Model-based UI Engineering of Advanced and Creativity-based UIs

by

KOTSALIS DIMITRIOS

B.Sc., Technological Education Institution of Crete, 2007

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF INFORMATICS ENGINEERING

SCHOOL OF APPLIED TECHNOLOGY

TECHNOLOGICAL EDUCATION INSTITUTION OF CRETE

2015

Approved by:

Major Professor
Demosthenes Akoumianakis

Copyright

Abstract

With the advent of new interaction platforms, novel network-attachable devices, new input/output devices and mobile terminals the prevalent Graphical User Interface (GUI) design language for interacting with software has changed both in ontological scope and symbolic manifestation. Thus, more recent graphical toolkits signify some sort of departure from or enrichment of the original design commitments in an attempt to facilitate richer interactions with more complex software systems. This trend has brought about customized libraries, widgets of various types as well as toolkit-specific techniques for creating custom interaction elements. Examples of this sort of development include Google's Material Design, MetroUI by Microsoft but also earlier efforts such the CrossY system[11] which injects new affordances to traditional widgets so that they can be manipulated using crossing interactions instead of clicking, the OrthoZoom system[12] which turns a traditional scrollbar into a powerful multi-scale navigation tool for very large documents, and scent widgets[13]. More importantly, however, efforts such as the above have enabled the development of new toolkits addressing cutting edge issues in 2D graphical interaction. Amongst the most prominent representatives are Jazz[14] and Piccolo[15] and a variety of information visualization toolkits such as prefuse[16] and JGraph (http://www.jgraph.com/).

Such a variety has both positive and negative implications. On the positive site, it enables exploitation of interaction facilities available through different toolkits so as to build creative but also more complex user interfaces. Such interactions are typically engineered using toolkit-based programming and custom techniques, which however bind the resulting user interface to a specific platform. As a result, portability, interoperability and scalability become harder, if at all possible. Another shortcoming is the difficulty associated with the integration of these toolkits with other user interface engineering methods such as model-based user interface engineering. Consequently, the benefits of the latter approach are sacrificed.

The basic premise of this thesis is that there is benefit to be gained by appropriating the offerings of both perspectives. Specifically, some of the advantages of toolkit programming-based techniques include the capability to build novel interaction facilities by combining radically different interaction object hierarchies. On the other hand, model-based UI engineering

offers powerful abstraction mechanisms to frame UI construction as model transformation rather than code manipulation. Therefore, it would be useful to establish bridges between toolkit-based offerings and abstract notations, device-independent mark-up and the mapping of abstract components to platform- specific toolkit libraries so as to appropriate the benefits of each for the benefit of both UI engineers and end users.

In light of the above, the proposed work is motivated by some shortcomings characterizing all recent approaches to model-driven UI engineering. These are summarized as follows:

- Model-based UI engineering assumes standard native vocabularies, thus it does not support novel or custom widgets built on top of native components
- Model-based UI engineering does not support mixed interaction object hierarchies combining interaction components from different toolkit libraries.

By relaxing these constraints, the current proposal aims to extend the capabilities of the current generation of model-driven user interface engineering methods by investigating and proposing:

- Techniques for modeling the interaction affordances of a designated interaction platform in a manner compliant to a model-driven development framework such as usiXML[17]
- New techniques for specifying (as opposed to programming) and modeling interactive behaviors so as to utilize novel interaction affordances which may be implemented through platform-specific custom widgets
- New tools to enhance the capacity of existing model-based development frameworks such as usiXML to utilize (i.e., link to) implemented (non-native) interaction components

Περιεχόμενα

Εικόνες

Πίνακες

Ευχαριστίες

Καταρχήν, θα ήθελα να ευχαριστήσω θερμά τον επιβλέπων καθηγητή μου κ. Ακουμιανάκη Δημοσθένη, για την καθοδήγησή και άμεση βοήθειά του κατά την διάρκεια εκπόνησης αυτής της διπλωματικής εργασίας αλλά και για την υποστήριξη του όλα τα χρόνια της συνεργασίας μας. Στη συνέχεια, θα ήθελα να ευχαριστήσω τον φίλο και επί χρόνια συνεργάτη Βελλή Γιώργο για την πολύτιμη βοήθεια του κατά τη τεκμηρίωση αυτής της εργασίας. Επίσης, θα ήθελα να ευχαριστήσω το Εργαστήριο Τεχνολογίας & Συστημάτων Λογισμικού (iSTLab) για τη φιλοξενία, το κ. Βιδάκη Νικόλαο και όλα τα παιδιά που συμμετείχαν/ουν στο εργαστήριο για την οποιαδήποτε συνεισφορά τους. Πάνω απ' όλα, ένα μεγάλο ευχαριστώ στους γονείς μου, για την υποστήριξή τους σε κάθε μου βήμα και τέλος θα ήθελα να ευχαριστήσω και το Ίδρυμα Κρατικών Υποτροφιών για την επιλογή να χρηματοδοτήσει με υποτροφία τη συγκεκριμένη διπλωματική εργασία.

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.

Κεφάλαιο 1 – Εισαγωγή

Η παρούσα εργασία παρακινείται από τη ραγδαία αύξηση όσον αφορά τον τύπο (type), εύρος (range) και φάσμα (scope) υπολογιστικών πόρων, που είναι διαθέσιμοι σε διαφορετικές υπολογιστικές συσκευές και επιτρέπουν την ανάπτυξη εφαρμογών που προσαρμόζονται στις απαιτήσεις διαφορετικών πλαισίων χρήσης (contexts of use). Όσον αφορά τον τύπο, από επαναχρησιμοποιήσιμα διαδραστικά αντικείμενα (των πρώιμων PC-oriented εργαλειοθηκών), πλέον έχουμε εξειδικευμένα αντικείμενα (customized components), αντικείμενα ενδεδειγμένα ως προς τη συσκευή (device-specific objects), σύνθετα web widgets είτε ως αυτόνομες εφαρμογές μικρής κλίμακας (π.χ. Google Gadgets[1]), είτε ως προϊόντα προηγμένων web-oriented εργαλειοθηκών (π.χ. AngularJS[2], GWT[3], κοκ.), κ.α.. Όσον αφορά το εύρος, αυτό σχετίζεται με τις αυξημένες επιλογές που προσφέρονται στους χρήστες όσο αφορά την τεχνοτροπία αλληλεπίδρασης που συχνά υπερβαίνει τις συμβατικές συνήθειες της κατάδειξης (pointing), επιλογής (selecting), είσοδο κειμένου (text-entry), κοκ..

Εικόνα 1: Ενδεικτικά στιγμιότυπα διαφόρων ενδεικτικών τύπων διαδραστικών στοιχείων

Οι χρήστες κερδίζουν από αυτή την ποικιλία, μιας και τους δίνεται η δυνατότητα να εκτελέσουν σειρά καθηκόντων σε εύρος πλαισίων χρήσης (contexts of use), διαμέσου του χειρισμού ευρείας γκάμας αντικειμένων 'σχετικών με το πεδίο' ('domain-specific'), τα οποία

[1] https://developers.google.com/gadgets/

[2] https://angularjs.org/

[3] http://www.gwtproject.org/

μπορεί να έχουν αναπτυχθεί από διαφορετικούς χρήστες ή να είναι συμβατά με διαφορετικές γλώσσες προδιαγραφών (π.χ. Yahoo widgets spec, Google Gadgets spec., κοκ.). Είναι λοιπόν εμφανές ότι εισερχόμαστε σε μια νέα φάση όσον αφορά την ανάπτυξη διεπαφών, όπου τα διαδραστικά στοιχεία διατίθενται κάτω από διαφορετικά καθεστώτα (π.χ. δωρεάν, ανοιχτού λογισμικού βιβλιοθήκες, κατανεμημένα σε 'αποθήκες' – repositories, κοκ.), ενώ η αποτελεσματική αξιοποίηση τους (κατά τη σχεδίαση διεπαφών) δεν εξυπηρετείται από τη συναρμολόγηση αντικειμένων (process of assembling components) . Ωστόσο, η διαχείριση ποικίλων συλλογών αντικειμένων (diverse collections of elements), επιβάλει ουσιαστικές προκλήσεις, μιας και υλοποιούνται από διαφορετικές πηγές, μπορεί να υιοθετούν διαφορετικές γλώσσες προγραμματισμού, πιθανότατα διαφορετικά APIs ή ακόμα και προγραμματιστικά μοντέλα (programming models), ενώ τουλάχιστον κάποιος βαθμός φορητότητας (portability) απαιτείται ούτως ώστε να διανέμονται και εκτελούνται απρόσκοπτα κατά μήκος διαφορετικών και ανά περίπτωση ριζικά (radically different) ετερογενών μεταξύ τους, πλαισίων χρήσης. Τα προηγούμενα μετασχηματίζουν τον τρόπο σχεδίασης των διεπαφών, εγείροντας σημαντικές επιπτώσεις στη φάση σχεδίασης η οποία με γοργό ρυθμό απομακρύνεται από μια κατά κανόνα μονοχρηστική διαδικασία (single-designer mode), σε μια πιο συνεργατική και κατανεμημένη διάταξη. Είναι πιο συνεργατική υπό την έννοια πως η σχεδίαση κάθε διεπαφής δεν αντανακλά μόνο τη φιλοσοφία του επικεφαλής σχεδιαστή (chief designer), αλλά επίσης τη φιλοσοφία των σχεδιαστών των αντικειμένων ή των οικείων (custom) αντικειμένων που χρησιμοποιούνται. Είναι πιο κατανεμημένη, μιας και η σχεδίαση αντανακλά τη συνεισφορά διαφορετικών ατόμων διασχίζοντας όρια ομάδων και φυσικών ορίων. Ως αποτέλεσμα, η σχεδίαση της διεπαφής νοείται ως διαδικασία συναρμολόγησης αντικειμένων (στα πλαίσια ενός product line), φτιαγμένων από διαφορετικές πηγές. Υπό αυτό το σκεπτικό είναι λοιπόν σημαντικό να εγκαθιδρύσουμε σχεδιαστικές ροές (processes), ικανές να υποστηρίξουν αποκλίσεις (variations) σε όλες τις όψεις τους. Υπό αυτό το πλαίσιο, ποιοτικά χαρακτηριστικά (ή affordances) και μη-λειτουργικές απαιτήσεις, αναδεικνύονται ως κρίσιμης σημασίας ιδιότητες όχι μόνο του σχεδιαστικού αποτελέσματος (π.χ. τεχνουργήματος και διεπαφής), αλλά και της διαδικασίας σχεδίασής τους.

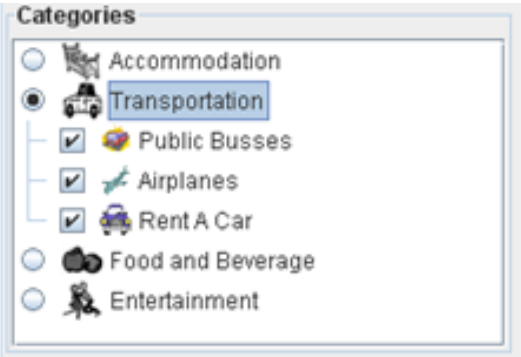
Στρατηγικές Ανάπτυξης Προηγμένων Αντικειμένων

Για την ανάπτυξη εξειδικευμένων αντικειμένων (π.χ.domain specific), όπως αυτών με τα οποία καταπιάνεται η εν λόγω εργασία και για την απόδοση συγκεκριμένων ποιοτικών

χαρακτηριστικών (affordances) σε αυτά (π.χ. κοινωνικής και ομαδικής ενημερότητας, κοινωνικής ημιδιαφάνειας, ενδιάμεσης διάδρασης, κοκ., βλ.), απαιτείται η υιοθέτηση συγκεκριμένων πρακτικών (Platform Administration Practices)[8]. Αυτά αφορούν την ικανότητα: α) επαύξησης των αλληλεπιδραστικών δυνατοτήτων υπαρχόντων αντικειμένων (component augmentation), β) της επέκτασης της αλληλεπιδραστικής ιεραρχίας εργαλειοθηκών (toolkit expansion), γ) της ενσωμάτωσης στοιχείων (integration) και δ) της αφαίρεσης (abstraction) ούτως ώστε τα στοιχεία να είναι πλήρως απεξαρτημένα από τα φυσικά τους χαρακτηριστικά μπορώντας κατ' αυτό τον τρόπο να προσαρμοστούν σε πολλαπλά πλαίσια χρήσης. Οι εν λόγω τακτικές ορίζονται κάτω από τον όρο 'Platform Administration Practices'[8].

Επαύξηση Διαδραστικών Αντικειμένων (Component Augmentation)

Η εν λόγω τακτική χρησιμοποιείται σε περιπτώσεις όπου απαιτείται βελτίωση – εμπλουτισμός της αλληλεπιδραστικής συμπεριφοράς αντικειμένων με στόχο την υποστήριξη αλληλεπιδραστικών ικανοτήτων, στα πλαίσια ικανοποίησης συγκεκριμένων αναγκών, που δεν προβλέπονταν ως μέρος της αρχικής υλοποίησης των στοιχείων.

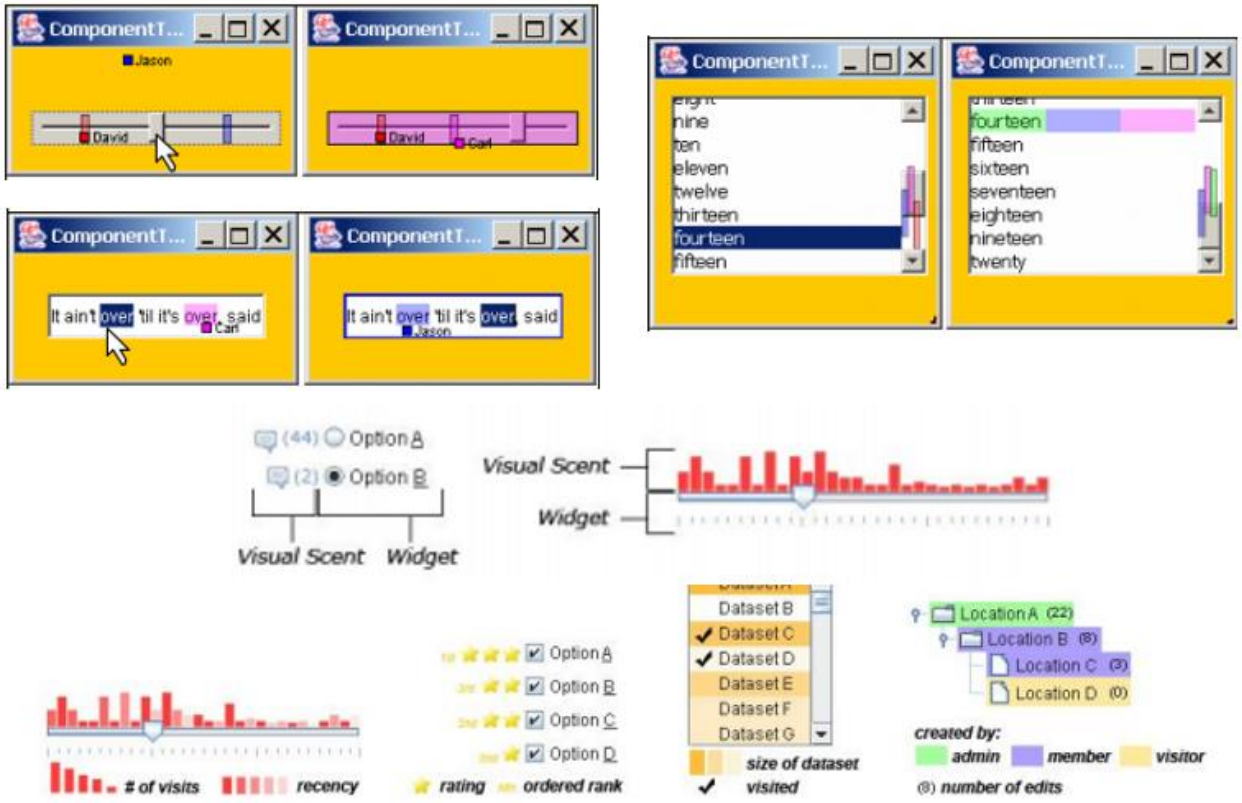


Εικόνα 2: Ενδεικτικό στιγμιότυπο επαύξησης του διαδραστικού αντικειμένου του δέντρου ('RadioCheckTree')

Στην Εικόνα 2, π.χ. παρουσιάζεται μια οικεία επαυξημένη εκδοχή του διαδραστικού στοιχείου του δέντρου (tree) της εργαλειοθήκης του java/swing, ώστε να υποστηρίξει συγκεκριμένες ανάγκες που αφορούν τη δυνατότητα προ-επιλογής (pre-selection) και μετά-επιλογής (meta-selection). Ενδεικτικά παραδείγματα περιπτώσεων επαύξησης αντικειμένων παρουσιάζονται επίσης στην Εικόνα 3. Στο πάνω μέρος απεικονίζεται το αποτέλεσμα της ενσωμάτωσης στους διαλόγους επιλεγμένων αντικειμένων, μέρους πληροφορίας (βλ. ομαδική

ενημερότητα) που στόχο έχει την υποβοήθηση του συντονισμού σύγχρονα και κατανεμημένα συνεργαζόμενων χρηστών.

Εναλλακτικά στην Εικόνα 3, φαίνονται ενδεικτικές αναπαραστάσεις επαυξημένες ώστε να άγουν σχετική με το υποκείμενο πεδίο κάθε φορά πληροφορία για την υποβοήθηση της επιλογής ή πλοήγησης ανάλογα, και γενικότερα τη υποστήριξη αυτών που ονομάζονται βασισμένων στην προαγωγής της δημιουργικότητας διεπαφών.



Εικόνα 3: Ενδεικτικά παραδείγματα περιπτώσεων επαύξησης αντικειμένων

Υποστήριξη για επαύξηση υποστηρίζεται στις περισσότερες εργαλειοθήκες μέσω της κληρονομικότητας, αποτελεί ωστόσο πολύ απαιτητικό καθήκον μιας και προϋποθέτει άριστη γνώση της φιλοσοφίας, αρχιτεκτονικής και δομής της εκάστοτε εργαλειοθήκης-στόχου.

Επέκταση Αλληλεπιδραστικής Ιεραρχίας Εργαλειοθήκης (Toolkit Expansion)

Επέκταση απαιτείται για την εισαγωγή ριζικά νέων διαδραστικών στοιχείων στα πλαίσια μιας συγκεκριμένης αλληλεπιδραστικής εργαλειοθήκης και με τρόπο που να καθιστά μη ορατό ότι πρόκειται περί τέτοιων διαμέσου της διατήρησης του προγραμματιστικού στυλ της

εργαλειοθήκης[9][10]. Η ικανότητα επέκτασης είναι ήσσονος σημασίας, ειδικά στην περίπτωση δημιουργίας νέων τύπων υποδοχέων και μεταφορών (π.χ. βιβλίου, δωματίου, κοκ.). Δεδομένου ότι η επέκταση στηρίζεται στην επινόηση νέων τύπων υποδοχέων εμπλέκει επίσης τη δημιουργία εξειδικευμένων πολιτικών τοπολογίας (topology policies - layouts).

Παραδείγματα τέτοιου είδους υποδοχέων φαίνονται στην Εικόνα 4. Στην πάνω πλευρά βλέπουμε ένα υποδοχέα που υλοποιεί την μεταφορά του πενταγράμμου για την αναπαράσταση μουσικού περιεχομένου, στη κάτω και αριστερή πλευρά ένα οικείο υποδοχέα τύπου γηπέδου ποδοσφαίρου για την υποστήριξη σχετικού σεναρίου, και στη κάτω και δεξιά πλευρά ένα εστιαζόμενο (zoomable) εξειδικευμένο υποδοχέα που δίνει τη δυνατότητα χρονικής αποτύπωσής πληροφορίας (Ημερολόγιο) του τύπου αντικειμένων που περιέχει.

Εικόνα 4: Παραδείγματα επέκτασης εργαλειοθήκης του swing για την εισαγωγή νέου τύπου υποδοχέων

Ενσωμάτωση (Integration)

Η ενσωμάτωση αφορά εξέχουσας σημασίας ιδιότητα στη συναρμολόγηση διεπαφών υπό την έννοια της υποστήριξης μίξης αντικειμένων με βάση ένα προτυποποιημένο τρόπο a)

ανεξαρτήτως προγραμματιστικού μοντέλου στο οποίο υπακούει κάθε widget, b) της εργαλειοθήκης στην οποία βασίζεται και c) των γλωσσών προγραμματισμού με βάση τα οποία είναι υλοποιημένα. Στην πραγματικότητα, η δυνατότητα της ενσωμάτωσης εξυπηρετεί την ανάγκη αξιοποίησης εναλλακτικών διαδραστικών βιβλιοθηκών, ανεξαρτήτως περιβάλλοντος υλοποίησης, με ενιαίο τρόπο. Στο πρόσφατο παρελθόν υπήρξαν προσπάθειες βελτίωσης της δυνατότητας ενσωμάτωσης μιας βιβλιοθήκης σε μια άλλη, έτσι ώστε να λειτουργούν ταυτόχρονα και στην περίπτωση αλληλοεξαρτήσεων, (βλέπε γραφικές εργαλειοθήκες και συμβατικές κλάσεις αντικειμένων), αλλά τα αποτελέσματα δεν γενικεύονται να καλύψουν κάθε είδους διαδραστικό αντικειμένου κυρίως όταν πρόκειται για μη-εγγενώς υποστηριζόμενα (δηλ. non-native).

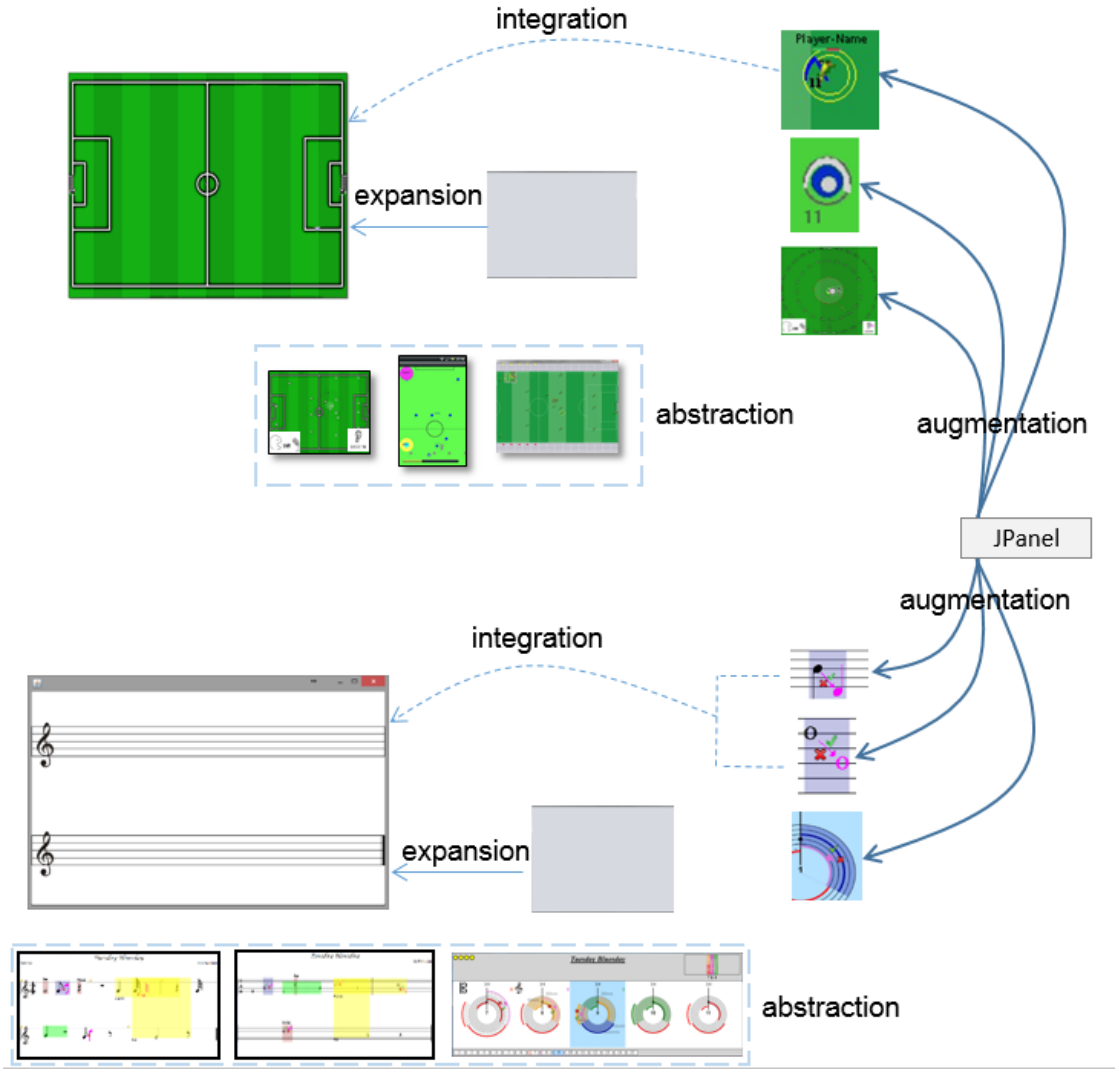
Αφαίρεση (Abstraction)

Η υποστήριξη κατάλληλου επιπέδου αφαίρεσης απαιτεί ειδική πρόνοια ώστε τα στοιχεία μιας διεπαφής να παρέχονται με τρόπο που να είναι πλήρως απαλλαγμένα των φυσικών τους χαρακτηριστικών, καθώς μόνο τότε μπορούν να προσαρμοστούν αναλόγως σε σχέση με διαφορετικά προφίλ χρηστών και πλαισίων χρήσης. Κατά το πρόσφατο παρελθόν, εργαλειο-κεντρικές προσεγγίσεις στην ανάπτυξη διεπαφών συχνά διεκδικούσαν την ιδιότητα της αφαίρεσης, ωστόσο η πρακτική υποδεικνύει ότι στην πραγματικότητα αυτό που υποστήριζαν ήταν κάποιας μορφή γενίκευση.

Σύνθεση τεχνικών και δυνατότητες

Στο σημείο αυτό ιδιαίτερο ενδιαφέρον παρουσιάζει η συμπληρωματική χρήση των ανωτέρω τακτικών έτσι ώστε να καθίσταται εφικτό να σχεδιαστούν και να υλοποιηθούν διεπαφές που να αξιοποιούν ταυτόχρονα επαύξηση, επέκταση και ενσωμάτωση βιβλιοθηκών με ενιαίο και αφηρημένο τρόπο. Δυστυχώς, παρότι υπάρχουν αρκετές προσεγγίσεις η δυσκολία παραμένει υψηλή και αυτό μπορεί εύκολα να διαπιστωθεί με ένα συγκεκριμένο παράδειγμα. Συγκεκριμένα, στο πάνω μέρος της διεπαφής στην Εικόνα 5, παρατηρούμε το αποτέλεσμα της εμπλοκής των ανωτέρω τακτικών για την υλοποίηση ενός ενδεικτικού σεναρίου που αφορά το παίγνιο του ποδοσφαίρου (Soccer game). Αναλυτικότερα, παρατηρούμε ενδεικτική επέκταση της εργαλειοθήκης του java/swing ώστε να υποστηρίξει ένα νέο τύπο υποδοχέα, αυτό του ‘Γήπεδο Ποδοσφαίρου’ (Soccer Field). Δεδομένου ότι, στα πλαίσια του οραματιζόμενου σεναρίου χρήσης, το εν λόγω παίγνιο προορίζεται για χρήση υπό ριζικά διαφορετικές συνθήκες:

α) ακουστική αλληλεπίδραση σε περίπτωση εμπλοκής τυφλού χρήστη και β) αποτύπωση με τρόπο ενδεδειγμένο για εκτέλεση διαμέσου κινητών συσκευών τύπου android στην περίπτωση κινητού χρήστη, θα έπρεπε να επεκταθούν και επεκτάθηκαν με ακριβώς το ίδιο σκεπτικό και οι εργαλειοθήκες ενός οικείου non-visual toolkit, αλλά και της εργαλειοθήκης του Android.



Εικόνα 5: Ενδεικτικά σενάρια χρήσης των στρατηγικών διαχείρισης πλατφόρμας

Σε κάθε περίπτωση, οι επεκτάσεις δεν προνόησαν μόνο ως προς τη διαχείριση ζητημάτων που αφορούν τη γραφική απεικόνιση των υποδοχέων, αλλά καταπιάστηκαν επίσης και θέματα που αφορούν την αλγοριθμική υλοποίηση της υποστηριζόμενης τοπολογίας. Επιπλέον, οι αναφερόμενοι υποδοχείς φέρουν εκτός των κοινών και ιδιαίτερα (μεμονωμένα – πολυμορφικά) χαρακτηριστικά για την εξυπηρέτηση αναγκών του πλαισίου χρήσης για το οποίο προορίζονται. Παραδείγματος χάριν, στην περίπτωση του swing, ο υποδοχέας υποστηρίζει

κοινωνική ενημερότητα (social awareness), διαμέσου των κερκίδων που υλοποιεί. Όσο περισσότερες οι ενεργές κερκίδες (αποτέλεσμα επαύξησης και ενσωμάτωσης), τόσο περισσότεροι οι παρατηρητές και το ενδιαφέρον στον ανάλογο αγώνα. Στην περίπτωση του Non-visual toolkit, το γήπεδο χωρίζεται σε συγκεκριμένο αριθμό νοητών γραμμών και στηλών ορίζοντας κελία, υποδεικνύοντας με αυτό τον τρόπο και άγοντας πληροφορία στο χρήστη σχετικά με τη θέση του παίχτη στο γήπεδο. Από την άλλη στην περίπτωση του android, υποστηρίζει μεταξύ άλλων τη γραφική αποτύπωση της επιλεγμένης τροχιάς κίνησης του παίχτη με βάση τη χειρονομία του χρήστη του παιγνίου.

Με την ίδια λογική ανάλογα διαδραστικά αντικείμενα των σχετικών εργαλειοθηκών (java/swing, NVT και Android/UI) έχουν επεκταθεί ώστε να υποστηρίξουν το αλληλεπιδραστικό στοιχείο του παίχτη ποδοσφαίρου (Soccer Player). Σε κάθε περίπτωση τα ποιοτικά χαρακτηριστικά των εναλλακτικών υλοποιήσεων διαφέρουν έτσι ώστε να ικανοποιήσουν τις ιδιαίτερες απαιτήσεις του πλαισίου χρήσης για να το οποίο προορίζεται η χρήση τους. Ομοίως και στην περίπτωση του ενδεικτικού σεναρίου που παρουσιάζεται στο κάτω μέρος της Εικόνα 5. Σε αυτό το πλαίσιο βλέπουμε την επαύξηση και επέκταση εργαλειοθηκών για την υποστήριξη εναλλακτικών μεταφορικών αναπαραστάσεων για την απεικόνιση μουσικού περιεχομένου. Και εδώ παρατηρούμε τις ιδιαίτερες ανάγκες που οδήγησαν ανά περίπτωση σε ριζικά ετερογενείς διεπαφές, τόσο σε επίπεδο γλώσσας προγραμματισμού, εργαλειοθηκών, φυσικών, συντακτικών, λεκτικών ιδιοτήτων, κοκ..

Από τα παραπάνω παραδείγματα προκύπτει η ιδέα μιας μορφής συναρμολόγησης εκδόσεων διεπαφής που παρότι διαφέρουν σε φυσικά χαρακτηριστικά, έχουν κοινό τύπο προγόνου. Ας σημειωθεί επίσης ότι οι επίγονοι του υποδοχέα δεν προέκυψαν τυχαία αλλά από την εφαρμογή τακτικών ανάπτυξης όπως αυτές που αναφέρθηκαν προηγούμενα. Το ερώτημα είναι κατά πόσο αυτή η έννοια της συναρμολόγησης μπορεί και με ποιο τρόπο να γενικευτεί και να καταστεί καθολική, ανεξαρτήτως πλατφόρμας ή τεχνολογικής γενιάς. Το ερώτημα αυτό αναλύεται και εξειδικεύεται παρακάτω.

Επισκόπηση προβλήματος

Η πρόσφατη εμπειρία στη διαχείριση συλλογών αντικειμένων καταδεικνύει ότι η επιστράτευση των παραπάνω τακτικών με την τρέχουσα γενιά τεχνολογικών εργαλείων εργαλειοθηκο-κεντρικού προγραμματισμού έχει σειρά επιπτώσεων. Συγκεκριμένα:

a. Επιφέρουν ιδιαιτέρως αυξημένο βαθμό πολυπλοκότητας μιας και εμπλέκουν την ανάγκη για δημιουργία και διαχείριση αφηρημένων υψηλού επιπέδου (abstract top-level) υποδοχέων,
b. Απαιτούν τον ορισμό κατάλληλων και εξειδικευμένων διαχειριστών διάταξης (layout managers), για την υλοποίηση κάθε φορά της επιθυμητής στρατηγικής υποδοχής (containment strategy) (βλ. π.χ. μεταφορά δωματίου),
c. Συνεπάγονται χρήση μη-εγγενώς υποστηριζόμενων αντικειμένων, περιορίζοντας τη φορητότητά τους κατά μήκος διαφορετικών πλατφορμών, μιας και τα οικεία αντικείμενα πιθανότατα δεν υποστηρίζονται σε άλλες πλατφόρμες.

