδραστικών αντικειμένων σε αυτά που είναι εγγενώς υποστηριζόμενα και κοινής φιλοσοφίας, καταλήγοντας κατ' αυτό τον τρόπο στην υποστήριξη απλών διεπαφών.

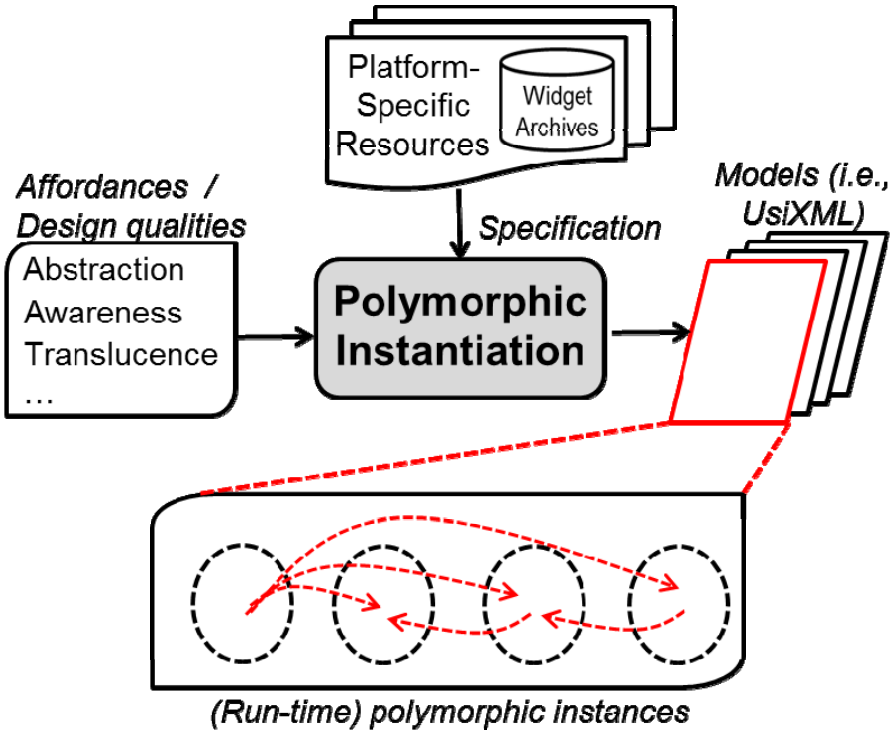
Για την αντιμετώπιση των περιορισμών αυτών, η παρούσα εργασία εδράζεται στην έννοια των σχημάτων πολυμορφικής στιγμιοτυποποίησης την οποία και εξειδικεύει στο πεδίο των συνεργατικών διεπαφών που παρουσιάζουν αυξημένες δυνατότητες για την αποτύπωση, διαμοιρασμό και σύγχρονη διαχείριση συνεργατικού έργου από εταίρους σε διαφορετικά γεωγραφικά σημεία που αξιοποιούν διαφορετικές τερματικές συσκευές.

'Σχήματα πολυμορφικής στιγμιοτυποποίησης': Περίγραμμα της Μεθόδου

Η εν λόγω προσέγγιση εστιάζει στην υιοθέτηση της πολυμορφικής στιγμιοτυποποίησης σε επίπεδο διεπαφής (polymorphic UI instantiation) για την επίτευξη του ζητούμενου βαθμού αφαίρεσης. Ο πολυμορφισμός, υποδηλώνει την ικανότητα μιας διεπαφής να αποδοθεί διαμέσου εναλλακτικών αλληλεπιδραστικών ενσαρκώσεων, ούτως ώστε να εξυπηρετεί τις εν εκτελέσει ανάγκες (οπότε και γίνονται γνωστές) είτε του χρήστη, είτε του περιβάλλοντος εκτέλεσης. Η ιδέα απεικονίζεται στην Εικόνα 11, αναδεικνύοντας ότι ο πολυμορφισμός (ως ποιοτικό χαρακτηριστικό) χαρακτηρίζεται από: α) συγκεκριμένα 'design-affordances[9]' τα οποία προσδιορίζονται κατά τρόπο ρητό, β) από προδιαγραφές που περιγράφουν με λεπτομέρεια πως τα εν λόγω affordances υποστηρίζονται στα πλαίσια ενός συγκεκριμένου λεξικού και γ) σχήματα

[9] Ποιοτικό χαρακτηριστικό ενός αντικειμένου το οποίο επιτρέπει σε κάποιον να εκτελέσει μια ενέργεια

στιγμιοτυποποίησης τα οποία παράγουν στιγμιότυπα συγκεκριμένων διαδραστικών στοιχείων με βάση τη συμπεριφορά που προσδιορίζεται στις προδιαγραφές τους. Υπό αυτή την έννοια λοιπόν, μια διεπαφή δεν είναι στατική αλλά συναρμολογείται εν εκτελέσει από ποικιλία διαδραστικών στοιχείων (τα οποία δεν χρειάζεται κατ' ανάγκη να είναι διαθέσιμα διαμέσου μιας και μόνο διαδραστικής εργαλειοθήκης ή πλατφόρμας), εκμεταλλευόμενη τους όποιους περιορισμούς ή δυνατότητες ανιχνεύονται εκείνη τη στιγμή.



Εικόνα 11: Πολυμορφική στιγμιοτυποποίηση (Polymorphic instantiation)

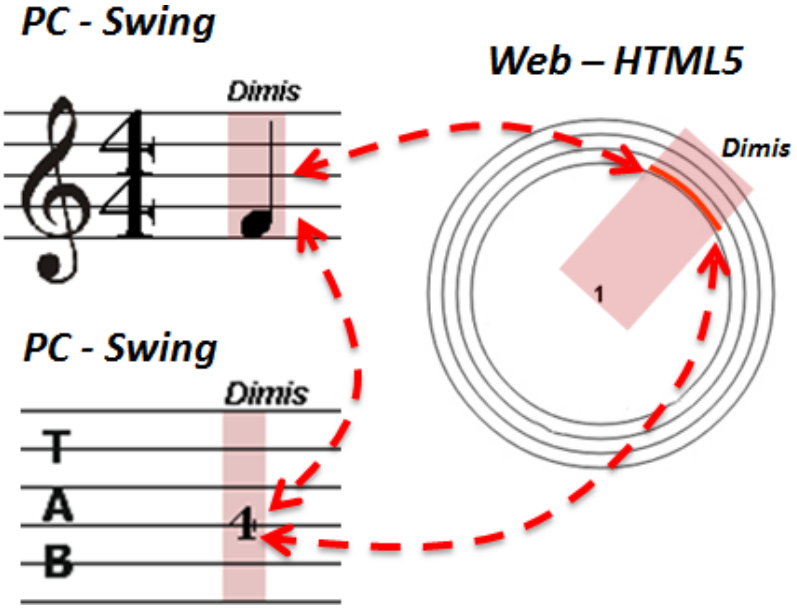
Για να καταστεί ωστόσο αυτό εφικτό, απαιτείται η ύπαρξη ενός σχήματος προδιαγραφών (specification scheme), που να εγκαθιδρύει τον απαιτούμενο βαθμό αφαίρεσης, σε σχέση με πόρους που σχετίζονται με τη συσκευή και την πλατφόρμα, κρύβοντας χαμηλού-επιπέδου τεχνικές λεπτομέρειες της υλοποίησης (technicalities). Ένα τέτοιο σχήμα, ονομαζόμενο Widget Specification Language (Γλώσσα Περιγραφής Widget), είναι επιφορτισμένο με το να περιγράφει όχι μόνο το look & feel των widgets αλλά και το εύρος των εναλλακτικών υλοποιήσεων όπως επίσης και τους μηχανισμούς σύνδεσης με τις βιβλιοθήκες κάθε εργαλειοθήκης, αντί της απευθείας ενσωμάτωσης αυτών στον κώδικα της διεπαφής.

Για να αναδείξουμε την ανάγκη και την αξία των σχημάτων πολυμορφικής στιγμιοτυποποίησης θα παρουσιάσουμε συνοπτικά ένα απαιτητικό σενάριο χρήσης που υποθέτει κατανεμημένη, σύγχρονη και συνεργατική εκτέλεση καθηκόντων και στη συνέχεια θα παρουσιάσουμε τόσο τις επιπλοκές που προβάλλει στις καθιερωμένες μεθόδους όσο και την προτεινόμενη λύση βάσει των σχημάτων πολυμορφικής στιγμιοτυποποίησης. Το σενάριο αφορά τη σύγχρονη και συνεργατική σύμπραξη χρηστών σε ένα μάθημα θεωρίας της μουσικής, ενώ αξίζει να σημειωθεί ότι η υλοποίηση του απαιτεί εξειδικευμένη (domain-specific) λειτουργικότητα η οποία δεν υποστηρίζεται εγγενώς από καμία εργαλειοθήκη διαδραστικών αντικειμένων γενικού σκοπού (π.χ. java/swing, Android/UI, MFC, κοκ.).

Οραματισμός αλληλεπιδραστικού σεναρίου χρήσης και προδιαγραφών

Τα μαθήματα μάθησης θεωρίας της μουσικής αποτελούν εποπτευόμενες (moderated) συνεργατικές συνεδρίες όπου ένας εκπαιδευτής και μια ομάδα μαθητών συμπράττουν (co-engage) για να εργαστούν διαμέσου ενδεδειγμένων μουσικών τεχνουργημάτων. Παρά το γεγονός ότι υπάρχει πλήθος διαθέσιμων τεχνολογιών για τη διευκόλυνση της επικοινωνίας εταίρων σε μαθήματα μουσικής (π.χ., conferencing και e-presence συστήματα), η υποστήριξη σύγχρονης συνεργατικής εργασίας πάνω σε διαμοιραζόμενες μουσικές αναπαραστάσεις παραμένει απαιτητική. Για το σκοπό μας, θα υποθέσουμε μια ομάδα χρηστών αποτελούμενη από ένα συντονιστή (moderator) και πολλούς μαθητές διεσπαρμένους σε διαφορετικά γεωγραφικά σημεία. Ο καθηγητής προετοιμάζει ένα κομμάτι μουσικής που εσκεμμένως περιέχει σειρά λαθών (π.χ. λάθος τοποθετημένες νότες, μη έγκυρο αριθμό νοτών σε ένα μέτρο, κοκ.) και ορίζει το χρόνο και την ημερομηνία για μια εικονική διάλεξη (σύγχρονη σύνοδο – synchronous session) όπου οι μαθητές από κοινού θα αναγνωρίσουν και διορθώσουν τα όποια λάθη. Κατά τη διάρκεια του μαθήματος, κάθε συμμετέχων λειτουργεί επί του ίδιου διαμοιραζόμενου πόρου (δηλ. μουσική), ωστόσο διαμέσου διαφορετικών αναπαραστάσεων και αλληλεπιδραστικών υλοποιήσεων. Παραδείγματος χάριν, ο καθηγητής και προχωρημένοι μαθητές μπορεί να επιθυμούν την αναπαράσταση του πενταγράμμου ενώ οι αρχάριοι χρήστες μπορεί να επιλέξουν μεταξύ της ταμπλατούρας ή άλλης γραφικής απόδοσης. Ενδεικτικά, η Εικόνα 12 σχηματοποιεί τα εναλλακτικά διαδραστικά σενάρια που είναι δυνατόν να προκύψουν καθώς και τις εργαλειοθήκες / πλατφόρμες που τα υποστηρίζουν. Επιπλέον, θα υποθέσουμε πως ένα μέλος της ομάδας επιθυμεί εναλλακτικό τρόπο προσβασιμότητας λόγω της ανεπάρκειάς του να

διαχειριστεί το οπτικό κανάλι αντίληψης. Ως ενδεικτικό παράδειγμα μπορούμε να υποθέσουμε ένα τυφλό χρήστη που η πρόσβαση του εξυπηρετείται με ομιλία και φωνητικές εντολές (δηλ., speech-based interaction). Τέλος, καθώς η σύμπραξη των μαθητών είναι σύγχρονη, κάθε ενέργεια ενός χρήστη θα πρέπει να 'καταγράφεται', να προωθείται και να διερμηνεύεται κατά το δοκούν από τα υπόλοιπα μέλη της ομάδας έτσι ώστε ο διαμοιραζόμενος πόρος να παραμένει πάντα συγχρονισμένος και να διευκολύνεται το feedthrough.



Εικόνα 12: Platform-specific στιγμιότυπα νοτών

Είναι προφανές ότι οι ανωτέρω προδιαγραφές μπορούν μόνο εν μέρη να υλοποιηθούν σήμερα από τις τεχνολογίες που είναι ευρέως διαθέσιμες. Ειδικότερα, κάθε επιμέρους στιγμιότυπο διεπαφής είναι δυνατόν να υλοποιηθεί με συγκεκριμένα εργαλεία ως ανεξάρτητη εφαρμογή που να εξυπηρετεί μεμονωμένους χρήστες. Η δυστοκία συνίσταται στη περίπτωση της ταυτόχρονης και συγχρονισμένης σύμπραξης των εταίρων στην εκτέλεση των καθηκόντων.

Προσέγγιση από πλευράς Μηχανικής (Engineering Approach)

Έχοντας περιγράψει τη μέθοδο θα εστιάσουμε στην προτεινόμενη προσέγγιση από πλευράς μηχανικής η οποία χρησιμοποιήθηκε για να υποστηρίξει και το εν λόγω σενάριο. Θα προσπαθήσουμε να αιτιολογήσουμε τόσο τις δεσμεύσεις που έγιναν σε γενικό επίπεδο όσο και τον τρόπο με τον οποίο επιστρατεύτηκαν τα οικεία εργαλεία για την υποστήριξη του σεναρίου της συνεργατικής μάθησης θεωρίας της μουσικής. Στη βάση της, η προσέγγιση αγκιστρώνεται

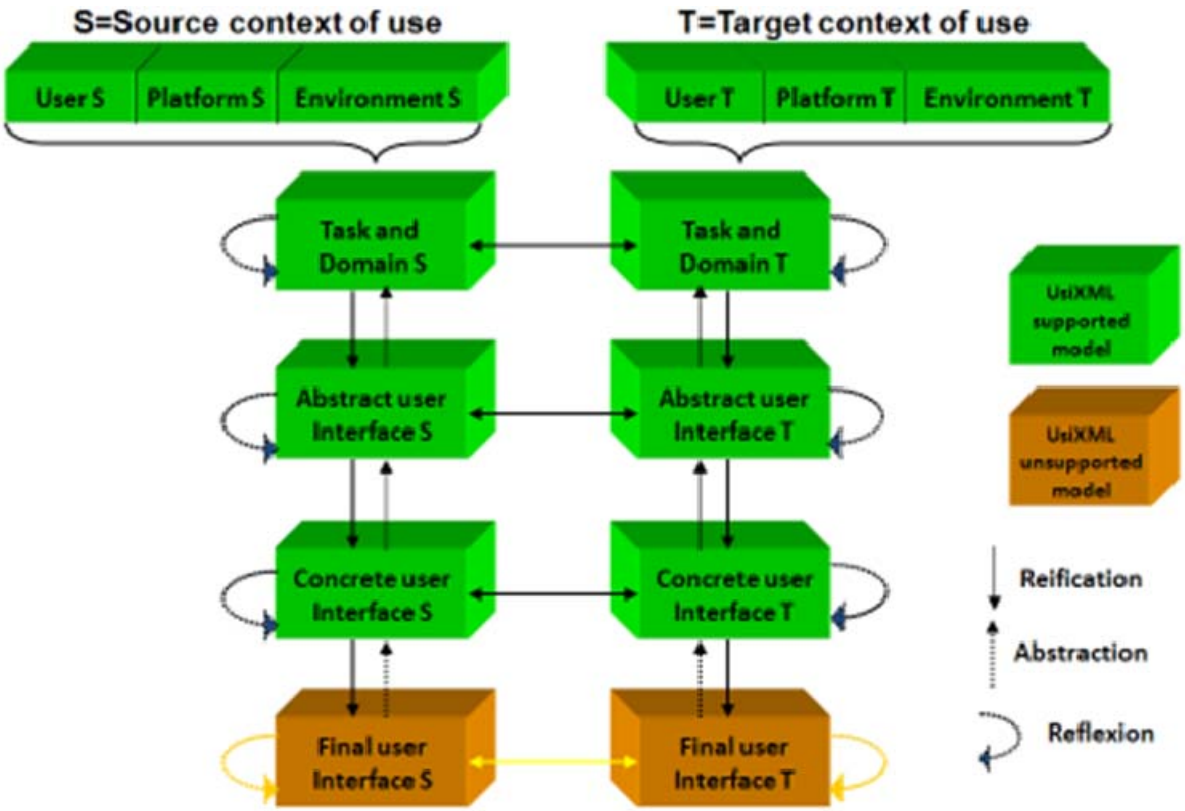
από (α) την επέκταση της UsiXML[6][29], ώστε να παραχωρηθεί πλήρης υποστήριξη πολυμορφικής στιγμιοτυποποίησης και (β) την παροχή σχεδιαστικής όπως και runtime υποστήριξης πολυμορφικών σχημάτων στιγμιοτυποποίησης ώστε να είναι εφικτή η διαχείριση ποικιλίας διαδραστικών λεξικών (συμπεριλαμβανομένων μη-εγγενώς υποστηριζόμενων widgets) τα οποία προορίζονται για ριζικά διαφορετικά (ετερογενή) πλαίσια χρήσης.

Σύντομη περιγραφή της UsiXML

Η UsiXML [6] αποτελεί την υλοποίηση του πλαισίου αναφοράς (reference-implementation) 'Cameleon Reference Framework', το οποίο επινοήθηκε ώστε να παρέχει υποστήριξη πλαστικότητας[10] (plasticity) σε επίπεδο διεπαφής[2]. Πέραν τούτου είναι βασικό να τονιστεί η ενοποιημένη διαδικασία ανάπτυξης πανταχού παρόντων διεπαφών (unified ubiquitous UI development process) την οποία εγγενώς προάγει το εν λόγω πλαίσιο αναφοράς, μειώνοντας δραστικότατα την πολυπλοκότητα σχεδίασης και ανάπτυξης για πολλαπλά περιβάλλοντα, η οποία προφανώς κληροδοτείται σε όλες τις υλοποιήσεις της, άρα και στη UsiXML. Η UsiXML προτείνει τον ορισμό μιας διεπαφής σε τέσσερα επίπεδα αφαίρεσης, με το κάθε ένα να εστιάζει σε διαφορετικές πλευρές της διαδικασίας ανάπτυξης. Πιο συγκεκριμένα, στο επίπεδο των 'tasks and concepts' (δηλ. το πιο αφηρημένο που υποστηρίζεται), μια διεπαφή προσδιορίζεται υπό την έννοια της αυξητικής (incremental) ιεραρχικής αποσύνθεσης καθηκόντων (tasks) σε υπο-καθήκοντα (sub-tasks) συσχετισμένα με τελεστές που ορίζουν την ακολουθία εκτέλεσης. Το συγκεκριμένο επίπεδο περιγραφής δεν εμπεριέχει κατ' οποιονδήποτε τρόπο κάποια δέσμευση, αναφορά ή εξάρτηση σε σχέση με υπολογιστικούς πόρους, γι' αυτό και χαρακτηρίζεται ως 'Computation Independent Model'. Στο AUI επίπεδο [2] (το αμέσως πιο συγκεκριμένο επίπεδο) δίνεται η δυνατότητα περιγραφής μιας διεπαφής ανεξαρτήτως εξαρτήσεων από αλληλεπιδραστικά κανάλια (modality independent), υπό την έννοια μιας αλληλεπιδραστικής ιεραρχίας αποτελούμενης από αφηρημένα διαδραστικά αντικείμενα (objects) και υποδοχείς (containers). Μιας και σε αυτό το επίπεδο δεν γίνεται καμία αναφορά σε πλατφόρμα, ορίζεται ως 'Platform Independent Model'. Τέλος, στο CUI επίπεδο, μια διεπαφή ορίζεται ως ένα σύνολο συγκεκριμένων διαδραστικών αντικειμένων (concrete interaction objects) καταλλήλως συναρμολογημένα με τρόπο ανεξάρτητο σε κάθε περίπτωση από λεπτομέρειες της

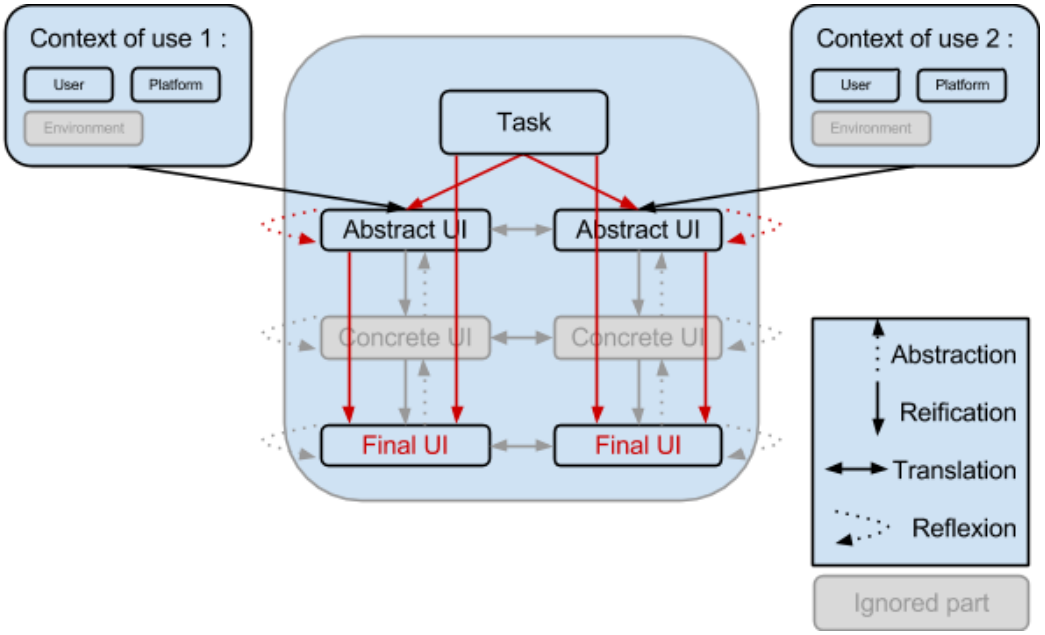
[10] Η ικανότητα μιας διεπαφής να προσαρμόζει πλήρως τα χαρακτηριστικά της εν εκτελέσει εξασφαλίζοντας σε κάθε περίπτωση υπόψιν τη διατήρηση της ευχρηστίας του αλληλεπιδραστικού συστήματος.

υπολογιστικής υλοποίησης σε επίπεδο toolkit-library (Platform Specific Model, αλλά toolkit-independent) (βλ. παράδειγμα Εικόνα 34 παραρτήματος). Στην Εικόνα 13 αποτυπώνονται όλα τα επίπεδα του 'Cameleon Reference Framework' άρα και της UsiXML.



Εικόνα 13: Model-Driven-Engineering Cameleon reference framework

Μεταβάσεις μεταξύ διαφορετικών επιπέδων αφαίρεσης υποστηρίζονται υιοθετώντας μια ημι-αυτοματοποιημένη προσέγγιση μετασχηματισμών (μετασχηματισμοί γράφων[45], βλ. Εικόνα 14). Ένα σημαντικότατο μειονέκτημα της UsiXML είναι η υποστήριξη που παρέχει όσον αφορά τη χρησιμοποίηση διαφορετικών διαδραστικών λεξικών (target presentation vocabularies). Συγκεκριμένα, στην τρέχουσα έκδοσης της η γλώσσα περιορίζεται σε διαδραστικά αντικείμενα που υποστηρίζονται από τα πιο δημοφιλή toolkits. Προφανώς, αυτό είναι σημαντικό μειονέκτημα μιας και περιορίζει την εκφραστικότητα της γλώσσας περιορίζοντάς τον τύπο διεπαφών που μπορεί να υποστηριχθούν.



Εικόνα 14: Πιθανές διατάξεις ανάπτυξης πανταχού παρόντων διεπαφών στη UsiXML[11]

Επεκτάσεις στη UsiXML

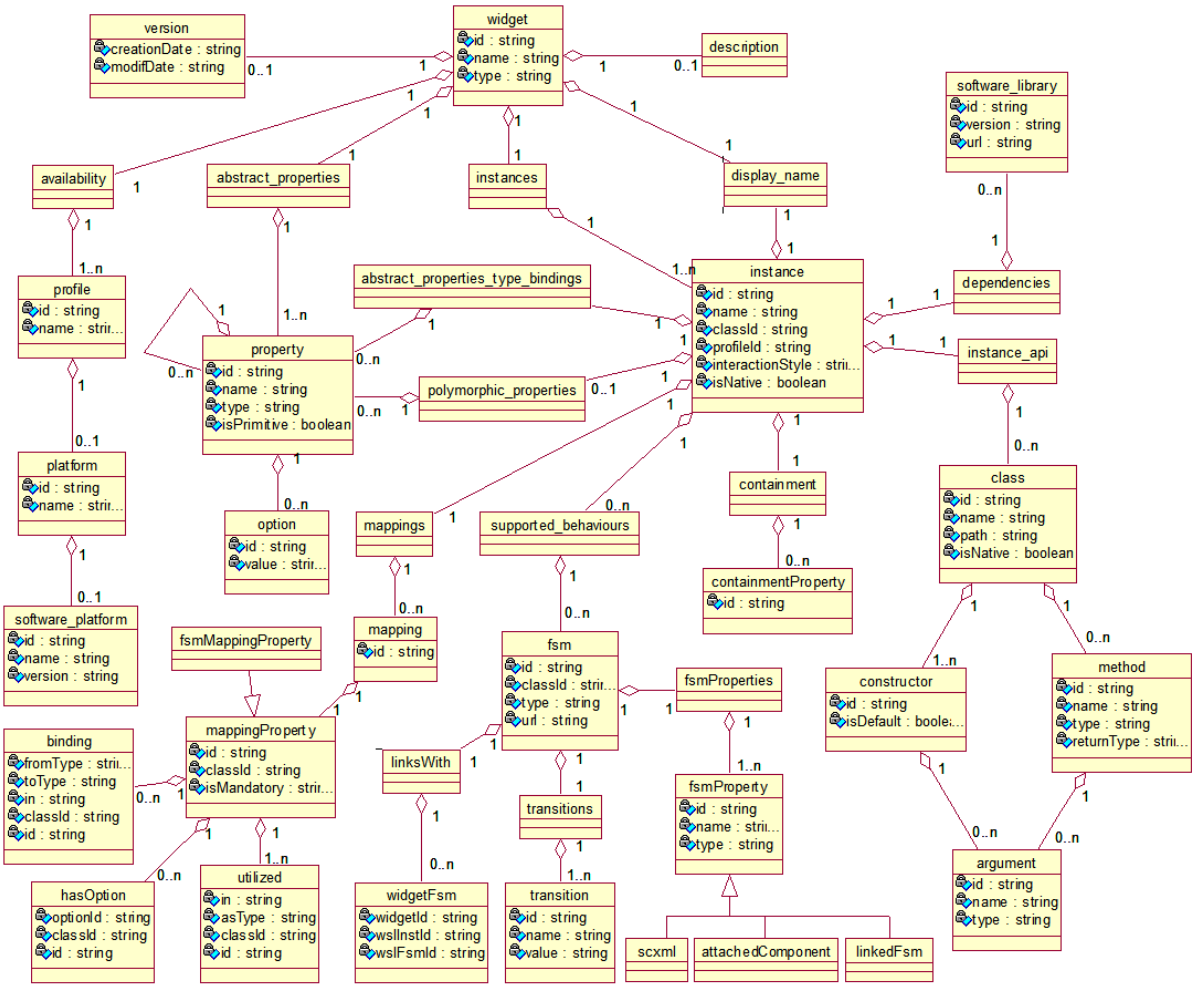
Οι επεκτάσεις της UsiXML που δρομολογήθηκαν για τις ανάγκες της παρούσας έρευνας περιλαμβάνουν μια νέα ροή εργασίας που ονομάζεται 'Polymorphic Widget Specification' καθώς επίσης βελτιώσεις (επαυξήσεις, επεκτάσεις, προσθέσεις) στο βασικό σύνολο των μοντέλων της γλώσσας. Εν συνεχεία, περιγράφονται και τα δύο επίπεδα προσθηκών αναδεικνύοντας τόσο το σκεπτικό όσο και την υποκείμενη σημασιολογία τους (semantics).

Ροή Περιγραφής Πολυμορφικών Αντικειμένων (Polymorphic Widget Specification Workflow)

Για την παροχή υποστήριξης ποικιλίας συλλογών αντικειμένων, εγγενώς υποστηριζόμενων ή εξειδικευμένων (custom), επινοήθηκε μια ροή για την προδιαγραφή widgets ('Widget Specification Workflow') που είναι συμβατή με XMLSchema[12] περιγραφές. Η ροή εργασίας εναπόκειται σε μια ενδεδειγμένη γλώσσα περιγραφής (WSL)[3] η οποία επιτρέπει στα οικεία μοντέλα και εργαλεία να χρησιμοποιήσουν βιβλιοθήκες τρίτων (3rd party libraries). Η δομή της και οι συσχετίσεις μεταξύ των ορισμένων οντοτήτων φαίνονται στο διάγραμμα κλάσεων στην Εικόνα 15.

[11] Από: http://romainvv.ddns.net/hyp-models-of-ui/

[12] http://www.w3.org/2001/XMLSchema/



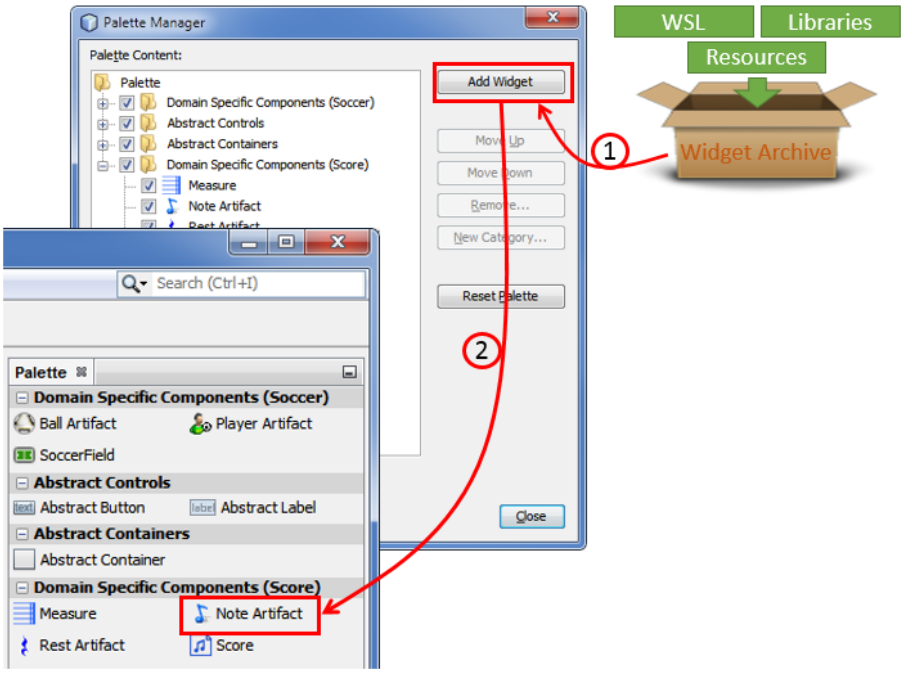
Εικόνα 15: Δομή γλώσσας περιγραφής widgets (WSL)

Στην πράξη το παραπάνω μοντέλο αντιστοιχίζεται σε ένα αρχείο συμβατό με 'XMLSchema'. Για να μπορέσει ένα οποιοδήποτε αντικείμενο (και οι εναλλακτικές αναπαραστάσεις αυτού) να χρησιμοποιηθεί – ενσωματωθεί στη γλώσσα θα πρέπει πρώτα να προσδιοριστεί κατάλληλα διαμέσου ενός στιγμιοτύπου XML (δηλ. Xml αρχείου) συμβατού με το WSL-XMLSchema. Όντας προτυποποιημένος ο τρόπος περιγραφής - κωδικοποίησης των widgets (και του συνόλου του API τους), δίνει τη δυνατότητα ανάπτυξης γενικού σκοπού συστατικών λογισμικού (runtime components), τα οποία μπορούν αξιοποιώντας τις reflective[13] δυνατότητες των γλωσσών προγραμματισμού να γεννήσουν και να διαχειριστούν εν εκτελέσει το οποιοδήποτε διαδραστικό

[13] https://en.wikipedia.org/wiki/Reflection_(computer_programming)/

στοιχείο χωρίς a priori γνώση της δομής (API) ή των τυχόν εξαρτήσεων σε πόρους (εξαρτώμενες βιβλιοθήκες, εικόνες, κοκ.).

Η Εικόνα 16 δείχνει ένα στιγμιότυπο του εργαλείου που αναπτύχθηκε για την υποστήριξη της ενσωμάτωσης ενός νέου widget στη γλώσσα WSL.



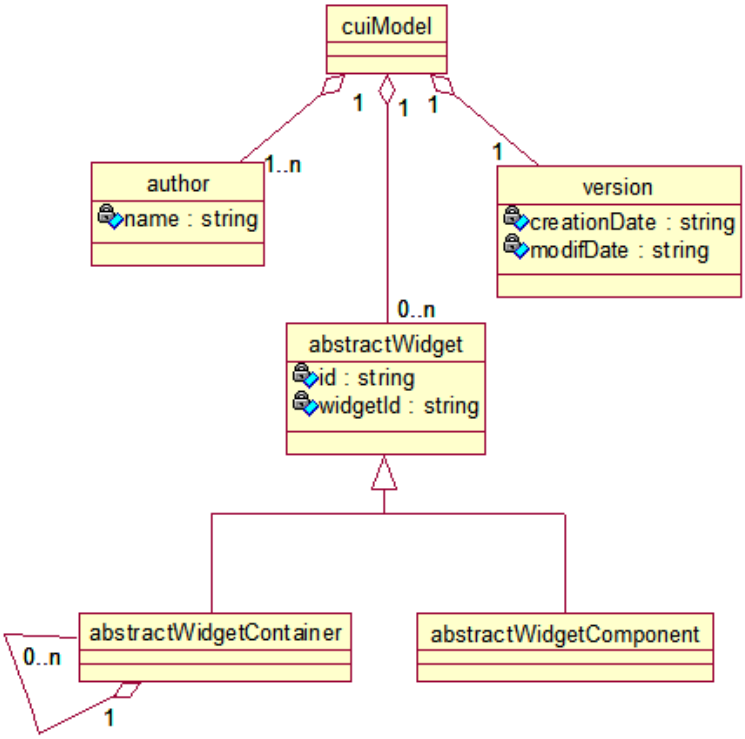
Εικόνα 16: Επισκόπηση της ροής περιγραφής (specification) ενός 'αφηρημένου' widget

Το 'widget archive' είναι ένα 'packaging format' (συμβατό με το πρότυπο zip) το οποίο επιτρέπει την ομαδοποίηση όλων των πληροφοριών που απαιτούνται από το WSL, συμπεριλαμβανομένων τυχόν πόρων που είναι απαραίτητοι για την απρόσκοπτη χρησιμοποίησή του.

Το WSL επεκτείνει τα παραδοσιακά μοντέλα 'CUI' (το πιο συγκεκριμένο επίπεδο περιγραφής πριν τη γέννηση της τελικής-εκτελέσιμης διεπαφής), δεδομένου ότι αντλεί δυναμικά τα υποστηριζόμενα semantics του από το σύνολο των widgets που έχουν ενσωματωθεί στη γλώσσα (δηλ. σύνολο widget-archives). Αυτό είναι σε πλήρη αντίθεση με τα συμβατικά CUI μοντέλα όπου τόσο το συντακτικό όσο και τα υποστηριζόμενα semantics είναι hardcoded και ενσωματωμένα στην επίσημη περιγραφή – ορισμό (formal specification) της γλώσσας.

Με βάση το νέο CUI, μια διεπαφή ορίζεται πλέον (στο πιο concrete επίπεδο δηλ.) διαμέσου της ιεραρχικής αποσύνθεσης των widgets που είναι διαθέσιμα στη γλώσσα (έχουν περιγράφει με τη

βοήθεια του WSL και ενσωματωθεί μέσω των widget-archives). Η Εικόνα 17, δείχνει με τη βοήθεια ενός διαγράμματος κλάσεων τον αναθεωρημένο ορισμό του CUI. Είναι αυτός ο τρόπος με τον οποίο προσεγγίζεται και υποστηρίζεται ο εμπλουτισμός (δηλ. επαύξηση και επέκταση) της εκφραστικής επάρκειας της γλώσσας. Γι' αυτό, και όπως φαίνεται, η κλάση 'abstractWidget' φέρει μόνο δύο ιδιότητες: α) είναι το id το οποίο αφορά τον μοναδικό αναγνωριστικό του στιγμιοτύπου του widget ('abstractWidget') στη CUI ιεραρχία, και β) το μοναδκό id του widget (widgetId) σε επίπεδο διαδραστικού λεξικού της γλώσσας (widget archives), όπως αυτό έχει οριστεί στο WSL με το οποίο έχει γίνει περιγραφεί και γίνει deployed.



Εικόνα 17: Το αναθεωρημένο μοντέλο CUI

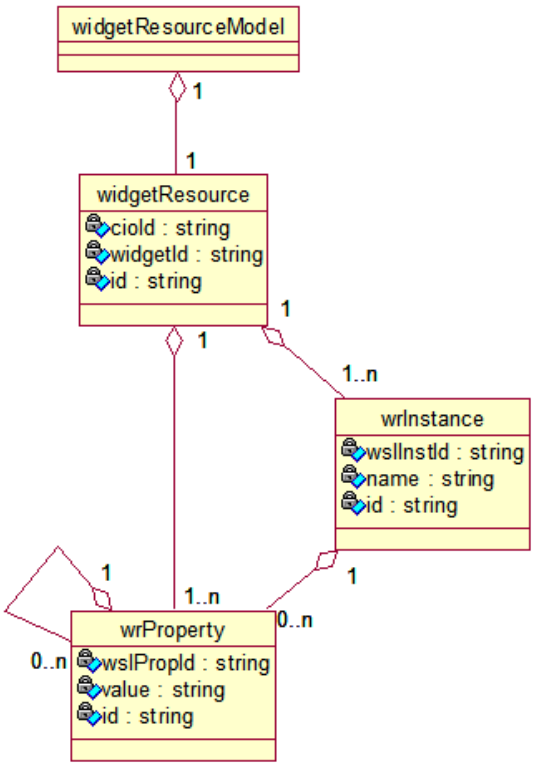
Ένας δεύτερος περιορισμός που επιβάλλεται στην πράξη με βάση τις εγγενείς προδιαγραφές του CUI μοντέλου της UsiXML είναι πως δεν μπορεί να υποστηρίξει την έννοια του πολυμορφισμού που παρουσιάστηκε προηγούμενα. Αυτό οφείλεται στο γεγονός πως το συντακτικό και τα semantics των προδιαγραφών του CUI μοντέλου της UsiXML επιβάλλουν στα πλαίσια κάθε ετικέτας (άρα και του σχετιζόμενου με αυτή διαδραστικού τύπου αντικειμένου όπως 'κουμπί', 'ετικέτα', κοκ), τη διαθεσιμότητα προδιαγεγραμμένου πλήθους και εύρους ιδιοτήτων που αντιστοιχούν μόνο στα κοινά (common) και όχι τα πολυμορφικά χαρακτηριστικά των όποιων

τύπων αντικειμένων υποστηρίζει. Παραδείγματος χάριν, στην περίπτωση της ετικέτας ‘Button’, επιβάλλεται (εκτός της δυνατότητας ορισμού τιμής της ετικέτας), ο ορισμός του επιθυμητού πλάτους (width) και ύψους (height). Αν και αυτή η γενίκευση είναι βολική για συμβατικά κουμπιά με ορθογώνια γραφική αναπαράσταση, εμποδίζει τη χρήση πιθανών εναλλακτικών αναπαραστάσεων όπως π.χ. στρογγυλά κουμπιά (‘RoundButton’), όπου αντί πλάτους και ύψους απαιτείται ένα σημείο στο επίπεδο και η ακτίνα. Το ίδιο πρόβλημα θα υπήρχε και στην περίπτωση των νοτών στην Εικόνα 12, όπου κάθε εναλλακτική αλληλεπιδραστική αναπαράσταση, της ίδιας ακριβώς νότας, επιδεικνύει σε σχέση με τις άλλες ριζικά διαφορετικά χαρακτηριστικά σε όλα τα επίπεδα, δηλ. συντακτικό (syntactic), λεκτικό (lexical), φυσικό (physical) και σημασιολογικό (semantic).

Αυτός ο εγγενής περιορισμός αποδίδεται στην αδυναμία υποστήριξης πραγματικής αφαίρεσης (‘true abstraction’). Να επισημανθεί ξανά ότι η πολυμορφική στιγμιοτυποποίηση (polymorphic instantiation) συνιστά κλειδί ως προς την υποστήριξη πραγματικής αφαίρεσης, επιβάλλοντας το ρητό διαχωρισμό της περιγραφής (specification) των αφηρημένων (δηλ. κοινών) και πολυμορφικών χαρακτηριστικών από τον τρόπο που ενσαρκώνονται και υλοποιούνται σε μια βιβλιοθήκη.