Από την άλλη πλευρά, εναλλακτικές προσεγγίσεις ανάπτυξης διεπαφών όπως η μοντελοκεντρική μηχανική προσφέρουν μεν μια αδιαμφισβήτητα καλύτερη βάση για την επίλυση του προβλήματος, αλλά υστερεί σε εργαλεία που αντιμετωπίζουν επιτυχώς τις παραπάνω δυσλειτουργίες. Τα βασικά πλεονεκτήματα απορρέουν από την εναπόθεση στη χρήση αφηρημένης σημειολογίας και γλωσσών σήμανσης, επιτρέποντας τον ορισμό αφηρημένων αντικειμένων και τη μετέπειτα διασύνδεσή τους με διαδραστικά λεξικά υποστηριζόμενα από την πλατφόρμα-στόχο. Τέτοιου είδους διασυνδέσεις προϋποθέτουν σχήματα μετασχηματισμών τα οποία έχουν ως αποτέλεσμα την ανάθεση της παρουσίασης σε σχετικούς με την πλατφόρμα renderers (platform-specific renderers). Ωστόσο, έχοντας ως αποκλειστικό γνώμονα την υποστήριξη φορητότητας και στόχο την απρόσκοπτη εκτέλεση των αφηρημένων specifications τους, οι μοντελο-κεντρικές προσεγγίσεις περιορίζονται στην υποστήριξη αποκλειστικά και μόνο εκείνων των διαδραστικών στοιχείων που είναι εγγενώς υποστηριζόμενα από τις πιο δημοφιλείς εργαλειοθήκες. Αυτό συνεπάγεται πως η εκφραστική τους επάρκεια περιορίζεται στην υποστήριξη αποκλειστικά και μόνο απλών (form-based) διεπαφών.

Κίνητρα και Συνεισφορά

Η παρούσα εργασία στοχεύει να αντιμετωπίσει τις αδυναμίες που χαρακτηρίζουν τις παραπάνω προσεγγίσεις. Συγκεκριμένα στη βάση υιοθέτησης μιας μοντελο-κεντρικής οπτικής, η οποία θεωρείται ως το ενδεδειγμένο πλαίσιο αναφοράς ως προς την υποστήριξη φορητότητας επιδιώκεται η αντιμετώπιση των παρακάτω αδυναμιών:

- Η μοντελο-κεντρική μηχανική διεπαφών υποθέτει συμβατικά εγγενώς υποστηριζόμενα λεξικά, ως εκ τούτου δεν υποστηρίζει καινοφανή ή οικεία widgets υλοποιημένα στην κορυφή εγγενών αντικειμένων.
- Η μοντελο-κεντρική μηχανική διεπαφών δεν υποστηρίζει ανάμεικτες (mixed) αλληλεπιδραστικές ιεραρχίες συνδυάζοντας διαδραστικά αντικείμενα από διαφορετικές εργαλειοθήκες (π.χ. Swing, JGraph, prefuse).

Τα παραπάνω παρακινούν την εν λόγω εργασία, η οποία επεκτείνει την τρέχουσα γενιά μοντελο-κεντρικών μεθόδων μηχανικής διεπαφών εξετάζοντας:

- Τεχνικές για τη μοντελοποίηση των αλληλεπιδραστικών δυνατοτήτων (affordances) κατά τρόπο συμβατό με μοντελο-κεντρικά πλαίσια ανάπτυξης (model-driven development framework), όπως η UsiXML.
- Νέες τεχνικές για την προδιαγραφή (εν αντιθέσει του προγραμματισμού) και μοντελοποίηση διαδραστικών συμπεριφορών έτσι ώστε να αξιοποιούν καινοφανή διαδραστικά affordances τα οποία μπορεί να έχουν υλοποιηθεί διαμέσου οικείων σχετικών-πλατφόρμας widgets
- Νέα εργαλεία για τη βελτίωση της ικανότητας των υπαρχόντων μοντελο-κεντρικών πλαισίων ανάπτυξης, όπως η UsixML, να χρησιμοποιούν (δηλ. link-to) υλοποιημένα (non-native) διαδραστικά αντικείμενα.

Δομή και οργάνωση εργασίας

Το υπόλοιπο της εργασίας δομείται ως εξής: Στο **Κεφάλαιο 2**, εξετάζονται οι δυνατότητες των state of the art γλωσσών περιγραφής αντικειμένων (widgets) και μεθόδων περιγραφής διεπαφών. Επίσης, γίνεται αναφορά στη φιλοσοφία της προτεινόμενης προσέγγισης και στην ανάδειξη μειονεκτημάτων των δημοφιλέστερων μεθόδων υπολογιστικής υλοποίησης διεπαφών. Στο **Κεφάλαιο 3**, περιγράφονται τα βασικά χαρακτηριστικά και οι δυνατότητες μιας νέας γλώσσας περιγραφής widgets, WSL (Widget Specification Language), ενώ αιτιολογούνται οι σχεδιαστικές επιλογές με βάση γνωστές αδυναμίες των γνωστών μεθόδων ανάπτυξης και απαιτήσεις. Στο **Κεφάλαιο 4**, παρουσιάζονται εργαλεία που έχουν υλοποιηθεί στα πλαίσια της νέας γλώσσας (WSL) και πως αυτά μπορούν να αλλάξουν τον τρόπο με τον οποίο σχεδιάζουμε και παράγουμε διεπαφές. Στο **Κεφάλαιο 5**, μέσα από ενδεικτικά και πολύ απαιτητικά σενάρια χρήσης αναδεικνύονται οι δυνατότητες που απορρέουν από τη συγκεκριμένη προσέγγιση και τα

εργαλεία της. Τέλος, στο **Κεφάλαιο 6**, γίνεται σύνοψη της συγκεκριμένης προσέγγισης αναδεικνύοντας τις δυνατότητες της και πως αυτές απαντούν στα ερευνητικά ερωτήματα που τέθηκαν.

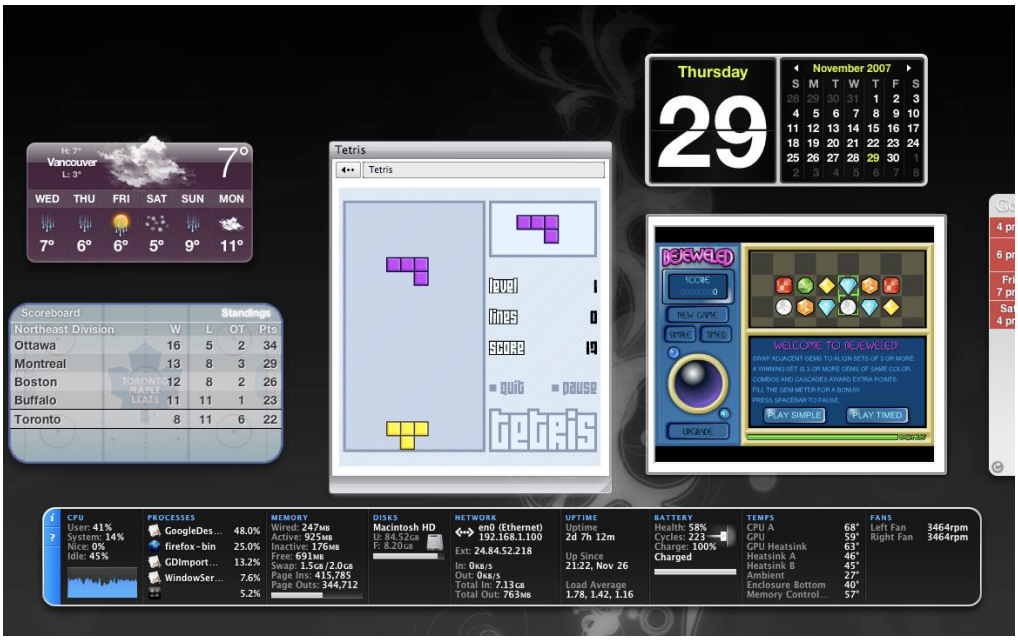
Κεφάλαιο 2 – State of the Art

Γλώσσες Περιγραφής Αντικειμένων

Οι γλώσσες περιγραφής αντικειμένων αφορούν την περιγραφή βασικών χαρακτηριστικών αυτόνομων εφαρμογών μικρής κλίμακας, ενδεδειγμένων για χρήση είτε στο web είτε στις πιο δημοφιλείς πλατφόρμες PC (Windows και MacOS). Η υλοποιούμενη λειτουργικότητα περιορίζεται εκ φύσεως στην υποστήριξη απλών καθηκόντων όπως παρουσίαση δεδομένων καιρού, νέων, σημειωματάριο, rss feeds, ρολόι, ημερολόγιο, κοκ. Η τεχνολογίες περιγραφής των διεπαφών περιορίζονται στο σύνολο στην HTML, CSS και JavaScript. Σε ορισμένες περιπτώσεις, υποστηρίζεται και η γλώσσα flash (βλ. Google gadgets). Σε όλες τις περιπτώσεις τα αντικείμενα ορίζονται στα πλαίσια ενός archive (συμπιεσμένου αρχείου με γνωστή επέκταση), και αποτελούνται από την υλοποίηση της διεπαφής (html code) και ένα xml αρχείο στο οποίο περιγράφονται βασικές πληροφορίες που αφορούν χαρακτηριστικά τους, όπως, α) όνομα, β) περιγραφή, γ) ελάχιστη έκδοση runtime για την εκτέλεσή τους, και σε καμία περίπτωση η οποιαδήποτε αναφορά ή κωδικοποίηση τμήματος του API τους. Για την εκτέλεσή τους απαιτείται η χειροκίνητη προ-εγκατάσταση ειδικών περιβαλλόντων εκτέλεσης (widget servers). Από σχεδίασής τους, δεν υποστηρίζεται η οποιαδήποτε δυνατότητα επικοινωνίας ή διασύνδεσης αναμεταξύ τους.

Οι πιο γνωστές γλώσσες περιγραφής widgets είναι τα Google Gadjets, τα Yahoo! Widgets, και α opera widgets. Τα google gadgets ενδείκνυνται για χρήση σε περιβάλλον web διαμέσου ενός plugin που πρέπει να εγκατασταθεί πρώτα στον browser του χρήστη, τα yahoo ! widgets ενδείκνυνται για χρήση σε περιβάλλον σταθερού υπολογιστή με λειτουργικό σύστημα Windows ή MacOS και τα Opera widgets τα οποία επίσης ενδείκνυνται για χρήση σε περιβάλλον PC.

Σε καμία περίπτωση οι εν λόγω γλώσσες περιγραφής δεν μπορούν να υποστηρίξουν την κωδικοποίηση αντικειμένων των συμβατικών εργαλειοθηκών, όπως της java/swing, SWT, Qt, κοκ.. Όπως γίνεται σαφές από τις προδιαγραφές τους, οι σχετικές γλώσσες ενδείκνυνται κατ' αποκλειστικότητα για: α) την απλή περιγραφή αυτόνομων εφαρμογών και όχι αλληλεπιδραστικών αντικειμένων κλάσεων, β) την υποστήριξη μόνο web προτύπων για την περιγραφή της διεπαφής, γ) και τη συμπίεση και δόμηση της εσωκλειόμενης πληροφορίας τους, με τρόπο που να είναι διαχειρίσιμο από ενδεδειγμένα runtime environments.



Εικόνα 6: Google Gadgets

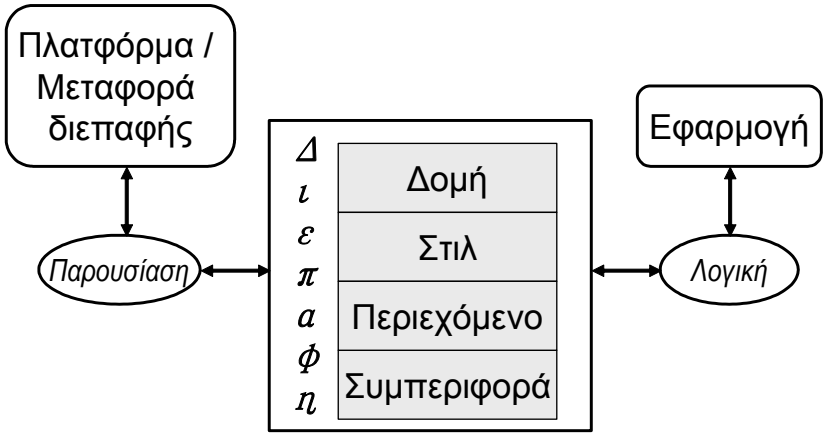
Εικόνα 7: Yahoo! widgets

Γλώσσες βασισμένες στη μοντελοκεντρική μηχανική διεπαφών

Η ανάπτυξη γλωσσών σήμανσης ανεξάρτητων συσκευής, όπως η XML, επέτρεψε την ανάπτυξη εξειδικευμένων εργαλείων προδιαγραφής διεπαφών υπό τον τύπο μοντέλων που αναπαριστούν δομές και δυνατότητες που παραδοσιακά 'φωλιάζονται' σε βιβλιοθήκες διαδραστικών αντικειμένων. Οι γλώσσες αυτές συχνά ακολουθούν διαφορετικές προσεγγίζεις, όπως παρουσιάζεται παρακάτω, γεγονός που δρομολόγησε και προσπάθειες για την ανάπτυξη ενοποιημένων πλαισίων αναφοράς (π.χ. CAMELEON).

UIML

Η UIML[4][24] είναι μια σχετικά πρόσφατη εξέλιξη προς την κατεύθυνση δημιουργίας de facto προτύπων γλωσσών περιγραφής διεπαφών ανεξαρτήτου συσκευής[18]. Για την κατασκευή μιας διεπαφής απαιτείται η σύνταξη ενός εγγράφου UIML το οποίο περιλαμβάνει δομές παρουσίασης κατάλληλες για τις συσκευές για τις οποίες προορίζεται η διεπαφή. Στη συνέχεια, το έγγραφο της UIML αντιστοιχίζεται αυτόματα στη γλώσσα που χρησιμοποιεί η συγκεκριμένη συσκευή — για παράδειγμα στην XHTML, τη WML, τη VoiceXML, τη C++ (με τη βοήθεια μιας βιβλιοθήκης όπως η MFC), τη Java (με μια βιβλιοθήκη όπως το swing), κ.ο.κ. Το ιδιαίτερο χαρακτηριστικό της UIML είναι ότι διαχωρίζει τη διεπαφή σε έξι τμήματα (βλέπε Εικόνα 8), αντί των κλασικών τριών του μοντέλου MVC.



Εικόνα 8: Το μοντέλο της UIML[5]

[4] http://www.uiml.org/

[5] Από [18].

Συγκεκριμένα, μια διεπαφή προσεγγίζεται ως ένα σύνολο στοιχείων (elements) με τα οποία αλληλεπιδρά ο χρήσης. Κάθε στοιχείο της διεπαφής ονομάζεται τμήμα (part), με την ίδια λογική που ένα μηχανοκίνητο όχημα αποτελείται από τμήματα. Τα τμήματα της διεπαφής μπορεί να οργανώνονται διαφορετικά για διαφορετικές κατηγορίες τελικών χρηστών όπως και για διαφορετικές οικογένειες τελικών συσκευών. Κάθε τμήμα της διεπαφής έχει περιεχόμενο (π.χ. κείμενο, ήχους, εικόνες, κοκ.), το οποίο χρησιμοποιείται για την επικοινωνία πληροφορίας προς το χρήστη. Κάποια τμήματα της διεπαφής μπορεί να λαμβάνουν είσοδο από το χρήστη. Αυτό επιτυγχάνεται διαμέσου της χρήσης τεχνουργημάτων όπως μιας λίστας, κουμπιών, κοκ. Μιας και τα τεχνουργήματα ποικίλουν από συσκευή σε συσκευή, η ουσιαστική διασύνδεση (mapping) μεταξύ ενός τμήματος της διεπαφής και του συσχετιζόμενου τεχνουργήματος (widget), γίνεται διαμέσου τη χρήσης άλλων στοιχείων της UIML, τα οποία ορίζονται ανεξάρτητα (μέσω των <presentation> ή <propery> ετικετών). Ο προσδιορισμός μιας διεπαφής στα πλαίσια της UIML απαντά στα ακόλουθα έξι ερωτήματα:

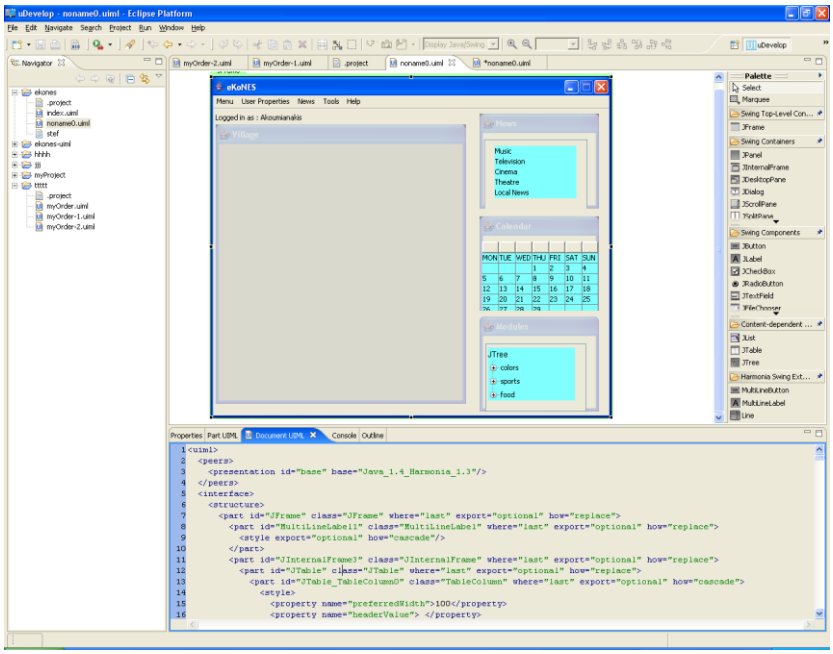
- Ποια τμήματα (parts) απαρτίζουν (*Δομή*) τη διεπαφή?
- Τι *στυλ* παρουσίασης πρέπει να συσχετιστεί με κάθε τμήμα ? (μέγεθος φόντου, χρώμα, κοκ.)
- Τι *περιεχόμενο* για κάθε τμήμα? (κείμενο, ήχοι, εικόνες, κοκ)
- Τι συμπεριφορά έχουν τα τμήματα?
- Πως συνδέονται (connect) με το περιβάλλον της διεπαφής? (business logic, πηγές δεδομένων, αλληλεπιδραστική εργαλειοθήκη)
- Πως διασυνδέονται με μια εργαλειοθήκη στόχο?

Σήμερα βρίσκονται σε εξέλιξη σειρά ενεργειών που αποβλέπουν στην υποστήριξη της ανάπτυξης διεπαφών με την χρήση της UIML. Ενδεικτικά παραδείγματα τέτοιων δράσεων είτε υλοποιούνται στα πλαίσια είτε υποστηρίζονται από την κοινοπραξία Harmonia[6]. Αν και τα εργαλεία αυτά δεν είναι ευρέως διαθέσιμα ούτε ολοκληρωμένα συνεισφέρουν ουσιαστικά στην γρήγορη ανάπτυξη διεπαφών.

Ενδεικτικά, θα περιγράψουμε ένα από τα εργαλεία αυτά το οποίο διατίθεται μέσω της κοινοπραξίας Harmonia. Το uDevelop είναι ένα εργαλείο γρήγορης ανάπτυξης διεπαφών με την χρήση της UIML. Στην Εικόνα 9, παρουσιάζεται η βασική διεπαφή του εργαλείου η οποία είναι

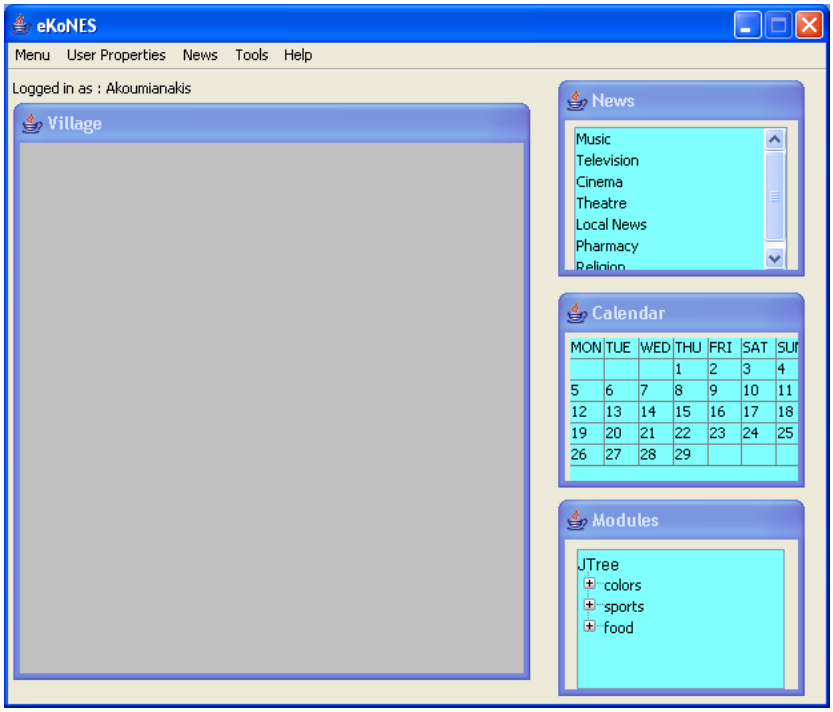
[6] http://www.harmonia.com/

παρόμοια με διεπαφές άλλων εργαλείων γρήγορης ανάπτυξης διεπαφών. Στο παράδειγμα της Εικόνα 9, συνοψίζεται μια πρωτότυπη διεπαφή για το έργο eKoNEΣ[25].



Εικόνα 9: Στιγμιότυπο του εργαλείου uDevelop

Στο κάτω υπο-παράθυρο που απεικονίζεται διαμέσου της Εικόνα 9, παρουσιάζεται τμήμα του αρχείου UIML από το οποίο μπορεί να δημιουργηθεί η τελική διεπαφή (βλ. Εικόνα 10).



Εικόνα 10: Πρωτότυπο διεπαφής με χρήση της UIML

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

```
<uiml>
  <peers>
    <presentation id="base" base="Java_1.4_Harmonia_1.3"/>
  </peers>
  <interface>
      <structure>
            <part id="JFrame" class="JFrame" where="last"
            export="optional" how="replace">
            <part id="MultiLineLabel1" class="MultiLineLabel" where="last"
            export="optional" how="replace">
            <style export="optional" how="cascade"/>
            </part>
            …
      </structure>
      <style>
            <property part-name="JFrame" name="size"
                  export="optional"
                  how="replace">620,527</property>
            <property part-name="JFrame" name="title"
                  export="optional"
                  how="replace"><[CDATA[eKoNES]></property>
            <property part-name="JFrame" name="layout"
                  export="optional"
                  how="replace">null</property>
<property part-name="JFrame" name="location">
63,8</property>
…
```

Η αντίστοιχη προδιαγραφή για τη λίστα στο εσωτερικό παράθυρο με τον τίτλο «News» είναι:

```
<part id="JList1" class="JList" where="last" export="optional" how="replace">
      <style>
            <property part-name="JList1"
                  name="location">15,9</property>
            <property part-name="JList1"
                  name="size">138,96</property>
            <property part-name="JList1" name="content">
                  <constant model="list">
                  <constant value="Music"/>
                  <constant value="Television"/>
                  <constant value="Cinema"/>
                  <constant value="Theatre"/>
                  <constant value="Local News"/>
                  <constant value="Pharmacy"/>
                  <constant value="Religion"/>
                  </constant>
            </property>
            <property part-name="JList1"
                  name="debugGraphicsOptions">FLASH_OPTION</property>
```

```
            <property part-name="JList1" name="layout">null</property>
            <property part-name="JList1"
            name="doubleBuffered">true</property>
            <property part-name="JList1"
            name="background">128,255,255</property>
      </style>
</part>
```

Οι παραπάνω προδιαγραφές είναι αποσπάσματα από το UIML αρχείο που αφορούν τις συγκεκριμένες συνιστώσες που αναφέραμε. Βοηθούν όμως στο να παρατηρήσουμε και στις δύο περιπτώσεις το λογικό επίπεδο στο οποίο εκφράζονται οι προδιαγραφές της UIML και ο σαφής διαχωρισμός των προδιαγραφών αυτών από τις υλοποιημένες κλάσεις μιας πλατφόρμας. Αυτό επιτρέπει στη UIML την αυτόματη μετατροπή / αντιστοίχιση προδιαγραφών σε συνιστώσες διαφορετικής πλατφόρμας γεγονός που εξυπηρετεί την ανάπτυξη πολλαπλών διεπαφών.

XIML

Η XIML[23] είναι μια γλώσσα περιγραφής διεπαφών η οποία αρχικά αναπτύχθηκε από το ερευνητικό εργαστήριο ‘RedWhale Software Corporation’. Πλέον υποστηρίζεται από το XIML φόρουμ (http://www.ximl.org/), ένα επιχειρηματικό (industrial) οργανισμό ο οποίος ασχολείται με την έρευνα, διάχυση, υιοθέτηση και προτυποποίηση της XIML. Η XIML, διαιρεί την περιγραφή (definition) μιας διεπαφής σε αντικείμενα (components) και υψηλού-επιπέδου δομικά στοιχεία μιας διεπαφής:

- Καθήκοντα (Tasks): business processes
- Πεδίο (Domain): Ορίζει μια ιεραρχία αντικειμένων (components)
- Χρήστη (User): Ορίζει μια ιεραρχία τελικώ-χρηστών
- Παρουσίαση (Presentation) και Διάλογο (Dialogue): Ορίζει τις δράσεις (actions) στα πλαίσια της διεπαφής.

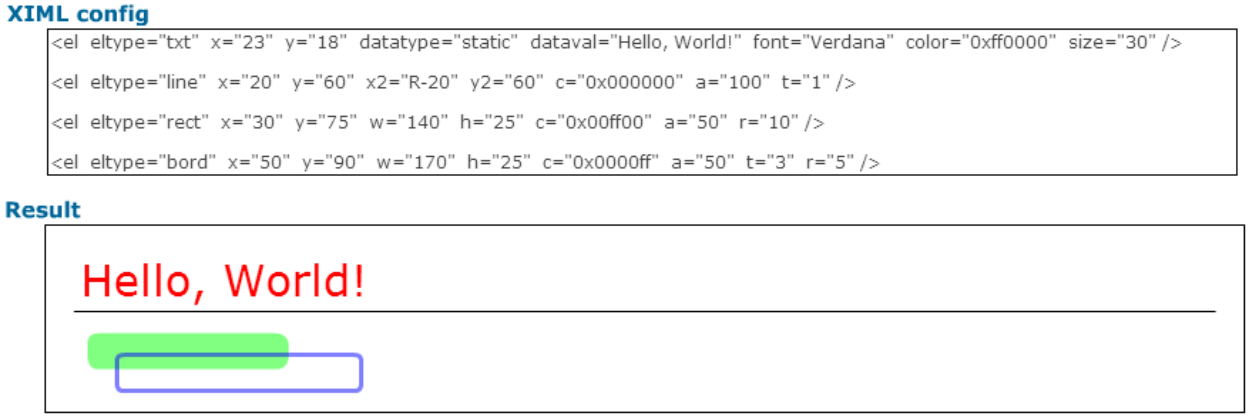
Τα αντικείμενα (components) αντιστοιχίζονται στη συνέχεια σε στοιχεία (concrete αναπαραστάσεις όπως widgets.), χρησιμοποιώντας:

- Συσχετίσεις (Relations)
- Attributes
- Statements
- Definitions

Στα μειονεκτήματα της XIML, συγκαταλέγονται:

- Δεν είναι εύκολη η συλλογή διαθέσιμης πληροφορίας σχετικά με τον ακριβή ορισμό της γλώσσας διαμέσου δωρεάν πηγών του διαδικτύου.
- Βασίζεται στην παραγωγή του εκτελέσιμου κώδικα κατά τη φάση της σχεδίασης και στη διερμήνευση κώδικα εν εκτελέσει, και δεν υπάρχουν υποστηρικτικά εργαλεία διαθέσιμα.
- Άλλο πιθανό πρόβλημα αφορά στο ότι δεν υποστηρίζει επεκτάσεις τριών (3rd party extensions) ή συγχρονισμό της κατάστασης της διεπαφής.
- Είναι ασαφές πως αυτά τα αντικείμενα αλληλεπιδρούν μεταξύ τους.

Στην Εικόνα 11, βλέπουμε την περιγραφή μιας απλής διεπαφής (‘Hello World’) σε XIML και το εν εκτελέσει αποτέλεσμά της.



Εικόνα 11: Ενδεικτικό παράδειγμα αντιστοίχισης XIML περιγραφής και παραγόμενης διεπαφής

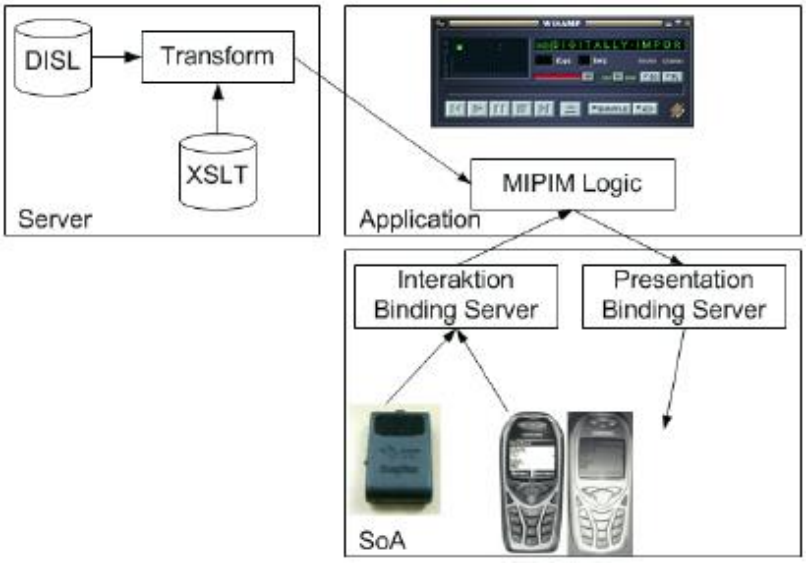
DISL[7]

Η DISL (Dialogue and Interface Specification Language)[19] αποτελεί υποσύνολο της UIML και στοχεύει στην υποστήριξη γενικευμένων (generic) και ανεξάρτητων καναλιού (modality-independent) περιγραφών διαλόγου (dialogue descriptions). Οι διαφοροποιήσεις από τη UIML αφορούν κυρίως την περιγραφή γενικευμένων αντικειμένων (generic widgets) και βελτιώσεις που σχετίζονται με τη συμπεριφορά των διαδραστικών αντικειμένων (behavioral aspects). Τα γενικευμένα αντικείμενα εισήχθησαν έτσι ώστε να υποστηρίξουν το ρητό διαχωρισμό της παρουσίασης (presentation) από τη δομή (structure) και της συμπεριφορά (behavior), δηλαδή την απομόνωση των ιδιοτήτων σχετικών με το χρήστη και τη συσκευή και των εμπλεκόμενων

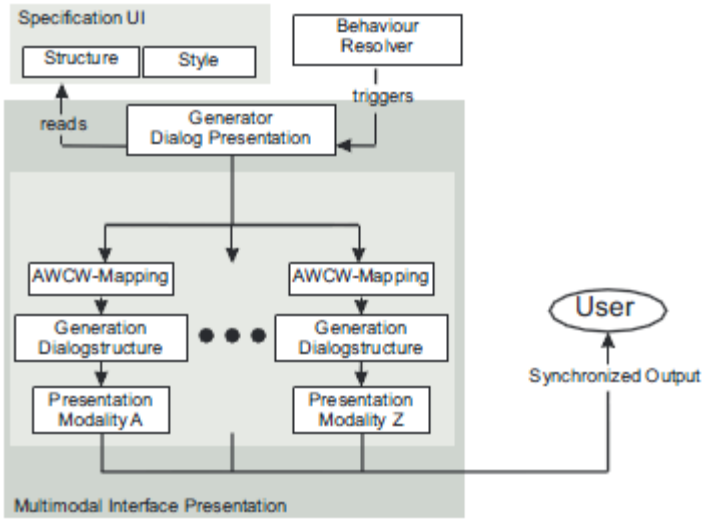
[7] http://it-bleul.de/publications/wm_rs_sb_hiccs37.pdf

καναλιών (modality channels) από μια ανεξάρτητη-καναλιών παρουσίαση. Η χρήση της γενικευμένης ιδιότητας αντικειμένου (generic widget attribute), επιτρέπει τη συσχέτιση κάθε widget με ένα συγκεκριμένο είδος λειτουργικότητας (π.χ. εντολή – command, πεδίο κειμένου – text field, κοκ.). Μηχανές ενδεδειγμένες ω προς την εκτέλεση των παραγόμενων προδιαγραφών (rendering engines), μπορούν να διερμηνεύσουν την κωδικοποιημένη σε αυτές πληροφορία, ώστε να δημιουργήσουν αντικείμενα της διεπαφής (interface components), που να δύνανται να χρησιμοποιηθούν στα πλαίσια του αλληλεπιδραστικού καναλιού που επιβάλλεται από το περιβάλλον εκτέλεσης.