Στην προσέγγισή μας αυτός ο διαχωρισμός επιτρέπεται διαμέσου του WSL (βλ. Εικόνα 15: κλάση ‘abstractProperties’ και ‘polymorphicProperties’). Όσον αφορά το επίπεδο του αναθεωρημένου CUI (Εικόνα 17), δεν γίνεται καμία αναφορά σε πολυμορφικές ή αφηρημένες ιδιότητες παρά μόνο ορίζεται μια αναφορά (βλ. ιδιότητα ‘widgetId’ Εικόνα 17) σε επίπεδο cui ετικέτας προς το στιγμιότυπο του WSL (XML αρχείο) το οποίο κωδικοποιεί το widget με το οποίο αυτή κάθε φορά σχετίζεται. Με βάση αυτή την αναφορά είναι εύκολο να συναχθεί το σύνολο των υποστηριζόμενων ιδιοτήτων που υποστηρίζονται στα πλαίσια κάθε cui ετικέτας (είτε είναι αφηρημένη είτε πολυμορφική). Το μοντέλο στο οποίο αποτυπώνονται οι τιμές που ορίζουν το ‘pre-instantiation configuration’ κάθε cui ετικέτας ονομάζεται ‘WidgetResource’. Ο λόγος που δεν γίνεται αυτό απευθείας μέσα στο CUI είναι για τον καλύτερο διαχωρισμό του ρόλου που εξυπηρετεί κάθε ένα. Το CUI ορίζει την ιεραρχική αποσύνθεση – δομή των widget archives που συνιστούν κάθε φορά την επιθυμητή διεπαφή–, ενώ το ‘WidgetResource’ κρατάει (κωδικοποιεί) όλες τις πληροφορίες για την ‘pre-instantiation’ παραμετροποίηση των

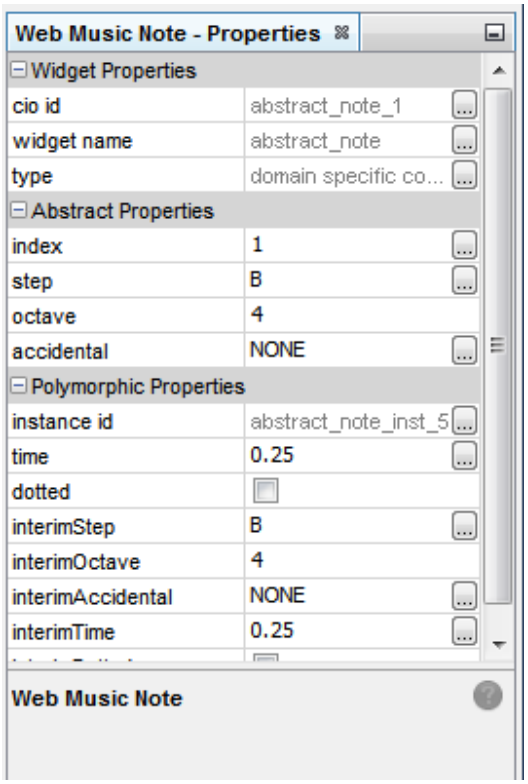
προηγούμενων και αφορά μη γραφικά αντικείμενα. Η υψηλού επιπέδου δομή του ‘WidgetResouce’ μοντέλου φαίνεται στην Εικόνα 18.



Εικόνα 18: Δομή του μοντέλου ‘widgetResource’

Όπως φαίνεται από τις συσχετίσεις μπορεί να δεχτεί απευθείας ορισμό μιας ιδιότητας (‘αφηρημένης’), είτε τον ορισμό της εντός των πλαισίων (βλ. κλάση ‘wrmInstance’) ενός συγκεκριμένου πολυμορφικού στιγμιοτύπου (‘πολυμορφική ιδιότητα’). Κάθε ιδιότητα μπορεί να οριστεί με (να της αποδοθεί τιμή από) τους ακόλουθους τρόπους: α) σε περίπτωση που η δομή της είναι επίπεδη (flat), με τη βοήθεια ενός και μόνο επιπέδου περιγραφής (inline definition), ενώ β) στην περίπτωση που η δομή της είναι ιεραρχική, ασχέτως βαθμού πολυπλοκότητας, μπορεί να περιγραφεί διαμέσου της ιεραρχικής αποσύνθεσης ιδιοτήτων (wrProperty), μέχρις ότου κάθε μια καταλήξει να μπορεί να οριστεί με τη βοήθεια απλών τύπων δεδομένων(δηλ. int, float, double, string, etc.). Το τελευταίο ισχύει για απόδοση αρχικών τιμών (παραμετροποίηση) τόσο σε αφηρημένες όσο και πολυμορφικές ιδιότητες. Επίσης, στα πλαίσια του ‘WidgetResouce’ μοντέλου κωδικοποιείται (και άρα μπορεί να συναχθεί) και το πλήθος των πολυμορφικών σχημάτων που είναι δυνατόν ή κρίνεται απαραίτητο να γίνουν engaged (ανά

περίπτωση) εν εκτελέσει στα πλαίσια κάθε στιγμιοτύπου στοιχείου του cui (cui element). Στιγμιότυπα του εν λόγω μοντέλου παράγονται και ορίζονται σταδιακά (incrementally), κατά τη φάση της σχεδίασης της διεπαφής με τη βοήθεια ενδεδειγμένων αλληλεπιδραστικών εργαλείων (βλ. Εικόνα 19).



Εικόνα 19: Παράδειγμα αλληλεπιδραστικής απόδοσης τιμών στα πλαίσια του πολυμορφικού στιγμιοτύπου μιας κυκλικής μουσικής νότας

Να τονιστεί πως η δομή αντικειμένων που ορίζεται διαμέσου του 'WidgetResource' μοντέλου είναι προτυποποιημένη με τέτοιο τρόπο που εκτός του εκφραστικού της πλουραλισμού, προνοεί για την γενικευμένη εν εκτελέσει αντιστοίχισή τους με τις λεξικές ιδιότητες (lexical), και τα semantics που κωδικοποιούνται στα πλαίσια κάθε WSL στιγμιοτύπου. Το τελευταίο σημαίνει πως δεν υπάρχει καμία a priori δέσμευση ως προς τη γνώση της δομής των implementation-specific σχημάτων.

Συμπερασματικά, με τη χρήση των έως τώρα μοντέλων επιτρέπεται α) δημιουργία ενός δυναμικού διαδραστικού λεξικού που είναι επεκτάσιμο και συγκροτείται από ποικιλία αντικείμενων (γηγενώς ή μη υποστηριζόμενων από τα συμβατικά toolkits), κατά κύριο λόγο με

την επιστράτευση του WSL και ροής ενσωμάτωσης που περιγράφηκε, β) χρήση του υποκείμενου αυτού λεξικού στα πλαίσια προσδιορισμού μιας διεπαφής διαμέσου του CUI μοντέλου, και γ) Pre-instantiation απόδοση και καθορισμός τιμών, σε επίπεδο κάθε cui αντικειμένου μιας διεπαφής όσον αφορά τόσο τις κοινές (αφηρημένων) όσο και τις τυχόν πολυμορφικές ιδιότητες που υποστηρίζονται.

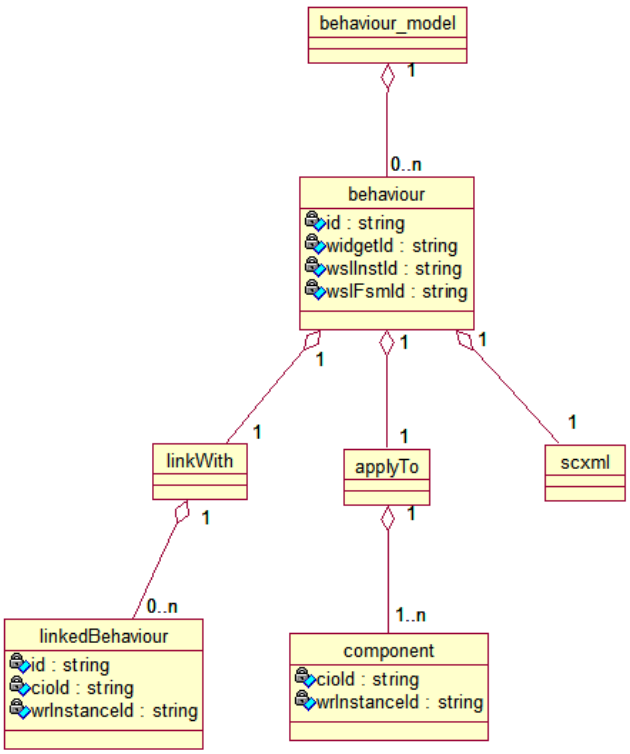
Μοντελοποίηση των δυναμικών πτυχών της διεπαφής: Το μοντέλο 'Behaviour'

Για τη μοντελοποίηση των δυναμικών πτυχών (dynamic aspects) της διεπαφής χρησιμοποιήθηκε η SCXML[14], μια γενικού σκοπού οδηγούμενη από γεγονότα (event-driven) γλώσσα μηχανών καταστάσεων (Finite State Machine). Η εν λόγω γλώσσα είναι προτυποποιημένη από τον οργανισμό W3C, είναι αρκετά ώριμη, υποστηρίζει ορισμό εμφολευμένων καταστάσεων, ενώ υπάρχει ευρεία υποστήριξη από εργαλεία και χρησιμοποιείται σε πλήθος περιπτώσεων. Ο λόγος που χρειάστηκε να υιοθετηθεί οφείλεται στο γεγονός πως η UsiXML δε μεριμνά όσον αφορά το εν λόγω πεδίο, εκτός της υποστήριξης της μετάβασης μεταξύ διαφορετικών οθονών με το πάτημα ενός κουμπιού και των απευθείας κλήσεων συναρτήσεων του λεγόμενου domain model.
Η ροή με βάση την οποία υποστηρίζεται η μοντελοποίηση και ο ορισμός (specification) της δυναμικής συμπεριφοράς κάθε widget είναι ως εξής:

1) Κατά την περιγραφή ενός υπολογιστικού σχήματος στιγμιοτυποποίησης (implementation-specific instantiation scheme), πρέπει να συνταχθεί ένα FSM αρχείο συμβατό με την SCXML, αν όχι ορίζεται αυτόματα ένα default, στο οποίο κωδικοποιείται το σύνολο των πιθανών καταστάσεων (states) και μεταβάσεων (transitions) μεταξύ τους που μπορεί να υποστηριχθούν. Αυτό συντάσσεται αλληλεπιδραστικά με τη βοήθεια ενδεδειγμένων γραφικών εργαλείων υψηλού επιπέδου που παρέχονται ώστε να διευκολυνθεί η διαδικασία. Για κάθε εναλλακτικό σχήμα στιγμιοτυποποίησης συντάσσεται ένα ξεχωριστό scxml file.
2) Το εν λόγω scxml αρχείο συσχετίζεται με μια custom υπολογιστική κλάση (π.χ. java class), η οποία αναλαμβάνει να κάνει grab (μέσω συμβατικών implementation-specific listeners) τα implementation specific events και να τα μεταφράσει στα scxml transitions.

[14] http://www.w3.org/TR/scxml/

3) Ορίζεται μια ‘εγγραφή’ στο WSL που σχετίζεται με το instantiation-scheme στο οποίο αναφερόμαστε, ορίζοντας μια αναφορά προς τα δύο προηγούμενα resources.
4) Κατά τη φάση σχεδίασης μια διεπαφής, αν ο designer επιλέξει να κάνει bind ένα action σε ένα συγκεκριμένο transition ή σε ένα state-specific action (‘onEnter’, ‘onLeave’), τότε αυτό αποθηκεύεται διαμέσου του ‘behavior’ μοντέλου (βλ. Εικόνα 20). Στην πράξη η εγγραφή που θα δημιουργηθεί (‘behaviour’) θα ορίζει ένα link μεταξύ του ‘widgetArchiveId’ (ώστε να ξέρουμε σε ποιο αφηρημένο widget αναφερόμαστε), του σχήματος στιγμιοτυποποίησης αυτού στο οποίο αναφέρεται. Στα πλαίσια αυτής της εγγραφής μπορεί αν προστεθούν εντολές μιας οικείας γλώσσας πεδίου η οποία παρέχει ένα σημαντικό εύρος εντολών για τη διαχείριση πληθώρας παραμέτρων της διεπαφής και του συστήματος γενικότερα.



Εικόνα 20: Δομή του μοντέλου ‘Behaviour’

Επίσης παρέχεται η δυνατότητα συσχέτισης (linking) FSMs που αφορούν διαφορετικά widget archives (βλ. linkWith κλάση). Αυτό αποτελεί ήσσονος σημασίας χαρακτηριστικό καθώς προσφέρει τη δυνατότητα όταν ένας υποδοχέας αλλάξει κατάσταση (μετά από ρητή εντολή του χρήστη) να είναι εφικτό να πυροδοτηθούν αλλαγές και στην κατάσταση των εσωκλειόμενων

αντικειμένων (contained objects). Παραδείγματος χάριν, όταν αλλάζει η κατάσταση ενός score από ‘single-user’ σε ‘collaborative’, τότε πρέπει να αλλάξει και η κατάσταση των νοτών (από ‘single user’ σε ‘collaborative’), μιας και οι ενέργειες οι οποίες μπορεί να επιδεχθούν - υποστηρίξουν διαφέρουν (social awareness, ενδιάμεση διάδραση, group-awareness, κοκ.).

Μοντέλο ‘Abstraction’

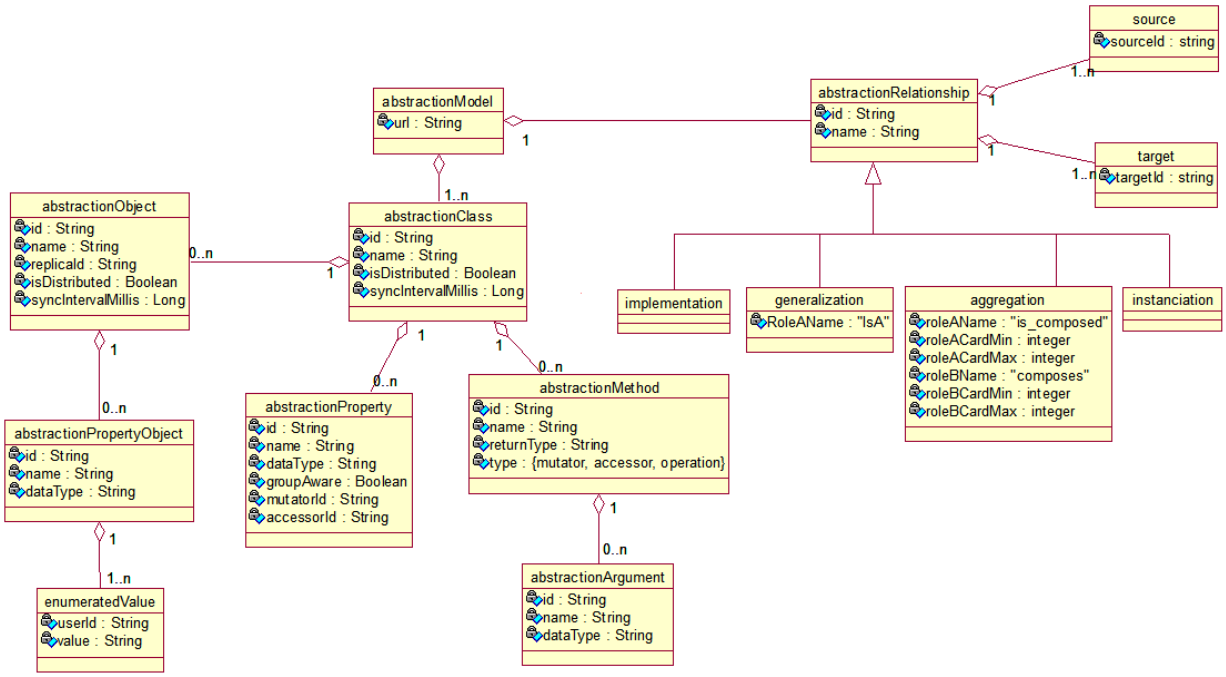
Κύριος ρόλος του εν λόγω μοντέλου είναι να υποστηρίξει το διαμοιρασμό δεδομένων (data-sharing) μεταξύ των παραγόμενων διεπαφών. Συγκεκριμένα παρέχει τη δυνατότητα ορισμού ενός κοινόχρηστου μοντέλου (κωδικοποιημένο xml), το οποίο ορίζει τα κοινά semantics πάνω στα οποία γίνονται bind και μπορεί να συγχρονίζονται σχήματα στιγμιοτυποποίησης του ίδιου ή διαφορετικών widgetArchives.

Μιας και τα σχήματα στιγμιοτυποποίησης που είναι να συγχρονιστούν κάθε φορά μπορεί να διαφέρουν ριζικά μεταξύ τους (σε σημασιολογικό, λεκτικό, συντακτικό, φυσικό επίπεδο), βλ. π.χ. τις νότες του σεναρίου αναφοράς, εναπόκειται κάθε φορά στο σχεδιαστή να βρει και να ορίσει μόνο τα κοινά χαρακτηριστικά τα οποία απαιτούνται και με βάση τα οποία έχει νόημα και μπορεί να επιτευχθεί συγχρονισμός.

Στην πράξη ένα στιγμιότυπο του abstraction μοντέλου αποτελείται από ορισμούς κλάσεων (‘abstractionClass’) και στιγμιοτύπων αυτών (‘abstractionObject’), όπως αναλόγως θα μοντελοποιούνταν και αναπαρίστατο πληροφορία με βάσει το αντικειμενοστραφές μοντέλο προγραμματισμού. Η δομή των κλάσεων και των στιγμιοτύπων που προκύπτουν από αυτές, γίνεται προοδευτικά κατά τη φάση της σχεδίασης χρησιμοποιώντας ενδεδειγμένα αλληλεπιδραστικά εργαλεία. Η δομή των παραγόμενων πληροφοριών κωδικοποιείται σε κάθε περίπτωση ως xml με βάση το υψηλού επιπέδου σχήμα που αποτυπώνεται στην Εικόνα 21. Η λογική αυτή προσιδιάζει αυτή του UML-G [46][47], μόνο που στην εν λόγω περίπτωση το κοινόχρηστο μοντέλο συνδέεται με μια ενοποιημένη μεθοδολογία ανάπτυξης που παρέχει υποστήριξη περιγραφής της διεπαφής (πολυμορφικής) όπως επίσης εργαλείων μόνο για την υποστήριξη όχι μόνο της σχεδιαστικής αλλά και της εν εκτελέσει διαχείρισης της διεπαφής.

Ο διαμοιρασμός των κοινόχρηστων κλάσεων γίνεται μέσω ενός υβριδικού σχήματος κατά το οποίο το κοινόχρηστο μοντέλο αντιγράφεται (replication) σε κάθε χρήστη ενώ ο συγχρονισμός, ακολουθώντας το μοτίβο του NotificationServer [83], γίνεται διαμέσου ενός κεντρικοποιημένου server-side component (SSF) το οποίο εξυπηρετεί τη σειριοποίηση (serialization) των γεγονότων

αλλαγής του κοινόχρηστού μοντέλου επιλύοντας έτσι τη διαχείριση θεμάτων που αφορούν τον έλεγχο ταυτοχρονισμού (concurrency control, βλ. Κεφ.2).



Εικόνα 21: Δομή του μοντέλου 'Abstraction'

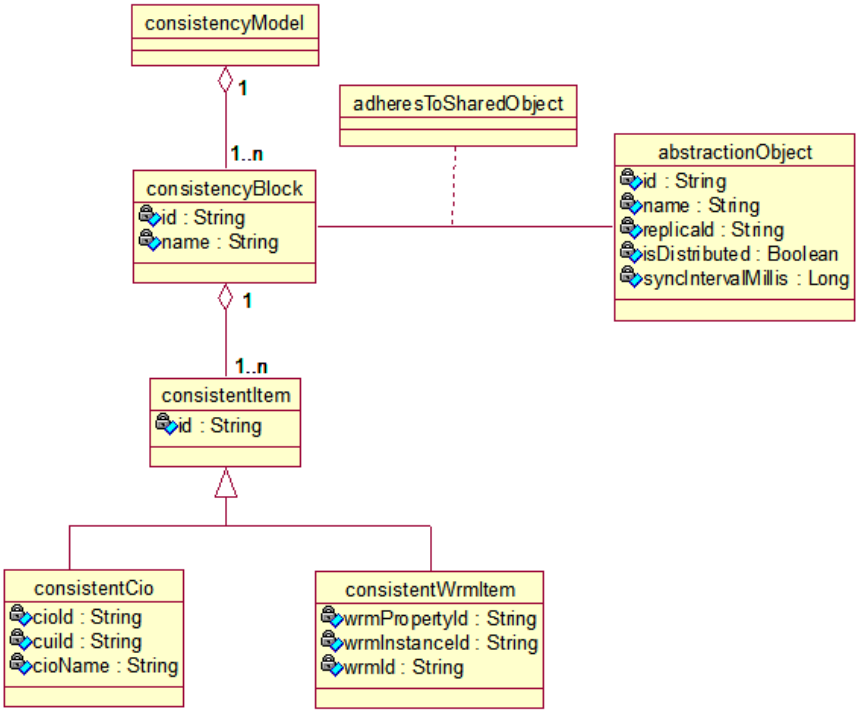
Ενημέρωση του κοινόχρηστου μοντέλου (Shared model update)

Για να καταχωρηθεί τοπικά ένα αίτημα αλλαγής ενός κοινόχρηστου αντικειμένου ('Shared Object'), χρησιμοποιείται το API της οικείας γλώσσας πεδίου που έχει αναπτυχθεί και υποστηρίζεται. Η κλήση αυτή μπορεί αν καταχωρηθεί στα πλαίσια αλλαγής στην κατάσταση ενός αντικειμένου (state change) ή κατά τη μετάβαση (Transition) από τη μια κατάσταση σε μια άλλη. Η πληροφορία αυτή αποθηκεύεται διαμέσου του 'behavior' μοντέλου. Να σημειωθεί ότι όπως σε κάθε αρχιτεκτονική αντιγράφων, για την επικοινωνία αλλαγών του κοινόχρηστου μοντέλου χρειάζεται η αναφορά στο 'replicaId' το οποίο είναι απαραίτητο σε κάθε DSL κλήση καταχώρησης αλλαγής στο διαμοιραζόμενο μοντέλο.

Υποστήριξη Κατανεμημένης διάδρασης (Feedthrough) – Το μοντέλο 'Consistency'

Με το που καταχωρείται μια αλλαγή στο κοινόχρηστο μοντέλο αυτή επικοινωνείται αυτόματα διαμέσου του τοπικού περιβάλλοντος εκτέλεσης, στο ενδεδειγμένο server-side component, το οποίο με τη σειρά του ενημερώνει αυτόματα (μέσω τακτικής relay) όλους τους συνδεδεμένους clients. Αυτή είναι μια διαδικασία που αποκρύπτεται σε κάθε περίπτωση ως ροή κατά το

σχεδιασμό του κοινόχρηστου μοντέλου το οποίο θεωρείται σε κάθε περίπτωση ως αυτόματα συγχρονισμένο από το σύστημα εκτέλεσης. Ωστόσο, αυτό που πρέπει ο σχεδιαστής να ορίσει είναι το πώς μια διεπαφή αποκρίνεται στις αλλαγές των τιμών των ιδιοτήτων των διαμοιραζόμενων αντικειμένων (shared objects) ως αποτέλεσμα της αλληλεπίδρασης ενός άλλου χρήστη με αυτό (feedthrough). Αυτό υλοποιείται με τη βοήθεια του consistency model στα πλαίσια του οποίου ορίζονται συσχετίσεις μεταξύ των ιδιοτήτων των abstractionObjects και των ιδιοτήτων των πολυμορφικών σχημάτων. Το περιβάλλον εκτέλεσης με το που ανιχνεύσει ένα αίτημα αλλαγής του μοντέλου, ως αποτέλεσμα της αλληλεπίδρασης ενός άλλου χρήστη με τον κοινόχρηστο χώρο, ανιχνεύει διαμέσου του consistency μοντέλου την ιδιότητα του πολυμορφικού στιγμιοτύπου που έχει γίνει bind σε αυτή και συμβουλευόμενο το WSL του στιγμιοτύπου εκτελεί το ανάλογο API call το οποίο και το ενημερώνει.



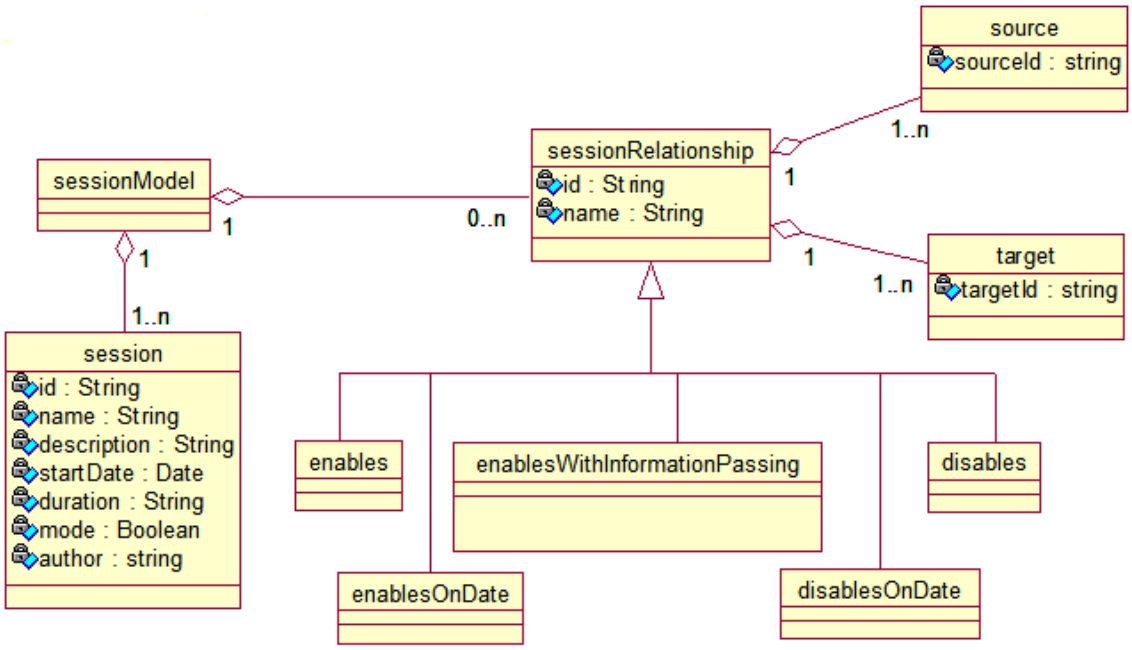
Εικόνα 22: Δομή του μοντέλου 'Consistency'

Στην Εικόνα 22 παρουσιάζεται η βασική δομή του consistency μοντέλου.

Μοντέλο 'Session'

Το εν λόγω μοντέλο αφορά τον ορισμό σύγχρονων συνόδων. Οι σύνοδοι μπορεί να είναι ολικά, μερικά ή και καθόλου επικαλυπτόμενοι χρονικά. Κάθε σύνοδος σχετίζεται με ένα συνολικό

μοντέλο (UI Μοντέλο) που περιλαμβάνει στιγμιότυπα όλων των προηγουμένων που αναφέραμε και που ορίζουν διάφορες πλευρές τις πολυχρηστικής διεπαφής. Το εν λόγω μοντέλο τροφοδοτείται (γίνεται deployed) με αυτοματοποιημένο τρόπο (διαμέσου του ολοκληρωμένου περιβάλλοντος ανάπτυξης που παρέχεται) στο server-side framework (το οποίο περιγράφεται στη συνέχεια).



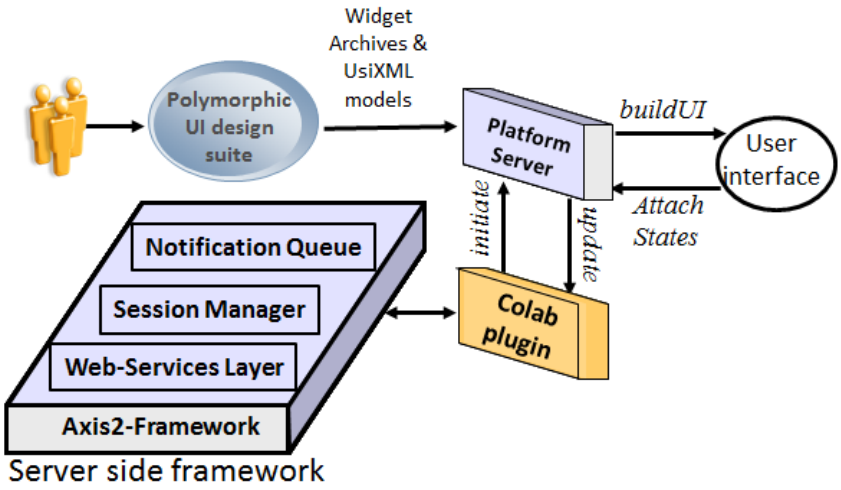
Εικόνα 23: Δομή του μοντέλου 'session'

Η σύνδεση σε μια σύνοδο γίνεται επί της παρούσης με την αποστολή προσκλήσεων (invitations) όπως στην περίπτωση του GroupKit [20].

Το Περιβάλλον Εκτέλεσης (The Runtime Environment)

Για να υποστηρίξουμε τα καινοφανή χαρακτηριστικά που περιγράφηκαν στις προηγούμενες ενότητες, αναπτύχθηκαν πολλαπλού σκοπού εξειδικευμένα αντικείμενα λογισμικού τόσο όσον αφορά το client-side όσο και το server-side κομμάτι. Μια υψηλού επιπέδου εικόνα του παρουσιάζεται στην παρακάτω εικόνα. Αυτό που πρέπει να ειπωθεί σε αυτό το σημείο είναι πως με το που συντάσσονται όλα τα μοντέλα και φτάνουν στην τελική τους μορφή, με τη βοήθεια του ολοκληρωμένου περιβάλλοντος ανάπτυξης, συμπιέζονται σε ένα κοινό φορμάτ και γίνονται deployed ως project στο server side framework. Αυτή η διαδικασία είναι που γεννά shortcuts τα οποία στην ουσία περιέχουν ένα link το οποίο υποδεικνύει το id του session στο οποίο

αναφέρεται η πρόσκληση. Να σημειωθεί πως στο server side framework εναποτίθενται και όλα τα widget archives τα οποία είναι διαθέσιμα σε κάθε client όποτε ζητηθούν.



Εικόνα 24: Το runtime περιβάλλον

Client-side Components

Η πλευρά του client αποτελείται από μια εξιδεικευμένη runtime υποδομή η οποία ονομάζεται 'Platform Server' (PS) [40]. Ο PS είναι ένα πολύ-λειτουργικό αντικείμενο λογισμικού το οποίο αναλαμβάνει να εξασφαλίσει την απρόσκοπτη, συνεπή και διασυνοριακά εκτεινόμενη ικανότητα να διαχειρίζεται όλες τις αντιστοιχήσεις (mappings) από το αφηρημένο στο τοπικό (device-specific) και αντίστροφα. Για να υποστηριχθούν αυτές οι αντιστοιχήσεις ο PS στην ουσία υλοποιεί ένα εικονικό επίπεδο μεταξύ των μοντέλων της UsiXML και των εγγενώς βιβλιοθηκών της εκάστοτε πλατφόρμας. Ο ρόλος του 'περιορίζεται': α) στην κατανεμημένα μεταφόρτωση κλάσεων (στην περίπτωση διαχείρισης μη εγγενώς στοιχείων), β) στη διαχείριση γεγονότων (ως τμήματος της υποστήριξης τοπικού ή/και κατανεμημένου συγχρονισμού), καθώς επίσης και γ) την εν εκτελέσει μεταγλώττιση και διερμήνευση των UsiXML μοντέλων. Στα πλαίσια του τελευταίου συμπεριλαμβάνονται η διερμήνευση του WSL για την αρχική στιγμιοτυποποίηση της διεπαφής αλλά και της πυροδότησης κάθε φορά της ενδεδειγμένης ροής συναρμολόγησης ώστε κάθε φορά να χρησιμοποιούνται μόνο οι ενδεδειγμένες αλληλεπιδραστικές ενσαρκώσεις.

Server-Side Components

Το server-side Framework (SSF), υλοποιεί αρκετά γενικευμένα αντικείμενα συμπεριλαμβανομένου του session manager, του μηχανισμού ειδοποιήσεων καθώς και ένα web-

services layer μέσω του οποίου δίνεται προγραμματιστική πρόσβαση στα προηγούμενα αλλά και σε πρόσθετες επικουρικές λειτουργίες. Το SSF, χειρίζεται επίσης χαμηλού επιπέδου διαχείριση συνόδων υλοποιημένων στηριζόμενων στο apache axis2 framework, υλοποιώντας πλήθος λειτουργιών όπως δημιουργία, εγγραφή, καταστροφή, κοκ.. Όσον αφορά τα μη εγγενή widgets, το SSF διατηρεί ένα κοινόχρηστο repository με όλα τα διαθέσιμα widget archives διευκολύνοντας την κατανεμημένη μεταφόρτωση κλάσεων.

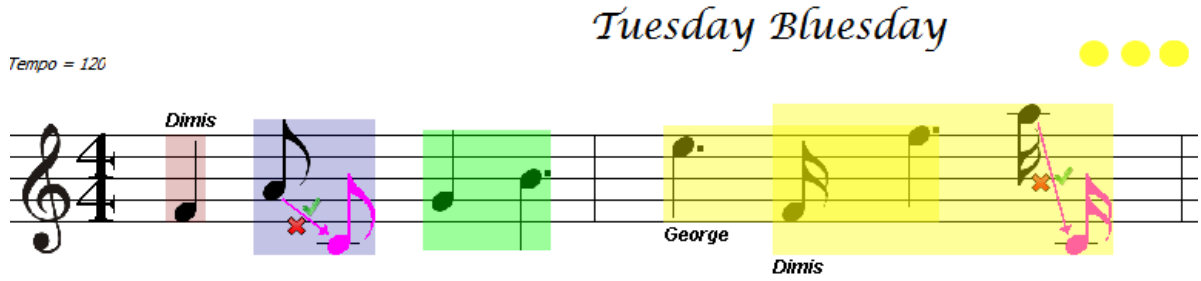
Κεφάλαιο 5 – Επίδειξη εφαρμογής & Σενάριο Χρήσης

Η συγκεκριμένη ενότητα παρουσιάζει την πλήρη ροή που πρέπει να ακολουθηθεί για την υποστήριξη του σεναρίου αναφοράς που σκιαγραφήθηκε στο κεφάλαιο 4 και αφορά το συνεργατικό μάθημα θεωρίας της μουσικής. Για λόγους απλοποίησης της παρουσίασης δεν θα γίνει αναφορά σε επιμέρους ζητήματα που σχετίζονται με τη χρήση βιβλιοθηκών για την ανάπτυξη των widgets που συνθέτουν τις μεταφορές αλληλεπίδρασης που εμπλέκονται. Αντί αυτού, θα τις θεωρήσουμε ως ήδη έτοιμες και θα εστιάσουμε στο τρόπο αξιοποίησης τους από την προτεινόμενη προσέγγιση για την εξυπηρέτηση του στόχου της συνεργατικής εκτέλεσης ενός μαθήματος θεωρίας της μουσικής. Οι βασικές διεπαφές που εμπλέκονται απεικονίζονται στην Εικόνα 25.

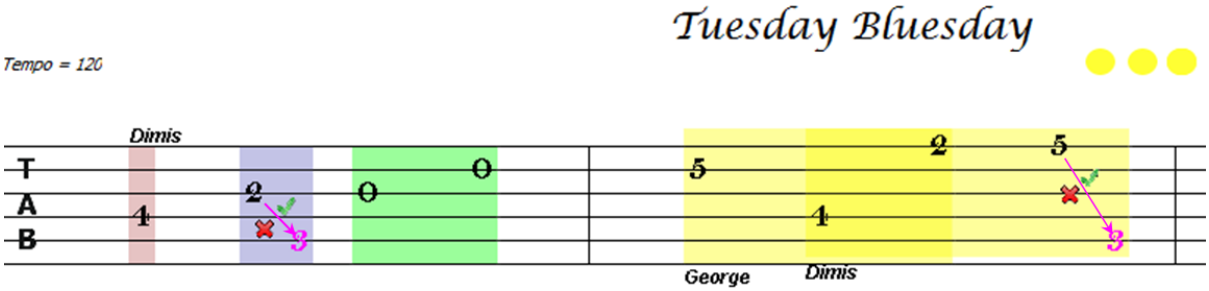
Συνοπτική περιγραφή των αλληλεπιδραστικών αναπαραστάσεων

Η αναπαράσταση του πενταγράμμου (Εικόνα 25 a) έχει υλοποιηθεί σε java/swing και υποθέτουμε ότι είναι ενδεδειγμένη για έμπειρούς μουσικούς - εκπαιδευτές. Το στιγμιότυπο της ταμπλατούρας (Εικόνα 25 b) αποδίδει μια εναλλακτική αναπαράσταση μουσικής για αρχάριους χρήστες. Έχει και αυτή υλοποιηθεί σε java/swing και αποτελείται στο σύνολό της από σειρά εξειδικευμένων μη εγγενώς υποστηριζόμενων από την κύρια βιβλιοθήκη αντικειμένων. Ο λόγος που ενδείκνυται για αρχάριους με τη μουσική χρήστες αφορά στο γεγονός ότι εστιάζει στη καθοδήγηση του χρήστη αναφορικά με την ενδεδειγμένη θέση των δαχτύλων στο τάστο, αγνοώντας τη χρονική διάσταση, κάτι που απλοποιεί κατά πολύ τη συνολική πολυπλοκότητα. Η τρίτη αναπαράσταση αποδίδει τη μουσική υπό μορφή κυκλικών μέτρων (Εικόνα 25 c) και υλοποιήθηκε με τη χρήση HTML5 τεχνολογιών και advanced JavaScript. Η εν λόγω μεταφορά εστιάζει στην ανάδειξη της παραμέτρου του χρόνο στην μουσική. Συγκεκριμένα, όσο μεγαλύτερο το μήκος του τόξου, τόσο περισσότερο θα πρέπει η νότα να παραμείνει πατημένη. Η απόσταση της νότας από το κέντρο των αξόνων φανερώνει τη σειρά της χορδής που πρέπει να παραμείνει πατημένη. Όσον αφορά τους διαφορετικούς χρωματισμούς, αυτοί χρησιμοποιούνται για να αποδώσουν τα ακόλουθα διαδραστικά καθήκοντα: (α) Λαχανί χρωματισμός αποδίδει ανάδειξη (highlighting) από μεμονωμένο χρήστη (single user) εύρους νοτών (β) Μωβ χρωματισμός αποδίδει επιλογή μιας νότας από μεμονωμένο χρήστη (single user) (γ) Βυσσινί

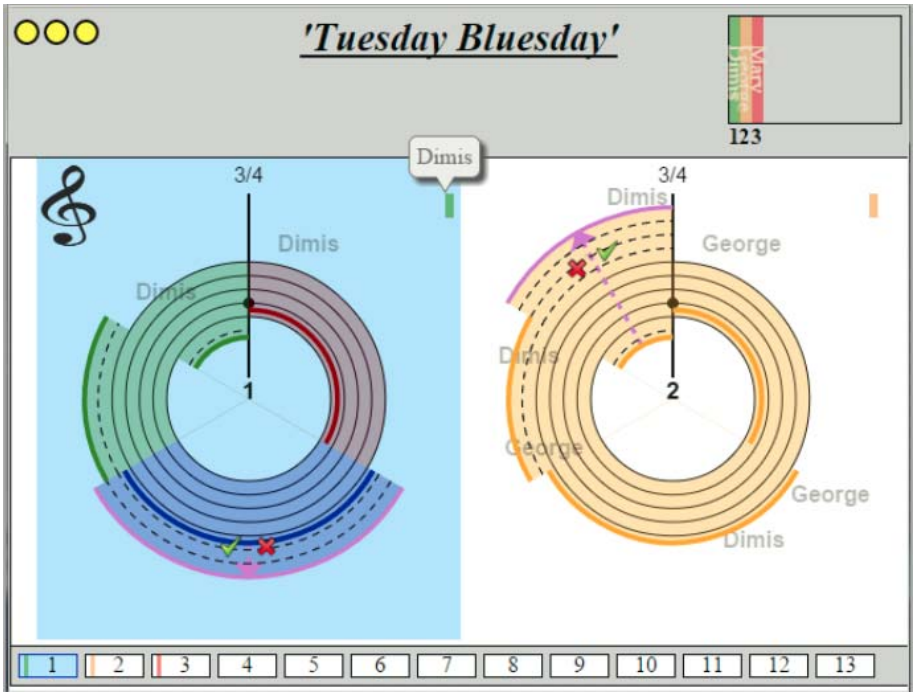
απόχρωση αποδίδει την πολυχρηστική επιλογή (multi-user selection) μιας νότας και (δ) Κίτρινη απόχρωση αποδίδει την πολυχρηστική επιλογή εύρους νοτών (multi-user highlighting).