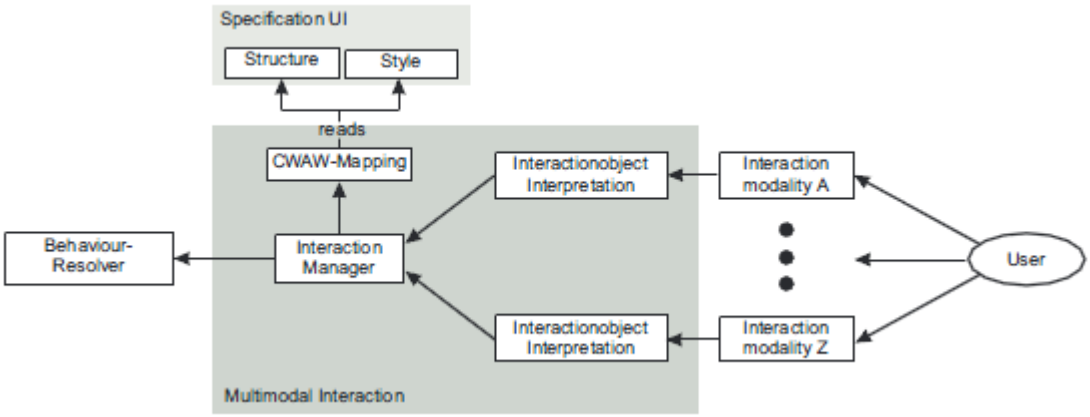
Η συνολική δομή ενός DISL εγγράφου αποτελείται από μια προαιρετική ετικέτα '*Head*', η οποία αφορά μέτα-πληροφορίες (meta-data) και μια συλλογή από πρότυπα (templates), και διεπαφές (interfaces), από τις οποίες μόνο μια διεπαφή μπορεί να καταστεί ενεργή κάθε χρονική στιγμή. Οι διεπαφές χρησιμοποιούνται για να περιγράψουν το διάλογο (Dialogue), τη δομή (structure), στιλιστικές πληροφορίες (style) και τη συμπεριφορά (behavior), ενώ τα πρότυπα (templates), μόνο για να περιγράψουν τη δομή (structure) και στιλιστικές πληροφορίες (style), έτσι ώστε να δύναται να επαναχρησιμοποιηθούν κατά μήκος και άλλων αντικειμένων διαλόγου (dialogue components). Τρέχουσες υλοποιήσεις υποστηρίζουν 'media players' για αρχεία τύπου mp3 σε κινητές συσκευές με περιορισμένους πόρους ή στιγμιότυπα ενός 'player' ενδεδειγμένου για χρήση σε συμβατικούς υπολογιστές (PC) ελεγχόμενου εξ αποστάσεως διαμέσου κινητής συσκευής.



Εικόνα 12: Στιγμιότυπο διεπαφής ενδεικτικού σεναρίου χρήσης



Εικόνα 13: The presentation component της DISL



Εικόνα 14: Interaction component της DISL

TeresaXML

Η TeresaXML[20] είναι μια συμβατή με XML γλώσσα (XML compliant) περιγραφής διεπαφών (User Interface Description Language -UIDL), η οποία αναπτύχθηκε στα πλαίσια του ερευνητικού έργου 'Teresa', από την ομάδα HCI του ISTI-C.N.R.[8]. Η Teresa παρέχει ένα ενοποιημένο περιβάλλον το οποίο υποστηρίζει τη σχεδίαση και παραγωγή 'Concrete' Διεπαφών για ένα φάσμα προκαθορισμένων πλατφορμών (PC, HTML-Web, Mobile).

Η γλώσσα TeresaXML, αποτελείται από δύο τμήματα: α) μια XML περιγραφή της CTT σημειογραφίας (notations), που είναι η πρώτη XML γλώσσα για μοντέλα καθηκόντων, και β) μια γλώσσα για την περιγραφή διεπαφών. Η TeresaXML, για την περιγραφή διεπαφών

[8] http://giove.cnuce.cnr.it/

περιγράφει πως ποικίλα AIO (Αφηρημένα Διαδραστικά Αντικείμενα – Abstract Interaction Objects), οργανώνονται μεταξύ τους για να συνθέσουν μια διεπαφή, υποβοηθώντας παράλληλα την κωδικοποίηση πληροφοριών που αφορούν τον προσδιορισμό (specification) λεπτομερειών του διαλόγου (UI Dialogue).

Η διεπαφή προσεγγίζεται ως σύνολο ενός ή περισσοτέρων στοιχείων παρουσίασης (presentation elements). Κάθε στοιχείο παρουσίασης (presentation element), χαρακτηρίζεται από μια δομή (structure), η οποία περιγράφει τη στατική δομή (static organization) της διεπαφής (τα AIOs) και καμία ή περισσότερες συνδέσεις, οι οποίες παρέχουν πληροφορίες όσον αφορά τις συσχετίσεις μεταξύ των εναλλακτικών σχημάτων παρουσίασης τμήματος της παρεχόμενης αλληλεπιδραστικής συμπεριφοράς της διεπαφής. Κάθε στοιχείο τύπου δομής (structure element), μπορεί να είναι είτε ένα βασικό-απλό AIO (elementary AIO), είτε ένα συνονθύλευμα (composition) τέτοιων. Κάθε AIO μπορεί να είναι είτε ένα αλληλεπιδραστικό AIO (interactive AIO), είτε ένα AIO εφαρμογής (application AIO), καθοριζόμενο από το αν εμπλέκεται ή όχι αλληλεπίδραση μεταξύ του χρήστη και της εφαρμογής.

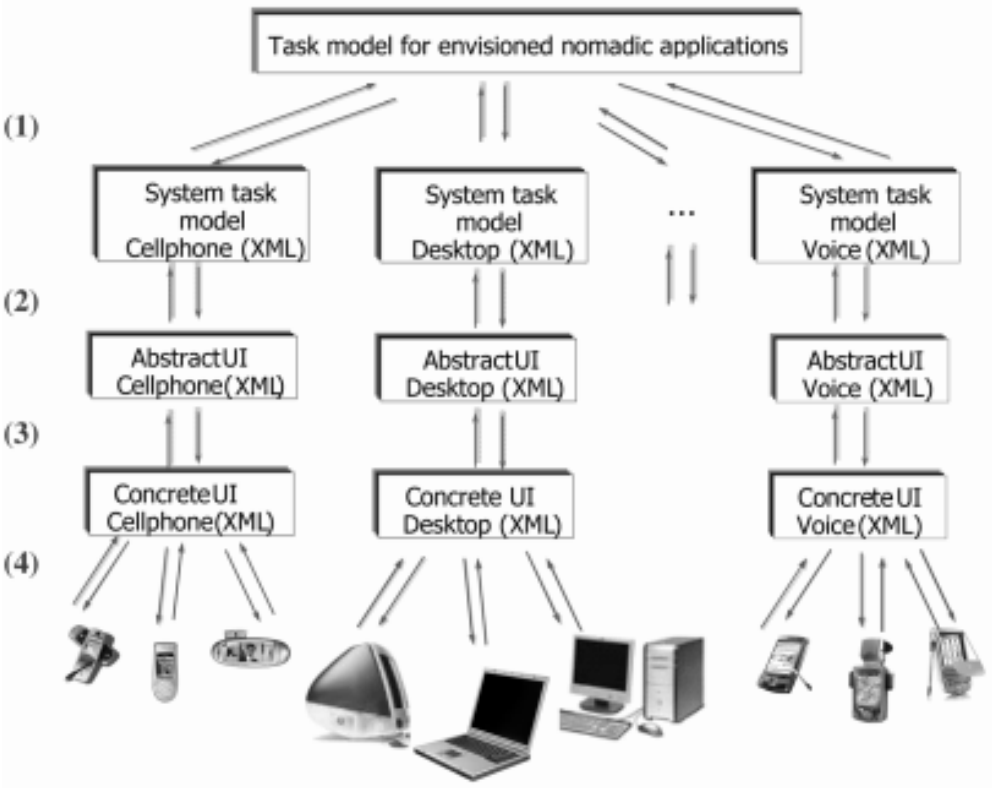
Το εργαλείο TeresaXML, υποστηρίζει το βαθμιδωτό αλληλεπιδραστικό προσδιορισμό των εμπλεκόμενων μοντέλων (καθηκόντων, αφηρημένης διεπαφής), την παραγωγή κώδικα και την υποστήριξη εκτέλεσης της διεπαφής στα πλαίσια ενός web browser, κινητής συσκευής περιορισμένων υπολογιστικών δυνατοτήτων και συμβατικών desktop υπολογιστών ως ‘stand alone’ εφαρμογής.

Η Εικόνα 15, παρουσιάζει ένα υψηλού επιπέδου στιγμιότυπο το οποίο απεικονίζει την υποκείμενη λογική της TeresaXML, που αφορά τη συγγραφή ενός μοντέλου από το οποίο μπορεί να πραχθούν εναλλακτικά στιγμιότυπα προσαρμοσμένα σε διαφορετικά πλαίσια χρήσης.

UsiXML

Είναι από τις πιο δημοφιλείς γλώσσες περιγραφής διεπαφών[17], με ευρύτατη υποστήριξη από πλευράς εργαλείων αλλά και διάχυση πληροφοριών σχετικά με τη δομή της. Συγκεκριμένα, αποτελεί υλοποίηση του Cameleon Reference Framework για την ανάπτυξη πλαστικών διεπαφών. Υποστηρίζει, την περιγραφή μιας διεπαφής σε τέσσερα διακριτά επίπεδα αφαίρεσης: α) των ‘καθηκόντων και εννοιών’ (Tasks and Concepts), όπου περιγράφεται μια διεπαφή ανεξαρτήτως αναφορών σε λεπτομέρειες που αφορούν την υπολογιστική υλοποίηση β) το AUI επίπεδο, όπου η διεπαφή προσδιορίζεται ως ιεραρχική αποσύνθεση αφηρημένων αντικειμένων

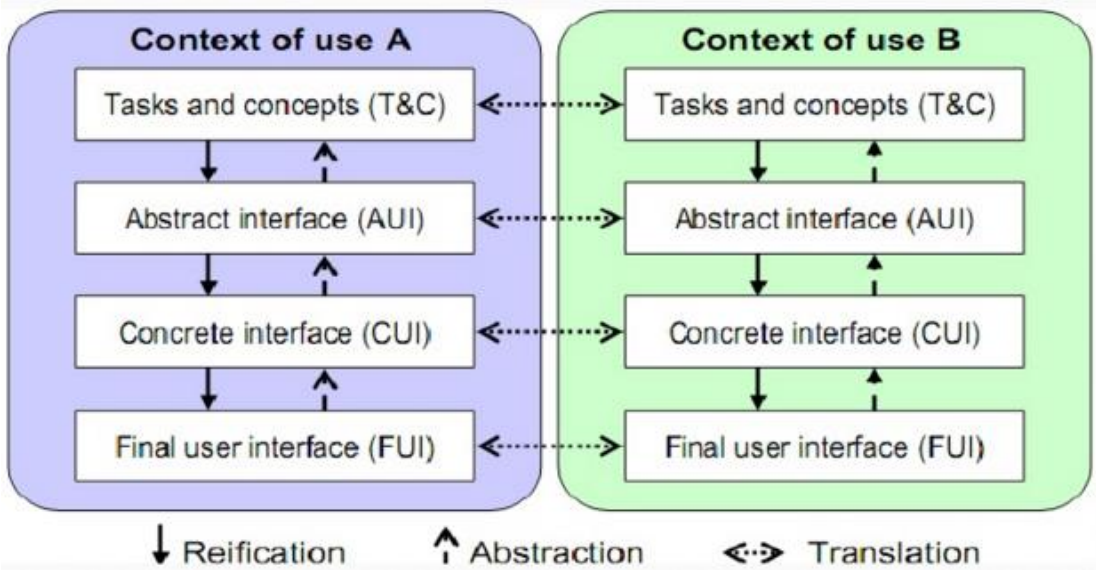
τύπου υποδοχέα (Abstract Interaction Container) ή στοιχείου αλληλεπίδρασης (Abstract Interaction Object), όντας περιγραφές ανεξάρτητες καναλιού αλληλεπίδρασης και υποκείμενης πλατφόρμας αναφοράς και τέλος, το c) CUI επίπεδο όπου η διεπαφή προσδιορίζεται ανεξαρτήτων πλατφόρμας αλλά με σαφείς εξαρτήσεις ως προς τα επιθυμητά κανάλια αλληλεπίδρασης αλλά και το βασικό τύπο αλληλεπιδραστικών στοιχείων (κουμπί, ετικέτα, κοκ.).



Εικόνα 15: Το παράδειγμα 'ενός μοντέλου – Πολλαπλών διεπαφών'[9]

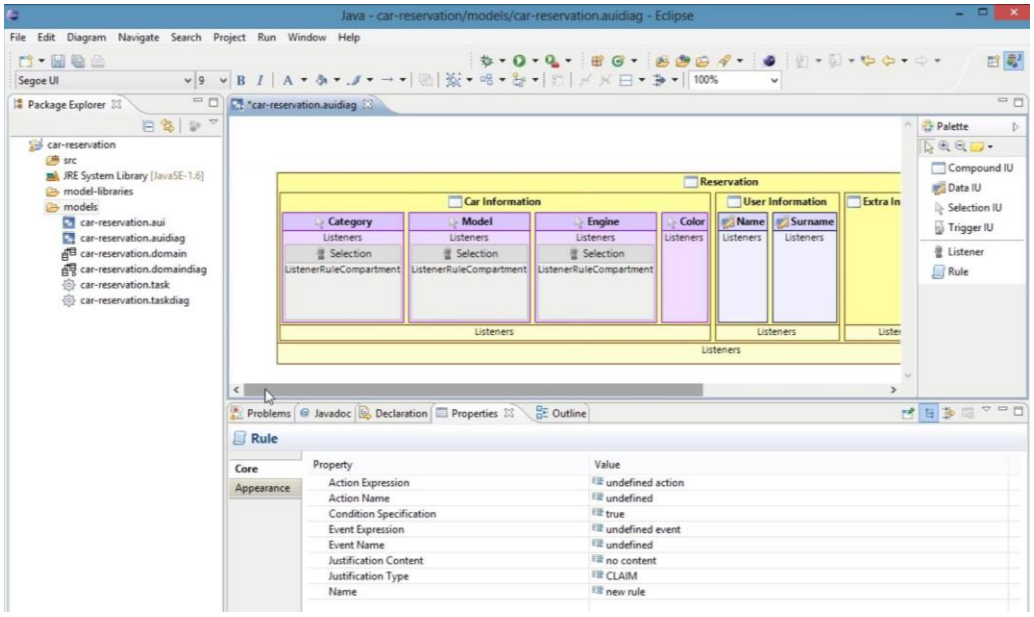
Η μετάβαση μεταξύ των επιπέδων υποστηρίζεται στη βάση προκαθορισμένων σχημάτων μετασχηματισμών, και μπορεί να είναι είτε με φορά από το CUI στο T&C επίπεδο συνιστώντας μετάβαση τύπου 'αφαίρεσης', είτε με αντίθετη φορά συνιστώντας μετάβαση τύπου 'συγκεκριμενοποίησης'. Επίσης επιτρέπεται μετάβαση στα πλαίσια του ίδιου επιπέδου οπότε συνιστά μεταγλώττιση (translation process - retargeting). Από οποιοδήποτε επίπεδο μπορεί να γεννηθεί είτε αυτομάτως ή ημι-αυτόματα ο τελικός εκτελέσιμος κώδικας (Final User Interface). Η Εικόνα 16 παρουσιάζει τη γενικότερη φιλοσοφία αυτή.

[9] Από: IEEE Transaction on software engineering, vol. 30, no. 8, August 2004.

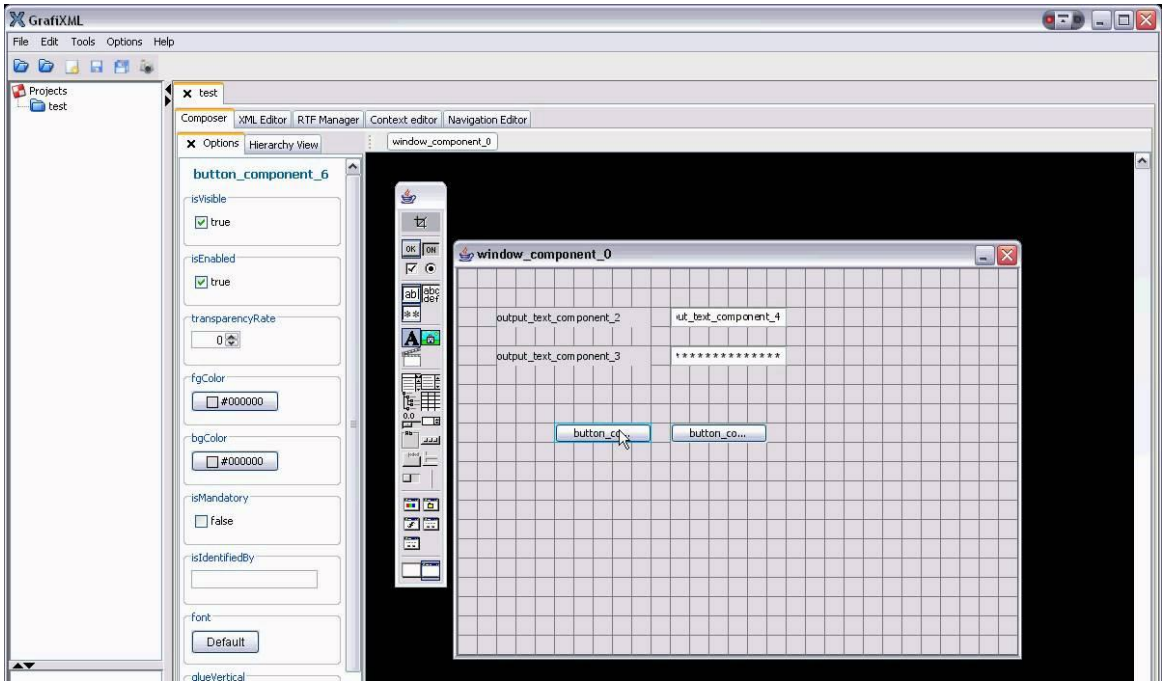


Εικόνα 16: Τα επίπεδα περιγραφής μιας διεπαφής στη UsiXML.

Ενδεικτικά, στην Εικόνα 17 παρουσιάζεται ο αλληλεπιδραστικός προσδιορισμός μιας διεπαφής στο επίπεδο αφαίρεσης AUI με τη βοήθεια του ενδεδειγμένου εργαλείου στα πλα΄σια της πλατφόρμας του Eclipse, ενώ στην Εικόνα 18 στο επίπεδο αφαίρεσης CUI με τη βοήθεια του εργαλείου GraphiXML.



Εικόνα 17: Στιγμιότυπο προσδιορισμού ενδεικτικής διεπαφής στο AUI επίπεδο αφαίρεσης με τη βοήθεια του εργαλείου GraphiXML



Εικόνα 18: Στιγμιότυπο προσδιορισμού ενδεικτικής διεπαφής στο CUI επίπεδο αφαίρεσης με τη βοήθεια του εργαλείου GraphiXML

MARIA XML

Η εν λόγω γλώσσα[21] έχει αναπτυχθεί από την ίδια ομάδα που ανέπτυξε την TeresaXML, αποτελώντας μάλιστα απόγονό της. Η εν λόγω γλώσσα υποστηρίζει δυναμικές συμπεριφορές (dynamic behaviours), γεγονότα (events), εμπλουτισμένες διαδικτιακές εφαρμογές (rich internet applications), πολλαπλών-στόχων διεπαφές (multi-target user interfaces), με σαφή προσανατολισμό στην αυτοματοποιημένη γέννηση της διεπαφής για τη διαχείριση λειτουργικότητας που εκτίθεται διαμέσου, προτυποποιημένης δομής, διαδικτυακών υπηρεσιών (web-services). Η MARIAXML, εναπόκειται, όπως και η UsiXML, στην υιοθέτηση και χρήση των προτεινόμενων επιπέδων αφαίρεσης του ‘Cameleon Reference Framework’ για την περιγραφή μιας διεπαφής. Γι’ αυτό εμπλέκει και περιγράφει έννοιες όπως: α) ‘αντικείμενα δεδομένων΄ (Data objects), διαχειριζόμενα από Interactors’ β) Γεγονότα (events) αφηρημένων και concrete διεπαφών, γ) Μοντέλο διαλόγου (dialogue model) εμπλουτισμένο με συνθήκες και CTT τελεστές στα πλάισια των χειριστών γεγονότων, δ) scripts, όπως Ajax και ε) ένα προκαθορισμένο σύνολο διαδραστικών στοιχείων. Όλα τα μοντέλα της γλώσσας, υποστηρίζονται πλήρως από υψηλού επιπέδου μονάδες λογισμικού (modules), ενσωματωμένων

και παρεχόμενω διμέου του MariaEnvironment, ενός υψηλού επιπέδου ενοποιημένου συστηματος σχεδίασης.

Στο AUI επίπεδο, το εργαλείο υποστηρίζει τους σχεδιαστές όσον αφορά την αλληλεπιδραστική ανάπτυξη προδιαγραφών, βάσει των υποστηριζόμενων μοντέλω, για κάθε μια από τις υποστηριζόμενες πλατφόρμες που είναι να υποστηριχθούν στα πλάισια μιας εφαρμογής. Το AUI, περιγράφεται υπό την έννοια AIOs (Αφηρημένων Αλληλεπιδραστικών Αντικειμένων - Abstract Interaction Objects)[22], τα οποία εν συνεχεία μετασχηματίζονται σε CIOs (Συγκεκριμενοποιημένα Διαδραστικά Αντικείμενα – Concerete Interaction Objects), με την επιλογή ενός συγκεκριμένου στόχου. Το MariaXML AUI, αποτελείται από ποικίλες παρουσιάσεις (Presentations), ομαδοποιώντας κάθε μια τα στοιχεία, τα οποία παρουσιάζονται στο χρήστη ανά περίπτωση.Οι υποστηριζόμενοι τύποι ετικετών είναι δύο τύπων: 'Interactors' και 'InteractorComposition' (Σύνθεση Interactors). Το πρώτο αναπαριστά οποιοδήποτε τύπο διαδραστικού στοιχείου της διεπαφής, ενώ το δεύτερο, ομαδοποιεί τα στοιχεία που παρουσιάζουν μεταξύ τους λογικές εξαρτήσεις. Ανάλογα με τις υποστηριζόμενες σημασιολογικές ιδιότητές του (semantics), ένας Interactor μπορεί να ανήκει σε μια από τις ακόλουθες κατηγορίες;

- **Επιλογής** (Selection): Επιτρέπει σε ένα χρήστη να επιλέξει μια ή περισσότερες τιμές μεταξύ στοιχείωνν μιας προκαθορισμένης (predefined) λίστας. Περιλαμβάνει την επιλεγμένη τιμή καθώς επίσης πληροφορία σχετικά με την πληθικότητα της λίστας. Ανάλογα με τον αριθμό των επιθυμητών υποστηριζόμενων επιλογών, ο interactor μπορέι να αποδοθεί διαμέσου μονής ή πολλαπλής επιλογής.

- **Επεξεργασίας** (Edit): Επιτρέπει στο χρήστη τη χειροκίνητη παραμετροποίηση του αντικειμένου που υλοποιείται διαμέσου του interactor, το οποίο μπορέι να είναι κείμενο (TextEdit), ένας αριθμός (numericalEdit), θέση (PositionEdit), ή γενικό αντικείμενο (ObjectEdit).

- **Χειρισμού** (Control): Επιτρέπει στο χρήστη την εναλλαγή μεταξύ αναπαραστάσεων (Navigator) ή την ενεργοποίηση λειτουργικότητας της διεπαφής (Activator)

- **Αποκλειστικής Εξόδου** (Onlyoutput): Αναπαριστά πληροφορία που άγεται προς το χρήστη και δεν επηρεάζεται απότ ις ενέργειες του χρήστη. Μπορεί να είναι κείμενο, μια περιγραφή που αναπαριστά διαφορετικούς τύπους μέσων (media), μια ειδοποίηση (alarm), ή ένα γενικευμένο αντικείμενο (generic object).

Οι διαφορετικοί τύποι ομαδοποίησης των interactors είναι:

- **Ομαδοποίησης** (Grouping): ένα γενικευμένο (generic) σύνολο interactors.

- **Συσχέτισης** (Relation): ένα σύνολο όπου δύο ή περισσότεροι στοιχεία (elements) συσχετίζονται μεταξύ τους.

- **Σύνθετης Περιγραφής** (Composite Description): αναπαριστά ένα σύνολο το οποίο στοχεύει στην παρουσίαση περιεχομένων διαμέσου της σύνθεσης στοιχείων περιγραφής (Description elements) και πλοήγησης (Navigation elements).

- **Επανάληψης** (Repeater): Χρησιμοποιείται, για την απανάληψη περιεχομένου σε σχέση με δοδομένα τα οποία αντλούνται διαμέσου μιας γενικευμένης πηγής δεδομένων (generic data-source).

Το MARIA δεν επιτρέπει μόνο τον προσδιορισμό και περιγραφή θεμάτων και ζητημάτων που σχετίζονται με την παρουσίαση (presentation), αλλά επιτρέπει επίσης την περιγραφή της αλληλεπιδραστικής συμπεριφοράς. Γι' αυτό το λόγο, υποστηρίζει ποικίλα χαρακτηρτιστικά (features):

- **Μοντέλο Δεδομένων** (Data Model): Η περιγραφή της διεπαφής εμπεριέχει περιγραφές τύπων δεδομένων οι οποίες διαχειρίζονται (manipulated) διαμέσου της διεπαφής. Οι interactors μπορεί να συσχετιστούν και προσδεθούν (bound), με στοιχεία ενός μοντέλου δεδομένων, το οποίο σημαίνει ότι, η εν εκτελέσει, αλλαγή της κατάστασης ενός Interactor θα οδηγήσει στη συνεπακόλουθη αλλαγή και της τιμής των προσδεμένων ιδιοτήτων και το αντίθετο. Αυτός ο μηχανισμός, επιτρέπει τη μοντελοποίηση της συσχέτισης (corellation) μεταξύ των διαδραστικών στοιχείων μιας διεπαφής, κατά συνθήκη διατάξεις (conditional

layouts), κατά συνθήκη συνδέσεις μεταξύ των αναπαραστάσεων και διαμόρφωση των τιμών των δεδομένων εισόδου. Το μοντέλο δεδομένων, ορίζεται χρησιμοποιώντας στοιχεία του προτύπου XML Schema.

- **Γενικευμένο Backend** (Generic Back End): Ο προδιορισμός της διεπαφής εμπεριέχει ένα σύνολο εξωτερικά οριζόμενων συναρτήσεων (external functions declaration), που αναπαριστά λειτουργικότητα που χρησιμοποιείται από τη διεπαφή αλλά είναι υλοποιημένη από ένα γενικευμένο backend μιας εφαρμογής (π.χ. web-services, βιβλιοθήκες, βάσεις δεδομένων, κοκ.). Ένας ορισμός (declaration), περιγράφει την υπογραφή των εξωτερικών συναρτήσεων στα πλάισια αναφοράς του ονόματος αυτών, των τύπων των ορισμάτων που κάθε μια δέχεται αλλά και επιστρέφει.

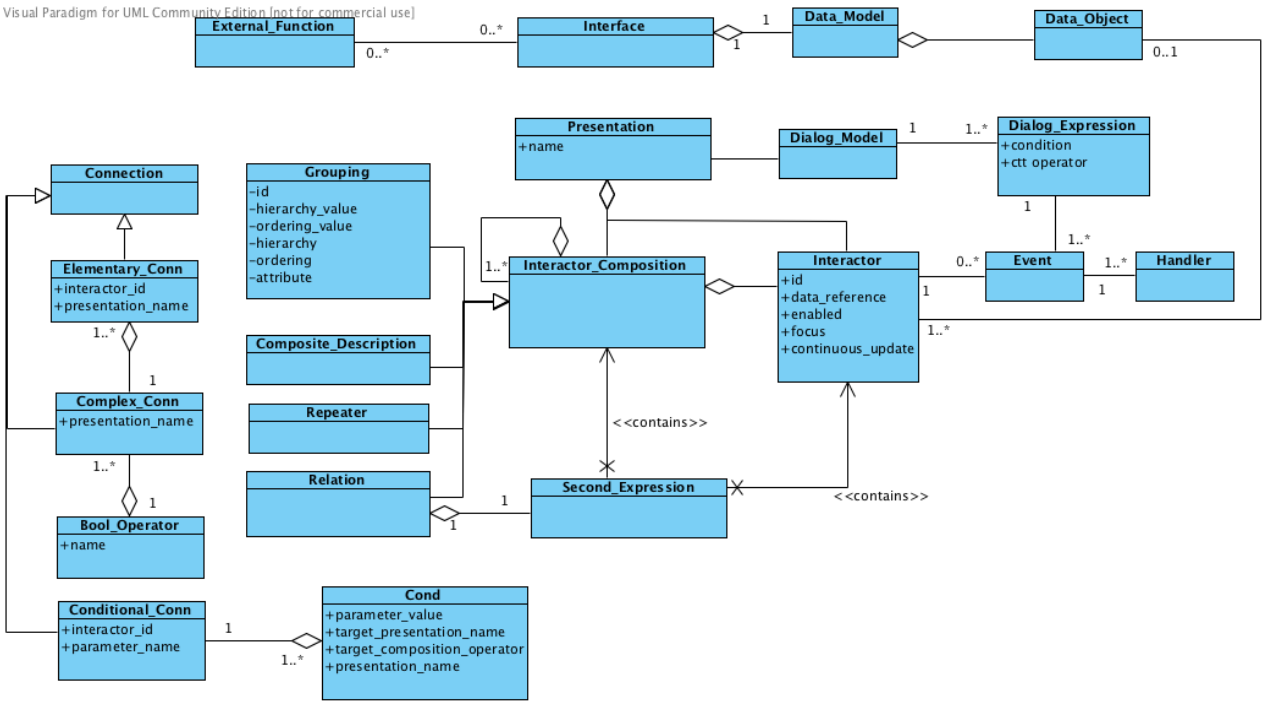
- **Μοντέλο Γεγονότων** (Event Model): Κάθε ορισμός (definition) ενός Interactor, συσχετίζεται με ένα αριθμό γεγονότων στα πλάισια των οποίων μπορεί να οριστεί η λογικής της ανάδρασης της διεπαφής ως προς τις εν εκτελέσει αλληλεπιδραστικές ενέργιες του χρήστη.

- **Μοντέλο Διαλόγου** (Dialog Model): Το μοντέλο διαλόγου, εμπεριέχει εργαλεία (constructs), για την περιγραφή της δυναμικής συμπεριφοράς της παρουσίασης, καθορίζοντας τι είδους γεγονότα μπορούν να εκπεμφθούν σε μια δεδομένη χρονική στιγμή.

- **Συνεχής Ενημέρωση Πεδίων** (Continuous update of fields): Είναι δυνατό να οριστεί ότι ένα συγκεκριμένο πεδίο (field), πρέπει να ανανεώνεται περιοδικά καλώντας μια εξωτερική συνάρτηση.

- **Δυναμικό Σύνολο Διαδραστικών Στοιχείων Διεπαφής** (Dynamic Set of User Interface Elements): Η γλώσσα εμπεριέχει-υποστηρίζει εργαλεία για τον προσδιορισμό μερικών ενημερώσεων της παρουσίασης και τη δυνατότητα ορισμού κατά συνθήκη μεταβάσεων μεταξύ των παρουσιάσεων.

Το σύνολο αυτών των χαρακτηριστικών επιτρέπουν, μόλις κιόλας από το εν λόγω αφηρημένο επίπεδο, την περιγραφή της διεπαφής ανεξαρτήτως πληροφοριών που σχετίζονται με τη διάταξη των αντικειμένων (layout details). Είναι ωστόσο πλήρες όσον αφορά τη δυνατότητα αιτιολόγησης (reasoning), ως προς το πώς η διεπαφή υποστηρίζει τόσο την αλληλεπίδραση του χρήστη με τη διεπαφή όσο και την αλληλεπίδραση με το backend της εφαρμογής.



Εικόνα 19: Το MariaXML AUI μέτα-μοντέλο

Ένα Concrete User Interface (CUI) στα πλαίσια της MARIA, υποστηρίζει την εξαρτημένη πλατφόρμας (platform-dependent), αλλά ανεξάρτητη λεπτομερειών της υπολογιστικής γλώσσας υλοποίησης (implementation language-independent details), περιγραφή της διεπαφής. Ως πλατφόρμα ορίζεται το σύνολο των πόρων λογισμικού και υλικού που χαρακτηρίζουν ένα σύνολο συσκευών. Η Maria προς το παρόν υποστηρίζει τις ακόλουθες πλατφόρμες:

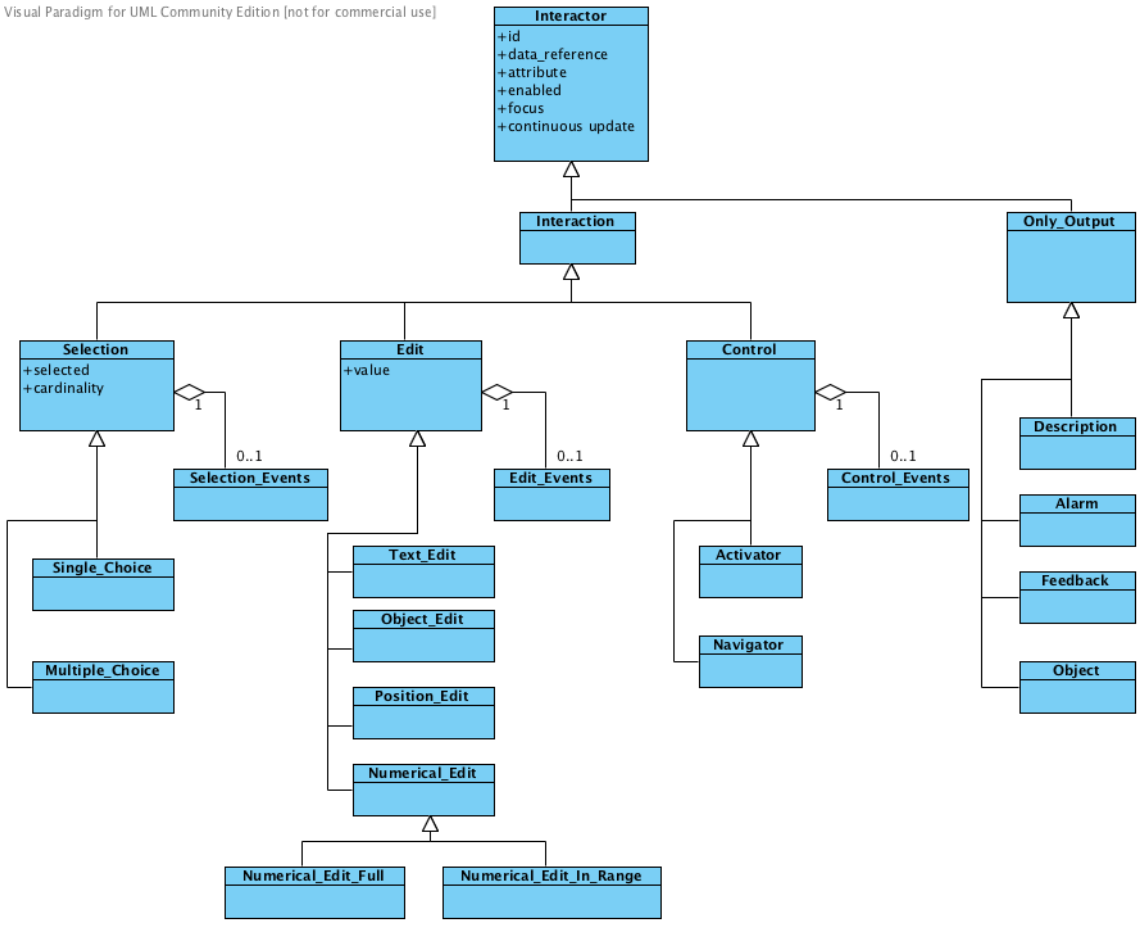
- **CUI σταθερών υπολογιστών** (Desktop CUIs): μοντελοποιούν γραφικές διεπαφές για σταθερούς υπολογιστές.