(α) Η αλληλεπιδραστική μεταφορά του πενταγράμμου ('PC' / 'Εμπειροι χρήστες')



(β) Η αλληλεπιδραστική μεταφορά της ταμπλατούρας ('PC' / 'Αρχάριοι χρήστες')



(γ) Η αλληλεπιδραστική μεταφορά του πενταγράμμου ('Web' / 'Αρχάριοι χρήστες')

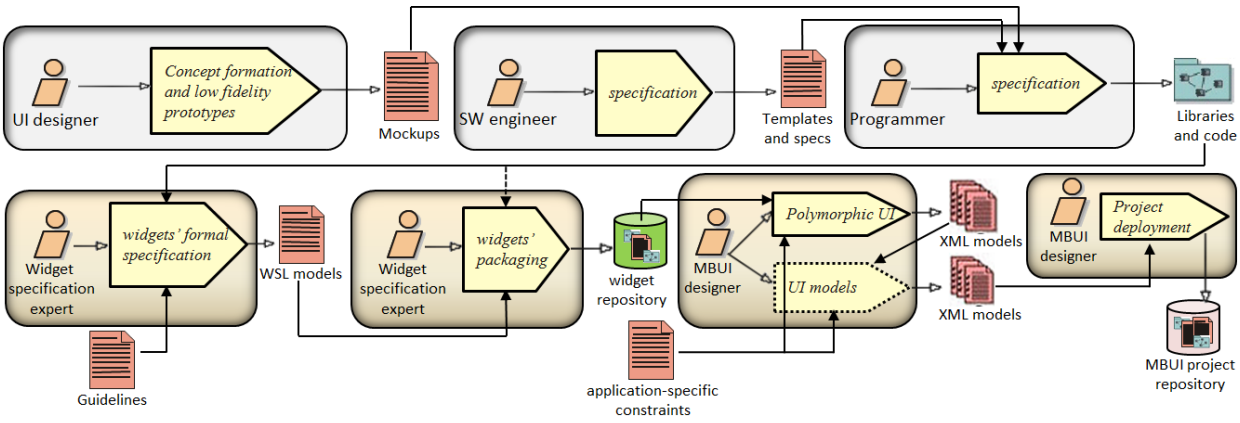
Εικόνα 25: Διαφορετικές ενσαρκώσεις της πολυμορφικής διεπαφής

Ενέργειες που διεκπεραιώνονται από μεμονωμένους χρήστες (single-user) είναι τοπικής εμβέλειας, δεν προωθούνται στους λοιπούς εμπλεκόμενους στη συνεδρία και εξυπηρετούν στόχους του μεμονωμένου / τοπικού χρήστη. Αντίθετα, ενέργειες που είναι πολύ-χρηστικές (multi-user) προωθούνται σε όλους φέροντας ανά περίπτωση και το όνομα του χρήστη που την προκάλεσε. Με αυτό τον τρόπο υποστηρίζεται ομαδική ενημερότητα (group-awareness) μεταξύ των χρηστών επιτρέποντας τον καλύτερο συντονισμό των δράσεών τους.

Στο σημείο αυτό είναι σημαντικό να υπερθεματίσουμε το βαθμό ετερογένειας μεταξύ των διεπαφών οι οποίες παρά τις ασυνέπειές τους, σε όλα τα επίπεδα (συντακτικό, σημασιολογικό, λεκτικό και φυσικό), πρέπει να παραμένουν συγχρονισμένες κατά τη διάρκεια μιας σύγχρονης συνεδρίας. Επίσης, για την περίπτωση των νοτών άξιο αναφοράς είναι το γεγονός ότι μπορούν να υποστηρίξουν μια εξελιγμένη μορφή διάδρασης, μέσω της απόδοσης ενδιάμεσης ανάδρασης (interim-feedback) στην οποία υπεισέρχεται μια νότα κατά τη διάρκεια σύγχρονης συνεργατικής διαπραγμάτευσης των παραμέτρων της. Σε αυτή την περίπτωση οι αλλαγές στο βασικό μοντέλο της νότας κατοχυρώνονται μόνο σε περίπτωση που πλειοψηφήσει η αποδοχή της αλλαγής. Σε αντίθετη περίπτωση, τα δεδομένα του μοντέλου παραμένουν ως έχουν και αυτά του ενδιαμέσου μοντέλου απορρίπτονται.

Ροή εργασίας υποστήριξης σεναρίου χρήσης

Είναι προφανές ότι κάθε μία από τις παραπάνω διεπαφές μπορούν να αναπτυχθούν με συμβατικά εργαλεία και μεθόδους, να ενσωματωθούν σε κλασικά μοντέλα κύκλου ζωής λογισμικού και να αποδοθούν ως εναλλακτικές αναπαραστάσεις μουσικής σε διαφορετικές ή ακόμη και στην ίδια εφαρμογή. Ωστόσο, αυτή η προσέγγιση δεν προσεγγίζει το στόχο της σύγχρονης αξιοποίησης τους από διαφορετικούς χρήστες στο πλαίσιο του επιλεγμένου σεναρίου αναφοράς. Η προσέγγιση που περιγράψαμε σε προηγούμενα κεφάλαια ανταποκρίνεται στις προηγμένες απαιτήσεις του σεναρίου μας μέσω της ροής εργασίας που συνοψίζεται στο παρακάτω σχήμα. Η σημειογραφία που επιλέγεται είναι αυτή της Rationale Unified Process (RUP) προκειμένου να αναδείξουμε τους ρόλους που εμπλέκονται, τις δραστηριότητες που εκτελούνται, τα τεχνουργήματα που αξιοποιούνται ή παράγονται καθώς και την μεταξύ τους αλληλουχία.



Εικόνα 26: Ενδεδειγμένη ροή εργασίας για την υποστήριξη του σεναρίου χρήσης

Ειδικότερα, αναγνωρίζονται συνολικά πέντε ρόλοι:

1) *UI Designer*: μέσω χαμηλής συνήθως πιστότητας πρωτοτύπων επικοινωνεί τις βασικές ιδέες της σχεδίασης
2) *SW engineer*: συλλέγει τις προδιαγραφές του συστήματος τις οποίες τεκμηριώνει, κωδικοποιεί και επικοινωνεί στους προγραμματιστές.
3) *(Toolkit-level) Programmer*: αναπτύσσουν λογισμικό επενδύοντας σε χαμηλού επιπέδου (toolkit-level) προγραμματιστικά εντατικές προσπάθειες για την ανάπτυξη του λογισμικού με βάση τις προδιαγραφές του συστήματος.
4) *Widget Specification Expert*: κωδικοποιεί το αποτέλεσμα της προγραμματιστικής διαδικασίας (API αλληλεπιδραστικών στοιχείων), αν και εφόσον αυτό χρειάζεται συντάσσοντας ή επαυξάνοντας ένα υπάρχων WSL, το οποίο μαζί με τυχόν λοιπούς πόρους (π.χ.: scxml FSMs, βλ. Εικόνα 35) πακετάρεται σε ένα widget archive το οποίο και κάνει deploy στο κοινό repository της πλατφόρμας ώστε να είναι διαθέσιμο για μελλοντική χρήση
5) *MBUI Designer*: Συντάσσει βαθμιδωτά διαμέσου ενδεδειγμένων εργαλείων το σύνολο των μοντέλων που χρειάζονται κάθε φορά για να υλοποιηθεί το επιθυμητό αποτέλεσμα. Υπό αυτό το πρίσμα συντάσσει την πολυμορφική διεπαφή (επιλέγοντας μεταξύ των διαθέσιμων εναλλακτικών αναπαραστάσεων που θέλει να γίνουν engaged) και τέλος κάνει deploy το project στο SSF ώστε να είναι προσβάσιμο από όλους τους κατανεμημένους χρήστες.

Αναφορικά με τις δραστηριότητες που εκτελούνται (πεντάγωνα σχήματα), αυτές περιλαμβάνουν:

1) *Concept formation & low-fidelity prototypes*: Ανάπτυξη πρωτοτύπων χαμηλής πιστότητας που στόχο έχουν να άγουν πληροφορία σχετικά με την επιθυμητή δομή και λειτουργικότητα της διεπαφής.
2) *Specification*: Συλλογή και κωδικοποιημένη καταγραφή διαφόρων οπτικών του συστήματος για την επικοινωνία πληροφορίας προς τους προγραμματιστές, οι οποίοι εν συνεχεία
3) *Specification*: Αναπτύσσουν με βάση τις προδιαγραφές αυτές αντικείμενα λογισμικού για την υποστήριξη της αλληλεπιδραστικής συμπεριφοράς, υπό τη μορφή πηγαίου κώδικα και βιβλιοθηκών.
4) *Widget's formal specification*: Η διαδικασία περιγραφής των widgets, είναι το πρώτο βήμα της επέκτασης του υποκείμενου διαδραστικού λεξικού της γλώσσας. Στόχο έχει την ενσωμάτωση (integration) νέων διαδραστικών αντικειμένων ή την επαύξηση υπαρχόντων σχημάτων στιγμιοτυποποίησης για την εξυπηρέτηση των αναγκών σεναρίων και την περαιτέρω 'εξειδίκευση' του λεξικού.
5) *Widgets' packaging:* Η εν λόγω δραστηριότητα συνιστά το τελευταίο βήμα της εισαγωγής ενός νέου στοιχείου στη γλώσσα. Συγκεκριμένα συνιστά εκείνη τη διαδικασία κατά τη οποία ένα στιγμιότυπο του WSL, συμπιέζεται με λοιπούς πόρους από τους οποίους εξαρτάται η λειτουργία του, σε ένα κοινό format (zip) το οποίο το καθιστά διαχειρίσιμο από τα οικεία εργαλεία (σχεδιαστικά και runtime).
6) *Polymorphic UI*: Σε αυτό το πλαίσιο με βάση τις προδιαγραφές του συστήματος, καθορίζεται η αφηρημένη πολυμορφική δομή της διεπαφής και το εύρος της προσαρμοστικής συμπεριφοράς.
7) *UI models*: Παράλληλα με τη δραστηριότητα της πολυμορφικής περιγραφής της διεπαφής, εκκινείται η διαδικασία κατά την οποία προσδιορίζονται (αλληλεπιδραστικά και βαθμιδωτά), λοιπές προδιαγραφές (μοντέλα) του συστήματος που αφορούν π.χ. δυναμικές πτυχές της διεπαφής, το διαμοιραζόμενο μοντέλο (shared model), κοκ.
8) *Project deployment*: Τελευταίο βήμα, αποτελεί το deployment του Project στο SSF, οπότε και δημιουργούνται οι συνεργατικές προσκλήσεις.

Τα τεχνουργήματα, περιλαμβάνουν σκαριφήματα διεπαφών (mockups), πρότυπα (templates) και προδιαγραφές, υλοποιημένες βιβλιοθήκες, μοντέλα διαφόρων μορφών και αποθετήρια

κωδικοποιημένης πληροφορίας. Σε αυτά ωστόσο θα αναφερθούμε αναλυτικότερα στην επόμενη ενότητα.

Πολυμορφική κωδικοποίηση των εναλλακτικών αλληλεπιδραστικών σχημάτων στιγμιοτυποποίησης

Η προτεινόμενη μέθοδος διακρίνεται μεταξύ άλλων από τους ρόλους του Widget Specification Expert και του MBUI designer. Παρότι οι ρόλοι αυτοί είναι γενικής φύσεως, η κατεξοχήν ενεργοποίηση τους πραγματοποιείται όταν έχουν ολοκληρώσει οι συμβατικοί ρόλοι (UI designer, SW engineer, Programmer) και έχουν υλοποιηθεί τα συστατικά των βιβλιοθηκών που υλοποιούν τις αλληλεπιδραστικές μεταφορές που φαίνονται στην Εικόνα 25. Για τις ανάγκες του σεναρίου μας θα αναφερθούμε πρώτα στο ρόλο του Widget Specification Expert και στη συνέχεια σε αυτόν του MBUI expert που είναι άμεσα εμπλεκόμενοι.

			Property Name	Property Type
Abstract 'Music Note'	Abstract Properties		index	int
			step	char {A, B, C, D, E, F, G}
			octave	int
			accidental	String {Double_flat, Flat, Natural, Sharp, Double_sharp}
	Polymorphic Properties	Sheet Note (*java/swing*)	time	String {Whole, Half, Quarter, Eighth, Sixteenth, Thirty_second, Sixty_fourth}
			dotted	boolean
			interimStep	char {A, B, C, D, E, F, G}
			interimOctave	int
			interimAccidental	String {Double_flat, Flat, Natural, Sharp, Double_sharp}
			interimTime	String {Whole, Half, Quarter, Eighth, Sixteenth, Thirty_second, Sixty_fourth}
			interimDotted	boolean
		Tablature Note (*java/swing*)	interimStep	char {A, B, C, D, E, F, G}
			interimOctave	int
			interimAccidental	String {Double_flat, Flat, Natural, Sharp, Double_sharp}
		Circular Note (*web/prototype*)	time	String {Whole, Half, Quarter, Eighth, Sixteenth, Thirty_second, Sixty_fourth}
			dotted	boolean
			interimStep	char {A, B, C, D, E, F, G}
			interimOctave	int
			interimAccidental	String {Double_flat, Flat, Natural, Sharp, Double_sharp}
			interimTime	String {Whole, Half, Quarter, Eighth, Sixteenth, Thirty_second, Sixty_fourth}
			interimDotted	boolean

Πίνακας 1: Πολυμορφική περιγραφή εναλλακτικών σχημάτων στιγμιοτυποποίησης μιας μουσικής νότας

Η πρώτη αρμοδιότητα του Widget Specification Expert είναι να προχωρήσει με βάσει τα υποστηριζόμενα χαρακτηριστικά των επιλεγμένων widgets και στο πλαίσιο των διαθέσιμων υλοποιήσεων να διαχωρίσει τα αφηρημένα (κοινά σε όλες τις υλοποιήσεις) από τα πολυμορφικά χαρακτηριστικά. Για το σενάριο μας η ανάλυση αυτή αφορά τη νότα (βλέπε Πίνακας 1), το 'score' (βλ. Πίνακας 2) και το 'μέτρο'(βλ. Πίνακας 3).

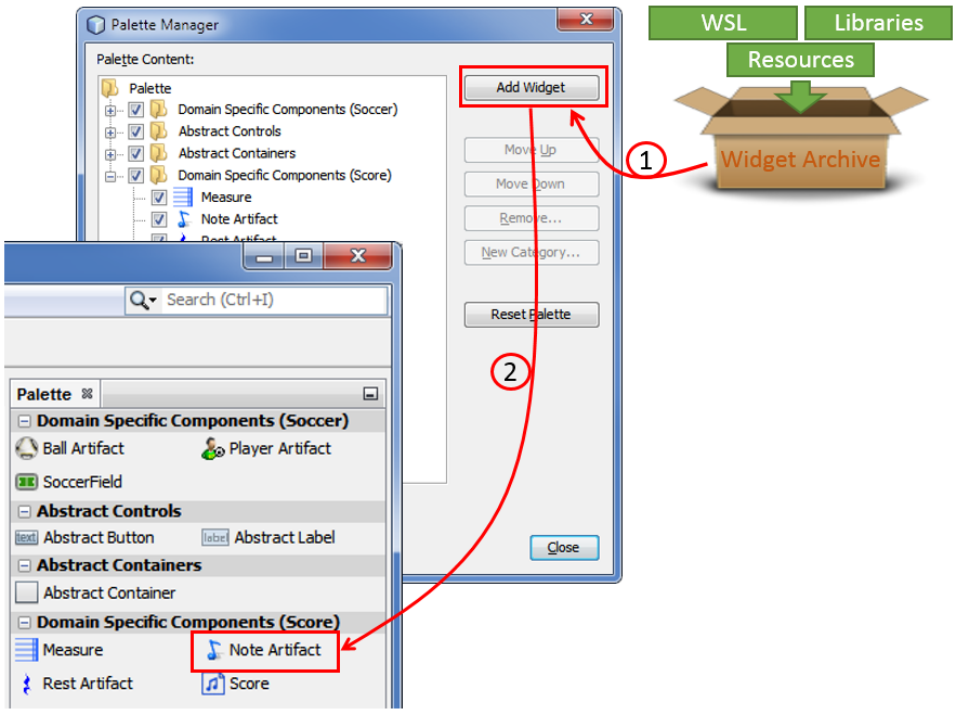
			Property Name	Property Type
Abstract 'Music Score'	Abstract Properties		title	String
			author	String
	Polymorphic Properties	Sheet Score (*java/swing*)	tempo	int
			duration	int
			highlighting	[int, int]
			multiuserHighlighting	complex { String, [int, int] }
			selection	int
			multiuserSelection	complex { String, [int, int] }
		Tablature Score (*java/swing*)	highlighting	[int, int]
			multiuserHighlighting	complex { String, [int, int] }
			selection	int
			multiuserSelection	complex { String, [int, int] }
		Circular Score (*web/prototype*)	tempo	int
			duration	int
			highlighting	[int, int]
			multiuserHighlighting	complex { String, [int, int] }
			selection	int
			multiuserSelection	complex { String, [int, int] }
			setScoreMiniViewEnabled	boolean
			setFilmViewEnabled	boolean
			setMultiuserMeasureSelectionInFilmViewEnabled	boolean

Πίνακας 2: Πολυμορφική περιγραφή ιδιοτήτων του 'MusicScore'

			Property Name	Property Type
Abstract 'Music Measure'	Abstract Properties		index	int
	Polymorphic Properties	Sheet Measure (*java/swing*)	timeNumerator	int
			timeDenominator	int
			clef	char {C, F, G}
			clefLine	int {1-5}
		Tablature Measure (*java/swing*)	totalChords	int
		Circular Measure (*web/prototype*)	timeNumerator	int
			timeDenominator	int
			clef	char {C, F, G}
			clefLine	int {1-5}
			focused	boolean

Πίνακας 3: Πολυμορφική περιγραφή ιδιοτήτων του 'MusicMeasure'

Με γνώμονα τον παραπάνω πίνακα ο Widget Specification Expert συντάσσει το WSL (όντας ευθυγραμμισμένος με το αφηρημένο μοντέλο συσχετίσεων της Εικόνα 15) για το αντικείμενο της νότας (λόγου του αρκετά μεγάλου μεγέθους του δεν είναι δυνατόν να παρατεθεί). Στο WSL κωδικοποιούνται-περιγράφονται όλες οι λεκτικές (lexical), συντακτικές (syntactic) και φυσικές (physical) ιδιότητες όλων των στιγμιοτύπων, δηλ. το σύνολο του API τους. (Οι ίδιες ακριβώς ενέργειες αναλαμβάνονται και για την περίπτωση του 'score' και του 'μέτρου'). Ακολούθως, το παραγόμενο στιγμιότυπο του WSL πακετάρεται μαζί με τις βιβλιοθήκες και με τυχόν άλλους πόρους (π.χ. εικόνες) που κάθε ένα στιγμιότυπο χρειάζεται για να λειτουργήσει απρόσκοπτα, σε ένα συμπιεσμένο αρχείο ('abstract_note.wdr') με κατάληξη 'wdr' (από το WiDgetArchive). Το archive με βάση τη ροή που φαίνεται και στην Εικόνα 27 ενσωματώνεται κι είναι διαθέσιμο στη γλώσσα. Εν συνεχεία ο MBUIdesigner με τη βοήθεια του ολοκληρωμένου περιβάλλοντος [3][40], σχεδιάζει την πολυμορφική διεπαφή του σεναρίου χρήσης (σε επίπεδο CUI), ως μιας ιεραρχία αφηρημένων CUI στοιχείων, το εύρος και οι ιδιότητες των οποίων αντλούνται δυναμικά από τα συνολικά διαθέσιμα widgetArchives που έχουν ενσωματωθεί. Τμήμα του παραγόμενου CUI μοντέλου φαίνεται στην Εικόνα 28.

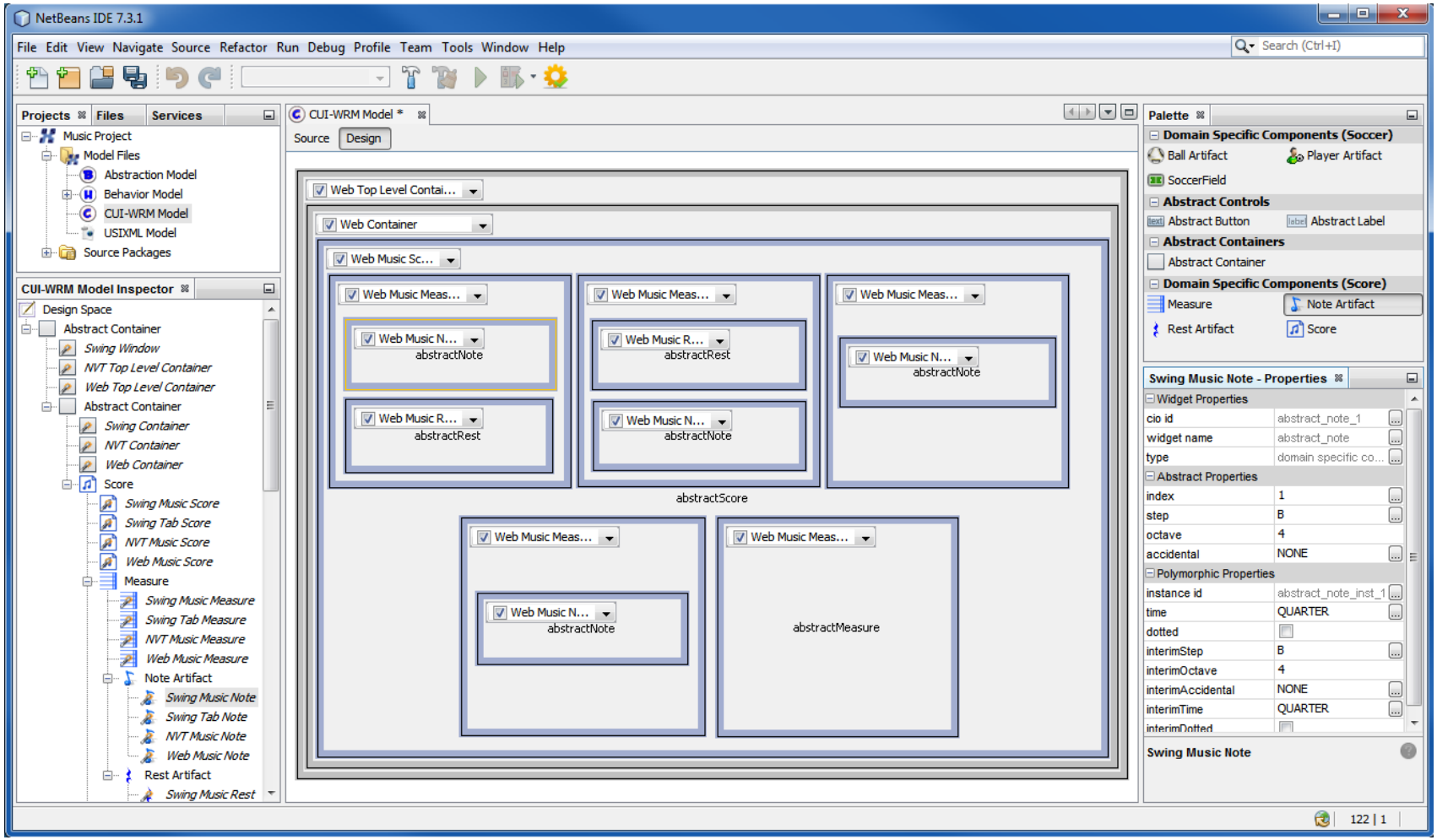


Εικόνα 27: Διαδικασία εισαγωγής αντικειμένου νότας ('Note Artifact') στο εργαλείο

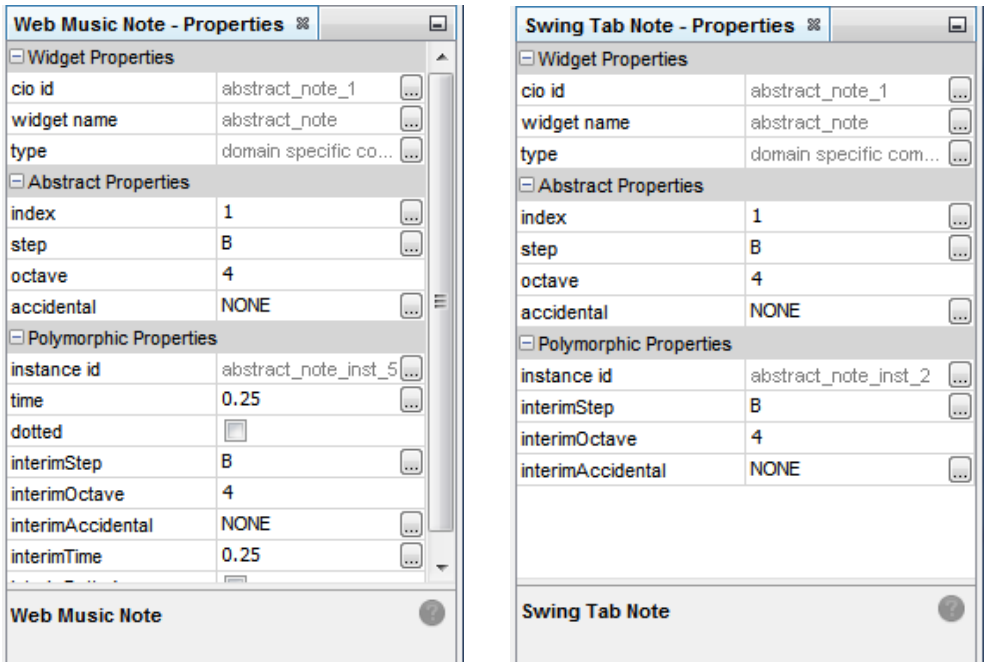
```
<cuiModel>
    ...
    <AbstractContainer id="abstract_container_1" widgetId="abstract_container" >
        <AbstractContainer id="abstract_container_2" widgetId="abstract_container" >
            <AbstractScore id="abstract_score_1" widgetId="abstract_score" >
                <AbstractMeasure id="abstract_measure_1" widgetId="abstract_measure" >
                    <AbstractNote id="abstract_note_1" widgetId="abstract_note"></AbstractNote>
                    <AbstractNote id="abstract_note_2" widgetId="abstract_note"></AbstractNote>
                    ...
                </AbstractMeasure>
                ...
            </AbstractScore>
        </AbstractContainer>
    </AbstractContainer>
</cuiModel>
```

Εικόνα 28: Τμήμα της CUI περιγραφής της διεπαφή του σεναρίου χρήσης της μουσικής

Στην Εικόνα 29 καταγράφεται ένα στιγμιότυπο του ολοκληρωμένου περιβάλλοντος σχεδίασης της πολυμορφικής διεπαφής του σεναρίου αναφοράς. Στο κάτω αριστερό μέρος της οθόνης φαίνεται καλύτερα η ιεραρχική αποσύνθεση των αντικειμένων που συνιστούν την εφαρμογή, καθώς και το εύρος των επιλεγμένων σχημάτων στιγμιοτυποποίησης που έχουν ενεργοποιηθεί στα πλαίσια κάθε CUI αντικειμένου. Τονίζεται πως το συνολικό εύρος των διαθέσιμων διαδραστικών στιγμιοτύπων δεν είναι καρφωμένο (εγγεγραμμένο - inscribed) στη λογική του κώδικα της υλοποίησης ενός widget, αλλά με τον τρόπο που περιγράφεται στην εν λόγω προσέγγιση, εκτίθεται κατά τη φάση σχεδίασης και εναπόκειται στη βούληση του σχεδιαστή και τις ανάγκες του σεναρίου να ενεργοποιήσει τις κατάλληλες αλληλεπιδραστικές ενσαρκώσεις κάθε φορά. Η δυνατότητα ορισμού του εύρους των ενεργοποιημένων αλληλεπιδραστικών σχημάτων δίνεται διαμέσου του ‘combobox’ (πάνω αριστερή γωνία κάθε CUI block) στο οποίο απαριθμούνται όλες οι επιλογές. Στο δεξί μέρος της οθόνης εμφανίζεται ο ‘polymorphic properties editor’, με τη βοήθεια του οποίου παραμετροποιούνται (pre-instantiation configuration) όλα τα πολυμορφικά αντικείμενα. Στην Εικόνα 30 φαίνονται δύο εναλλακτικά στιγμιότυπα (screenshots) του editor, όπου στην αριστερή πλευρά προσδιορίζονται τιμές για το κυκλικό στιγμιότυπο μιας νότας του web ενώ στη δεξιά ενός συγκεκριμένου πολυμορφικού στιγμιοτύπου που αφορά τη νότα μιας ταμπλατούρας. Και στις δύο περιπτώσεις όπως φαίνεται τα αφηρημένα χαρακτηριστικά είναι κοινά (ίδιες τιμές), ενώ στην περίπτωση των πολυμορφικών δεν αλλάζει μόνο ο τύπος αλλά και το εύρος των επιλογών ανά στιγμιότυπο.



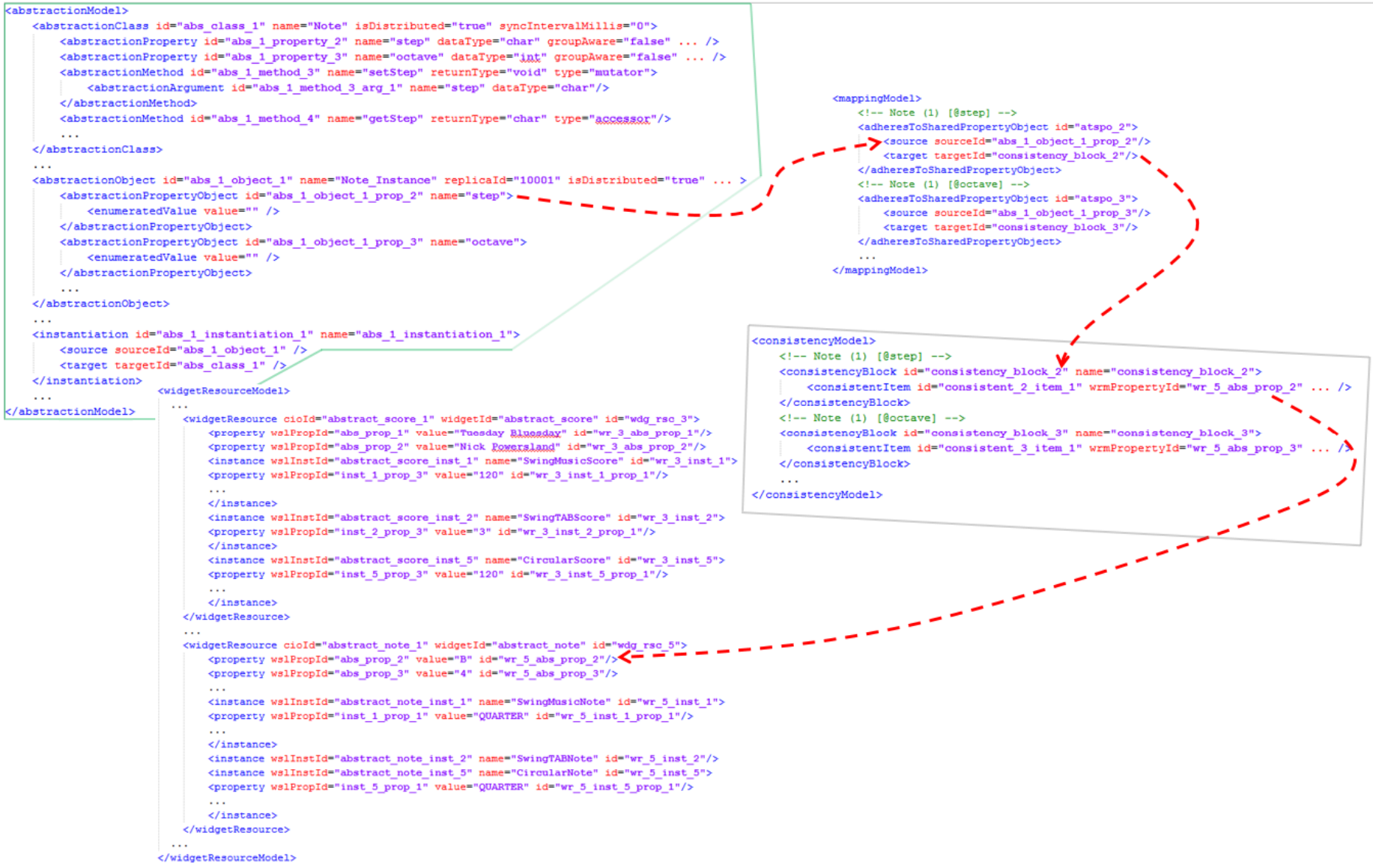
Εικόνα 29: Σχεδιαστική σουίτα για την περιγραφή πολυμορφικής διεπαφής του σεναρίου της μουσικής



Εικόνα 30: Στιγμιότυπα παραμετροποίησης αφηρημένων και πολυμορφικών χαρακτηριστικών μιας νότας (αριστερά: η περίπτωση της κυκλικής νότας, δεξιά της νότας μιας ταμπλατούρας)

Ορισμός του διαμοιραζόμενου μοντέλου (Shared model)

Στη συνέχεια επιλέγονται τα κοινά semantics (π.χ. ‘step’, ‘octave’, κοκ.) με βάσει τα οποία επιθυμείται ο συγχρονισμός μεταξύ των ριζικά ετερογενών διαδραστικών σχημάτων στιγμιοτυποποίησης. Είναι λογικό πως αν οριστούν semantics που δεν υποστηρίζονται από ένα πλυμορφικό σχήμα, τότε αυτό απλά δεν θα μπορεί να συγχρονιστεί με βάση αυτό. Αφού καθοριστούν τα κοινά semantics, συντάσεται αλληλεπιδραστικά ένα στιγμιότυπου του ‘abstraction’ μοντέλου το οποίο κωδικοποιεί όλη αυτή την πληροφορία (βλ. Εικόνα 31, πράσσινο μπλοκ). Στην Εικόνα 31 ενδεικτικά απεικονίζεται τμήμα του abstraction μοντέλου στο οποίο φαίνεται ένα διαμοιραζόμενο αντικειμένο της κλάσης ‘Note’ (αυτό που φέρει replicaId: 10001). Το συγκεκριμένο αντικείμενο μπορεί και προσδιορίζεται μοναδικά κατά μήκος των κατανεμημένων κόμβων διαμέσου του ‘replicaId’ που είναι κοινό σε κάθε περίπτωση για το εν λόγω αντίγραφο. Εν εκτελέσει αλλαγή της τιμής της συγκεκριμένης ιδιότητας του κοινόχρηστου μοντέλου, που αφορά το βήμα μιας νότας (step), μπορεί να πυροδοτηθεί διαμέσου της σχετικής DSL κλήσης η οποία θα έχει οριστεί να εκτελείται ως αποτέλεσμα κάποιου ‘state transition’ εξ αυτών που περιγράφηκαν στο scxml του WSL (βλ. Εικόνα 31) του τρέχοντος πολυμορφικού σχήματος της νότας η οποία πρόκειται να το πυροδοτήσει.

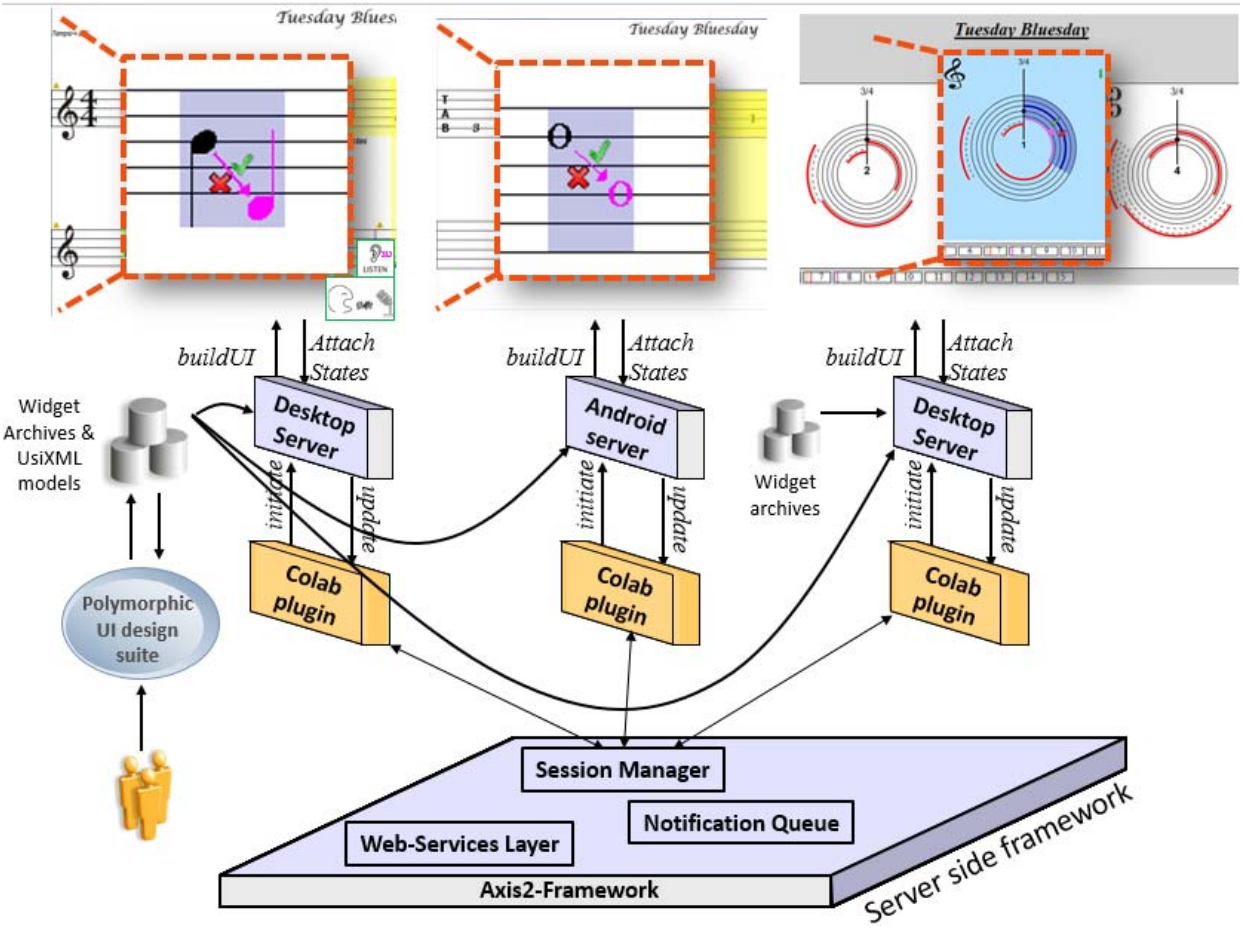


Εικόνα 31: Υποστήριξη Feedthrough

Προσδιορισμός Feedthrough

Για κάθε μια από τις ιδιότητες των κοινόχρηστων αντικειμένων που έχουν δηλωθεί διαμέσου το ‘abstraction model’, θα πρέπει ο MBUIdesigner να ορίσει (με τη βοήθεια του ‘consistency’ μοτνέλου), το ποιες ιδιότητες ποιών πολύμορφικών στιγμιοτύπων θα επηρεάζονται ως αποτέλεσμα της αλλαγής τους από ένα άλλο χρήση. Η ύπαρξη αυτής της συσχέτισης είναι απαραίτητη ώστε να υλοποιηθεί το feedthrough και όλα τα πολυμορφικά στιγμιότυπα τα οποία έχουν γίνει bind στο σημείο αυτό του κοινόχρηστου μοντέλου να παραμείνουν συγχρονισμένα.

Στην Εικόνα 31 απεικονίζεται όλη η ακολουθία βημάτων - ελέγχων που πρέπει να γίνουν ώστε να συναχθεί ποια ιδιότητα του WidgetResource Μοντέλου πρέπει να αλλάξει ως αποτέλεσμα της αλλαγής της τιμής της ιδιότητας ‘step’ του διαμοιραζόμενου αντικειμένου. Με το που καταλήγουμε σε μια ιδιότητα του widgetResourceModel, μπορούμε εύκολα να συνάγουμε διαμέσου των γνωστών συσχετίσεων που ορίζονται μεταξύ των μοντέλων και που περιγράφηκαν στο προηγούμεν κεφάλαιο: α) το ποιο πολυμορφικό στιγμιότυπο είναι στη μνήμη, β) τι τύοπυ είναι και γ) ποιο API call πρέπει να γίνει ώστε να κάνει re-set τη νέα τιμή με βάση αυτή του κοινόχρηστου μοντέλου. Με αυτό τον τρόπο και έχοντας ορίσει bindings μεταξύ κάθε μιας διαμοιραζόμενης ιδιότητας του κοινόχρηστου μοντέλου με αυτές των διαδραστικών σχημάτων οι οποίες πρέπει να ενημερώνονται, οι διεπαφές θα παραμένουν συγχρονισμένες παρά τις ασυνέπειές τους σε κάθε περίπτωση.Αφού υλοποιηθούν τα προηγούμενα βήματα της διαδικασίας, ο MBUIDesigner κάνει deploy το project στο SSF. Αποτέλεσμα αυτής της διαδικασίας είναι η γέννηση συγκεκριμένων shortcuts τα οποία υποδεικνύουν με την εκτέλεσή τους στο τοπικό runtime environment (PS), το ποιο Project πρέπει να φορτωθεί και εκτελεστεί. Στην παρακάτω εικόνα (Εικόνα 32) φαίνεται ένα στιγμιότυπο της εν εκτελέσει ροής του σεναρίου. Σε συνέχεια του προηγούμενου βήματος ο PS επικοινωνεί με το SSF ώστε να κατεβάσει όλα τα σχετικά μοντέλα, τα οποία θα του υποδείξουν ποια dependency libraries (και λοιπούς πόρους) πρέπει να κατεβάσει (δεδομένου ότι όλα τα διαδραστικά αντικείμενα είναι μη εγγενώς υποστηριζόμενα), εκτελώντας τη λειτουργία της κατανεμημένης μεταφόρτωσης κλάσεων, να κάνει attach τις όποιες συμπεριφορές στα engaged πολυμορφικά στιγμιότυπα με βάση το behavior μοντέλο και να εκκινήσει τέλος μια διαδικασία συνεχούς παρακολούθησης της διεπαφής ώστε αυτή να παραμένει συγχρονισμένη κάθε στιγμή και ευθυγραμμισμένη με τις απαιτήσεις του δυναμικού περιβάλλοντος εκτέλεσης.



Εικόνα 32: Εν εκτελέσει ανάπτυξη του σεναρίου

Σύνοψη αποτελεσμάτων και σχολιασμός

Το σενάριο που παρουσιάστηκε είναι πλήρως υλοποιημένο, λειτουργικό και διαθέσιμο προς επίδειξη. Όπως πρέπει να έχει καταστεί ήδη σαφές, δεν μπορεί να υποστηριχθεί διαμέσου των καθιερομένων μεθόδων ανάπτυξης για μια σειρά λόγων οι οποίοι και έχουν αναπτυχθεί (π.χ. υποθέσεις σε επίπεδο κλάσεων, διαλόγου, φορητότητας, κοκ.). Η προτεινόμενη προσέγγιση στο σύνολό της δεν έχει επικυρωθεί μόνο διαμέσου του εν λόγω σεναρίου χρήσης, αλλά και στα πλάισια άλλων καινοφανών σεναρίων χρήσης όμορης πολυπλοκότητας, ένα εκ των οποίων περιγράφεται σε πρόσφατα δημοσιευμένη εργασία [3]. Η εν λόγω προσέγγιση να σημειωθεί επίσης πως ακολουθείται και ως οδηγός στο ερευνητικό έργο πρόγραμμα MusicNet[15], για

[15] "MusicNet: Ολοκληρωμένος Σχεδιασμός και Υλοποίηση Συστήματος Δικτυακής Μουσικής Εκτέλεσης", ΘΑΛΗΣ, Πανεπιστήμιο-Κρήτης, ΔΙΕΣΗ, Κ.Α 3579.