- **CUI κινητών συσκευών** (Mobile CUIs): μοντελοποιούν γραφικές διεπαφές για κινητές συσκευές.

- **Πολυκαναλικά CUI σταθερών υπολογιστών** (Multimodal Desktop CUIs): μοντελοποιούν διεπαφές που συνδυάζουν το γραφικό (graphical modality) και ακουστικό κανάλι (vocal modality) για σταθερούς υπολογιστές.

- **Πολυκαναλικά CUI κινητών συσκευών** (Multimodal Mobile CUIs): μοντελοποιούν διεπαφές που συνδυάζουν το γραφικό (graphical modality) και ακουστικό κανάλι (vocal modality) για κινητές συσκευές.

- **Ακουστικά CUI** (Vocal CUIs): διεπαφές με ακουστική μετάδοση πληροφορίας και αναγνώριση φωνής.



Εικόνα 20: Κατηγοριοποίηση (taxonomy) των υποστηριζόμενων interactors τη MariaXML

Το μετα-μοντέλο κάθε πλατφόρμας αποτελεί μια συγκεκριμενοποίηση (refinement) εκδοχή του AUI μοντέλου, η οποία προσδιορίζει πως ένας αφηρημένος interactor, μπορεί να

αναπαρασταθεί στη συγκεκριμένη πλατφόρμα. Παραδείγματος χάριν, ένας αφηρημένος interactor μονής επιλογής (Single Choice Interactor), μπορεί να υλοποιηθεί (σε μια πλατφόρμα σταθερού υπολογιστή με γραφικά), μέσω ενός radioButton, ενός dropdown μενού, ή μιας λίστας (listbox), ενώ σε μια ακουστική πλατφόρμα μπορεί να αποδοθεί μέσω μιας λίστας μηνυμάτων όπου κάθε επιλογή θα έχει συσχετιστεί με μια λέξη κλειδί.

Το ίδιο συμβαίνει και στην περίπτωση των συνθέσεων interactors. (Interactor compositions): Σε μια πλατφόρμα σταθερού υπολογιστή ένα σύνολο (group) μπορεί να υλοποιηθεί χρησιμοποιώντας συγκεκριμένου χρώματος φόντο, σύνορα (borders), κοκ. Ενώ σε μια ακουστική είναι δυνατό να χρησιμοποιηθούν εναλλακτικά ήχοι πριν το πρώτο στοιχείο του συνόλου. Ο ορισμός του μοντέλου μπορεί να χρησιμοποιηθεί για τη δημιουργία τελικών υλοποιήσεων (final implementations) σε διαφορετικές γλώσσες προγραμματισμού. Επίσης, είναι δυνατόν, το ίδιο στιγμιότυπο του CUI μοντέλου να χρησιμοποιηθεί για την αναπαράσταση μιας διεπαφής τόσο στα πλαίσια μιας Android όσο και μιας iPhone συσκευής.

Ανοιχτά ζητήματα και προσέγγιση

Μετά την επισκόπηση των βασικών γλωσσών προδιαγραφής διεπαφών είναι προφανές ότι η έμφαση στην εν λόγω προσέγγιση είναι στη χρήση μοντέλων για τον προοδευτικό μετασχηματισμό αφηρημένων δομών σε στοιχεία που υλοποιούνται από μια εργαλειοθήκη διαδραστικών αντικειμένων. Αξίζει επίσης να ανακαλέσουμε ότι οι ευρέως διαθέσιμες εργαλειοθήκες συχνά διαφέρουν τόσο σε φυσικό και συντακτικό επίπεδο όσο και σε εννοιολογικό / σημασιολογικό. Δεδομένων αυτών των διαφορών, οι γλώσσες που προαναφέραμε περιορίζουν τη δυνατότητα προδιαγραφής προηγμένων πανταχού παρόντων διεπαφών (advanced ubiquitous UIs) με ενιαίο τρόπο. Ένας τρόπος άρσης αυτών των περιορισμών είναι η βελτίωση των γλωσσών προδιαγραφής έτσι ώστε να υποστηρίζουν διεπαφές που (α) συντίθεται από ποικίλες συλλογές αντικειμένων (diverse collection of elements, native or not), εγγενώς υποστηριζόμενων και μη (β) παρουσιάζουν ‘πλαστικότητα’ εν εκτελέσει με αυξημένη προσαρμοστικότητα σε κάθε επίπεδο (σημασιολογικό, λεκτικό, συντακτικό και φυσικό) έτσι ώστε να αξιοποιούν εναλλακτικές συλλογές αντικειμένων (diverse collections of elements). Τα παραπάνω χαρακτηριστικά ωστόσο δεν υποστηρίζονται στα πλαίσια των state-of-the-art μεθόδων ανάπτυξης. Ειδικότερα, στην περίπτωση των εργαλειοθηκο-κεντρικών προσεγγίσεων, αυτό οφείλεται σε υποθέσεις σε επίπεδο κλάσεων και διαλόγου που περιορίζουν το εύρος της

υποστήριξης που προσφέρεται μόνο σε ομογενείς οικογένειες πλατφορμών. Ένα άλλο μειονέκτημα εστιάζει στη λογική υλοποίησης της προσαρμοστικής συμπεριφοράς, η οποία στο σύνολο των περιπτώσεων αποτελεί χαρακτηριστικό ενσωματωμένο στον κώδικα είτε της γλώσσας προγραμματισμού (βλ. PIM[1], HOMER[2]), είτε της εργαλειοθήκης (βλ. Meta-Widgets[3], Context-Independent Toolkit[4]). Ως εκ τούτου δεν εκτίθεται προς σχεδίαση.

Από την άλλη πλευρά, στην περίπτωση της μοντελοκεντρικής μηχανικής διεπαφών, θέματα που αφορούν την υποστήριξη ετερογενών περιβαλλόντων χρήσης είναι εγγενώς λυμένα διαμέσου της φιλοσοφίας που εξ' ορισμού τα διατρέχει (προσκόλληση στη χρήση μοντέλων και λοιπών αφαιρετικών δομών). Πλήθος προσεγγίσεων έχουν επίσης αποδείξει την επάρκεια των σχετικών μεθόδων για υποστήριξη εν εκτελέσει προσαρμοστικότητας σε όλα τα επίπεδα (συντακτικό, λεκτικό, φυσικό) (βλ. π.χ., COMMETS[5], Rainbow[6]). Ωστόσο, λόγω υποθέσεων σχετικά με την υποστήριξη του μεγαλύτερου δυνατού βαθμού φορητότητας (portability) των παραγόμενων περιγραφών (specifications), το εύρος των αντικειμένων που υποστηρίζονται περιορίζεται αποκλειστικά και μόνο στα κοινά μεταξύ των πιο δημοφιλών εργαλειοθηκών (κουμπιά, λίστες, κοκ) και στα εγγενώς υποστηριζόμενα. Ως εκ τούτου οι διεπαφές που μπορούν να περιγραφούν και παραχθούν είναι μάλλον απλοϊκές.

Προσέγγιση

Η προτεινόμενη προσέγγιση στοχεύει στην προσπάθεια γεφύρωσης των πλεονεκτημάτων των δύο κύριων μεθόδων προγραμματισμού, της εργαλειοθηκο-κεντρικής (εκφραστικότητα, σχεδιαστικός πλουραλισμός, ευελιξία) και της μοντελο-κεντρικής (πανταχού παρούσα ενοποιημένη μεθοδολογία ανάπτυξης, διαχείρισης ετερογενών περιβαλλόντων, προσαρμοστικότητα, κοκ.). Συγκεκριμένα, ένας τρόπος για την υποστήριξη του ζητούμενου επιπέδου αφαίρεσης που απαιτείται ως προς την υποστήριξη των παραπάνω, είναι η επινόηση σχημάτων περιγραφής τα οποία θα μπορεί να κωδικοποιήσουν με ενοποιημένο (π.χ. xml-based) και επίσημο τρόπο (formal specification, π.χ. xml-schema), α) το πως υλοποιείται κάθε εναλλακτικό αλληλεπιδραστικό στιγμιότυπο, σε ένα συγκεκριμένο πλαίσιο χρήσης, και β) πως τα εγγεγραμμένα σε κάθε ένα σχήμα affordances (qualities του object) υλοποιούνται ώστε να μπορεί να εκτίθενται προς διαχείριση κατά τη φάση της σχεδίασης. Κλειδί για την επίτευξη των παραπάνω, αφορά στην ικανότητα υποστήριξης πολυμορφισμού[1][2]. Ο πολυμορφισμός ορίζει τη δυνατότητα ενός αφηρημένου αντικειμένου, να διαχωρίζει το ρόλο του (για ποιο λόγο δηλαδή

χρησιμοποιείται) από τις εναλλακτικές υπολογιστικές αναπαραστάσεις με βάσει τις οποίες μπορεί να ενσαρκωθεί. Αυτό συνεπάγεται την ύπαρξη σχημάτων περιγραφής που θα προάγουν το ρητό διαχωρισμό μεταξύ κοινών (αφηρημένων) και πολυμορφικών ιδιοτήτων ενώ μέσω της κωδικοποίησης και περιγραφής λεπτομερειών της υπολογιστικής τους υλοποίησης θα υποβοηθούν στη συνολική διαχείρισή τους τόσο σχεδιαστικά όσο και εν εκτελέσει. Τέτοιου είδους σχήματα ονομάζονται γλώσσες περιγραφής widget (WSL). Η γλώσσα η οποία παρουσιάζεται στο επόμενο κεφάλαιο είναι οικεία και αρκετά ώριμη, μιας και έχει δοκιμαστεί επιτυχώς για την υποστήριξη πλήθους απαιτητικών σεναρίων, ενώ είναι η μοναδική καταγεγραμμένη του είδους στη διεθνή βιβλιογραφία των μεθόδων μοντελο-κεντρικής μηχανικής διεπαφών.

Κεφάλαιο 3 – Γλώσσα περιγραφής αντικειμένων (WSL)

Η γλώσσα περιγραφής Widgets (WSL) επινοήθηκε με γνώμονα να παρέχει τη δυνατότητα: α) προδιαγραφής επεκτάσιμων διαδραστικών λεξικών που υπακούουν σε συγκεκριμένους κανόνες αλλά δεν περιορίζονται ως προς το περιβάλλον χρήσης τους, β) υποστήριξης πολυμορφισμού σε όλα τα επίπεδα (φυσικό, συντακτικό και σημασιολογικό) μέσω της προδιαγραφής πολυμορφικών σχημάτων στιγμιοτυποποίησης και γ) κωδικοποίησης εγγενών πληροφοριών που αφορούν την υπολογιστική φύση των εναλλακτικών σχημάτων στιγμιοτυποποίησης με στόχο τη συνολική διαχείρισή τους κατά τη φάση της σχεδίασης. Προτείνεται λοιπόν μια γλώσσα η οποία είναι επίσημη (formal), υπό την έννοια ότι η δομή της είναι προτυποποιημένη με απόλυτο τρόπο και κωδικοποιημένη με χρήση XMLSchema[10]. Για την ευκολότερη απόδοση της δομής της παρουσιάζεται ένα υψηλού επιπέδου μοντέλο κλάσεων της UML (βλ. Εικόνα 27), η δομή είναι ίδια με αυτή που αποτυπώνεται και στο XMLSchema.

Πολυμορφικές περιγραφές

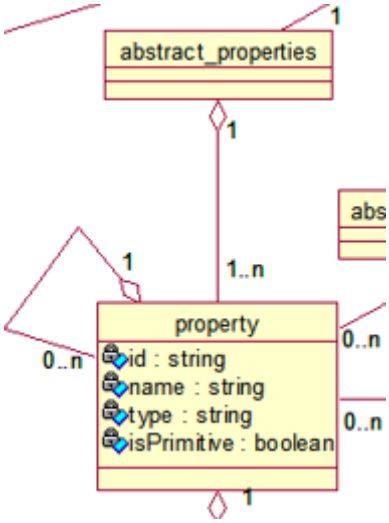
Τα πολυμορφικά στιγμιότυπα που υποστηρίζονται από τα τις δομές της γλώσσας κωδικοποιούνται στα πλαίσια ενός κοινού στιγμιοτύπου του WSL (xml αρχείο). Κάθε WSL χωρίζεται σε τρεις βασικές ενότητες με βάση την ακόλουθη προσέγγιση. Καταρχήν, κωδικοποιούνται πληροφορίες που αφορούν το abstractWidget στο οποίο αναφερόμαστε (‘abstractButton’, ‘abstractConatiner’, κοκ..). Οι πληροφορίες αυτές περιλαμβάνουν το όνομά του (name), ένα μοναδικό αναγνωριστικό σε επίπεδο λεξικού (id), καθώς και το αν είναι τύπου container ή όχι (type). Επιπλέον απαριθμούνται καταλλήλως όλες οι αφηρημένες ιδιότητες (‘abstractProperties’), δηλ. οι ιδιότητες που είναι κοινές ανεξαρτήτως σχήματος στιγμιοτυποποίησης. Για κάθε ένα από τα σχήματα στιγμιοτυποποίησης που ορίζονται, δημιουργείται ένα ξεχωριστό section (‘polymorphic_instance’), κάτω από το οποίο ορίζονται όλες οι πληροφορίες που αφορούν την υλοποίησή του (δηλ. το public API του). Αυτές αφορούν τους διαθέσιμους constructors, σύνολο μεθόδων, καθώς επίσης αναφορές σε 3rd party βιβλιοθήκες που χρειάζονται για τη λειτουργία τους, αλλά και αναφορές σε λοιπούς σχετικούς πόρους (εικόνες, αρχεία, κοκ.).

Παρακάτω αναλύονται λεπτομέρειες αυτών.

[10] http://www.w3.org/XML/Schema/

Αφηρημένες ιδιότητες (abstract properties)

Όσον αφορά τις αφηρημένες ιδιότητες, τα χαρακτηριστικά που κρατούνται είναι ένα id (μοναδικό αναγνωριστικό σε επίπεδο WSL), το όνομά της και ο τύπος της.



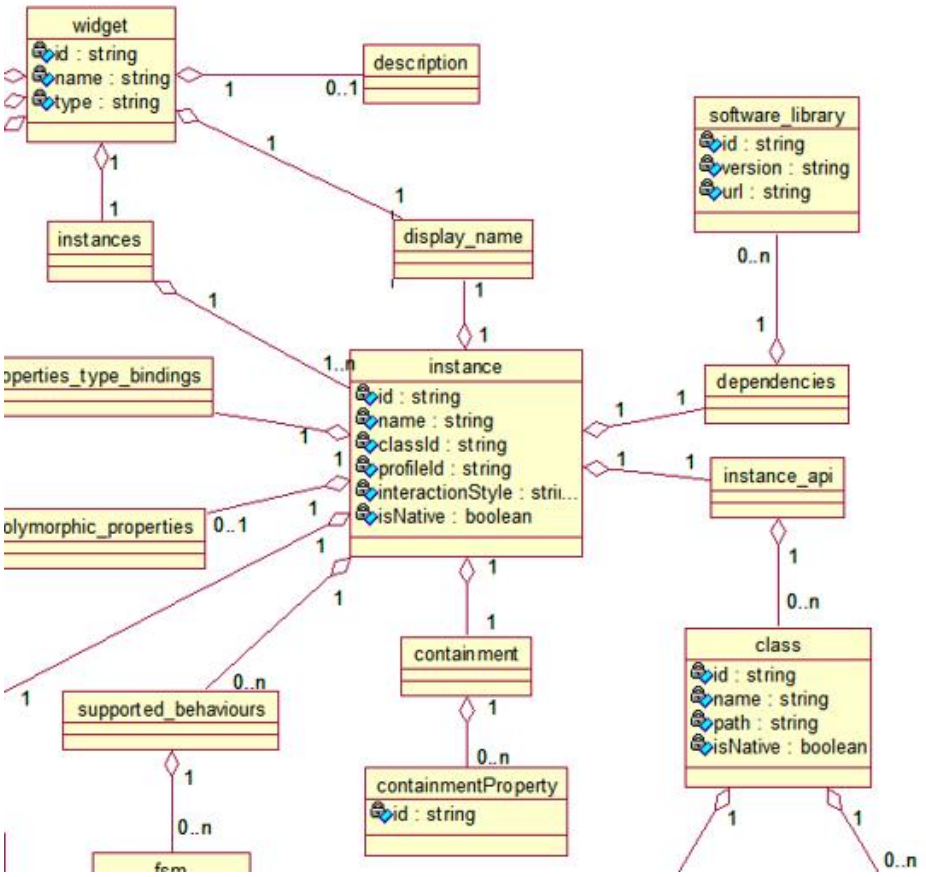
Εικόνα 21: Απόσπασμα WSL με εστίαση στην Κωδικοποίηση αφηρημένων ιδιοτήτων

Το id χρειάζεται ώστε να μπορεί να προσδιοριστεί η εν λόγω ιδιότητα με μοναδιαίο τρόπο στα πλαίσια του εγγράφου ενώ το όνομα χρειάζεται για προφανής λόγους (π.χ. 'label', 'title', 'playerName', κοκ.). Όσον αφορά τον τύπο, αφορά ειδική περίπτωση, και μπορεί να δεχτεί μια εκ των προκαθορισμένων τιμών: '*boolean'*, '*char'*, '*byte'*, '*short'*, '*int'*, '*long*', '*float*', '*double*' και '*string*'. Ο τύπος αυτός παρά το γεγονός ότι χαρακτηρίζει μια κοινή ιδιότητα, με αποτέλεσμα η τιμή του να είναι κοινή σε κάθε περίπτωση, μπορεί σε επίπεδο υπολογιστικού σχήματος να αποδίδεται-κωδικοποιείται με χρήση διαφορετικού τύπου του αφηρημένου. Π.χ. στην περίπτωση ενός κουμπιού η ετικέτα να μην αποθηκεύεται ως string που είναι ο δηλωθείς τύπος της αφηρημένης ιδιότητας αλλά ως enumeration. Έτσι υπάρχουν bindings μεταξύ των προκαθορισμένων τύπων που αναφέρθηκαν και των τύπων στους οποίους πρέπει να μεταφραστούν στα πλαίσια κάθε σχήματος (π.χ. java.lang.String, .net.String, κοκ). Στην περίπτωση εξειδικευμένων περιπτώσεων, όπως αυτής του κουμπιού, όπου ένας γενικός τύπος (string) πρέπει να μεταφράζεται σε κάποιο implementation-specific advanced data-type, δίνεται πάντα η δυνατότητα ορισμού ειδικών μεταφραστών (translators).

Στην άλλη εξειδικευμένη περίπτωση που μια αφηρημένη ιδιότητα μπορεί να πάρει προκαθορισμένο εύρος τιμών, αυτό υποστηρίζεται μέσω των λεγόμενων ‘options’ (βλ. ‘option’ class).

Πολυμορφικά σχήματα στιγμιοτυποποίησης (polymorphic instantiation schemes)

Οι πολυμορφικές αναπαραστάσεις που μπορεί να υποστηριχθούν στα πλαίσια ενός widget, κωδικοποιούνται κάτω από ένα στοιχείο τύπου ‘instance’.



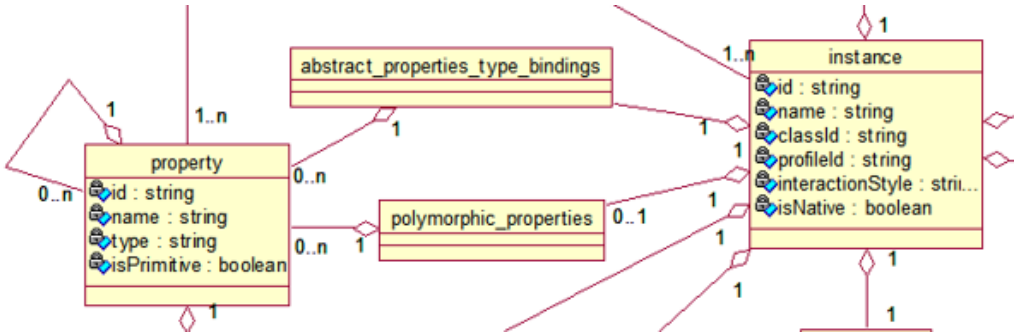
Εικόνα 22: Απόσπασμα WSL με εστίαση στην Κωδικοποίηση πολυμορφικών σχημάτων

Μέσα σε αυτό κωδικοποιούνται, εκτός από γενικής φύσεως πληροφορίες που αφορούν το μοναδικό id του σχήματος, το όνομά του κοκ, και όλες εκείνες οι πληροφορίες που αφορούν τυχόν εξαρτήσεις (βλ. κλάση ‘dependencies’) στις οποίες στηρίζεται η ομαλή λειτουργία του σχήματος. Τέτοιου είδους εξαρτήσεις μπορεί να αφορούν βιβλιοθήκες τρίτων ή άλλους πόρους όπως αρχεία εικόνων, ήχου, κοκ. Επίσης προσδιορίζεται το εύρος των προφίλ χρήσης στα πλαίσια των οποίων δύναται να λειτουργήσει απρόσκοπτα (π.χ. συγκεκριμένο τύπο και έκδοση

λειτουργικού συστήματος, λογισμικό που απαιτείται να είναι προ-εγκατεστημένο (π.χ. JRE, λοιπούς πόρους του συστήματος, κοκ). Σε περίπτωση που το widget έχει δηλωθεί γενικά ως τύπου container, πρέπει σε αυτό το επίπεδο να δηλωθεί επίσης το αν είναι τύπου 'top-level'. Τέλος, μέσω των ενδεδειγμένων κλάσεων (xml tags), κωδικοποιείται το σύνολο των λεκτικών (lexical), φυσικών (physical) και σημασιολογικών (semantic) ιδιοτήτων του πολυμορφικού σχήματος.

Πολυμορφικές ιδιότητες (polymorphic properties)

Όλες οι ιδιότητες του αντικειμένου που δεν συμπεριλαμβάνονται στις αφηρημένες καταγράφονται ως πολυμορφικές και αφορούν μόνο το συγκεκριμένο στιγμιότυπο και τα unique qualities (individual affordances) αυτού.



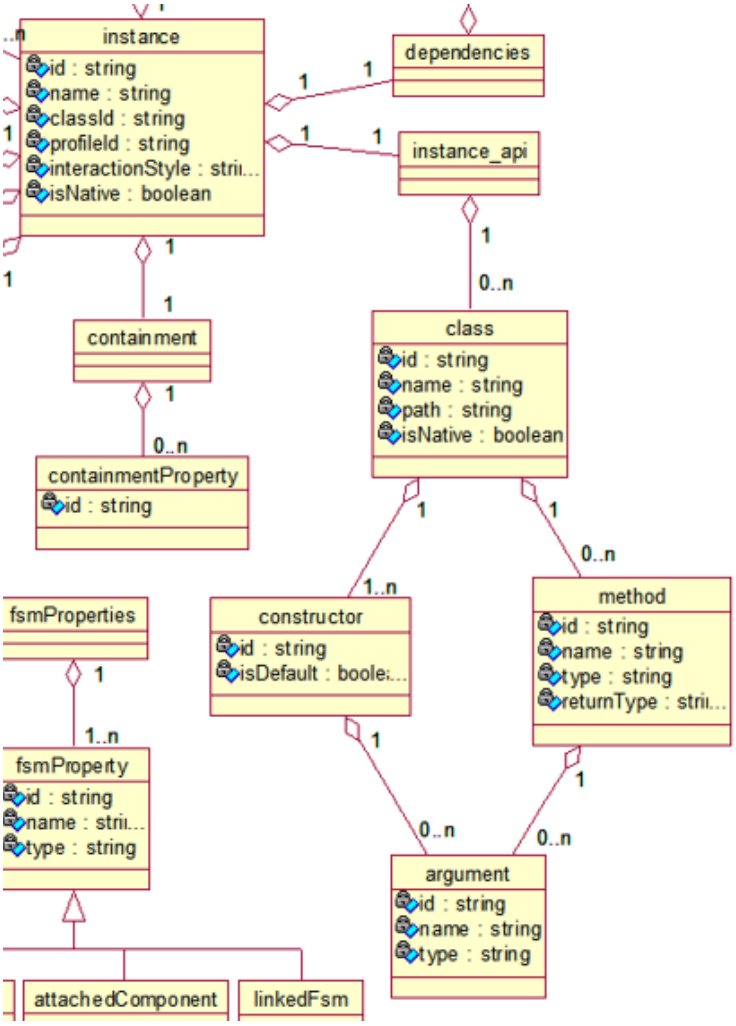
Εικόνα 23: Απόσπασμα WSL με εστίαση στην Κωδικοποίηση πολυμορφικών ιδιοτήτων

Όσον αφορά τις πολυμορφικές ιδιότητες, τα χαρακτηριστικά που κρατούνται είναι ένα id (μοναδικό αναγνωριστικό σε επίπεδο WSL), το όνομά της, ο τύπος της και αν αυτός είναι primitive. Το id χρειάζεται ώστε να μπορεί να προσδιοριστεί η εν λόγω ιδιότητα με μοναδιαίο τρόπο στα πλαίσια του εγγράφου ενώ το όνομα χρειάζεται για προφανής λόγους (π.χ. 'width', 'height', 'radius', 'voiceStrength', 'color', κοκ.). Ο τύπος του μπορεί να είναι είτε primitive στα πλαίσια της γλώσσας-πλατφόρμας στην οποία υποστηρίζεται, οπότε χειρίζεται ως τέτοιος, είτε custom. Στην περίπτωση των custom ιδιοτήτων ορίζεται ο τρόπος δημιουργίας και απόδοσής τιμών σε αυτές ενώ στην περίπτωση που είναι πολυσύνθετες υποστηρίζονται και εμφωλευμένες δομές (βλ. recursive loop στην κλάση 'polymorphic_properties' → 'property').

Implementation-specific (polymorphic) API

Σε επίπεδο πολυμορφικού στιγμιοτύπου, όπως ειπώθηκε επιγραμματικά προηγουμένως, κωδικοποιείται πληροφορία που αφορά το πλήρες public API του σχήματος (βλ. κλάση 'class'

κάτω από το ‘instance_api’ στην Εικόνα 24). Στην κορυφή βρίσκεται η κλάση που υλοποιεί το διαδραστικό σχήμα, και ένθετες βρίσκονται είτε μέθοδοι που μπορεί να εφαρμοστούν σε αυτό, ή εσωκλειόμενες κλάσεις με το (public) API τους. Να σημειωθεί ότι το εύρος των διαθέσιμων σχημάτων στιγμιοτυποποίησης συνάγεται από το πλήθος των σχημάτων που έχουν καταγραφεί συνολικά.

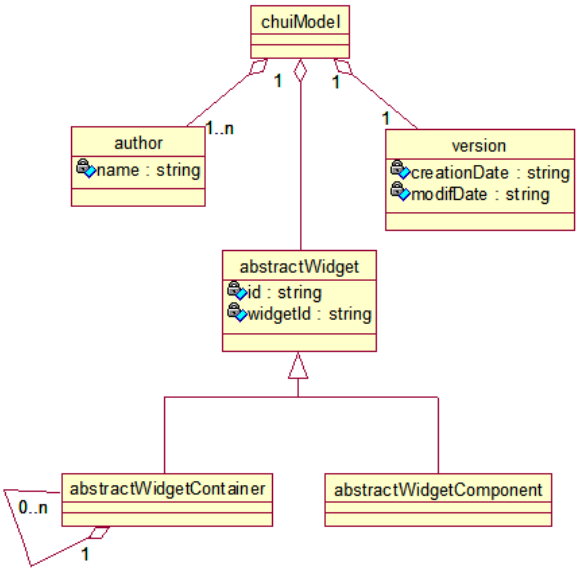


Εικόνα 24: Απόσπασμα WSL με εστίαση στο implementation-scheme API

Ορισμός της πολυμορφικής διεπαφής (CUI model)

Το WSL τροφοδοτεί με τα διαθέσιμα semantics των widget (abstract), στα πλαίσια του οποίου έχουν οριστεί, το μοντέλο CUI (βλέπε Εικόνα 25). Ρόλος του συγκεκριμένου μοντέλου είναι η περιγραφή της πολυμορφικής διεπαφής, δηλαδή μιας περιγραφής της ιεραρχικής αποσύνθεσης της διεπαφής, υπό την έννοια αφηρημένων widget τύπου υποδοχέα (abstractWidgetContainer) ή μη (abstractWidgetComponent), με τρόπο ανεξάρτητο πλατφόρμας

και διαδραστικού λεξικού στόχου (target implementation-specific vocabulary). Κάθε αφηρημένο (πολυμορφικό) αντικείμενο που ορίζεται σε αυτά τα πλαίσια φέρει μοναδικό ένα id (widgetId) του τύπου αντικειμένου που συμβολίζει, διατηρώντας ανά πάσα στιγμή μια σύνδεση με το κατάλληλα κωδικοποιημένο (σε WSL) API του.



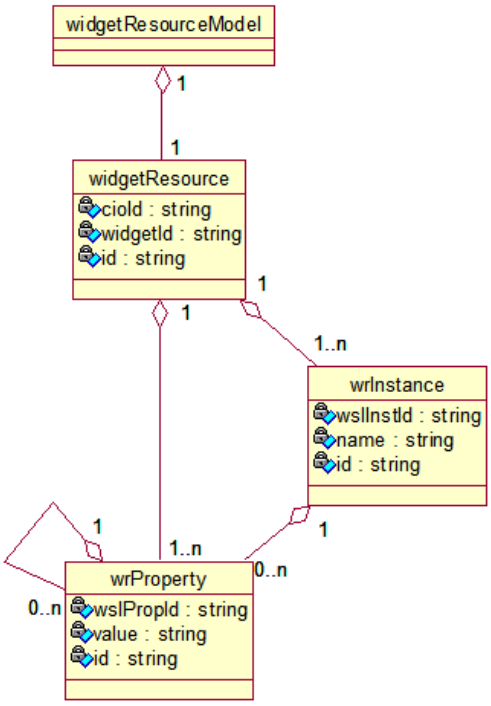
Εικόνα 25: Το μοντέλο 'CUI

Σε αυτό το επίπεδο δεν γίνεται καμία αναφορά σε συντακτικές ιδιότητες, ή προσδιορισμός τιμών σε αυτές. Αυτές προσδιορίζονται σε ένα δεύτερο μοντέλο, το 'WidgetResource'.

Προσδιορισμός ιδιοτήτων αντικειμένων (WidgetResource model)

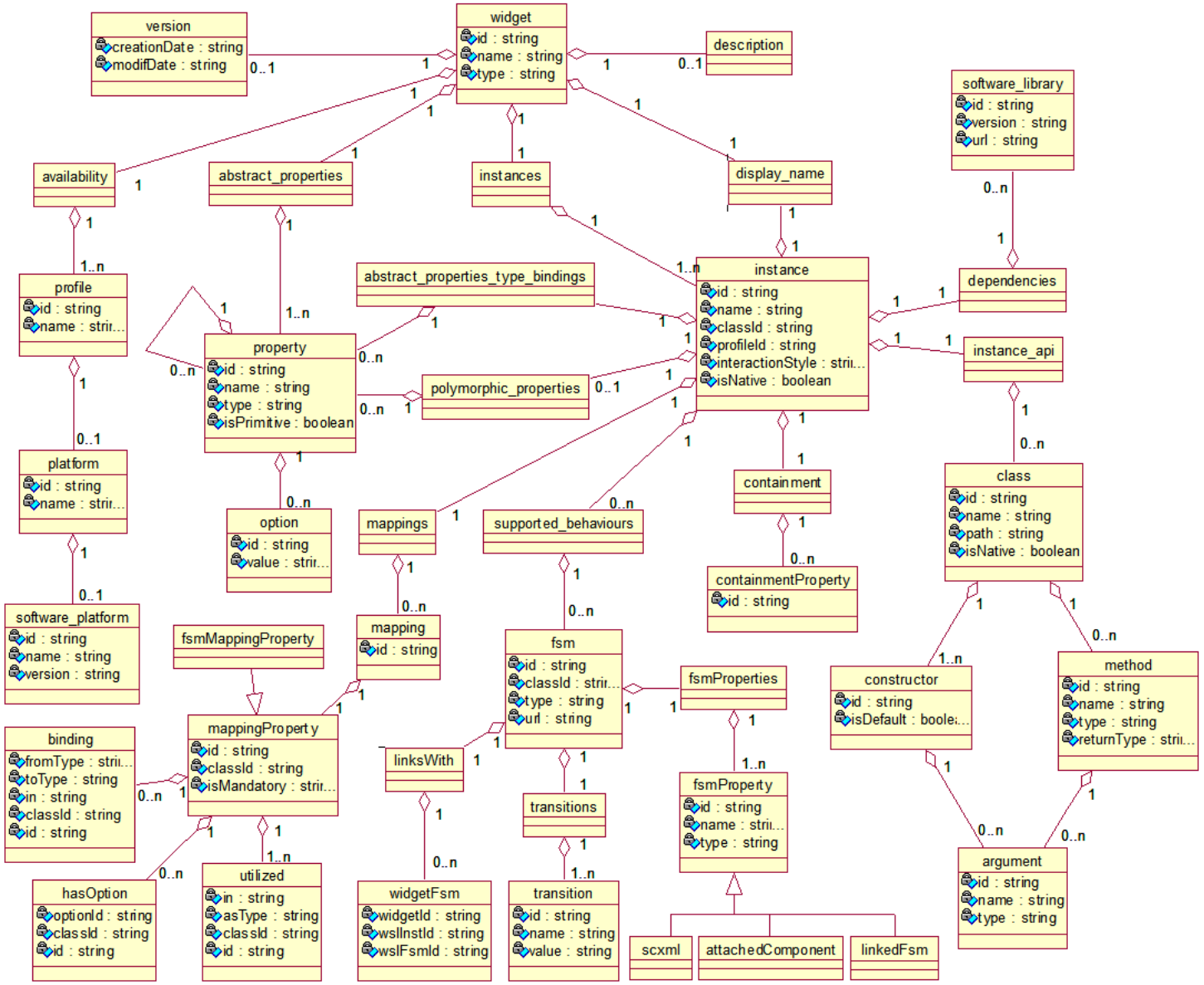
Το συγκεκριμένο μοντέλο (βλέπε Εικόνα 26) μπορεί να κωδικοποιήσει οποιαδήποτε ιεραρχική δομή αντικειμένων μπορεί να συνιστά ιδιότητα ενός αντικειμένου και να αποδώσει αρχικές τιμές σε αυτές. Η εν λόγω πολυπλοκότητα καθορίζεται από την υπολογιστική υλοποίηση ενός αλληλεπιδραστικού σχήματος και είναι αντίστοιχη με αυτή που κωδικοποιείται στα πλαίσια του WSL. Οι τιμές ιδιοτήτων μπορεί να οριστούν στα πλαίσια κάθε αντικειμένου του CUI.

Οι αφηρημένες ιδιότητες ορίζονται σε πρώτο ιεραρχικό επίπεδο ενώ οι πολυμορφικές κάτω από το σχήμα κάθε εναλλακτικού στιγμιοτύπου στο οποίο αναφέρονται.



Εικόνα 26: Το μοντέλο 'WidgetResouce'

Να σημειωθεί πως στα πλαίσια του εν λόγω μοντέλου καταγράφονται ως διαθέσιμα και αρχικοποιούνται μόνο εκείνα τα πολυμορφικά στιγμιότυπα ενός CUI αντικειμένου, από το σύνολο αυτών που υποστηρίζει και έχουν καταγραφεί στο WSL του, τα οποία επιθυμεί ο σχεδιαστής να είναι διαθέσιμα. Το ποια θα επιλεγούν εξαρτάται από τις ανάγκες του σεναρίου. Π.χ. στην περίπτωση που δεν υπάρχει πρόνοια για υποστήριξη ακουστικής αλληλεπίδρασης, οι μη γραφικές αναπαραστάσεις ίσως έχει νόημα να μην επιλεγούν ακόμα και αν υπάρχουν ως διαθέσιμες (ως δυνατότητα). Επίσης να τονιστεί πως στα πλαίσια του WSL μια ιδιότητα μπορεί να οριστεί ως υποχρεωτική σε σχέση με τον προσδιορισμό της. Π.χ. στην ενδεικτική περίπτωση του αντικειμένου soccerPlayer η ιδιότητα που καθορίζει την πλευρά στην οποία ανήκει (side1 ή side2) είναι υποχρεωτικό από τις κατά τόπους αλληλεπιδραστικές υλοποιήσεις του να προσδιοριστεί. Τέτοιου είδους ιδιότητες που ορίζονται ως "isMandatory=true" στο WSL είναι υποχρεωτικό να προσδιορίζονται σε κάθε περίπτωση όταν αυτές είναι αφηρημένες, ενώ στην περίπτωση που υποχρεωτικές έχουν οριστεί και ιδιότητες που είναι πολυμορφικές, αυτές θα πρέπει να προσδιορίζονται μόνο αν και εφόσον το ανάλογο σχήμα στιγμιοτυποποίησης έχει επιλεγεί.



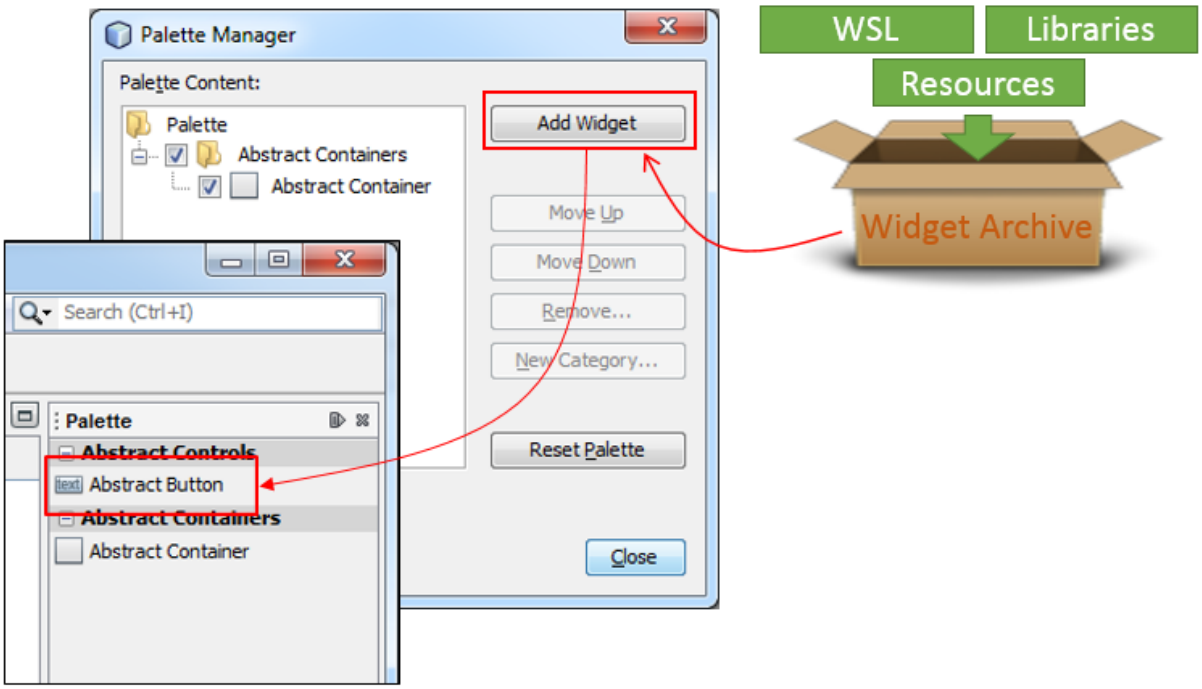
Εικόνα 27: Υψηλού επιπέδου επισκόπηση του WSL

Κεφάλαιο 4 – Σχεδιαστικό Περιβάλλον

Για τη διευκόλυνση της σύνταξης των παραπάνω μοντέλων υλοποιήθηκε ένα εξειδικευμένο IDE πάνω στην πλατφόρμα του NetBeans. Στο μεγαλύτερο μέρος τους τα υλοποιημένα υποσυστήματα (NetBeans modules), έχουν ως στόχο την υποβοήθηση της σταδιακής (incremental) και αλληλεπιδραστικής σύνταξης (interactive specification) στιγμιοτύπων των ανωτέρω μοντέλων (xml files). Παρακάτω περιγράφεται κάθε ένα με λεπτομέρεια.

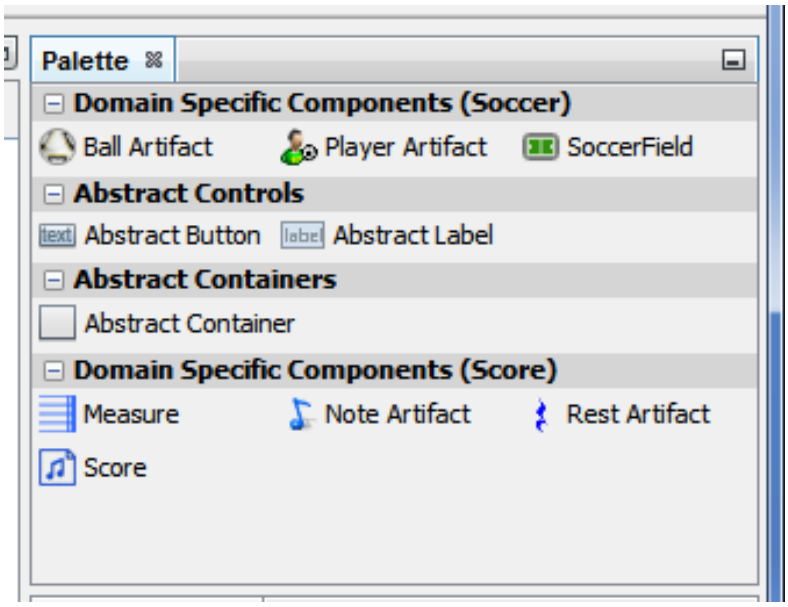
Widget Specification Workflow

Για την εισαγωγή ενός widget στο σύστημα και τη μετέπειτα αξιοποίησή του ορίστηκε ένα archiving φορμάτ (zip), το οποίο έχει προκαθορισμένη δομή και στο οποίο εσωκλείονται το στιγμιότυπο του WSL αρχείου που περιγράφει το widget καθώς και σύνολο συνοδευτικών πόρων (implementation-libraries, εικόνες, κοκ.) που απαιτούνται για την απρόσκοπτη στιγμιοτυποποίησή του εν εκτελέσει. Στην Εικόνα 28, αποτυπώνεται η εν λόγω ροή βημάτων για την εισαγωγή ενός widgetArchive για την περίπτωση του κουμπιού.



Εικόνα 28: Διαδικασία εισαγωγής widgetArchive στο ολοκληρωμένο περιβάλλον ανάπτυξης

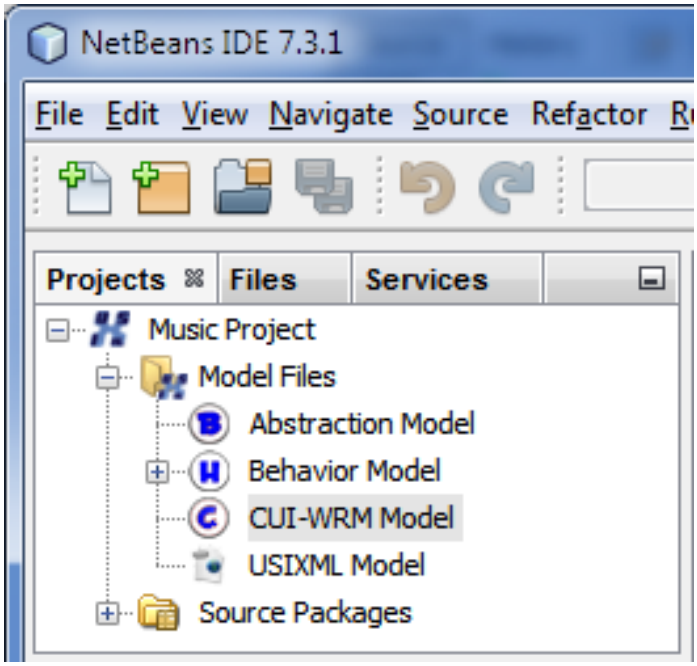
Στην Εικόνα 29 παρουσιάζεται η τρέχουσα κατάσταση της παλέτας, με βάση το υποστηριζόμενο υποκείμενο αφηρημένο-πολυμορφικό λεξικό, όπως αυτό έχει διαμορφωθεί στα πλαίσια πλήθους καινοφανών σεναρίων που έχουν υποστηριχθεί κατά το πρόσφατο παρελθόν.



Εικόνα 29: Στιγμιότυπου παλέτας στα πλαίσια της τρέχουσας κατάστασης του υποστηριζόμενου διαδραστικού λεξικού

Υποσύστημα 'Διαχείρισης Projects' (Project Manager Module)

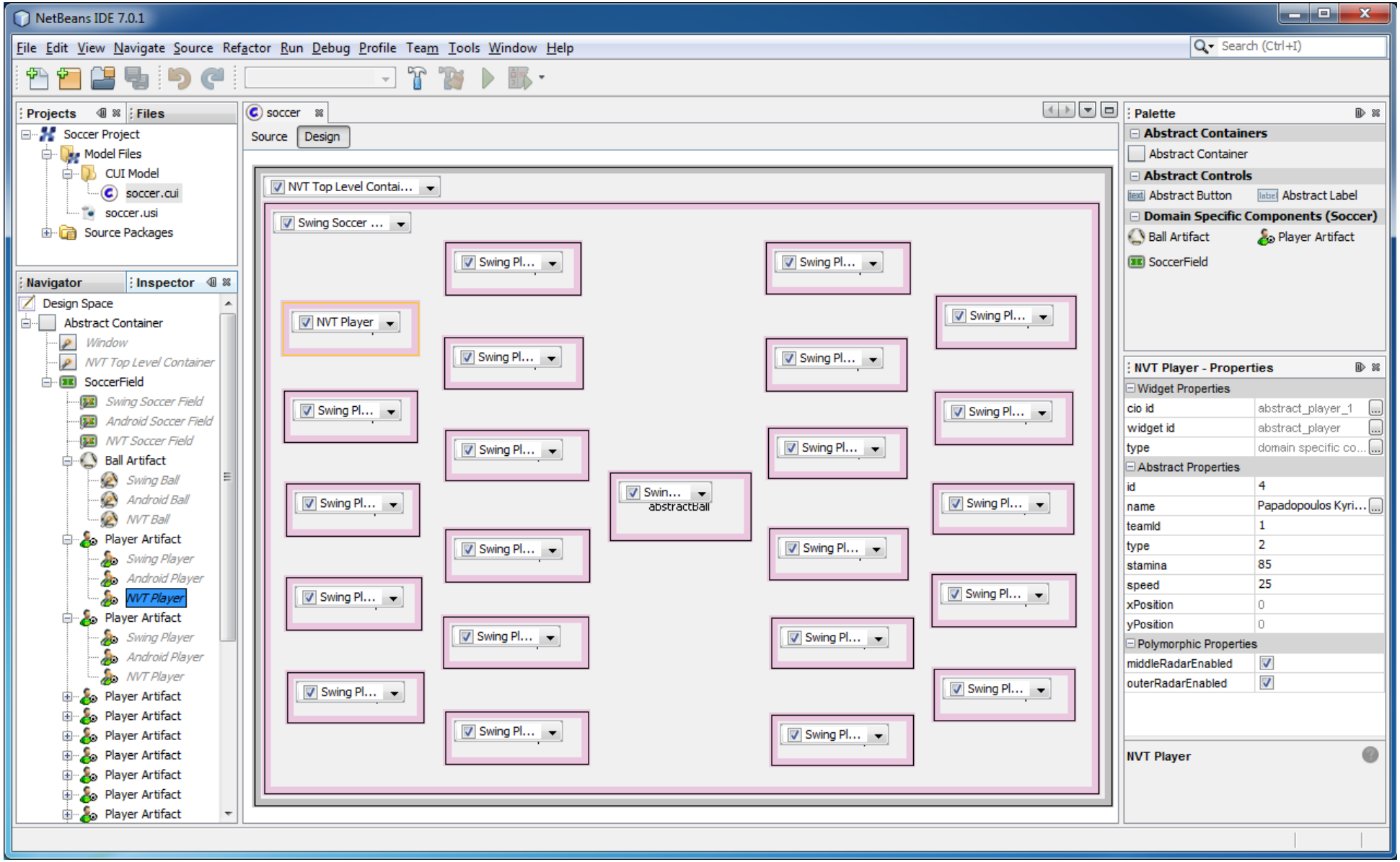
Για τη συνολική διαχείριση σε επίπεδο εργαλείου όλων των μοντέλων που αφορούν ένα συγκεκριμένο αλληλεπιδραστικό σύστημα υλοποιήθηκε η λογική των projects. Η έννοια δεν είναι καινούργια, ωστόσο η δομή και η ενσωματωμένη λειτουργικότητα είναι μοναδική. Το εν λόγω ενδεδειγμένο υποσύστημα του NetBeans, ο project manager, υλοποιεί όλη τη σχετική λειτουργικότητα διαχείρισης, π.χ. δημιουργία, διαγραφή, οργάνωση του συστήματος αρχείων στα πλαίσια κάθε project, ανίχνευση και φόρτωση υπάρχοντος, κοκ.. Με τη δημιουργία ενός νέου project, δημιουργείται ένα xml αρχείο το οποίο συμπεριλαμβάνει όλα τα υποστηριζόμενα μοντέλα μέσα του. Κάθε φορά που ο χρήστης επιλέγει το άνοιγμα ενός συγκεκριμένου τύπου μοντέλου (βλ. Εικόνα 30), γίνεται αυτόματα engaged το ανάλογο υποσύστημα το οποίο επιτρέπει τον αλληλεπιδραστικό προσδιορισμό του (interactive specification). Σε κάθε περίπτωση βέβαια, οποιοδήποτε από τους υποστηριζόμενους τύπους μοντέλων στιγμιότυπο, διατίθεται προς επεξεργασία στη φυσική του αναπαράσταση (xml) διαμέσου ενός ενδεδειγμένου xml editor.



Εικόνα 30: Διαχείριση Projects

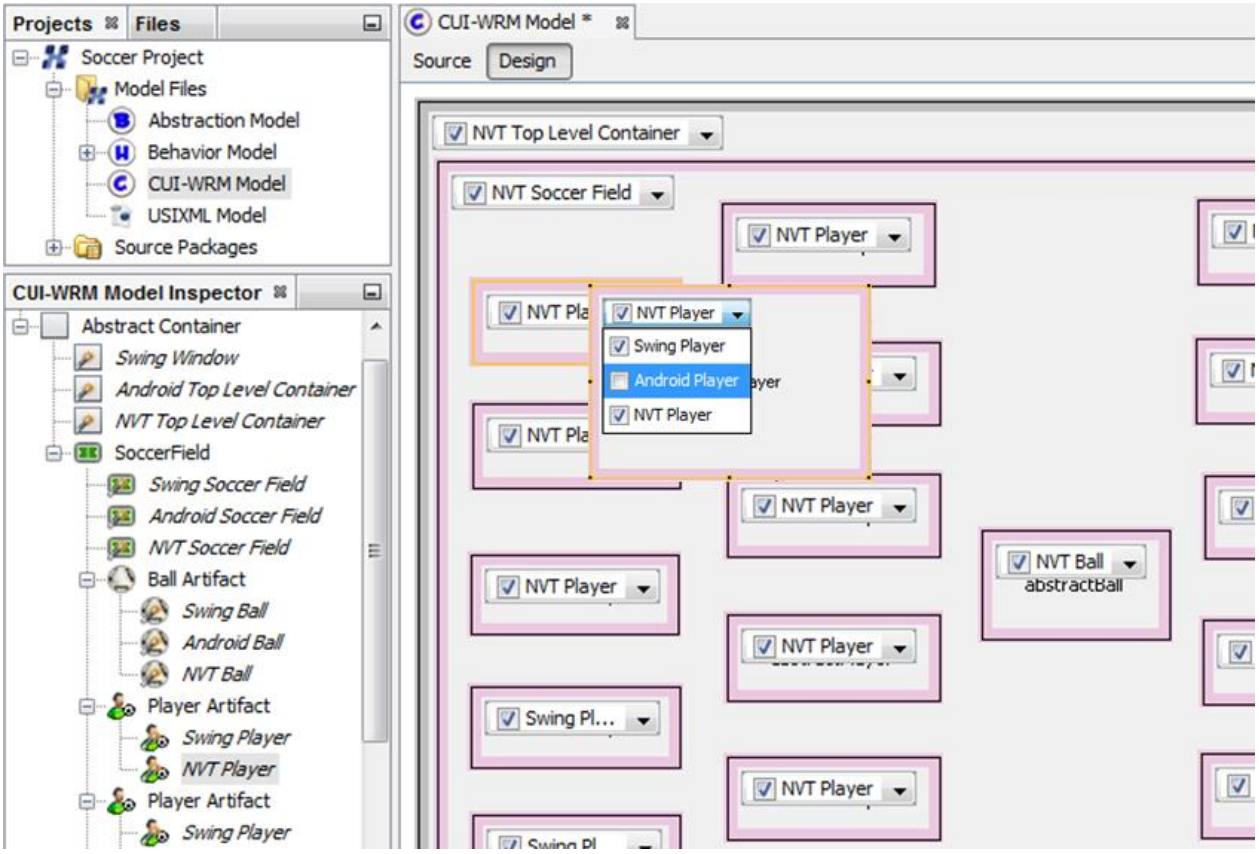
Υποσύστημα Διαχείριση της 'Πολυμορφικής Διεπαφής'(Polymorphic UI Designer Module)

Με τη δημιουργία ενός νέου project, το πρώτο βήμα που εκτελούμε αφορά τη δημιουργία ενός νέου μοντέλου τύπου CUI (βλ. 'CUI-WRM Model' Εικόνα 30), στα πλαίσια του οποίου θα οριστεί η δομή της διεπαφής υπό την έννοια του προσδιορισμού της ιεραρχικής αποσύνθεσης των widgetArchives (αφηρημένων αντικειμένων) που τη συνιστούν. Η διαδικασία αυτή υποστηρίζεται αλληλεπιδραστικά διαμέσου του drag-and-drop αντικειμένων της παλέτας (βλ. Εικόνα 31 πάνω δεξιά) στον κεντρικό σχεδιαστικό χώρο (βλ. Εικόνα 31 κέντρο). Με βάση την πληροφορία που υπάρχει καταγεγραμμένη στο WSL κάθε widgetArchive, μπορεί όπως περιγράφηκε προηγουμένως να συναχθεί εύκολα το αν και ποιους τύπους widgetArchives μπορεί να φιλοξενήσει (host-containment). Ως εκ τούτου, αν επιχειρηθεί το drop σε ένα widgetArchive που δεν υποστηρίζει containment, ή επιχειρηθεί drop ενός μη επιτρεπτού τύπου widgetArchive (child element) στα πλαίσιά του, τότε η ενέργεια αυτή απορρίπτεται.



Εικόνα 31: Ολοκληρωμένο περιβάλλον σχεδίασης (Polymorphic UI Designer)

Όπως ειπώθηκε ήδη η διεπαφή σε αυτό το επίπεδο, αποτελείται από ένα σύνολο στιγμιοτύπων widgetArchives (CUI αντικείμενα). Στο πάνω αριστερό μέρος κάθε στιγμιοτύπου CUI (βλ. εστιασμένο combobox Εικόνα 32), εμφανίζονται όλα τα διαθέσιμα διαδραστικά σχήματα που μπορεί να υποστηρίξει.



Εικόνα 32: Ενδεικτικό στιγμιότυπο προσδιορισμού πολυμορφικής συμπεριφοράς (Design Mode)

Η πληροφορία αυτή αντλείται εν εκτελέσει από το στιγμιότυπο του WSL το οποίο έχει ενσωματωθεί ως μέρος του σχετικού widgetArchive. Το πλήθος των αλληλεπιδραστικών ενσαρκώσεων που επιλέγονται οδηγεί αυτόματα στο σταδιακό specification του widgetResourceModel. Να σημειωθεί πως πατώντας το 'source-mode' στην κορυφή του Designer Module, ο σχεδιαστής μπορεί ανά πάσα στιγμή να έχει δει το παραγόμενο μοντέλο (βλ. Εικόνα 33).

model.cui

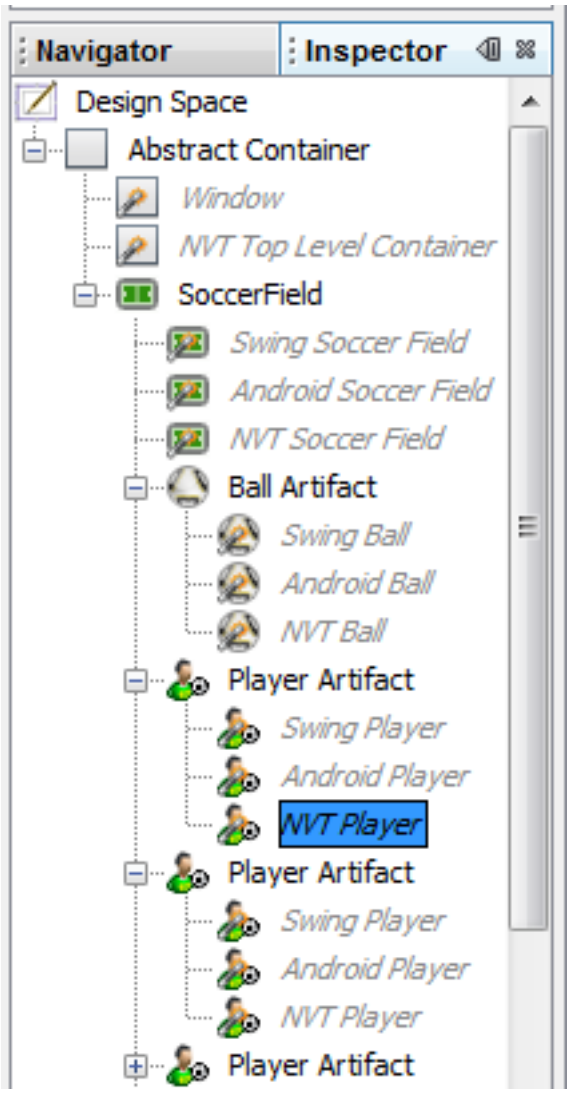
Source | Design

```xml
<?xml version="1.0" encoding="UTF-8"?>
<models id="cui_wr_1">
    <cuiModel id="cuiModel" name="cui-model">
        <version modifDate="2013-09-20T17:09:17.402+01:00">7</version>
        <authorName>Dimitris</authorName>
        <abstractContainer id="abstract_container_1" widgetId="abstract_container">
            <abstractSoccerField id="abstract_soccer_field_1" widgetId="abstract_soccer_field">
                <abstractPlayer id="abstract_player_1" widgetId="abstract_player"/>
            </abstractSoccerField>
        </abstractContainer>
    </cuiModel>
    <widgetResourceModel id="wrModel">
        <widgetResource cioId="abstract_container_1" id="wdg_rsc_1" widgetId="abstract_container">
            <instance id="wr_1_inst_1" name="Window" wslInstId="abstract_container_inst_1"/>
            <instance id="wr_1_inst_3" name="NVTTopLevelContainer" wslInstId="abstract_container_inst_6"/>
        </widgetResource>
        <widgetResource cioId="abstract_soccer_field_1" id="wdg_rsc_2" widgetId="abstract_soccer_field">
            <instance id="wr_2_inst_1" name="SwingSoccerField" wslInstId="abstract_soccer_field_inst_1"/>
            <instance id="wr_2_inst_2" name="AndroidSoccerField" wslInstId="abstract_soccer_field_inst_2"/>
            <instance id="wr_2_inst_3" name="NVTSoccerField" wslInstId="abstract_soccer_field_inst_3"/>
        </widgetResource>
        <widgetResource cioId="abstract_player_1" id="wdg_rsc_3" widgetId="abstract_player">
            <instance id="wr_3_inst_3" name="NVTPlayer" wslInstId="abstract_player_inst_3"/>
        </widgetResource>
    </widgetResourceModel>
</models>
```

Εικόνα 33: Ενδεικτικό στιγμιότυπο CUI μοντέλου (‘source-mode’)

Υποσύστημα για την εναλλακτική αναπαράσταση της Ιεραρχικής Αποσύνθεσης της πολυμορφικής Διεπαφής' (CUI Inspector Module)

Παράλληλα με το υποσύστημα 'Polymorphic UI Designer', υπάρχει η δυνατότητα εστιασμένης γραφικής εποπτείας της πολυμορφικής από- σύνθεσης της διεπαφής διαμέσου του 'CUI Inspector' (βλ. Εικόνα 34).

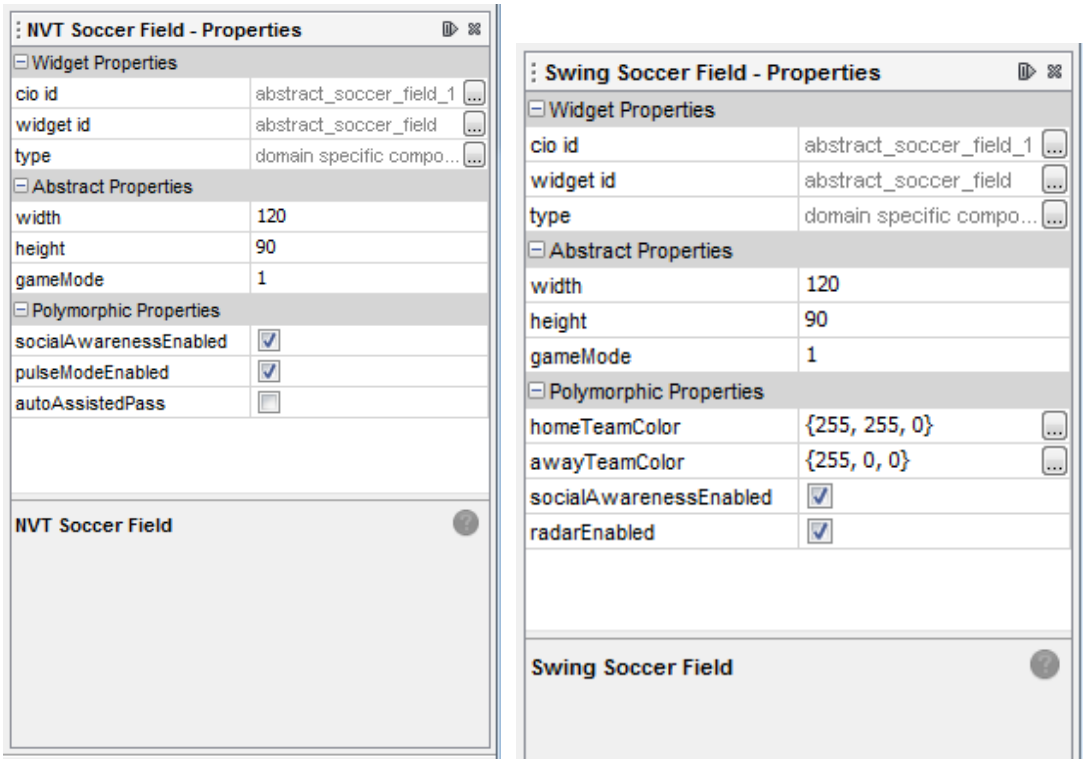


Εικόνα 34: Στιγμιότυπα του 'NetBeans CUI Inspector' module, στα πλαίσια προσδιορισμού ιδιοτήτων διαφορετικών σχημάτων στιγμιοτυποποίησης

Στην Εικόνα 34 παρουσιάζεται τμήμα της πολυμορφικής αποσύνθεσης της διεπαφής της Εικόνα 31.

Υποσύστημα Διαχείρισης Ιδιοτήτων CUI Αντικειμένων (Properties Editor Module)

Στα πλαίσια κάθε αλληλεπιδραστικού σχήματος που επιλέγεται δίνεται η δυνατότητα καθορισμού τιμών των υποστηριζόμενων ιδιοτήτων του (αφηρημένων, πολυμορφικών). Αυτές αντλούνται εν εκτελέσει από το υποσύστημα 'PropertiesEditor', διατρέχοντας το WSL με το οποίο είναι συσχετισμένο το εκάστοτε αντικείμενο που επιλέγεται στον 'Polymorphic UI Designer'.



Εικόνα 35: Στιγμιότυπα του 'NetBeans Properties Editor' module, στα πλαίσια προσδιορισμού ιδιοτήτων διαφορετικών σχημάτων στιγμιοτυποποίησης

Στην Εικόνα 35, φαίνεται ένα στιγμιότυπο στα πλαίσια του οποίου εμφανίζονται οι ιδιότητες δύο εναλλακτικών σχημάτων στιγμιοτυποποίησης του widgetArchive 'SoccerField'. Στη μια περίπτωση εκτίθενται προς προσδιορισμό οι ιδιότητες ενός μη-οπτικού σχήματος, υλοποιημένου με μια οικεία εργαλειοθήκη υποστήριξης μη-οπτικής αλληλεπίδρασης, ενώ στην άλλη οι ιδιότητες του αλληλεπιδραστικού σχήματος που είναι υλοποιημένο στο java/swing. Αυτό που αξίζει να σημειωθεί για τη συγκεκριμένη περίπτωση είναι πως οι δύο εργαλειοθήκες αφορούν domain-specific υλοποιήσεις που δεν είναι εγγενώς παρεχόμενες από τις εργαλειοθήκες στόχο.

Επίσης, δεδομένου ότι κάθε μια εξυπηρετεί ανάγκες διαφορετικού πλαισίου χρήσης κάθε μια μπορεί να επιδεχθεί διαφορετικές δυνατοτήτων (affordances). Π.χ. όσον αφορά τη γραφική αναπαράσταση του swing, υποστηρίζεται χρώμα ενώ στην περίπτωση του μη-οπτικού υποστηρίζονται άλλες δυνατότητες σχετικά με την απόδοση της αίσθησης της χωρικής ενημερότητας η οποία δεν είναι αυθυπάρκτως αγόμενη ως πληροφορία και απαιτεί ειδική πρόνοια.

Η δυνατότητα αυτή είναι ήσσονος σημασίας μιας και επιτρέπει την πλήρη διαχείρισης, κατά τη φάση της σχεδίασης, των δυνατοτήτων των διαχειριζόμενων αλληλεπιδραστικών σχημάτων όποια και να είναι αυτά, αρκεί να έχουν περιγραφεί καταλλήλως με το WSL και την εισαγωγή τους στο σύστημα ως WidgetArchives. Αυτό είναι ένα μοναδικό χαρακτηριστικό της προσέγγισης το οποίο δεν απαντάται στο state-of-the art ανάλογων IDEs, τα οποία επιτρέπουν στο σύνολό τους έκθεση (exposure) για προσδιορισμό μόνο των κοινών (common) χαρακτηριστικών των αντικειμένων. Αυτό σημαίνει ότι αν ένα αντικείμενο μπορεί να υποστηρίξει επιπλέον δράσεις (διαμέσου των μοναδικά υποστηριζόμενων affordances του), αυτές δεν θα παραμένουν ανεκμετάλλευτες μιας και δεν μπορεί να προσδιοριστούν-διαχειριστούν.