κάποια απο τα σενάρια που αφορούν σύγχρονη συνεργασία κατα τη μουσική εκτέλεση, όπου η μουσική παρτιτούρα χρησιμοποιείται ως γλωσσικό ιδίωμα ανταλλαγής απόψεων και καθοδήγησης μουσικών που αυτοσχεδιάζουν.

Κεφάλαιο 6 – Επίλογος

Η παρούσα πτυχιακή εργασία καταπιάστηκε με το θέμα της υποστήριξης σύγχρονης συνεργατικής κατανεμημένης εργασίας σε πανταχού παρόντα περιβάλλοντα. Σε αυτά τα πλαίσια ανέδειξε το μεθοδολογικό αλλά και γενικότερο υπολογιστικό κενό όσον αφορά την υποστήριξη ανάπτυξης συστημάτων που να ικανοποιούν τις προδιαγραφές και απαιτήσεις των αναδυόμενων διάχυτων περιβαλλόντων χρήσης (pervasive contexts of use), όπου η ριζική ετερογένεια μεταξύ τους δεσπόζει ως κύριο χαρακτηριστικό τους γνώρισμα. Η εκκωφαντική αυτή ανεπάρκεια στη διαχείριση ριζικά ετερογενών περιβαλλόντων χρήσης, αποδίδεται στο γεγονός πως η τρέχουσα τεχνολογική στάθμιση (state-of-the-art) και οι διαθέσιμες προσεγγίσεις (βλ. Κεφάλαιο 3), λόγω συγκεκριμένων υποθέσεων (βλ. Κεφάλαιο 1), αδυνατούν να υποστηρίξουν τον απαραίτητο βαθμό αφαίρεσης (‘true abstraction’).

Απαντώντας τα ερευνητικά ερωτήματα

Επιχειρώντας να αξιολογήσουμε τη συνεισφορά της παρούσας μεταπτυχιακής εργασίας, θα αναφερθούμε κατ’ αρχήν στα βασικά ερευνητικά ερωτήματα τα οποία τέθηκαν και στη συνέχεια θα κωδικοποιήσουμε ενδεικτικές προτάσεις/λύσεις οι οποίες σχεδιάστηκαν και υλοποιήθηκαν στο πλαίσιο της παρούσας έρευνας.

Ερώτημα Α) Είναι δυνατή η υποστήριξη της περιγραφής πανταχού παρόντων διεπαφών μέσω μιας ενοποιημένης περιγραφής, στα πλαίσια μιας ενοποιημένης (unified) μεθοδολογίας, με κοινά εργαλεία? Τι δομές περιγραφής χρειάζονται για την ενοποιημένη περιγραφή πανταχού παρόντων διεπαφών?

Αποδείξαμε ότι κάτι τέτοιο είναι εφικτό, δεδομένης της υποστήριξης μιας πραγματικά αφαιρετικής προσέγγισης αντί γενικεύσεων. Στη βάση αυτή η παρούσα πρόταση ανέπτυξε την επιχειρηματολογία της στηριζόμενη σε δύο βασικούς πυλώνες: α) Τον πολυμορφισμό, ως θεωρητικού εργαλείου για την υποστήριξη ‘υπολογιστικά-αγνωστικών σχημάτων στιγμιοτυποποίησης’ (implementation-agnostic instantiation schemes) και β) τη μοντελο-κεντρική μηχανική ως εργαλείου μηχανικής (engineering construct) για την υποστήριξη του πρώτου, διαμέσου μιας ενοποιημένης μεθοδολογίας ανάπτυξης, κατάλληλα υποστηριζόμενης

από γενικού σκοπού εργαλεία (σχεδιαστικά και runtime περιβάλλοντα), αλλά και αφηρημένων σχημάτων προδιαγραφών (abstract specification schemes).

Ερώτημα Β) Είναι δυνατό να υποστηριχθεί σύγχρονος συγχρονισμός μεταξύ ριζικά ετερογενών διεπαφών με τρόπο γενικευμένο τόσο ώστε να μπορεί να καλύψει το σύνολο των πιθανών περιπτώσεων?

Είδαμε αναλυτικά πως αυτό επετεύχθη στα πλαίσια ενός σεναρίου όπου οι εμπλεκόμενες διεπαφές επιδείκνυαν διαφορές σε όλα τα πιθανά επίπεδα: λεκτικό (lexical/physical), συντακτικό (syntactic), σημασιολογικό (semantic), αλλά και από πλευράς τεχνολογιών υλοποίησης. Οι αναπαραστάσεις του πενταγράμμου και της ταμπλατούρας είναι υλοποιημένες σε java/swing, ενώ η κυκλική αναπαράσταση με χρήση HTML5 και advanced JavaScript. Δείξαμε πως παρόλες τις ασυνέπειες, με τη βοήθεια ρητά ορισμένων αφηρημένων μοντέλων και υποστηρικτικών αρχιτεκτονικών, μπορεί να υποστηριχθεί συγχρονισμός και κατανεμημένη ανάδραση (feedthourgh). Η ικανότητα και επάρκεια της μεθόδου έχει επαληθευτεί κατ' επανάληψη σε ανάλογης πολυπλοκότητας σενάρια (π.χ. [3]). Σε κάθε περίπτωση τα μοντέλα τα οποία περιεγράφηκαν δεν κάνουν υποθέσεις ούτε επιβάλλουν περιορισμούς αναφορικά με την εσωτερική δομή και σύσταση των αλληλεπιδραστικών αντικειμένων, επιτρέποντας κατ' αυτό τον τρόπο, την κωδικοποίηση κατά κανόνα κάθε πιθανού πολυμορφικού σχήματος. Δεδομένου ότι το ίδιο συμβαίνει και στην περίπτωση των προδιαγραφών του διαμοιραζόμενου μοντέλου, αλλά και των λοιπών οικείων μοντέλων (consistency, widgetResource), καθιστά την προσέγγιση ως γενικού σκοπού.

Ερώτημα Γ) Τι είδους περιβάλλοντα εκτέλεσης, υπολογιστικά εργαλεία και αρχιτεκτονικές χρειάζονται ως προς την εκπλήρωση των ζητούμενων στόχων?

Απαιτούνται εξελιγμένα περιβάλλοντα εκτέλεσης τόσο στην πλευρά του server, όσο και σε αυτή του client. Όσον αφορά τον client, προαπαιτούμενο είναι κατάλληλη υποδομή που να εξασφαλίζει την απρόσκοπτη, συνεπή και διασυνοριακά εκτεινόμενη ικανότητα διαχείρισης όλων των αντιστοιχήσεων (mappings) από το αφηρημένο στο τοπικό (device-specific) και αντίστροφα, συστήνοντας ουσιαστικά ένα εικονικό επίπεδο μεταξύ των μοντέλων της UsiXML και των εγγενώς βιβλιοθηκών της εκάστοτε πλατφόρμας. Ενώ από την πλευρά του server, η ύπαρξη πολύ-λειτουργικού συστατικού και δομών που να αναλαμβάνουν την υποστήριξη και διαχείριση σειράς καθηκόντων, όπως αυτά που περιγράφηκαν, και αφορούν: τη διαχείριση

συνόδων, τη διατήρηση ενός κοινόχρηστού αποθετηρίου widgetArchives τα οποία να μπορεί να είναι προσβάσιμα ανά πάσα στιγμή, την υποβοήθηση κατανεμημένης μεταφόρτωσης κλάσεων στην περίπτωση μη εγγενώς υποστηριζόμενων αντικειμένων, κοκ.

Επεκτάσεις και Μελλοντική εργασία

Παρότι τα αποτελέσματα της παρούσας έρευνας υπήρξαν ιδιαιτέρως ενθαρρυντικά, αναδεικνύονται και ζητήματα που τελούν υπό διερεύνηση και απαιτούν επιπλέον προσπάθεια. Παραδείγματος χάριν, το θέμα της ενημερότητας των εταίρων σε σύγχρονα περιβάλλοντα διάδρασης είναι καθοριστικό για την οργάνωση του συνεργατικού έργου και την απρόσκοπτη σύμπραξη των εμπλεκομένων εταίρων. Αν και το ‘ακουμπήσαμε’ στο σενάριο που περιγράψαμε (μέσω των μηχανισμών feedthrough), η γενικευμένη υποστήριξη προηγμένων μορφών ενημερότητας (βλ. group-awareness, social translucence, κοκ.) παραμένει ανοιχτό ζήτημα. Επίσης, ανοιχτό ζήτημα παραμένει η χρήση της προσέγγισης σε εξειδικευμένα περιβάλλοντα συνεργασίας όπως παραδείγματος χάριν οι περιπτώσεις των συστημάτων τηλεδιάσκεψης που προϋποθέτουν ειδικού τύπου συστατικά (π.χ. codecs) για τη κωδικοποίηση/από-κωδικοποίηση πακέτων σημάτων οπτικοακουστικού περιεχομένου και video. Οι εφαρμογές αυτές ωστόσο, πέρα των ζητημάτων διεπαφής που θέτουν παρουσιάζουν και άλλα ζητήματα (π.χ. διαχείριση καθυστέρησης, προαπαιτούμενα σε επίπεδο τηλεπικοινωνιακής υποδομής, κλπ.) που δεν εμπίπτουν στο πεδίο της τρέχουσας εργασίας.

Βιβλιογραφία - Αναφορές

[1] Thomas P. Moran and Paul Dourish. 2001. Introduction to this special issue on context-aware computing. Hum.-Comput. Interact. 16, 2 (December 2001), 87-95.

[2] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., and Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces, Interacting with Computers, 15(3), pp. 289-308 (June 2003).

[3] Dimitrios Kotsalis, George Vellis, Demos Akoumianakis, and Jean Vanderdonckt. 2014. Implementation-agnostic instantiation schemes for ubiquitous, synchronous multi-user interfaces. In Proceedings of the 18th Panhellenic Conference on Informatics (PCI '14). ACM, New York, NY, USA, , Article 37 , 6 pages.

[4] Meixner, G., Paternó, F., and Vanderdonckt, J.: Past, Present, and Future of Model-Based User Interface Development. i-com, 10, 3 (2011), 2-11.

[5] L. Choonhwa, S. Helal, and L. Wonjun, "Universal interactions with smart spaces", IEEE Pervasive Computing, vol. 5, Jan. 2006, pp.16-21, doi:10.1109/MPRV.2006.19.

[6] Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, and Víctor López-Jaquero. 2004. USIXML: a language supporting multi-path development of user interfaces. In Proceedings of the 2004 international conference on Engineering Human Computer Interaction and Interactive Systems (EHCI-DSVIS'04), Rémi Bastide, Philippe Palanque, and Jörg Roth (Eds.). Springer-Verlag, Berlin, Heidelberg, 200-220.

[7] Fabio Paternò, Cristiano Mancini, and Silvia Meniconi. 1997. ConcurTaskTrees: A Diagrammatic Notation for Specifying Task Models. In *Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction* (INTERACT '97), Steve Howard, Judy Hammond, and Gitte Lindgaard (Eds.). Chapman & Hall, Ltd., London, UK, UK, 362-369.

[8] Q. Limbourg and J. Vanderdonckt. 2009. Multi-Path Transformational Development of User Interfaces with Graph Transformations. Journal Human-Centered Software Engineering, HCI Series, Springer, London, UK, pp. 109-140.

[9] Molina, A.I., Redondo, M.A., Ortega, M.*: A methodological approach for user interface development of collaborative applications: A case study.* Science of Computer Programming 74(9), 754–776 (2009).

[10] Fabio Paternò, ConcurTaskTrees: An engineered notation for task models, in: D. Diaper, N.A. Stanton. (Eds.), The Handbook of Task Analysis For HCI, LEA, Mahwah, NJ, 2004, pp. 483–501.

[11] David Pinelle, Carl Gutwin, and Saul Greenberg. 2003. Task analysis for groupware usability evaluation: Modeling shared-workspace tasks with the mechanics of

collaboration. ACM Trans. Comput.-Hum. Interact. 10, 4 (December 2003), 281-311.

[12] Garrido, J.L., Gea, M., Rodríguez, M.L.: *Requirements engineering in cooperative systems, In: Requirements Engineering for Sociotechnical Systems,* Idea Group, Inc., USA, pp. 226–244 (2005).

[13] Ana Isabel Molina, Miguel Ángel Redondo, Manuel Ortega*, A methodological approach for user interface development of collaborative applications: A case study, Science of Computer Programming,* Volume 74, Issue 9, 1 July 2009, pp. 754-776, ISSN 0167-6423.

[14] Victor M. Ruiz Penichet, María Dolores Lozano, José A. Gallud, Ricardo Tesoriero: User interface analysis for groupware applications in the TOUCHE process model. Advances in Engineering Software 40(12): 1212-1222 (2009).

[15] Antony Savidis, Constantine Stephanidis, Demosthenes Akoumianakis, Unifying toolkit programming layers: a multi-purpose toolkit integration module, In *Proceedings of the 4th Workshop on Design, Specification and Verification of Interactive Systems*, Granada, Spain, 1997, pp. 177–192.

[16] Anthony Savidis and Constantine Stephanidis. 1995. Developing dual user interfaces for integrating blind and sighted users: the HOMER UIMS. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '95), ACM PressNew York, NY, USA.

[17] Blattner, M.; Glinert, E.; Jorge, J. & Ormsby, G. Metawidgets: Towards a Theory of Multimodal Interface Design. In Proceedings of the 16th International Computer Software and Applications Conference (COMPSAC '92), 1992, 115-120.

[18] Murray Crease, Philip Gray, and Stephen Brewster. 2000. A toolkit of mechanism and context independent widgets. In *Proceedings of the 7th international conference on Design, specification, and verification of interactive systems* (DSV-IS'00), Philippe Palanque and Fabio Paterno (Eds.). Springer-Verlag, Berlin, Heidelberg, 121-133.

[19] Baha Jabarin and T. C. Nicholas Graham. 2003. Architectures for Widget-Level Plasticity. In Proceedings of Interactive Systems Design Specification and Verification (DSV-IS'03), Joaquim A. Jorge, Nuno Jardim Nunes and Joao Falcao e Cunha (Eds.). Springer-Verlang, Berlin, Heidelberg, 124-138

[20] Mark Roseman and Saul Greenberg. 1996. Building real-time groupware with GroupKit, a groupware toolkit. ACM Trans. Comput.-Hum. Interact. 3, 1 (March 1996), 66-106.

[21] Ralph D. Hill, Tom Brinck, John F. Patterson, Steven L. Rohall, and Wayne T. Wilner. 1993. The Rendezvous language and architecture. Commun. ACM 36, 1 (January 1993), 63-67.

[22] Banani Roy, Nicholas Graham, and Carl Gutwin. 2012. DiscoTech: a plug-in toolkit to improve handling of disconnection and reconnection in real-time groupware. In Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work (CSCW '12). ACM, New York, NY, USA, 1287-1296.

[23] Brian de Alwis, Carl Gutwin, and Saul Greenberg. 2009. GT/SD: performance and simplicity in a groupware toolkit. In Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems (EICS '09). ACM, New York, NY, USA, 265-274.

[24] Jason Hill and Carl Gutwin. 2004. The MAUI Toolkit: Groupware Widgets for Group Awareness.Comput. Supported Coop. Work 13, 5-6 (December 2004), 539-571.

[25] Matthias Wenzel, Lutz Gericke, Raja Gumienny, Christoph Meinel. 2013. Towards cross-platform collaboration - Transferring real-time groupware to the browser. In *Proceedings of the 17th IEEE International Conference on Computer Supported Cooperative Work in Design* (CSCWD '13). IEEE, Whistler, BC, Canada, 49-54.

[26] Weigang Wang. 2013. Integrating PowerMeeting into Blackboard Learning Environment: Synchronous Collaboration Support in Asynchronous Collaboration Context. In *Proceedings of the 10th IEEE International conference on e-Business Engineering* (ICEBE '10). IEEE, Coventry, UK, 484-489.

[27] Vivian Genaro Motti and Jean Vanderdonckt. 2013. A computational framework for context-aware adaptation of user interfaces. In *Proceedings of the IEEE 2013 conference on Research Challenges in Information Science* (RCIS '13). IEEE, Paris, France, 1-12.

[28] Lucio Davide Spano, Fabio Paternò, and Gianni Fenu. 2014. A gestural concrete user interface in MARIA. In Proceedings of the 2014 ACM SIGCHI symposium on Engineering interactive computing systems (EICS '14). ACM, New York, NY, USA, 179-184.

[29] Sophie Lepreux, Anas Hariri, José Rouillard, Dimitri Tabary, Jean-Claude Tarby, and Christophe Kolski. 2007. Towards multimodal user interfaces composition based on UsiXML and MBD principles. In Proceedings of the 12th international conference on Human-computer interaction: intelligent multimodal interaction environments (HCI'07), Julie A. Jacko (Ed.). Springer-Verlag, Berlin, Heidelberg, 134-143.

[30] Matthias Wenzel, Lutz Gericke, Raja Gumienny, Christoph Meinel. 2013. Towards cross-platform collaboration - Transferring real-time groupware to the browser. In *Proceedings of the 17th IEEE International Conference on Computer Supported Cooperative Work in Design* (CSCWD '13). IEEE, Whistler, BC, Canada, 49-54.

[31] Christopher Wolfe, T.C. Nicholas Graham, W. Greg Phillips, and Banani Roy. 2009. Fiia: user-centered development of adaptive groupware systems. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems* (EICS '09). ACM, New York, NY, USA, 275-284.

[32] Fernando Gallego, Ana I. Molina, Crescencio Bravo, and William J. Giraldo. 2010. A proposal for model-based design and development of group work tasks in a shared context. In Proceedings of the 7th international conference on Cooperative design, visualization, and engineering (CDVE'10), Yuhua Luo (Ed.). Springer-Verlag, Berlin, Heidelberg, 11-18.

[33] W. Greg Phillips, T. C. Nicholas Graham*: Workspaces: A Multi-level Architectural Style for Synchronous Groupware.* DSV-IS 2003: 92-106.

[34] Maria Luisa Rodriguez, Jose; Luis Garrido, Maria V. Hurtado, and Manuel Noguera. 2007. *An approach to the model-based design of groupware multi-user interfaces.* In *Proceedings of the 13th international conference on Groupware: design implementation, and use* (CRIWG'07), Jorg M. Haake, Sergio F. Ochoa, and Alejandra Cechich (Eds.). Springer-Verlag, Berlin, Heidelberg, 157-164.

[35] Josefina Guerrero Garcia, Jean Vanderdonckt, and Juan Manuel Gonzalez Calleros. 2008. FlowiXML: a step towards designing workflow management systems. *Int. J. Web Eng. Technol.* 4, 2 (May 2008), 163-182.

[36] David Pinelle, Carl Gutwin, and Saul Greenberg. 2003. Task analysis for groupware usability evaluation: Modeling shared-workspace tasks with the mechanics of collaboration. ACM Trans. Comput.-Hum. Interact. 10, 4 (December 2003), 281-311.

[37] Penichet, V., Lozano, M., Gallud, J., Tesoriero, R., Analysis Models for User Interface Development in Collaborative Systems, CADUI'08, available on-line from penichet.no-ip.com:8080/penichet/modules/CVitae/archivos/penichet_CADUI08-Springer.pdf

[38] Steven Xia, David Sun, Chengzheng Sun, David Chen, and Haifeng Shen. 2004. Leveraging single-user applications for multi-user collaboration: the coword approach. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work* (CSCW '04). ACM, New York, NY, USA, 162-171.

[39] Isenhour, P., Rosson, M-B., Carroll, J.*, Supporting interactive collaboration on the Web with CORK.* Interacting with Computers 13, 2001, 655—676.

[40] George Vellis, Dimitrios Kotsalis, Demosthenes Akoumianakis, Jean Vanderdonckt (2012). Model-based engineering of multi-platform, synchronous & collaborative UIs Extending UsiXML for polymorphic User Interface specification, 16th Panhellenic Conference on Informatics, Athens 5-7 October, IEEE

[41] Dewan, P. and Choudhary, R., A high-level and flexible framework for implementing multiuser user interfaces. ACM Transactions on Information Systems, 10(4):345–380, October 1992.

[42] Jacob W. Bartel and Prasun Dewan. 2012. Towards multi-domain collaborative toolkits. In Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work (CSCW '12). ACM, New York, NY, USA, 1297-1306.