Κεφάλαιο 5 – Σενάρια χρήσης

Σενάριο Χρήσης: Παίγνιο 'Tic-Tac-Toe'

Σύντομη περιγραφή

Πρόκειται για το δημοφιλές δι-χρηστικό (2-player) παίγνιο τρίλιζα ('Tic-tac-Toe'), το οποίο ωστόσο για λόγους της παρουσίασης αλλά και δοκιμής των αντοχών της υφιστάμενης μοντελοκεντρικής προσέγγισης διαμορφώθηκε κατά το δοκούν ώστε να μπορεί να υποστηρίξει τα ακόλουθα χαρακτηριστικά: α) οι παίκτες να δύναται να εμπλακούν στο παιχνίδι διαμέσου διαφορετικών τερματικών συσκευών όντας χωρικά διεσπαρμένοι, β) να δίνεται η δυνατότητα υποστήριξης παρατηρητών (observer), που θα μπορούν να παρακολουθήσουν την έκβαση του παιγνίου από δυνητικώς διαφορετικές τερματικές συσκευές (κινητό, desktop), και τέλος γ) να παρέχεται η δυνατότητα να μπορεί το παίγνιο να παραμετροποιηθεί σε τέτοιο βαθμό που να αναδεικνύονται επιπλέον δυνατότητες, χαρακτηριστικά (επίπεδο δυσκολίας, αριθμός κινήσεων κλπ.). Υπό αυτό το σκεπτικό, για το συγκεκριμένο σενάριο υλοποιήθηκε η αναπαράσταση της 'τετράλιζας' αντί της κλασικής 'τρίλιζας' και για τους παρακάτω συνδυασμούς συμμετεχόντων και πλατφορμών (Πίνακας 1).

Συμμετέχων	**Ρόλος**	**Πλατφόρμα**
Παίχτης 1	Παίχτης (player)	Desktop - Native Swing
Παίχτης 2	Παίχτης (player)	Mobile - Android
Παρατηρητής 1	Παρατηρητής (observer)	Desktop - Custom Swing

Πίνακας 1: Ρόλοι - Πλατφόρμες ανά συμμετέχων στο σενάριο του παιγνίου 'tic tac toe'

Στη συνέχεια εξηγείται η υποστήριξή του στα πλαίσια της παρούσης προσέγγισης.

Πολυμορφική Κατηγοριοποίηση Αλληλεπιδραστικών Σχημάτων

Η λογική της προσέγγισης εστιάζει στη χρήση αλληλεπιδραστικών στοιχείων τύπου κουμπί ('Button'), στα πλαίσια ενός εξειδικευμένου οικείου container που υποστηρίζει μεταξύ άλλων την ενδεδειγμένη διάταξη (4x4). Να σημειωθεί πως a) για την κατασκευή της διεπαφής του πρώτου παίχτη (player) χρησιμοποιήθηκε το τετράγωνο κουμπί, όπως αυτό εγγενώς παρέχεται από τη βιβλιοθήκη του java/swing, ενώ b) για την περίπτωση του παρατηρητή στα πλαίσια χρήσης και επέκτασης της ίδιας βιβλιοθήκης χρησιμοποιήθηκε ένα μη-εγγενές και

εξειδικευμένο (custom) αντικείμενο το στρογγυλό κουμπί. Για τον δεύτερο παίκτη χρησιμοποιήθηκε ως κύριο συστατικό επίσης το στρογγυλό κουμπί που σε αυτή όμως τη περίπτωση αποτελεί μη-εγγενές συστατικό μιας άλλης βιβλιοθήκης (android) σε μια τελείως διαφορετική συσκευή (mobile). Σε όλες τις περιπτώσεις για την φιλοξενία των αντίστοιχων εγγενών ή μη αντικειμένων (κουμπιών) χρησιμοποιήθηκαν οι αντίστοιχοι, εξειδικευμένοι containers στα πλαίσια των ανάλογων βιβλιοθηκών.

Για την κωδικοποίηση των παραπάνω αντικειμένων με τη βοήθεια του WSL, έπρεπε να δημιουργηθούν δύο widgetArchives, ένα για το αφηρημένο αντικείμενο τύπου κουμπί (abstractButtonWidgetArchive) και ένα για τους containers (abstractContainerWidgetArchive). Δεδομένου ότι η ουσιαστική αξία του σεναρίου εστιάζει στην πολυμορφική συμπεριφορά των κουμπιών, οι containers δεν θα αναλυθούν περεταίρω. Στην περίπτωση των εναλλακτικών αλληλεπιδραστικών σχημάτων των κουμπιών ο πολυμορφικός διαχωρισμός έγινε όπως φαίνεται στον Πίνακας 2. Όπως φαίνεται μοναδικό αφηρημένο χαρακτηριστικό είναι η ετικέτα, αφού παραμένει ίδια σε κάθε περίπτωση ανεξαρτήτως εναλλακτικής αλληλεπιδραστικής ενσάρκωσης.

			Property Name	Property Type	
Abstract ' Button'	Abstract Properties		text	String	
	Polymorphic Properties	Round Button (*java/swing*)	radius	int	' Tic-Tac-Toe' – specific instantiations
			backgroundColor	String	
			borderThikness	int	
			socialAwareness	boolean	
		Rectangular Button (*java/swing*)	width	int	
			height	int	
		Round Button (*Android*)	radius	int	
			fontColor	String	
			borderColor	String	
			borderSize	int	

Πίνακας 2: Πολυμορφική κατηγοριοποίηση ιδιοτήτων εναλλακτικών αναπαραστάσεων κουμπιών

Αξίζει να σημειωθεί η διαφορά στην κωδικοποίησης ενός τετράγωνου συμβατικού κουμπιού και στις ιδιότητες που φέρει (width, height) και σε αυτές του κυκλικού (radius). Οι διαφορές δεν έγκειται μόνο στο σχήμα αλλά και στις ικανότητες που ξεχωριστά μπορεί να επιδεχθεί κάθε υλοποίηση. Π.χ. στην περίπτωση του Android το κουμπί δύναται να επιδεχθεί αλλαγής χρώματος, κάτι που στην περίπτωση του οικείου κουμπιού δεν γίνεται. Παρά τη σχετική απλότητα του εν λόγω παραδείγματος η εν λόγω δυνατότητα είναι εξαιρετικά σημαντική μιας και επιτρέπει την έκθεση και πλήρης παραμετροποίηση και εκμετάλλευση των ιδιαιτέρων

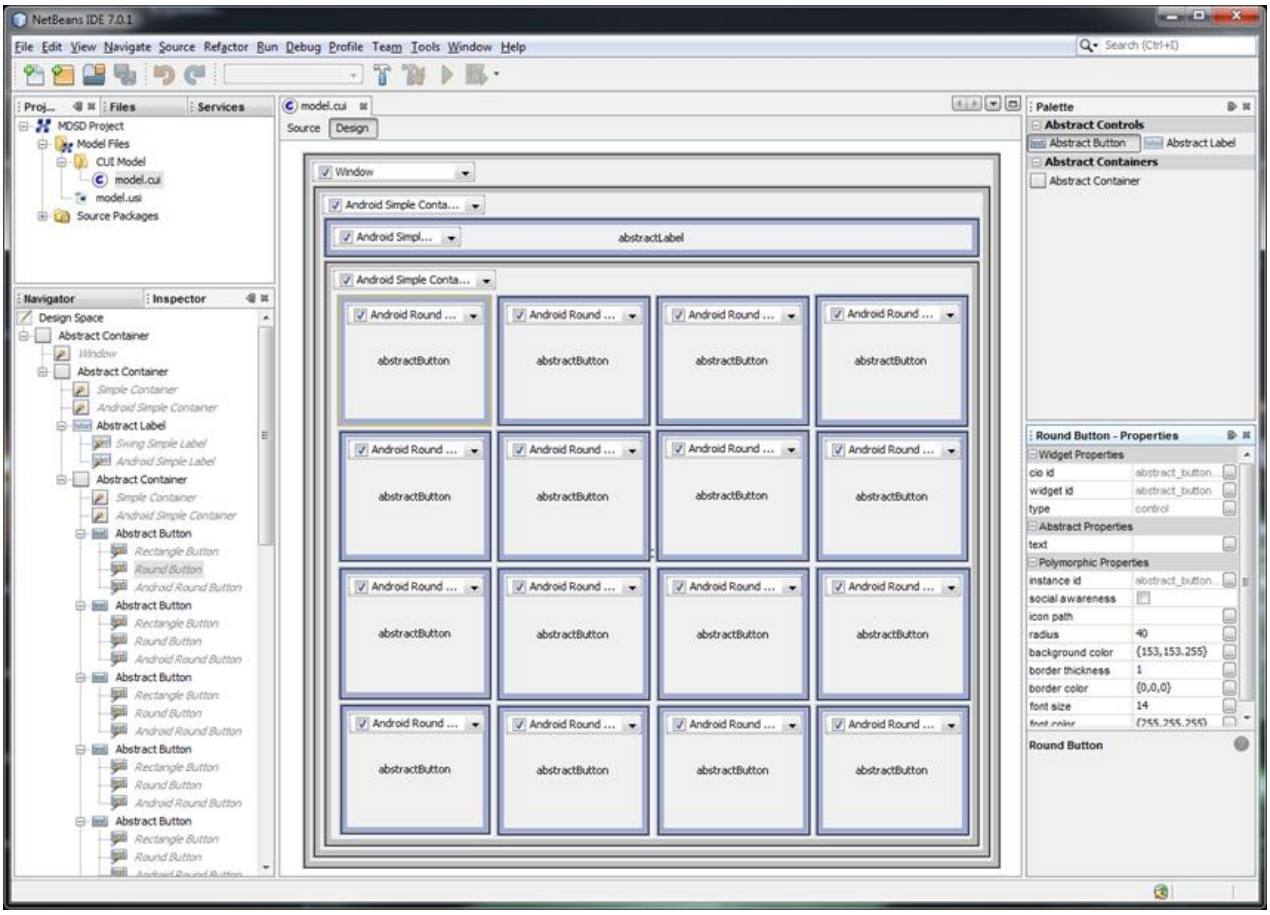
ικανοτήτων κάθε υλοποίησης. Αυτές οι ιδιότητες όπως θα εξηγηθεί στη συνέχεια φορτώνονται και μπορεί να προσδιοριστούν διαμέσου και του οικείου IDE. Στο παρακάτω σχήμα φαίνεται απόκομμα του στιγμιοτύπου του WSL το οποίο συντάχθηκε. Δεδομένης της ακολουθίας που περιγράφηκε στο Κεφάλαιο 5, το εν λόγω WSL, μαζί με τις βιβλιοθήκες (jar και apk αρχεία) των οικείων υλοποιήσεων, συμπιέζονται σε ένα widgetArchive, ονόματι 'abstractButton.wdr'. Με αυτό τον τρόπο γίνεται δυνατή η εισαγωγή τους στην παλέτα του οικείου IDE και η πλήρης εκμετάλλευσή τους από τα προτεινόμενα μοντέλα.

```
<widget id = "abstract_button" name = "AbstractButton" ... >
    <display_name>Abstract Button</display_name>
    ...
    <abstract_properties>
        <property id="abs_prop_1" name="text" type="string"/>
    </abstract_properties>
    <instances>
        <instance id="abstract_button_inst_1" name="SwingRectangleButton" ... >
            <display_name>Swing Rectangle Button</display_name>
            ...
            <polymorphic_properties>
                <property id="inst_1_prop_1" name="icon" type="ImageIcon">
                    <property id="inst_1_prop_2" name="iconPath" type="string" isPrimitive="true"/>
                </property>
                ...
            </polymorphic_properties>
            ...
        </instance>
        <instance id="abstract_button_inst_2" name="SwingRoundButton" classId="inst_2_cl_1" ... >
            <display_name>Swing Round Button</display_name>
            ...
            <polymorphic_properties>
                <property id="inst_2_prop_1" name="icon" type="ImageIcon">
                    <property id="inst_2_prop_2" name="iconPath" type="string" isPrimitive="true"/>
                </property>
                ...
            </polymorphic_properties>
            ...
        </instance>
        <instance id="abstract_button_inst_5" name="AndroidRoundButton" ... >
            <display_name>Android Round Button</display_name>
            ...
            <polymorphic_properties>
                <property id="inst_5_prop_1" name="pressedColor" type="string" isPrimitive="true"/>
                ...
            </polymorphic_properties>
            ...
        </instance>
        ...
    </instances>
</widget>
```

Εικόνα 36: Τμήμα WSL 'abstractButton'

Σχεδίαση του σεναρίου με τη βοήθεια του οικείου IDE

Μετά την εισαγωγή των widgetArchives στο σύστημα εμφανίζονται ως διαθέσιμα για χρήση στην παλέτα (βλ. πάνω δεξιά γωνία Εικόνα 37) του IDE, η οποία αντανακλά την τρέχουσα κατάσταση του εύρους του υποκείμενου διαδραστικού λεξικού (σύνολο widgetArchives). Για τις ανάγκες της 'τετράλιζας', συντάχθηκε μέσω της αλληλεπιδραστικής διαδικασίας drag-and-drop η διεπαφή της Εικόνα 37.



Εικόνα 37: Στιγμιότυπο του εργαλείου πολυμορφικής σχεδίασης για το παιχνίδι της 'τετράλιζας'

Αναλυτικότερα, επελέγην δύο containers μέσα στους οποίους τοποθετήθηκαν δεκαέξι αφηρημένα κουμπιά. Στη συνέχεια, στα πλαίσια κάθε κουμπιού, ενεργοποιήθηκαν οι κατάλληλες μορφές σύμφωνα με το σενάριο (όσες και οι διαθέσιμες εν προκειμένω, στα πλαίσια ενός άλλου σεναρίου θα μπορούσε να είναι υποσύνολο των διαθέσιμων) και προσδιορίστηκαν τιμές σε όλα τα πολυμορφικά χαρακτηριστικά που απαιτούνται για την αρχικοποίηση έκαστου αλληλεπιδραστικού σχήματος (width, height, radius, colors, etc). Ως αφηρημένο (κοινό)

χαρακτηριστικό της ετικέτας ορίστηκε το κενό string, ώστε να αλλάζει μετέπειτα κατά την λειτουργία (κίνηση παίχτη) του παιχνιδιού ανάλογα με τις επιλογές των παιχτών.

Η διαδικασία αυτή υποστηρίχθηκε αλληλεπιδραστικά από το οικείο υποσύστημα 'PropertiesEditor', ο οποίος στα πλαίσια της Εικόνα 37, εστιάζει στα πλαίσια του πάνω αριστερά (top-left) κουμπιού και συγκεκριμένα στις ιδιότητες του αλληλεπιδραστικού σχήματος του κυκλικού κουμπιού του Swing. Τμήμα του αυτόματα παραγόμενου xml αρχείου των μοντέλων CUI και widget-resource φαίνεται στην Εικόνα 38.

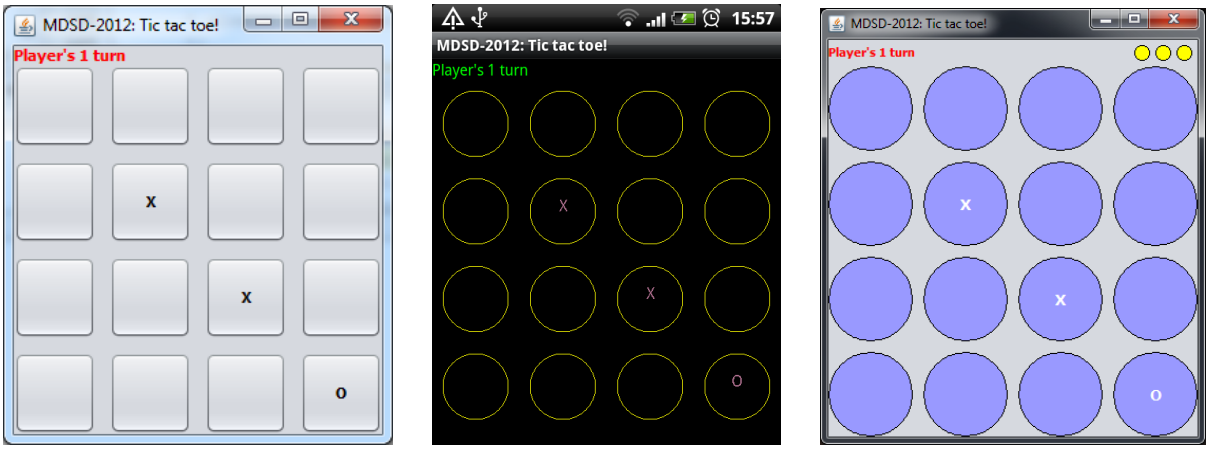
```
<cuiModel id="cuiModel">
    ...
    <abstractContainer id="cio_abstractContainer_4" widgetId="abstract_container">
        <abstractContainer id="cio_abstractContainer_7" widgetId="abstract_container">
            <abstractLabel id="cio_abstractLabel_11" widgetId="abstract_label"/>
            <abstractContainer id="cio_abstractContainer_15" widgetId="abstract_container">
                <abstractButton id="cio_abstractButton_29" widgetId="abstract_button"/>
                ...
            </abstractContainer>
        </abstractContainer>
    </abstractContainer>
</cuiModel>
```

```
<widgetResourceModel id="wrModel">
    ...
    <widgetResource id="wdg_rsc__8" cioId="cio_abstractContainer_7" ... >
        <instance id="inst__9" name="SwingSimpleContainer" ... />
        <instance id="inst__10" name="AndroidSimpleContainer" ... />
    </widgetResource>
    <widgetResource id="wdg_rsc__30" cioId="cio_abstractButton_29" ... >
        <property id="cio_29_prop_1" value="" ... />
        <instance id="cio_29_inst_1" name="SwingRectangleButton" ... >
            <property id="cio_29_prop_2" ... >
                <property id="cio_29_prop_3" value="false" ... />
            </property>
            ...
        </instance>
        <instance id="cio_29_inst_2" name="SwingRoundButton" ... >
            <property id="cio_29_prop_22" ... >
                <property id="cio_29_prop_23" value="false" ... />
            </property>
            ...
        </instance>
        <instance id="cio_29_inst_3" name="AndroidRoundButton" ... >
            <property id="cio_29_prop_34" value="false" ... />
            ...
        </instance>
    </widgetResource>
    ...
</widgetResourceModel>
```

Εικόνα 38: Τμήματα των μοντέλων αναπαράστασής για το παιχνίδι της 'τετράλιζας'

Εν εκτελέσει Στιγμιότυπα σεναρίου Χρήσης

Από τα μοντέλα που δημιουργήθηκαν κατά την σχεδίαση του συγκεκριμένου σεναρίου και μέσω ενδεδειγμένων μηχανισμών [7] προκύπτουν οι κατάλληλες μορφές διεπαφών για τους ρόλους Παίχτης 1, 2 και Παρατηρητής. Στην Εικόνα 39 αναπαρίσταται ένα τέτοιο στιγμιότυπο των διεπαφών (Παίχτης 1, 2 και Παρατηρητής από αριστερά προς τα δεξιά αντίστοιχα) όπου μόλις έχει τελειώσει η σειρά του παίχτη 1 (X ενδείξεις).



(α) (β) (γ)

Εικόνα 39: Εναλλακτικά πολυμορφικά(context-aware) στιγμιότυπα για το παιχνίδι της 'τετράλιζας', α) java/swing εγγενή κουμπιά, β) android μη-εγγενή στρογγυλά κουμπιά, γ) java/swing μη-εγγενή στρογγυλά κουμπιά

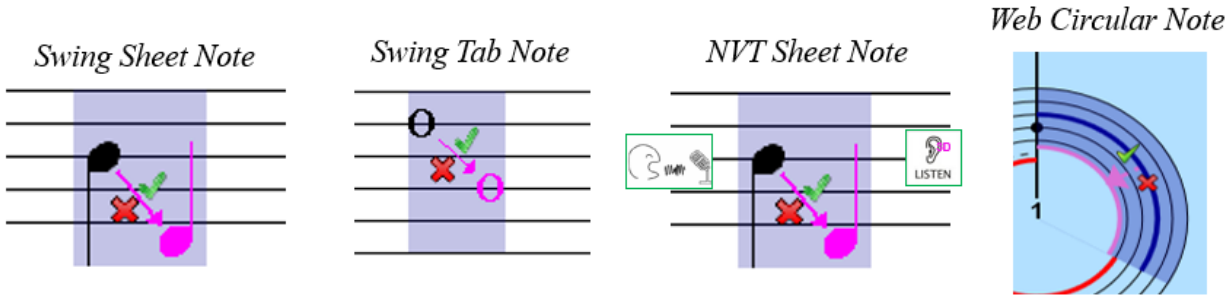
Η υλοποίηση των εν λόγω διεπαφών αξιοποίησε περιβάλλον εκτέλεσης (run-time environment) το οποίο υλοποιήθηκε στα πλαίσια συμπληρωματικής μεταπτυχιακής εργασίας[26] έτσι ώστε να αξιοποιεί τις προδιαγραφές WSL που σχεδιάστηκαν.

Σενάριο Χρήσης: 'Κατανεμημένη Μάθηση Μουσικής'

Σύντομη περιγραφή

Το συγκεκριμένο σενάριο εστιάζει στα πλαίσια της σύγχρονης συνεργατικής κατανεμημένης μάθησης μουσικής. Συγκεκριμένα, εξετάστηκε το σενάριο όπου χρήστες διαφόρων επιπέδων στη μουσική (αρχάριοι, προχωρημένοι), εμπλέκονται διαμέσου διαφορετικών πλαισίων χρήσης, σε μια σύγχρονη συνόδου (synchronous collaborative session), εποπτευόμενοι και καθοδηγούμενοι ανά πάσα στιγμή από ένα καθηγητή μουσικής, με σκοπό τη διόρθωση εσκεμμένων λαθών επί ενός διαμοιραζόμενου τραγουδιού. Τέτοιου είδους διατάξεις να τονιστεί ότι τέτοιου είδους διατάξεις δεν υποστηρίζονται από ευρέως διαδεδομένα εργαλεία διαμοιρασμού πόρων και επικοινωνίας πληροφορίας (βλ. forums, portals, chat-based εργαλεία, κοκ.). Αναλυτικότερα, οι χρήστες που είναι οικείοι με προχωρημένες έννοιες της μουσικής (π.χ. καθηγητής) ενεργούν είτε διαμέσου της μεταφοράς του πενταγράμμου (SwingSheetScore), η οποία αναπτύχθηκε ως επέκταση στη βάση της εργαλειοθήκης java/swing, είτε διαμέσου του ίδιου τύπου μεταφοράς υλοποιημένου ωστόσο στα πλαίσια μιας οικείας οπτικής εργαλειοθήκης

(Non-Visual Toolkit), για την υποστήριξη χρηστών με βλάβες στο οπτικό κανάλι. Από την άλλη πλευρά, οι αρχάριοι με τη μουσική χρήστες μπορούν να χρησιμοποιήσουν μια εκ των μεταφορών της ταμπλατούρας και του κυκλικού μέτρου. Η πρώτη δίνει έμφαση στη θέση των δαχτύλων στο τάστο πάνω στο οποίο πρέπει να πατηθεί η χορδή και είναι υλοποιημένη ως επέκταση του java/swing, ενώ η δεύτερη (μεταφορά κυκλικών μέτρων), δίνει έμφαση στη χρονική διάρκεια που μια νότα πρέπει να παραμείνει πατημένη έτσι ώστε να παράγει τον επιθυμητό κάθε φορά ήχο. Συγκεκριμένα, το μεγαλύτερο τόξο, το περισσότερο που η χορδή πρέπει να παραμείνει πατημένη. Πρωτότυπα της υλοποίησης των αλληλεπιδραστικών μεταφορών για την υποστήριξη του σεναρίου φαίνονται στην Εικόνα 47.



Εικόνα 40: Εναλλακτικές αλληλεπιδραστικές μεταφορές παρουσίασης μουσικού περιεχομένου

Να σημειωθεί πως όλες οι υλοποιήσεις υλοποιούν την κατάσταση ενδιάμεσης διάδρασης (interim-feedback), στο οποίο υπεισέρχεται μια νότα κατά την αλλαγή κάποιου εκ των βασικών χαρακτηριστικών της (π.χ. αλλαγή οκτάβας, τόνου, κοκ.). Σε αυτή την περίπτωση το προτεινόμενο μοντέλο της νότας συνυπάρχει με το αρχικό, έως ότου είτε: α) υπερψηφιστεί από όλους τους συμμετέχοντας η αποδοχή της αλλαγής, οπότε οι προτεινόμενες αλλαγές κατοχυρώνονται και το αρχικό μοντέλο αποδεσμεύεται και καταστρέφεται, είτε β) σε περίπτωση καταψήφισης, οπότε το προτεινόμενο μοντέλο καταστρέφεται και η νότα παραμένει στην αρχική της κατάσταση. Στην Εικόνα 47, ο ροζ χρωματισμός υποδεικνύει στο χρήστη την προτεινόμενη αλλαγή-θέση.

Πολυμορφική Κατηγοριοποίηση Αλληλεπιδραστικών Σχημάτων

Τα διακριτά διαδραστικά στοιχεία των τριών υλοποιήσεων σε κάθε περίπτωση είναι: 1) ο υποδοχέας (container) των μέτρων, 2) ο υποδοχέας των νοτών, και 3) η νότα. Στην περίπτωση των μεταφορών πενταγράμμου και μόνο υπάρχει και ένα τέταρτο, 4) η παύση (rest). Ως εκ

τούτου, για την εισαγωγή των εν λόγω υπολογιστικών υλοποιήσεων στο σύστημα (μοντέλα, IDE), πρέπει να δημιουργηθούν τέσσερα νέα widgetArchives, ένα που θα αφορά τις εναλλακτικές αναπαραστάσεις των containers (‘*AbstractScore.wdr*’), ένα τις εναλλακτικές ενσαρκώσεις των μέτρων (‘*AbstractMeasure.wdr*’), ένα των νοτών (‘*AbstractNote.wdr*’) και ένα τέταρτο τα αλληλεπιδραστικά σχήματα της παύσης (‘*AbstractRest.wdr*’). Στα πλαίσια καθενός από τα τέσσερα πρέπει να δημιουργηθεί ένα στιγμιότυπο του WSL διαχωρίζοντας κοινές και πολυμορφικές ιδιότητες και θα κωδικοποιεί τόσο τα διαθέσιμα σχήματα, όσο και το API τους. Παρακάτω φαίνεται ο πολυμορφικός διαχωρισμός των ιδιοτήτων των widgetArchives της νότας (βλ. Πίνακας 3), του μέτρου (βλ. Πίνακας 4) και της παρτιτούρας (βλ. Πίνακας 5).

			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}
		Sheet Note (*java/JNVT2*)	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
		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 3: Πολυμορφική κατηγοριοποίηση ιδιοτήτων της Νότας ‘Music Note’

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
		Sheet Measure (*java/JNVT2*)	timeNumerator	int
			timeDenominator	int
			clef	char {C, F, G}
			clefLine	int {1-5}
		Circular Measure (*web/prototype*)	timeNumerator	int
			timeDenominator	int
			clef	char {C, F, G}
			clefLine	int {1-5}
			focused	boolean

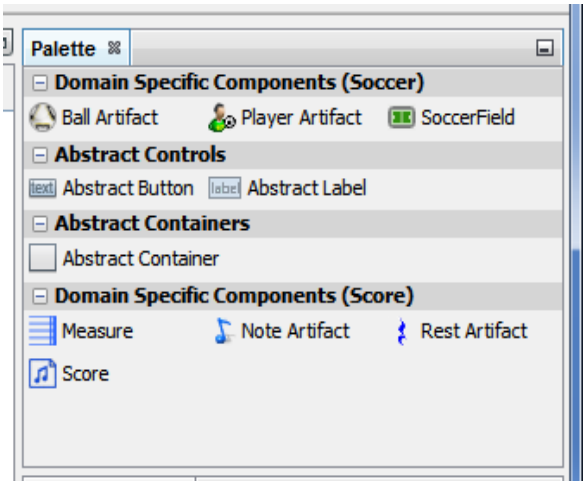
Πίνακας 4: Πολυμορφική κατηγοριοποίηση ιδιοτήτων του Μέτρου'Music Measure'

			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] }
		Sheet Score (*java/JNVT2*)	tempo	int
			duration	int
			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 5: Πολυμορφική κατηγοριοποίηση ιδιοτήτων της Παρτιτούρας 'Music Score'

Τα εν λόγω στιγμιότυπα του WSL πακετάρονται σε τέσσερα διαφορετικά widgetArchives, μαζί με τις βιβλιοθήκες και τους λοιπούς πόρους (εικόνες) που είναι απαραίτητοι σε κάθε περίπτωση για τη στιγμιοτυποποίηση των αλληλεπιδραστικών σχημάτων. Εν συνεχεία φορτώνονται μέσω

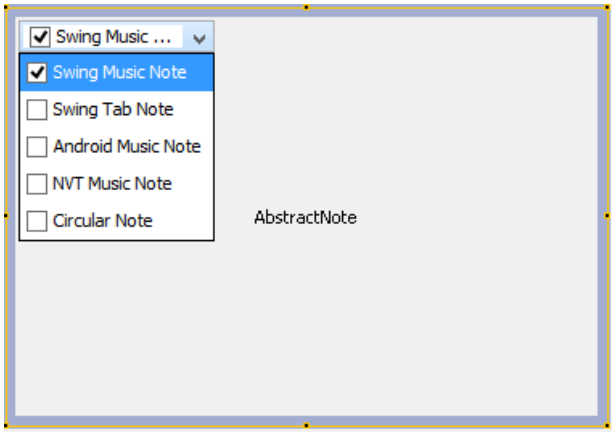
του ενδεδειγμένου διαλόγου στο οικείο IDE, επεκτείνοντας το διαθέσιμο υποκείμενο πολυμορφικό λεξικό.



Εικόνα 41: Ενημερωμένη Παλέτα IDE στα πλαίσια υποστήριξης του σεναρίου της μουσικής

Σχεδίαση του σεναρίου με τη βοήθεια του οικείου IDE

Στην Εικόνα 43 φαίνεται πως μέσω drag-and-drop αντικειμένων από την παλέτα (pallete) στο σχεδιαστικό χώρο (design space), ξεκινάει η σχεδίαση της διεπαφής, υπό την έννοια ιεραρχικά αποσυντιθέμενων αφηρημένων αντικειμένων (widgetArchives). Στα πλαίσια καθενός επιλέγεται το σύνολο των διαδραστικών σχημάτων που επιθυμείται να ενεργοποιηθούν (βλ. Εικόνα 42).



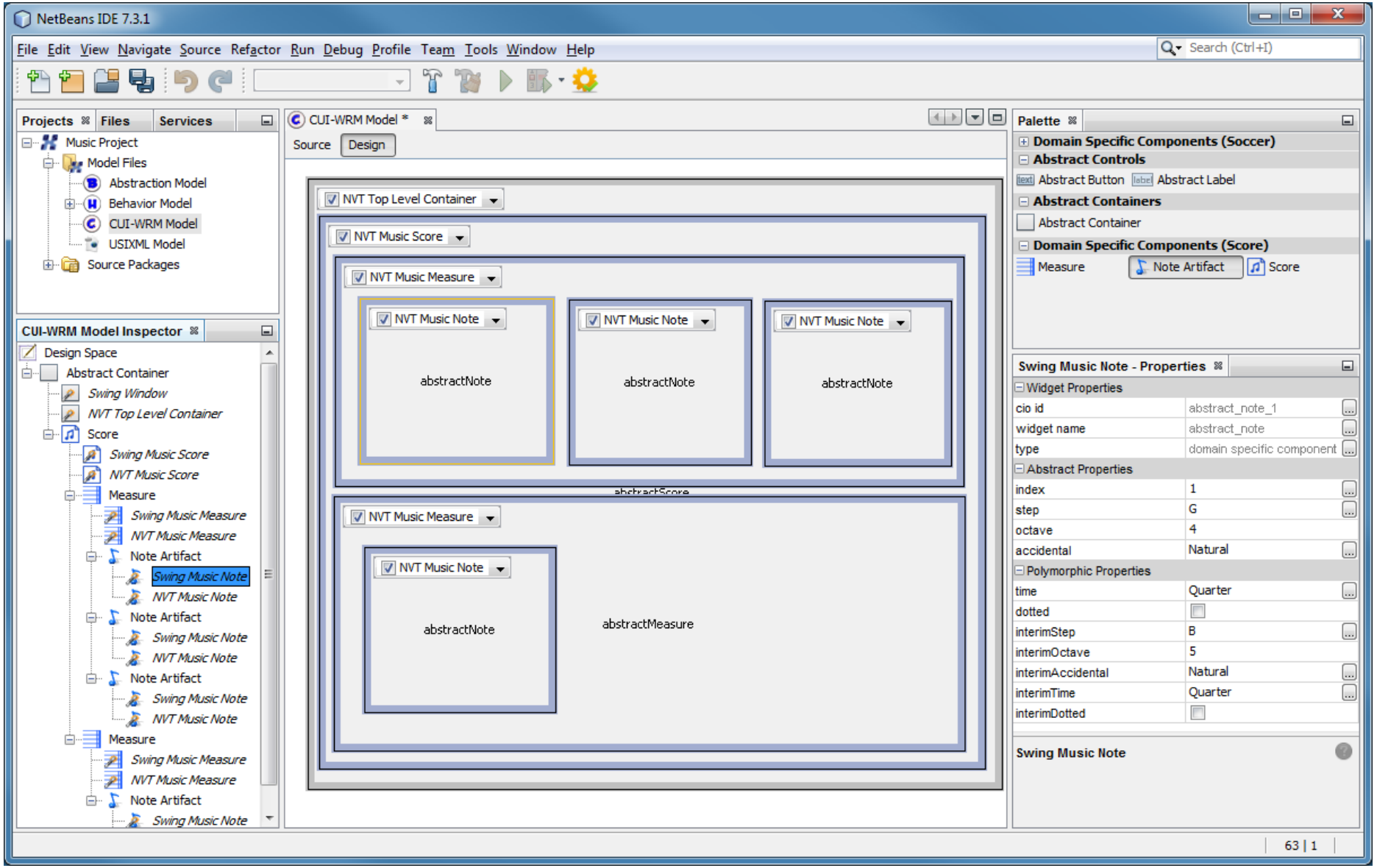
Εικόνα 42: Στιγμιότυπο επιλογής-ενεργοποίησης πολυμορφικών σχημάτων στα πλαίσια του AbstractNote widgetArchive

Τα διαθέσιμα σχήματα φορτώνονται εν εκτελέσει μέσα από το στιγμιότυπο του WSL του widgetArchive στο οποίο αναφέρεται το εκάστοτε abstractWidget που έχει επιλεγεί (‘AbstractNote), το όνομα του οποίου σεκάθε περίπτωση απεικονίζεται στο κέντρο της απεικόνισης του ‘widget’ στο σχεδιαστικό χώρο. Να σημειωθεί πως αυτή η δυνατότητα δεν θα ήταν δυνατή δεδομένης της μη ύπαρξης ενός αγνωστικού υλοποίησης xml σχήματος κωδικοποίησης (WSL).

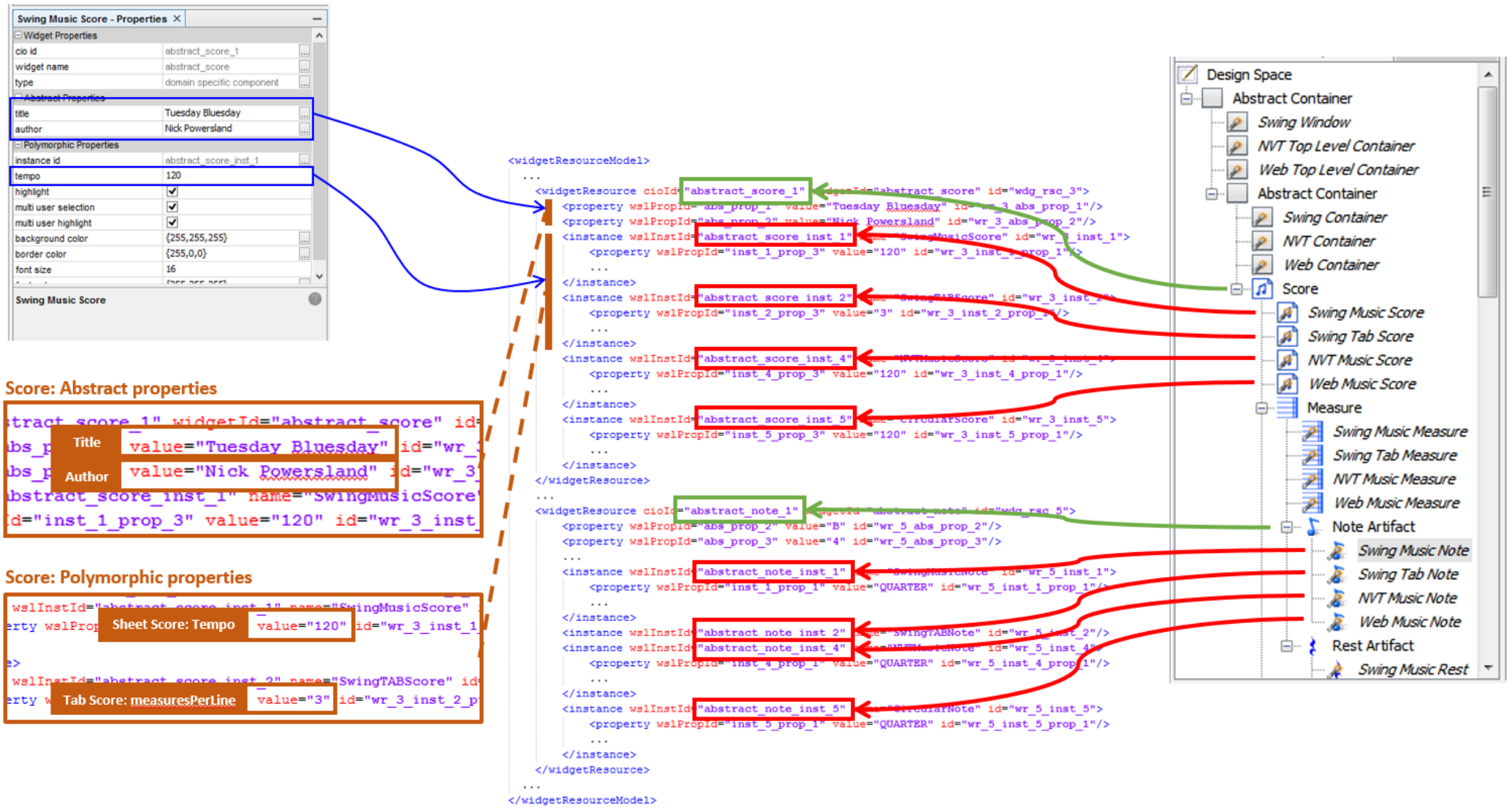
Εν συνεχεία στην Εικόνα 44, φαίνεται πως οι σχεδιαστικές αυτές επιλογές αποθηκεύονται-κωδικοποιούνται στα πλαίσια ενός στιγμιοτύπου του WidgetResource μοντέλου. Συγκεκριμένα φαίνεται πως αντιστοιχίζονται οι ενεργοποιημένες ενσαρκώσεις κάθε στιγμιοτύπου widgetArchive στα αντίστοιχα στοιχεία (βλ. ετικέτες ‘instance’) του WidgetResource μοντέλου. Αναλυτικότερα, με πράσινα βέλη εμφανίζονται οι δηλώσεις των widgetArchives μέσα στην ιεραρχία της διεπαφής, ενώ με κόκκινα βέλη απεικονίζονται τα ενεργοποιημένα στα πλαίσια των προηγούμενων αλληλεπιδραστικά σχήματα. Επιπλέον, με μπλε βέλη απεικονίζεται στιγμιότυπο το οποίο απεικονίζει το εργαλείο (υποσύστημα ‘PropertiesEditor’) με την οποία προσδιορίζονται τιμές στην ιδιότητες ενός σχήματος.

Όπως φαίνεται οι αφηρημένες ιδιότητες (τίτλος, συγγραφέας) δηλώνονται απευθείας κάτω από το WidgetResource στοιχείο, ενώ οι πολυμορφικές (tempo, measuresPerLine, κοκ.) κάτω από το στιγμιότυπο στο οποίο αναφέρονται (java/swing Πεντάγραμμο).

Στην Εικόνα 45, φαίνεται πως οι ιδιότητες και οι τιμές που προσδιορίζονται στα πλαίσια αυτών συσχετίζονται με συγκεκριμένες API κλήσεις ανάλογα με το σχήμα στο οποίο αναφέρονται. Συγκεκριμένα βλέπουμε πως το αφηρημένο στοιχείο τύπου ‘AbstractScore’ widgetArchive του CUI, παίρνει τιμή για τον τίτλο (title), ο οποίος είναι αφηρημένο χαρακτηριστικό, ‘Tuesday Bluesday’. Εν συνεχεία βλέπουμε πως μέσω του γνωρίσματος ‘wslPropId’ της ετικέτας ‘property’, αντιστοιχίζεται ο τίτλος του widgetResouce model με την σημασιολογικά ίδια ιδιότητα του WSL. Η συγκεκριμένη αφηρημένη ιδιότητα του WSL, όπως και κάθε ιδιότητα, αντιστοιχίζεται μέσω κατάλληλων συσχετίσεων (mappings) με τα κωδικοποιημένα τμήματα του API που χρειάζεται να κληθούν ώστε να διαχειριστεί (κλήση μεθόδου, αρχικοποίηση διαμέσου κατασκευαστή, κοκ.). Στη συγκεκριμένη περίπτωση, η ιδιότητα του τίτλου για την επιλεγμένη ενσάρκωση του πενταγράμμου του swing αρχικοποιείται διάμεσου της κλήσης του σχετικού κατασκευαστή.



Εικόνα 43: Στιγμιότυπο του 'Polymorphic UI Designer' στα πλαίσια του σεναρίου της μουσικής



Εικόνα 44:Απεικόνιση κωδικοποίησης πολυμορφικών σχημάτων και τιμών ιδιοτήτων στα πλαίσια του 'WidgetResource' μοντέλου

Με το ίδιο σκεπτικό η πολυμορφική ιδιότητα του ‘tempo’ που αφορά αποκλειστικά και μόνο ως χαρακτηριστικό την ενσάρκωση του java/swing πενταγράμμου (με ορισμένη τιμή 120), αρχικοποιείται, ακολουθώντας την τιμή του γνωρίσματος wslPropertyId στο WidgetResource Μοντέλο, μέσω της API κλήσης ‘setTempo’ στην οποία καταλήγουν οι εμφανιζόμενες συσχετίσεις από το WidgetResource μοντέλο στο WSL. Η διαδικασία αυτή αναλαμβάνεται εν εκτελέσει από ενδεδειγμένα runtime περιβάλλοντα (Platform Servers)[7]. Όπως έχει ήδη ειπωθεί η πολυπλοκότητας της χειροκίνητης διασύνδεσης των ιδιοτήτων στο σύνολό τους στα πλαίσια των ενεργοποιημένων στιγμιότυπων, αποκρύπτεται πλήρως από το σχεδιαστή υλοποιούμενη από το σχεδιαστικό περιβάλλον αυτόματα. Στην Εικόνα 46 παρουσιάζονται ενδεικτικά στιγμιότυπα όπως αυτά πάρθηκαν μετά τη διερμήνευση της ενοποιημένης περιγραφής της διεπαφής στα πλαίσια διαφορετικών πλαισίων χρήσης.

Σενάριο Χρήσης: ‘Ποδοσφαιράκι’

Σύντομη περιγραφή

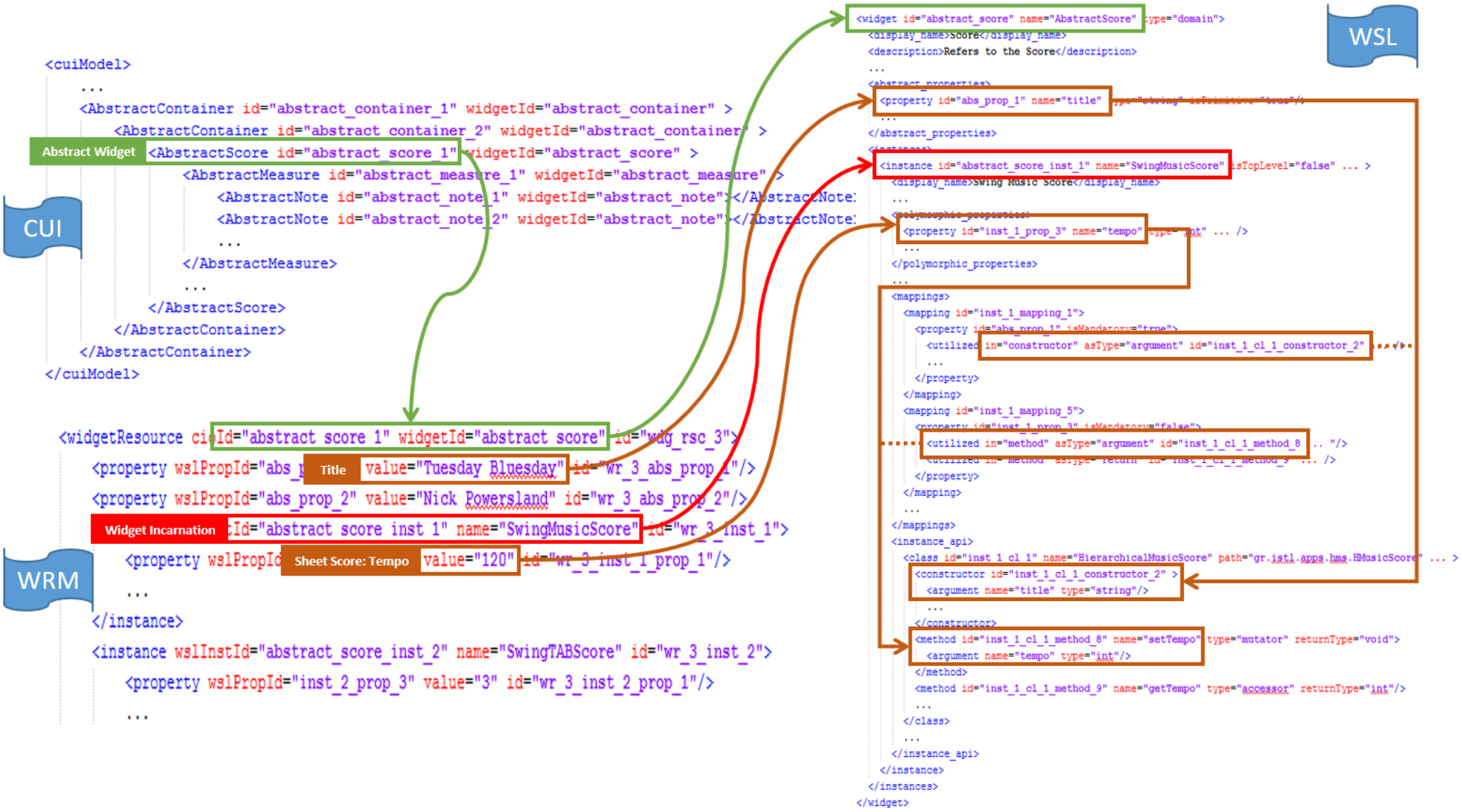
Στο συγκεκριμένο σενάριο στόχος είναι η υποστήριξη του παιγνίου ‘ποδοσφαιράκι’ υπό το πρίσμα ριζικά ετερογενών πλαισίων χρήσης. Να σημειωθεί ότι το εν λόγω παίγνιο είναι αδύνατο να υποστηριχθεί σε οποιαδήποτε μορφή του από οποιαδήποτε τρέχουσα μοντελοκεντρική προσέγγιση. Για τις ανάγκες της εργασίας ορίστηκε η ανάγκη υποστήριξης ενός σεναρίου κατά το οποίο ένας χρήστης μπορεί να εμπλακεί στο παίγνιο, είτε:

α) διαμέσου μιας συμβατικής γραφικής διεπαφής στα πλαίσια ενός PC, είτε

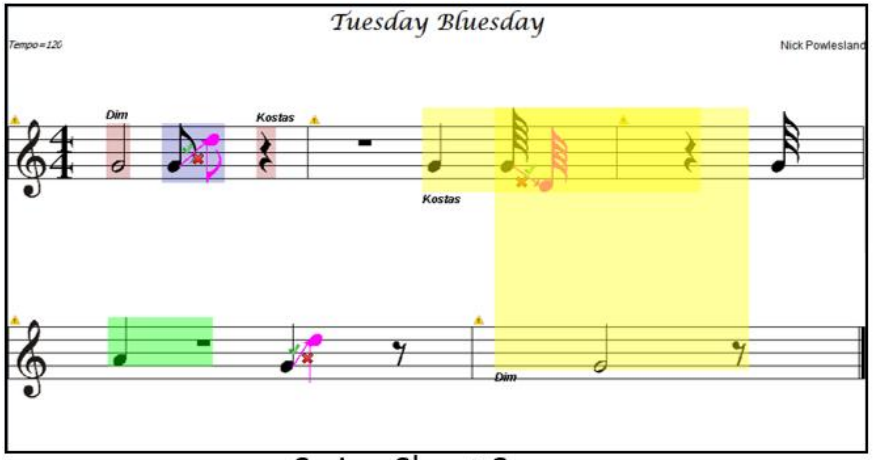
β) διαμέσου κινητής συσκευής Android, σε περίπτωση κινητού χρήστη, είτε τέλος σε περίπτωση προβλημάτων αξιοποίησης του οπτικού καναλιού

γ) διαμέσου πολυκαναλικής διεπαφής υλοποιημένης στην κορυφή μιας οικείας μη-οπτικής εργαλειοθήκης (non-visual toolkit).

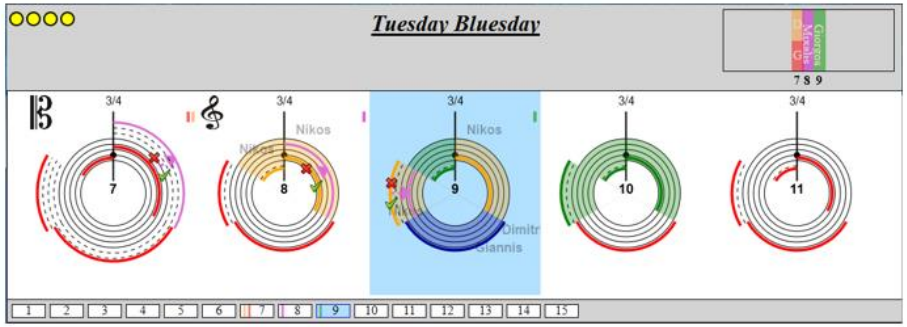
Υποστηρίζεται επίσης η εμπλοκή ενός χρήστη ως θεατή (ρόλος: observer). Ο Πίνακας 6 συνοψίζει όλες τις πιθανές διατάξεις του σεναρίου.



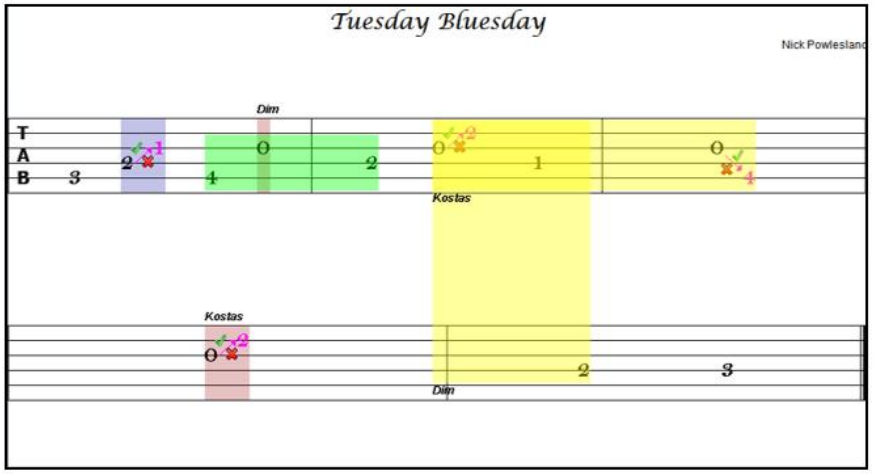
Εικόνα 45: Αντιστοίχιση ιδιοτήτων του 'widgetResource' με εγγενής API κλήσεις του πολυμορφικού αντικειμένου της παρτιτούρας, κωδικοποιημένες διαμέσου του WSL



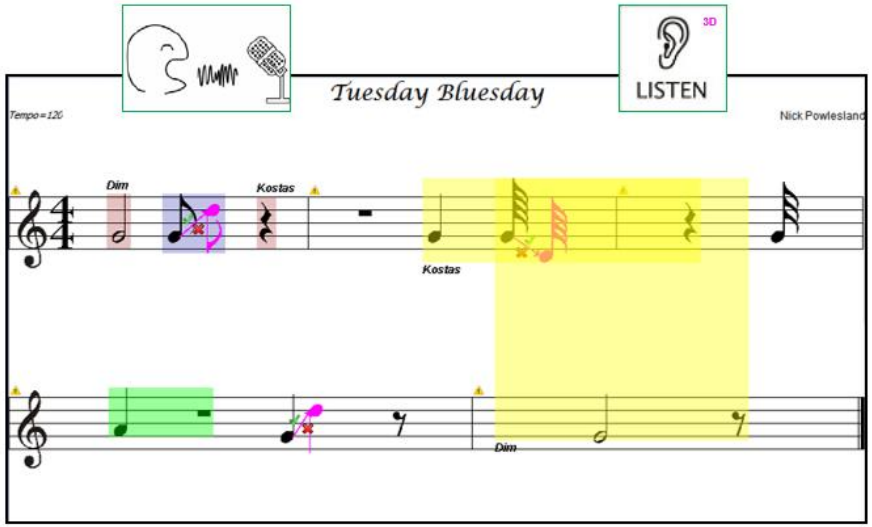
Swing Sheet Score



Web Circular Score



Swing Tab Score



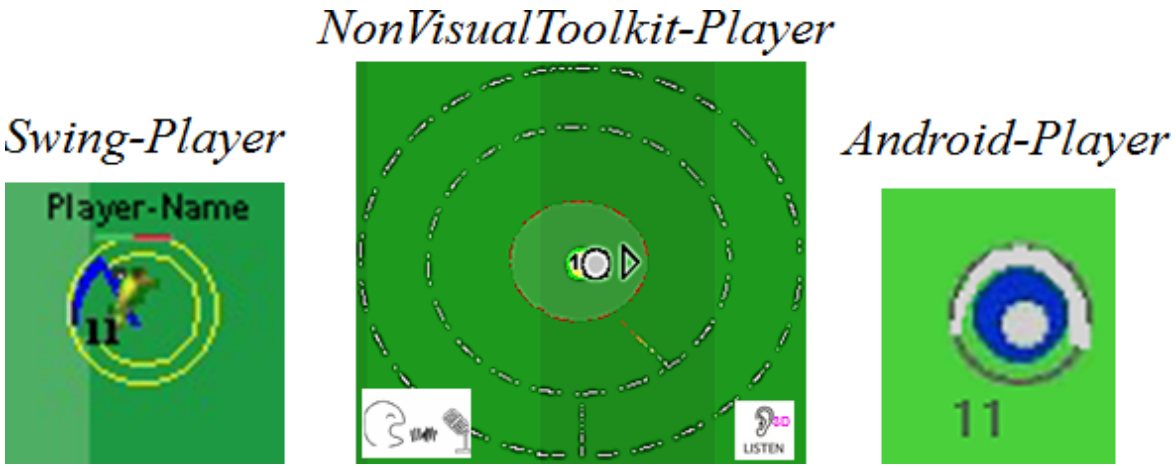
NVT Sheet Score

Εικόνα 46: Εν εκτελέσει πολυμορφικά στιγμιότυπα της αφηρημένης διεπαφής σε διαφορετικά πλαίσια χρήσης

Συμμετέχων	Ρόλος	Πλατφόρμα	Στερεότυπο χρήστη
Παίχτης 1	Παίχτης (player)	Desktop - NVToolkit	Με προβλήματα όρασης ή Τυφλός
Παίχτης 2	Παίχτης (player)	Mobile - Android	Με κανονική όραση
Παρατηρητής 1	Παρατηρητής (observer)	Desktop - Custom Swing	Με κανονική όραση

Πίνακας 6: Ρόλοι - Πλατφόρμες ανά συμμετέχων στο σενάριο του παιγνίου 'Ποδοσφαιράκι'

Η Εικόνα 47 παρουσιάζει τρία ενδεικτικά στιγμιότυπα των εναλλακτικών διεπαφών που ικανοποιούν κάθε περίπτωση. Και οι τρεις αναπαραστάσεις υλοποιήθηκαν καταβάλλοντας χαμηλού επιπέδου προγραμματιστικά-έντονο προγραμματισμό σε επίπεδο εργαλειοθήκης. Για λόγους οικονομίας της συζήτησης, δεν θα ασχοληθούμε με τις τεχνικές υλοποίησής τους παρά μόνο με το πώς ενσωματώνονται στη γλώσσα ως εναλλακτικά πολυμορφικά διαδραστικά σχήματα.



Εικόνα 47: Εξειδικευμένα ανά πλατφόρμα(platform-specific) στιγμιότυπα των ποδοσφαιριστών

Είναι φανερό ότι οι διεπαφές παρουσιάζουν ριζικές διαφορές σε όλα τα επίπεδα όσον αφορά τα χαρακτηριστικά τους (σημασιολογικό, φυσικό, λεκτικό). Π.χ. στην περίπτωση του τυφλού χρήστη (μεσαία αναπαράσταση), πρέπει να επιστρατευτούν ειδικά μέσα και τεχνικές για

την απόδοση της χωρικής διάταξης της πληροφορίας ενδιαφέροντος εντός του γηπέδου, στον τυφλό χρήστη. Γι' το λόγο αυτό κάθε παίχτης φέρει στην υλοποίησή του ειδικά χαρακτηριστικά τα οποία υποβοηθούν τη λειτουργία του radar. Συγκεκριμένα, υλοποιήθηκαν τρεις ομόκεντροι δίσκοι, με κάθε ένα να τον διατρέχει μια ξεχωριστή ακτίνα.

Κάθε φορά που η ακτίνα προσκρούει σε ένα άλλο παίχτη ή γραμμή εκπέμπει ένα συγκεκριμένο ήχο που αποδίδει το ανάλογο γεγονός άγοντας στον τυφλό παίχτη τη σχετική πληροφορία. Αυτή του πιο κοντά στον παίχτη ακτίνα, περιστρέφεται με μεγαλύτερη από τις άλλες δύο ταχύτητα μιας και είναι αμεσότερου ενδιαφέροντος η αντίληψη αλλαγών στον κοντινότερο χώρο. Με την ίδια λογική η ενδιάμεση ακτίνα κινείται αργότερα από την εσωτερική και γρηγορότερα από την περιφερειακή. Προφανώς τέτοιου είδους λεπτομέρειες δεν απασχολούν τον κώδικα των γραφικών αναπαραστάσεων μιας και εκεί η σχετική πληροφορία είναι αυθύπαρκτη – δεδομένη διαμέσου του καναλιού της όρασης. Διαφορές ωστόσο υπάρχουν μεταξύ των άλλων δύο υλοποιήσεων, π.χ. στην περίπτωση του android το γήπεδο μπορεί να επιδεχθεί δυνατότητας απεικόνισης της χειρονομίας του παίχτη ώστε να απεικονίζεται η τροχιά που έχει σχεδιαστεί να κινηθεί ο παίχτης διαμέσου της χειρονομίας του χρήστη, διευκολύνοντας πολύ το χειρισμό. Τέτοιου είδους λεπτομέρειες δεν είναι αρκετό μόνο να υποστηρίζονται σε επίπεδο υλοποίησης αλλά πρέπει να επιτρέπεται ο ρητός προσδιορισμός του κατά τη φάση της σχεδίασης, ειδάλλως παραμένει ανεκμετάλλευτη κάθε μοναδική δυνατότητα (πολυμορφική) που ανά περίπτωση υποστηρίζουν και με αυτό τον τρόπο χάνεται.

Πολυμορφική Κατηγοριοποίηση Αλληλεπιδραστικών Σχημάτων

Τα κύρια αλληλεπιδραστικά στοιχεία που υλοποιούν κάθε εναλλακτική υλοποίηση είναι τρία, αυτά 1) του γηπέδου (SoccerField), 2) της μπάλας (SoccerBall) και 3) των παιχτών (SoccerPlayer). Και για τις τρεις περιπτώσεις αναπτύχθηκαν και οι υπάρχουν υλοποιήσεις (libraries), των οποίων τα χαρακτηριστικά και οι δυνατότητες κωδικοποιούνται στον Πίνακας 7, Πίνακας 8 και Πίνακας 9. Για κάθε μια από τις τρεις περιπτώσεις συντάχθηκε ένα στιγμιότυπου WSL που κωδικοποιεί τις ιδιότητες και το API τους αναλυτικά. Η Εικόνα 48, δείχνει ένα απόσπασμα του WSL που συντάχθηκε στα πλαίσια του 'SoccerField'. Παρατηρούνται τα τρία διαφορετικά στοιχεία τύπου 'instance' στα πλαίσια των οποίων κωδικοποιείται το σύνολο των implementation-specific σχημάτων. Χωρίς καμία αναφορά σε στιγμιότυπο είναι οι αφηρημένες ιδιότητες που είναι κοινές σε κάθε περίπτωση (πλάτος, ύψος γηπέδου, κοκ.).

			Property Name	Property Type
Abstract ' Soccer Field'	Abstract Properties		width	int
			height	int
			team1Name	String
			team2Name	String
	Polymorphic Properties	SoccerField (*java/swing*)	fieldMiniViewEnabled	boolean
			socialAwarenessEnabled	boolean
			team1Color	String
			team2Color	String
		SoccerField (*Android*)	visualizeGestureEnabled	boolean
			team1Color	String
			team2Color	String
			vibrateOnTap	boolean
		Non-visual SoccerField (*JNVT2*)	voiceType	int
			middleRadarEnabled	boolean
			outerRadarEnabled	boolean
			enableStaminaAudioCues	boolean

Πίνακας 7: Πολυμορφική κατηγοριοποίηση ιδιοτήτων του widgetArchive 'SoccerField

			Property Name	Property Type
Abstract ' Soccer Player'	Abstract Properties		id	int
			name	String
			teamId	int {1, 2}
			type	String {GK, DEF, MID, STR }
			speed	int
			xPosition	int
			yPosition	int
			stamina	int
	Polymorphic Properties	Soccer Player (*java/swing*)	shootPowerIndicationColor	String
		Soccer Player (*Android*)	shootPowerIndicationColor	String
		Non-visual Player (*JNVT2*)	-	-

Πίνακας 8: Πολυμορφική κατηγοριοποίηση ιδιοτήτων του widgetArchive 'SoccerPlayer'.

			Property Name	Property Type
Abstract ' Ball'			xPosition	int
			yPosition	int
			velocity	double
	Polymorphic Properties	Soccer Player (*java/swing*)	color	String
		Soccer Player (*Android*)	color	String
		Non-visual Player (*JNVT2*)	enableSpecialAudioCues	boolean

Πίνακας 9: Πολυμορφική κατηγοριοποίηση ιδιοτήτων του widgetArchive 'SoccerBall'

WSL

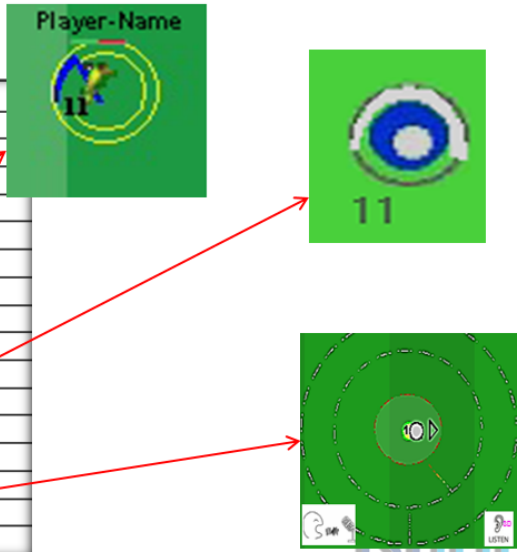
```
<widget id="abstract_soccer_field" name="abstractSoccerField" type="domainSpecificComponent">
    ...
    <availability>...</availability>
    <abstract_properties>
                <property id="abs_prop_1" name="width" ... />
                <property id="abs_prop_2" name="height" ... />
                <property id="abs_prop_3" name="team1Name" ... />
                <property id="abs_prop_4" name="team2Name" ... />
        </abstract_properties>
    <instances>
        <instance id="abstract_soccer_field_inst_1" name="SwingSoccerField" isTopLevel="false" isNative="false" ... >
            <display_name>Swing Soccer Field</display_name>
            <dependencies>...</dependencies>
            <polymorphic_properties>...</polymorphic_properties>
            <supported_behaviors>...</supported_behaviors>
            <mappings>...</mappings>
            <instance_api>...</instance_api>
        </instance>
        <instance id="abstract_soccer_field_inst_2" name="AndroidSoccerField" isTopLevel="false" isNative="false" ... >
            <display_name>Android Soccer Field</display_name>
            ...
        </instance>
        <instance id="abstract_soccer_field_inst_3" name="NVTSoccerField" isTopLevel="false" isNative="false" ... >
            <display_name>NVT Soccer Field</display_name>
            <dependencies>...</dependencies>
            <polymorphic_properties>
                <property id="inst_3_prop_1" name="voiceType" type="java.lang.Integer" isPrimitive="true" .../>
                <property id="inst_3_prop_2" name="middleRadarEnabled" type="java.lang.Boolean" isPrimitive="true" .../>
                <property id="inst_3_prop_3" name="outerRadarEnabled" type="java.lang.Boolean" isPrimitive="true" .../>
            </polymorphic_properties>
            ...
            <mappings>...</mappings>
            <instance_api>...</instance_api>
        </instance>
    </instances>
</widget>
```

Εικόνα 48: Τμήμα WSL 'abstractSoccerField'

Η Εικόνα 49 δείχνει πως οι ιδιότητες αντιστοιχίζονται στην πράξη με συγκεκριμένα φυσικά χαρακτηριστικά.

popular Object-Oriented languages.