[43] Ana I. Molina, William J. Giraldo, JesúS Gallardo, Miguel A. Redondo, Manuel Ortega, and Guillermo GarcíA. 2012. CIAT-GUI: A MDE-compliant environment for developing Graphical User Interfaces of information systems. Adv. Eng. Softw. 52 (October 2012), 10-29.

[44] Alexandre Demeure, Gaelle Calvary, Joelle Coutaz, and Jean Vanderdonckt. 2006. The COMETs inspector: towards run time plasticity control based on a semantic network. In *Proceedings of the 5th international conference on Task models and diagrams for users interface design* (TAMODIA'06), Karin Coninx, Kris Luyten, and Kevin A. Schneider (Eds.). Springer-Verlag, Berlin, Heidelberg, 324-338.

[45] Quentin Limbourg and Jean Vanderdonckt. 2004. Addressing the mapping problem in user interface design with UsiXML. In *Proceedings of the 3rd annual conference on Task models and diagrams* (TAMODIA '04). ACM, New York, NY, USA, 155-163.

[46] Jessica Rubart and Peter Dawabi. 2002. Towards UML-G: A UML Profile for Modeling Groupware. In*Proceedings of the 8th International Workshop on Groupware: Design, Implementation and Use*(CRIWG '02), Jörg M. Haake and José A. Pino (Eds.). Springer-Verlag, London, UK, UK, 93-113.

[47] Jessica Rubart and Peter Dawabi. 2004. Shared data modeling with UML-G. *Int. J. Comput. Appl. Technol.* 19, 3/4 (May 2004), 231-243.

[48] Carl A. Gutwin, Michael Lippold, and T. C. Nicholas Graham. 2011. Real-time groupware in the browser: testing the performance of web-based networking. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work* (CSCW '11). ACM, New York, NY, USA, 167-176.

[49] Oliver Schmid, Agnes Lisowska Masson, and Béat Hirsbrunner. 2014. Real-time collaboration through web applications: an introduction to the Toolkit for Web-based Interactive Collaborative Environments (TWICE). *Personal Ubiquitous Comput.* 18, 5 (June 2014), 1201-1211.

[50] Qi Huang, Daniel A. Freedman, Ymir Vigfusson, Ken Birman, and Bo Peng. 2010. Kevlar: a flexible infrastructure for wide-area collaborative applications. In *Proceedings of the ACM/IFIP/USENIX 11th International Conference on Middleware* (Middleware '10). Springer-Verlag, Berlin, Heidelberg, 148-168.

[51] Wen Li, Weicong Wang, and Ivan Marsic. 1999. Collaboration transparency in the DISCIPLE framework. In *Proceedings of the international ACM SIGGROUP conference on Supporting group work* (GROUP '99), Stephen C. Hayne (Ed.). ACM, New York, NY, USA, 326-335.

[52] Gall, U., Hauck, F.J.: Promondia: A Java-based framework for real-time group communication in the Web. In *Proceedings of the Sixth International World Wide Web Conference*, Santa Clara, California, USA, (1997) 917-926.

[53] Shirmohammadi, S., Oliveira, J.C., and Georganas, N.D., Applet-Based Telecollaboration: A Network-centric Approach, IEEE Multimedia Magazine, Volume 5, Number 2, (April-June 1998), pp. 64-73.

[54] Greenberg, S., Marwood, D. Real time groupware as a distributed system: Concurrency control and its effect on the interface. Proceedings of the ACM 1994 Conference on Computer Supported Cooperative Work (CSCW'94) (Chapel Hill, N.C., USA, October 22-26, 1994), 207-218.

[55] Dewan, P., Choudhary, R. Flexible user interface coupling in collaborative systems. Proceedings of ACM CHI'91 conference, (New Orleans, Lousiana, April 27 - May 2, 1991), 41-49.

[56] Christian Schuckmann, Lutz Kirchner, Jan Schümmer, and Jörg M. Haake. 1996. Designing object-oriented synchronous groupware with COAST. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work* (CSCW '96), Mark S. Ackerman (Ed.). ACM, New York, NY, USA, 30-38.

[57] Greenberg, S. and Marwood, D., Real time groupware as a distributed system: Concurrency control and its effect on the interface. In Proceedings of the ACM CSCW'94 Conference on Computer Supported Cooperative Work, pages 207–217, Chapel Hill, North Carolina, October 22–26, 1994.

[58] Mark Stefik, Gregg Foster, Daniel G. Bobrow, Kenneth Kahn, Stan Lanning, and Lucy Suchman, "Beyond the Chalkboard Computer Support for Collaboration and Problem Solving in Meetings," CACM 30:1 (January 1987), pp. 32-47.

[59] M. Stefik, D. G. Bobrow, G. Foster, S. Lanning, and D. Tatar, 'WYSIWIS Revised: Early Experiences with Multiuser Interfaces', ACM Transactions on Office Information Systems 5:2 (April 1987), pp. 147-167.

[60] Mark Stefik, Gregg Foster, Daniel G. Bobrow, Kenneth Kahn, Stan Lanning, and Lucy Suchman. 1987. Beyond the chalkboard: computer support for collaboration and problem solving in meetings. *Commun. ACM* 30, 1 (January 1987), 32-47.

[61] Stephan Lukosch. 2004. Flexible and transparent data sharing for synchronous groupware. *Int. J. Comput. Appl. Technol.* 19, 3/4 (May 2004), 215-230.

[62] Dewan, P. Architectures for collaborative applications. In Computer Supported Co-operative Work, M. Beaudouin-Lafon, Ed., vol. 7 of Trends in Software. John Wiley & Sons, 1999, pp. 169-193.

[63] Menges, J., Jeffay, K., "Inverting X: An Architecture for a Shared Distributed Window System", Workshop on Infrastructure for Collaborative Enterprises, 1994.

[64] Engelbart, D. and English, W. (1968). A research center for augmenting human intellect. In AFIPS Proceedings of the 1968 Fall Joint Computer Conference. AFIPS Press, 395-410.

[65] Surendar Chandra, John Boreczky, and Lawrence A. Rowe. 2014. High performance many-to-many intranet screen sharing with DisplayCast. *ACM Trans. Multimedia Comput. Commun. Appl.*10, 2, Article 19 (February 2014), 22 pages.

[66] Richardson, T., Stafford-Fraser, Q., Wood, K. R., and Hopper, A. (1998). Virtual Network Computing. IEEE Internet Computing, 2(1), IEEE Educational Activities Department, 33-38.

[67] Gregor McEwan and Saul Greenberg. 2005. Supporting social worlds with the community bar. In*Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*(GROUP '05). ACM, New York, NY, USA, 21-30.

[68] Saul Greenberg and Michael Rounding. 2001. The notification collage: posting information to public and personal displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '01). ACM, New York, NY, USA, 514-521.

[69] H. AbdelWahab, E. Stoica and M. AlAghbari, ``XTV: A Collaborative Computing System", proceedings of the Second Annual Joint Conference on Information Sciences, Writesville Beach, NC, pp. 107110, Sept. 1995.

[70] K. A. Lantz, “An Experiment in Integrated Multimedia Conferencing,” Proceedings of CSCW 86.

[71] Terrence Crowley, Paul Milazzo, Ellie Baker, Harry Forsdick, and Raymond Tomlinson, “MMConf: An Infrastructure for Building Shared Multimedia Applications,” Proceedings of CSCW 90.

[72] Nigay, Laurence; Coutaz, Joëlle (1997). "Software Architecture Modelling: Bridging Two Worlds Using Ergonomics and Software Properties" (PDF). Formal Methods in Human-Computer Interaction. Springer-Verlag. chapter 3, pp. 49–73. ISBN 3-540-76158-6.

[73] Greenberg, S. and Roseman, M. (1999). *Groupware Toolkits for Synchronous Work.* In "Computer-Supported Cooperative Work (Trends in Software 7)" (John Wiley & Sons Ltd; edited by M. Beaudouin-Lafon).

[74] Ahuja, S.R., Ensor, J.R. and Lucco, S.E. (1990) “A comparison of applications sharing mechanisms in real-time desktop conferencing systems.” In Proceedings of the ACM COIS Conference on Office Information Systems, p238–248, Boston, April 25–27.

[75] Dourish, P. (1996) “Consistency guarantees: Exploiting application semantics for consistency management in a collaboration toolkit.” in Proceedings of the ACM

CSCW'96 Conference on Computer Supported Cooperative Work, Boston, Mass., November 16–20, in press.

[76] Graham, T.C.N., Urnes, T. and Nejabi, R. (1996) "Efficient distributed implementation of semireplicated synchronous groupware." in Proceedings of the ACM UIST '96 User Interface Software and Technology, Seattle, Washington, November 6-8, In press.

[77] Greenberg, S. and Marwood, D. (1994) "Real time groupware as a distributed system: Concurrency control and its effect on the interface." in Proceedings of the ACM CSCW'94 Conference on Computer Supported Cooperative Work, p207–217, Chapel Hill, North Carolina, October 22–26.

[78] Lauwers, J.C. and Lantz, K.A. (1990) "Collaboration awareness in support of collaboration transparency." in Proceedings of the ACM SIGCHI'90 Conference on Human Factors in Computing Systems, p303-211, Seattle, Washington, April 1–5.

[79] Lauwers, J.C., Joseph, T.A., Lantz, K.A. and Romanow, A.L. (1990) "Replicated architectures for shared window systems: A critique." In Proceedings of the ACM COIS'90 Conference on Office Information Systems, p249–260, Boston, Mass., April 25–27.

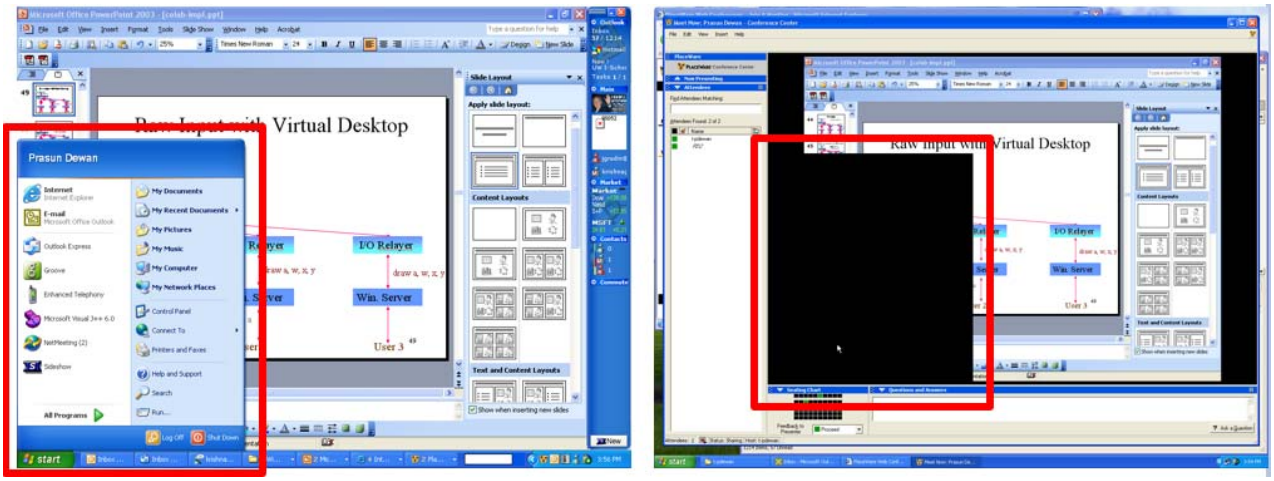
[80] Jahn, P. (1995) "Getting started with Share-Kit." Tutorial manual distributed with Share-Kit version 2.0. Communications and Operating Systems Research Group, Department of Computer Science, Technische Universitat, Berlin, Germany. Available via anonymous ftp from ftp.inf.fuberlin.de:/pub/misc/share-kit.

[81] Knister, M.J. and Prakash, A. (1990) "DistEdit: A distributed toolkit for supporting multiple group editors." in Proceedings of ACM CSCW'90 Conference on Computer Supported Cooperative Work, p343–355, Los Angeles, California, October 7–10.

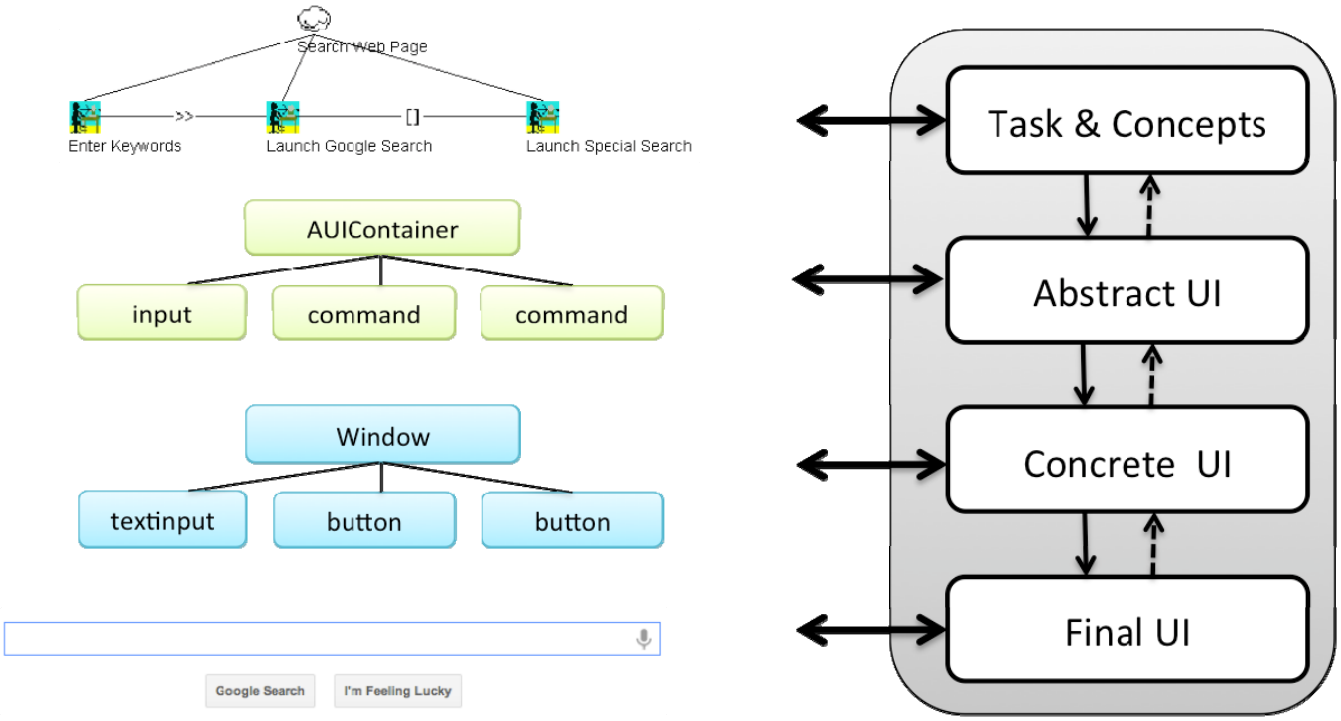
[82] Tou, I., Berson, S., Estrin, G., Eterovic, Y. and Wu, E. (1994) "Prototyping synchronous group applications." IEEE Computer, 27(5), p48–56, May.

[83] Patterson, J.F., Day, M. and Kucan, J. (1996) "Notification servers for synchronous groupware." in Proceedings of the ACM CSCW'96 Conference on Computer Supported Cooperative Work, Boston, Mass., November 16–20, in press.

Παράρτημα



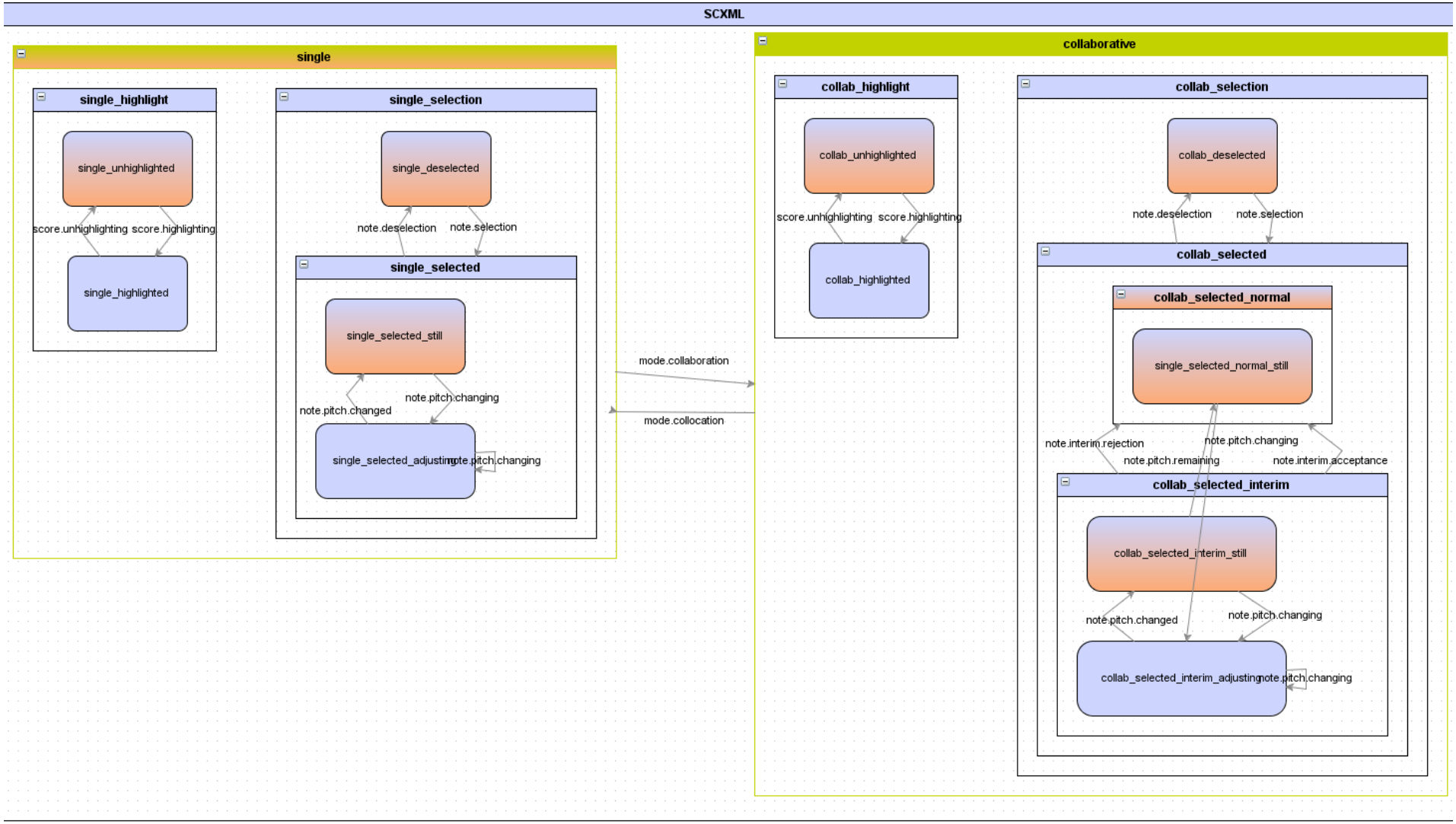
Εικόνα 33: Πρόβλημα επικάλυψης παραθύρων στην περίπτωση του διαμοιρασμού παραθύρου (Window-sharing)



Εικόνα 34: Παράδειγμα περιγραφής (specification) διεπαφής στη UsiXML σε όλα τα υποστηριζόμενα επίπεδα αφαίρεσης

Μέθοδος Ανάπτυξης		Υποστήριξη Ριζικά ετερογενών πλατφορμών	Προσαρμοστική διαχείριση διαλόγου	Ενοποιημένη πολύ-πλαισιακή μεθοδολογία ανάπτυξης	Διαδραστικό Λεξικό		Groupware Aspects support
					Επεκτασιμότητα	Εκφραστικότητα	
Toolkit-based programming	Single-user toolkits	ΟΧΙ	ΟΧΙ	ΟΧΙ	**ΝΑΙ** (κληρονομικότητα)	**ΝΑΙ** (μέγιστη δυνατή)	ΟΧΙ
	Adaptive toolkits	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ	Η προκαθορισμένη (κατά κανόνα ελάχιστη)	ΟΧΙ
	Groupware toolkits	ΟΧΙ	ΟΧΙ	ΟΧΙ	ΟΧΙ (εξαιρετικά περίπλοκη)	Η Προκαθορισμένη (κατά κανόνα ελάχιστη)	**ΝΑΙ**
Model-based UI engineering		**ΝΑΙ**	**ΝΑΙ**	**ΝΑΙ**	ΟΧΙ	Ελάχιστη (In principle form-based only)	ΟΧΙ

Πίνακας 4: Συγκριτικός πίνακας κύριων μεθόδων ανάπτυξης διεπαφών ως προς τις βασικές απαιτήσεις υποστήριξης synchronous distributed ubiquitous groupware



Εικόνα 35: Στιγμιότυπο του προσδιορισμού όλων των υποστηριζόμενων καταστάσεων (States) και transitions του σχήματος στιγμιοτυποποίησης της νότας πενταγράμμου του java/swing