			Property Name	Property Type
Abstract 'Soccer Field'	Abstract Properties		width	int
			height	int
			team1Name	String
			team2Name	String
	Polymorphic Properties	SoccerField (*java/swing*)	fieldMiniViewEnabled	boolean
			socialAwarenessEnabled	boolean
			team1Color	String
			team2Color	String
		SoccerField (*Android*)	visualizeGestureEnabled	boolean
			team1Color	String
			team2Color	String
			vibrateOnTap	boolean
		Non-visual SoccerField (*JNVT2*)	voiceType	int
			middleRadarEnabled	boolean
			outerRadarEnabled	boolean
			enableStaminaAudioCues	boolean

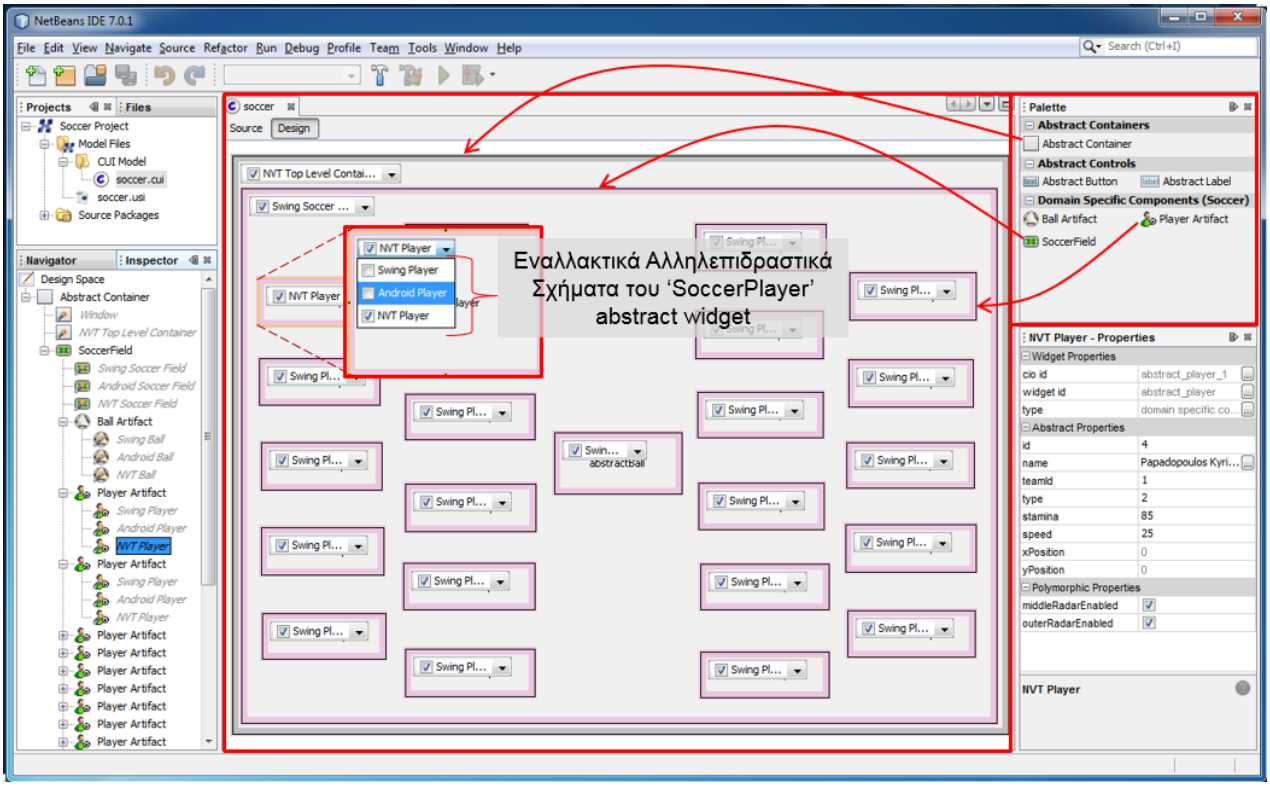


Εικόνα 49: Πολυμορφικές διαφοροποιήσεις εναλλακτικών σχημάτων 'SoccerField'

Για κάθε ένα από τα αντικείμενα τύπου γηπέδου, παίχτη και μπάλας δημιουργούνται τρία ξεχωριστά widgetArchives, συμπεριλαμβάνοντας το ανάλογο στιγμιότυπο WSL και τις σχετικές βιβλιοθήκες της υλοποίησης για κάθε περίπτωση εργαλειοθήκης. Τα widgetArchives εν συνεχεία εισάγονται μέσω της ενδεδειγμένης ροής στο IDE και γίνονται διαθέσιμα στη γλώσσα ως νέα διαδραστικά στοιχεία.

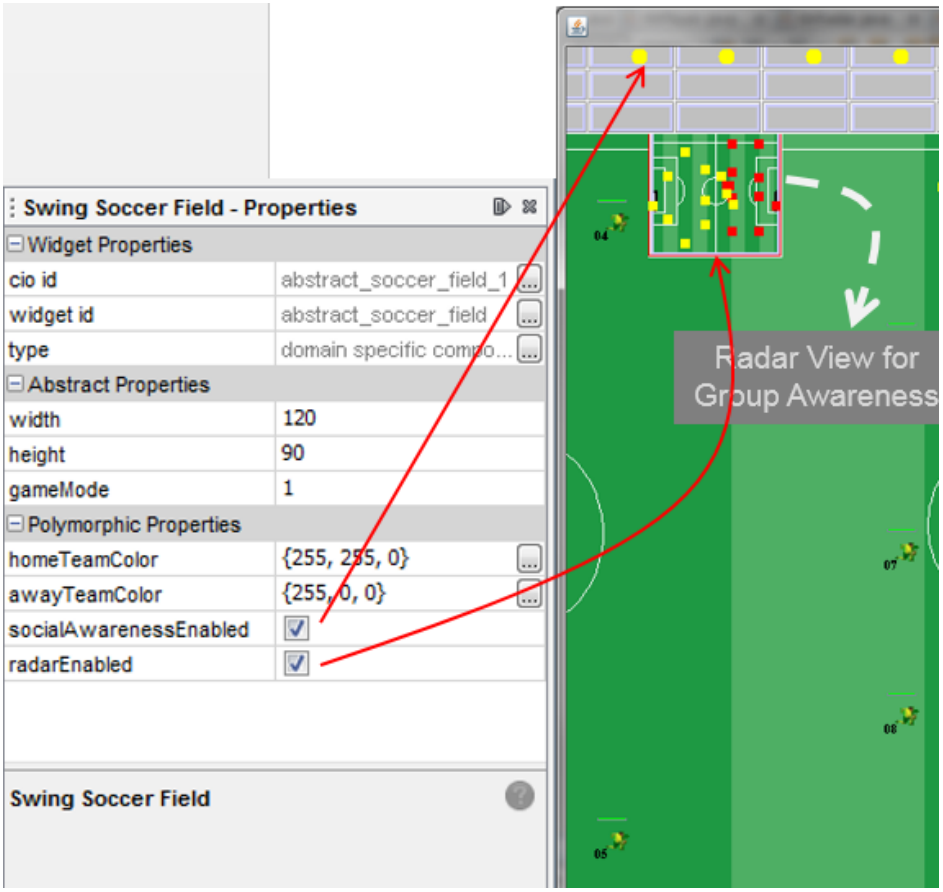
Σχεδίαση του σεναρίου με τη βοήθεια του οικείου IDE

Το στιγμιότυπο της Εικόνα 50 απεικονίζει τη στιγμή που ο σχεδιαστής έχει ολοκληρώσει τον αλληλεπιδραστικό προσδιορισμό της διεπαφής. Αυτό σημαίνει ότι έχει ολοκληρώσει όχι μόνο την εισαγωγή των αφηρημένων αντικειμένων και την επιλογή των κατάλληλων ενσαρκώσεων αλλά έχει προσδιορίσει και τα ιδιαίτερα χαρακτηριστικά κάθε πολυμορφικού σχήματος που επιθυμεί στα πλαίσια του σεναρίου να είναι ενεργά. Στην Εικόνα 50, π.χ. φαίνεται πως έχουν φορτωθεί από το σχετικό WSL όλα τα διαθέσιμα σχήματα, και όλες οι ιδιότητες του επιλεγμένου σχήματος (Non-visual) στα πλαίσια ενός παίχτη και επιτρέπεται ως ρητός προσδιορισμός τους (εν προκειμένω ενεργοποιήθηκαν και τα δύο εξωτερικά radar).



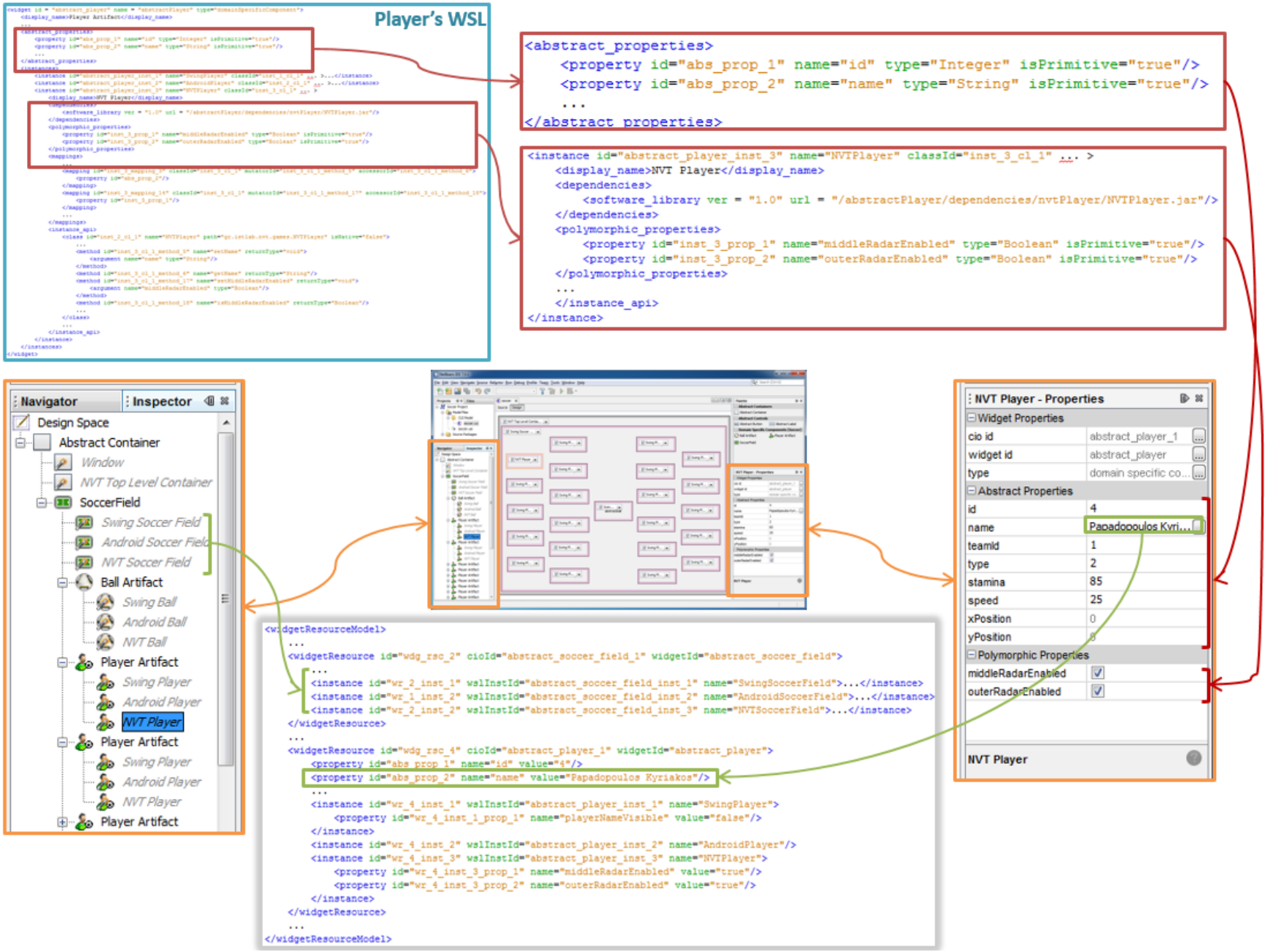
Εικόνα 50: Στιγμιότυπο του εργαλείου πολυμορφικής σχεδίασης για το παιχνίδι 'Ποδοσφαιράκι'

Στην ίδια εικόνα, όπως φαίνεται, δεδομένου τα εν λόγω χαρακτηριστικά (non-visual radars) είναι τύπου Boolean, ο PropertiesEditor υποβοήθησε αυτόματα τον προσδιορισμό του εύρους τιμών τους διαμέσου ενός checkbox. Στην Εικόνα 51, φαίνεται πως έχουν ενεργοποιηθεί ιδιότητες μοναδικές ως προς την υλοποίηση του διαδραστικού σχήματος το οποίο έχει υλοποιηθεί ως επέκταση της εργαλειοθήκης java/swing.



Εικόνα 51: Ενεργοποίηση radar γηπέδου στην περίπτωση του σχήματος γηπέδου της υλοποίησης του java/Swing

Στην Εικόνα 52 φαίνεται πως ο αλληλεπιδραστικός προσδιορισμός των ιδιοτήτων των αντικειμένων της διεπαφής, κωδικοποιείται στα πλαίσια του μοντέλου ‘widgetResource’. Όπως φαίνεται, το πλήθος των ετικετών ‘instances’ καταμαρτυρά το πλήθος των ενεργοποιημένων αλληλεπιδραστικών σχημάτων (εν προκειμένω τρία), ενώ στη δεξιά πλευρά φαίνεται πως αποθηκεύεται η τιμή μιας αφηρημένης ιδιότητας. Το πώς κάθε τιμή αντιστοιχίζεται εν συνεχεία με την ανάλογη κλήση στο API της σχετικής υπολογιστικής υλοποίησης συνάγεται διαμέσου των ενδεδειγμένων συσχετίσεων στο WSL.



Εικόνα 52: Υψηλού επιπέδου στιγμιότυπο συσχέτισης μεταξύ: WSL, CUI και WidgetResource μοντέλων, στα πλαίσια υποστήριξης του σεναρίου του παιγνίου ποδοσφαιράκι

Εν εκτελέσει Στιγμιότυπα σεναρίου Χρήσης

Στην Εικόνα 53 αναπαρίσταται ένα εν εκτελέσει στιγμιότυπο των πολυμορφικών διεπαφών για το ‘ποδοσφαιράκι’. Θα πρέπει να σημειωθεί ότι και το εν λόγω σενάριο αξιοποίησε περιβάλλον εκτέλεσης (run-time system) το οποίο υλοποιήθηκε (από άλλη μεταπτυχιακή εργασία) έτσι ώστε να διαλειτουργεί με τις προδιαγραφές WSL[26]. Τέλος, το συγκεκριμένο σενάριο παρουσιάζεται σε πραγματικές συνθήκες στο σύνδεσμο(https://www.youtube.com/watch?v=ApPWc1d03V4.

(α)

(β)

(γ)

Εικόνα 53: Εναλλακτικά πολυμορφικά(context-aware) στιγμιότυπα για το παιχνίδι ‘Ποδοσφαιράκι’, α: Non-visual toolkit components, β: android custom components, γ: java/swing custom components

Κεφάλαιο 6 – Επίλογος

Η μεταπτυχιακή εργασία εστίασε σε ένα βασικό ζήτημα της μοντελοκεντρικής προσέγγισης για την κατασκευή διεπαφών. Συγκεκριμένα, μελέτησε την ανάπτυξη γλωσσών προδιαγραφών ικανές να υποστηρίξουν σύνθετα σενάρια χρήσης και απαιτήσεις που δεν εξυπηρετούνται επαρκώς από τεχνικές της τρέχουσας τεχνολογικής στάθμισης. Η πολυπλοκότητα του θέματος των προδιαγραφών διεπαφών προκύπτει αφενός από τις συνεχώς αυξανόμενες αλλαγές που συντελούνται σε επίπεδο συσκευών και περιβαλλόντων χρήσης και αφετέρου από την ολοένα και μεγαλύτερη προσφορά βιβλιοθηκών εξειδικευμένων βιβλιοθηκών και εφαρμογών από την ευρύτερη κοινότητα των κατασκευαστών λογισμικού. Είναι λοιπόν προφανής η ανάγκη αξιοποίησης της συλλογικής εμπειρίας και διαθέσιμων πόρων έτσι ώστε η σχεδιαστική κοινότητα να εστιάζει σε κατεξοχήν σχεδιαστικά καθήκοντα, αντί να περιορίζεται από συμβάσεις και περιορισμούς προερχόμενους από τις ιδιαιτερότητες της εκάστοτε πλατφόρμας και περιβάλλοντος χρήσης.

Σύνοψη και συνεισφορά

Προς αυτή την κατεύθυνση, η εργασία προτείνει μια νέα γλώσσα WSL και ένα σχεδιαστικό περιβάλλον που επιτρέπει την αξιοποίηση των παραγόμενων προδιαγραφών από ένα ειδικού σκοπού περιβάλλον εκτέλεσης πραγματικού χρόνου (runtime environment). Η ίδια η γλώσσα WSL σηματοδοτεί σημαντικές επεκτάσεις και βελτιώσεις στις καθιερωμένες δομές της UsiXML αυξάνοντας ουσιαστικά τη εκφραστική της δύναμη όσο αφορά το είδος και το εύρος των διεπαφών που προδιαγράφονται. Το σχεδιαστικό περιβάλλον αποτελεί καινοτόμο σύστημα που βασίζεται στη WSL και ενσαρκώνει κατάλληλους μηχανισμούς και μοντέλα παραγωγής επίσημων προδιαγραφών των διεπαφών. Ιδιαίτερο χαρακτηριστικό και των δύο επιτευγμάτων είναι η δέσμευσή τους στην υλοποίηση πολυμορφικών σχημάτων που καθοδηγούν κατάλληλους μετασχηματισμούς στα μοντέλα της γλώσσας έτσι ώστε να επιτυγχάνονται σύνθετες διαδραστικές εμπειρίες. Τέλος, για την επίδειξη της επάρκειας της γλώσσας και του σχεδιαστικού περιβάλλοντος επιλέχθηκαν ενδεικτικά σενάρια τα οποία στο σύνολο τους δεν υποστηρίζονται από μοντελο-κεντρικές γλώσσες προδιαγραφών ή/και συστήματα. Τα σενάρια αυτά επιλέχθηκαν έτσι ώστε να καλύπτονται ενδεικτικά ένα εύρος τεχνολογιών όπως Java/Swing, Android και HTML5. Με τον τρόπο αυτό αποδεικνύεται αφενός η εκφραστική

δύναμη της γλώσσας WSL στο να καλύψει διαφορετικές πλατφόρμες και γενιές τεχνολογικών εφαρμογών, ενώ αφετέρου σηματοδοτούνται σειρά επεκτάσεων σε σχεδιαστικά περιβάλλοντα και εργαλεία της μοντελο-κεντρικής θεώρησης στην κατασκευή διεπαφών.

Περιορισμοί, Επεκτάσεις και Μελλοντική εργασία

Κάθε ερευνητική προσπάθεια περιορίζεται από παράγοντες, αντικειμενικούς ή/και υποκειμενικούς, που ταυτόχρονα συνιστούν και πιθανές επεκτάσεις της ερευνητικής ατζέντας και μελλοντική εργασία. Υπό αυτό πρίσμα, η παρούσα έρευνα δεν αποτελεί εξαίρεση. Αναγνωρίζονται περιορισμοί που σχετίζονται με (α) την εξοικείωση των σχεδιαστών με τις προτεινόμενες τεχνικές αλλά και (β) τις απαιτήσεις που τίθενται για την απρόσκοπτη εκτέλεση των παραγόμενων προδιαγραφών σε περιβάλλοντα και πλατφόρμες.

Το θέμα της εξοικείωσης των σχεδιαστών με μοντέλα και επίσημες γλώσσες προδιαγραφών δεν είναι καινούργιο, ιδιαίτερα για τους ερευνητές της μοντελο-κεντρικής θεώρησης των διεπαφών. Αφορά κυρίως τη σημειογραφία των τεχνικών (notations) και τη δυνατότητα ελέγχου αυτών που προδιαγράφονται με τη χρήση κατάλληλων εργαλείων. Δεδομένου ότι η γλώσσα WSL που παρουσιάσαμε αποτελεί επίσημη επέκταση της UsiXML και ακολουθεί τις πλέον αποδεκτές πρακτικές που υιοθετούνται από την κοινότητα της UsiXML είναι λογικό να γίνει δεκτή χωρίς ιδιαίτερο προβληματισμό από εμπειρογνώμονες του κλάδου. Ωστόσο, πρέπει να τονιστεί ότι η WSL δεν αποτελεί εργαλείο γενικού τύπου ή γλώσσα που αναμένεται να υιοθετηθεί για κάθε κατηγορία σχεδιαστικού προβλήματος, ιδιαίτερα αν αυτό δεν εμπεριέχει εναλλακτικούς ρόλους ποικιλία συσκευών και διαφορετικές πλατφόρμες.

Όσο αφορά τις απαιτήσεις απρόσκοπτης εκτέλεσης προδιαγραφών από ευρέως διαθέσιμες πλατφόρμες θα πρέπει να αναγνωρίσουμε τις ιδιαιτερότητες του προς επίλυση ζητήματος (δηλ. τη διαχείριση προδιαγραφών σύνθετων διεπαφών) που εξ' ορισμού εγείρει περιορισμούς. Με άλλα λόγια, για τις ανάγκες της παρούσας εργασίας αναπτύχθηκαν και υλοποιήθηκαν όλες οι επαυξήσεις, επεκτάσεις και ενσωματώσεις διαδραστικών βιβλιοθηκών που απατούνται για να προδιαγραφούν και να εκτελεστούν τα σενάρια που περιγράφτηκαν. Ωστόσο, γίνεται αντιληπτό ότι παρόμοιες προσπάθειες από άλλους μηχανικούς θα μπορούσαν να οδηγήσουν σε διαφορετικά διαδραστικά αντικείμενα, διαφορετικές τεχνικές και εναλλακτικές υλοποιήσεις. Για να μπορέσουν αυτές να αξιοποιηθούν από το περιβάλλον εκτέλεσης της WSL θα πρέπει να ανταποκρίνονται στις ειδικές απαιτήσεις της WSL, να είναι πλήρης ως προς το

εύρος των πολυμορφικών χαρακτηριστικών και να έχουν υλοποιηθεί σε μία από τις πλατφόρμες που παρουσιάσαμε.

Αναφορές

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] Garlan, S.-W. Cheng, A.-C. Huang, B. Schmerl, and P. Steenkiste. 2004. Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure. Computer 37(10), 46–54.

[7] 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.

[8] D. Akoumianakis, G. Milolidakis, D. Kotsalis, G. Vellis, Interaction platform administration strategies - Practice and experience, Proc. 11th Panhellenic Conference on Informatics, Patras, New Technologies Publications, 2007, 18--23.

[9] Heiko Paulheim. 2010. Seamlessly integrated, but loosely coupled: building user interfaces from heterogeneous components. In *Proceedings of the IEEE/ACM international conference on Automated software engineering* (ASE '10). ACM, New York, NY, USA, 123-126.

[10] Paulheim, H. (2013). Ontology-based Application Integration on the User Interface Level. it - Information Technology Methoden und innovative Anwendungen der

Informatik und Informationstechnik, 55(4), pp. 165-168. Retrieved 23 Oct. 2015, from doi:10.1524/itit.2013.1011

[11] Apitz G, Guimbretiere F., Cross, Y. (2004) A crossing-based drawing application. ACM Symposium on User Interface Software and Technology (UIST ’04), (pp. 3–12), ACM Press: New York, U.S.A.

[12] Appert C, Fekete JD. (2006) OrthoZoom scroller: 1D multi-scale navigation. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI ’06), (pp. 21–30), ACM Press: New York, U.S.A.

[13] Willett, W., Heer, J., & Agrawala, M. (2007) Scented Widgets: Improving Navigation Cues with Embedded Visualizations, IEEE Information Visualization (InfoVis), 13 (6), 1129-1136.

[14] Bederson, B.B., Meyer, J., & Good, L. (2000) Jazz: An Extensible Zoomable User Interface Graphics Toolkit in Java. ACM Symposium on User Interface Software and Technology (UIST’00), (pp. 171-180), ACM Press: New York, U.S.A.

[15] Bederson, B. B., Grosjean, J., & Meyer, J. (2004) Toolkit Design for Interactive Structured Graphics, IEEE Transactions on Software Engineering, 30 (8), 535-546.

[16] Heer, J., Card, S., Landay, J. (2005) prefuse: a toolkit for interactive information visualization, ACM Conference on Human Factors in Computing Systems (CHI ’05), (pp. 421 - 430), ACM Press: New York, U.S.A.

[17] Vanderdonckt, J., A MDA-Compliant Environment for Developing User Interfaces of InformationSystems, Proc. of 17th Conf. on Advanced Information Systems Engineering CAiSE'05 (Porto,13-17 June 2005), O. Pastor & J. Falcao e Cunha (eds.), Lecture Notes in Computer Science, Vol. 3520, Springer-Verlag, Berlin, 2005, pp. 16-31. Conference keynote address.

[18] Ακουμιανάκης, Δ. (2006): Διεπαφή Χρήστη-Υπολογιστή – Μια σύγχρονη προσέγγιση, Αθήνα, Εκδόσεις ΚΛΕΙΔΑΡΙΘΜΟΣ (ISBN 960-209-975-5.

[19] Schaefer, R., Steffen, B., Wolfgang, M., Task Models and Diagrams for User Interface Design, Proceedings of 5 International Workshop, TAMODIA'2006 (Hasselt, Belgium, October 2006), Lecture Notes in Computer Science, Vol. 4385, Springer Verlag Berlin, 2006, pp. 39-53.

[20] Mori, F. Paterno, and C. Santoro. Design and development of multidevice user interfaces through multiple logical descriptions. IEEE Transactions on Software Engineering, 30(8):507–520, 8 2004.

[21] Fabio Paterno', Carmen Santoro, and Lucio Davide Spano. 2009. MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments.*ACM Trans. Comput.-Hum. Interact.* 16, 4, Article 19 (November 2009), 30 pages.

[22] Vanderdonckt J., and Bodart F. (1993), Encapsulating knowledge for intelligent automatic interaction objects selection In Ashlund S., Mullet K., Henderson A., Hollnagel E., and White T. (Eds.), Proc. of the ACM Conference on Human Factors in Computing Systems InterCHI'93 (Amsterdam, 24-29 April 1993), ACM Press pages, New York, 1993, pp. 424-429.

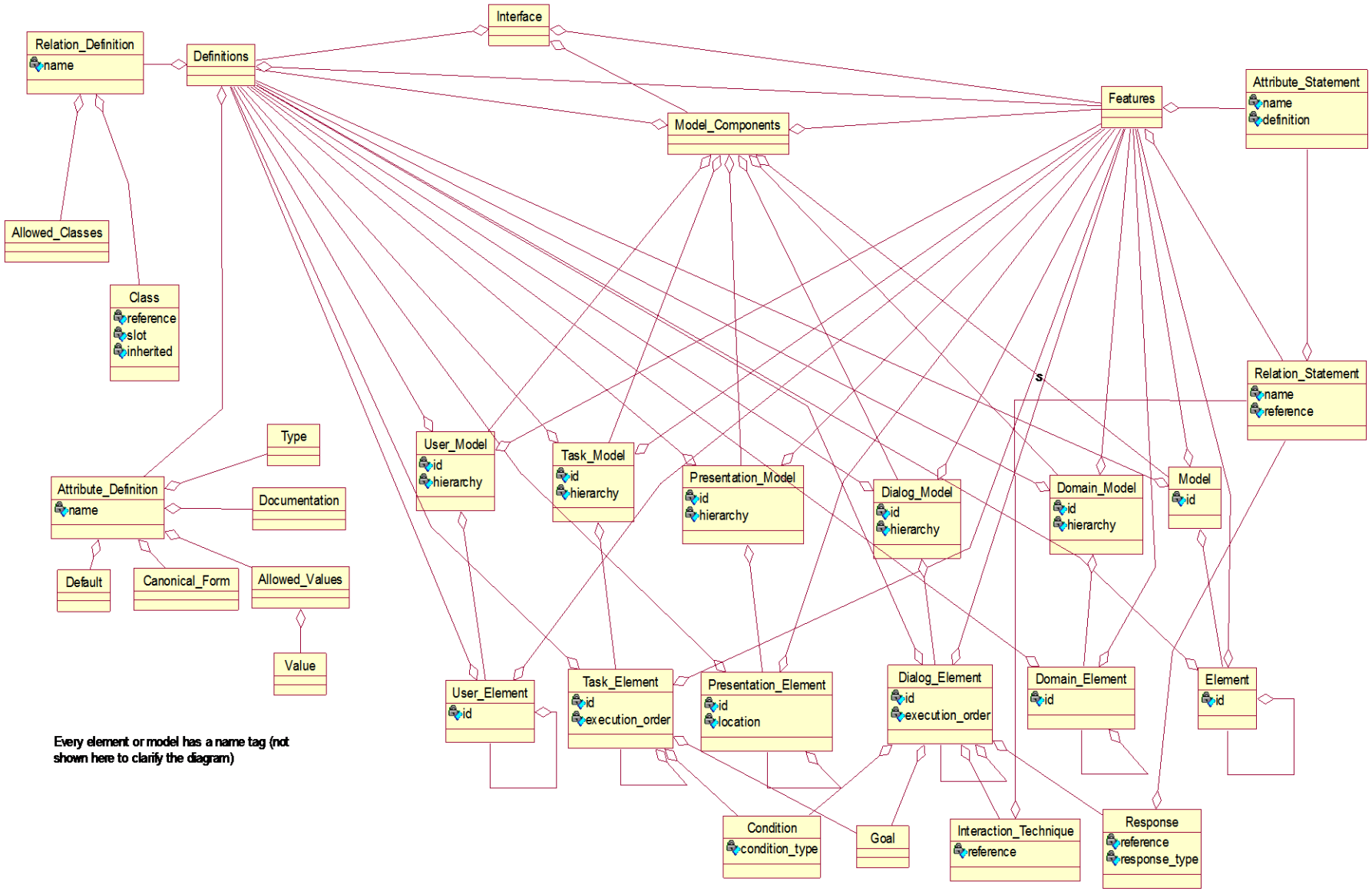
[23] A. Puerta, A., Eisenstein, J., XIML: A Common Representation for Interaction Data, Proceedings of 6th International Conference on Intelligent User Interfaces IUI'2002 (San Francisco, USA, January 13-16, 2002,), ACM Press, pp. 214-215

[24] M. Abrams, C. Phanouriou, A. L. Batongbacal, S. M. Williams, and J. E. Shuster. 1999. UIML: An Appliance-Independent XML User Interface Language. Computer Networks 31(11–16), 1695–1708.

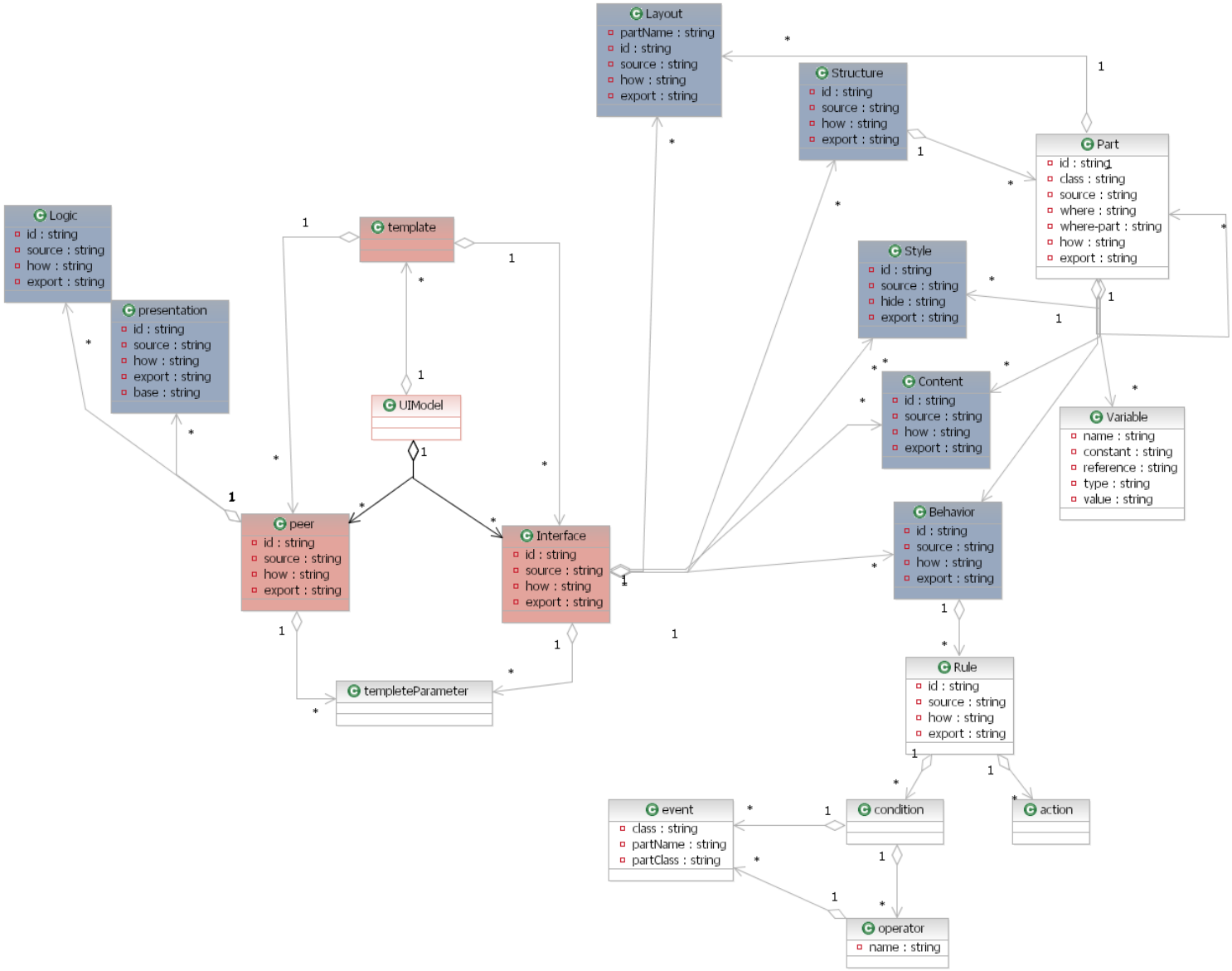
[25] Akoumianakis D., Vidakis N., Akrivos A., Milolidakis G., Kotsalis D., Vellis G. (2011): Building 'Flexible' vacation packages using collaborative assembly toolkits and dynamic packaging: The Case Study of the eKoNES, Journal of Vacation Marketing, Vol 17 Issue 1, pp 17-30.

[26] Vellis, G.: "Model –Based Development of Ubiquitous Synchronous Collaborative UIS". Master's thesis, Technological Education Institution of Crete, Hellas, 2015.

Παράρτημα



Εικόνα 54: Μεταμοντέλο XIML



Εικόνα 55: Μετα-μοντέλο UIML