Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή Εργασία

Μοντελοκεντρική ανάπτυξη διεπαφών σύγχρονης συνεργασίας: Η περίπτωση χρήσης του παιγνίου Soccer στο web

Προβιδάκης Αντώνιος
Α.Μ 3318

Επιβλέπων καθηγητής : Δημοσθένης Ακουμιανάκης

ΗΡΑΚΛΕΙΟ

2016

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την οικογένεια μου για την υποστήριξη που μου παρείχε κατά την διάρκεια των σπουδών μου. Επίσης, θα ήθελα να ευχαριστήσω τον Δρ. Ακουμιανάκη Δημοσθένη και τον Δρ. Βιδάκη Νίκο για την δυνατότητα που μου έδωσαν ώστε να εκπονήσω την πτυχιακή εργασία και την πρακτική άσκηση μου στο iSTLab. Τέλος, ευχαριστώ τους Βελλή Γιώργο και Κότσαλη Δημητρη για την πολύτιμη βοήθεια τους, γιατί χωρίς αυτήν δεν θα είχε ολοκληρωθεί η παρούσα εργασία, καθώς και τον Μήτσο για την υπομονή που δείχνει όλα τα χρόνια της συνεργασίας μας.

Abstract

The goal of this thesis is to assess and exploit the benefits of model-based user interface engineering for the development of advanced web-based interaction scenarios. Specifically, it examines the state of the art in the area of model-based user interface engineering and makes use of existing models and methods (e.g. UsiXML) to step towards interaction architectures that enable smooth and seamless collaboration between distributed users. As use case, the online game of Soccer will be used as it presents special requirements that extend beyond those of traditional GUI applications. The resulting implementation rests on UsiXML, a popular model-based user interface description language, and presents a dedicated web platform server for handling the intrinsic details of the actual game.

Σύνοψη

Στόχος της παρούσας πτυχιακής εργασίας είναι η αξιοποίηση των πλεονεκτημάτων που απορρέουν από την υιοθέτηση της μοντελοκεντρικής μηχανικής διεπαφών για την ενοποιημένη ανάπτυξη πανταχού παρόντων διεπαφών. Πιο συγκεκριμένα, στην αναφορά μελετάμε την τρέχουσα τεχνολογική στάθμιση στη μοντελοκεντρική μηχανική διεπαφών και εξειδικεύονται στην πράξη υπάρχοντα μοντέλα (π.χ. UsiXML) υπό το πρίσμα της υλοποίησης αρχιτεκτονικών διεπαφής χρήστη-υπολογιστή που επιτρέπουν την ομαλή και απρόσκοπτη συνεργασίας μεταξύ χωρικά διεσπαρμένων χρηστών. Ως μελέτη περίπτωσης θα χρησιμοποιηθεί το διαδικτυακό παίγνιο ποδοσφαίρου εξαιτίας των ιδιαίτερων απαιτήσεων που θέτει η υποστήριξή του. Για τις ανάγκες της πιλοτικής ανάπτυξης θα χρησιμοποιηθεί μια πολύ δημοφιλής μοντελο- κεντρική γλώσσα, η UsiXML.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

Εισαγωγή

1.1 Συνεργατική εκτέλεση καθηκόντων και διεπαφές χρήστη-υπολογιστή

Το πλήθος και η ποικιλία υπολογιστικών διατάξεων που απαντώνται σήμερα, σε συνδυασμό με τους εκθετικούς ρυθμούς της διείσδυσης και υιοθέτησής τους στα πλαίσια της εκτέλεσης σειράς καθηκόντων από πλευράς των χρηστών, αναδεικνύει νέες δυνατότητες αλλά και προκλήσεις. Δυνατότητες δεδομένου ότι οι χρήστες αξιοποιούν πλέον ένα ιδιαίτερα μεγάλο εύρος υπολογιστικών συσκευών και πλατφορμών το οποίο εξασφαλίζει αυξημένη ευελιξία, υπό την έννοια του πλουραλισμού επιλογών, αλλά και δυνατοτήτων. Από την άλλη ωστόσο κατακερματίζει την ομοιογένεια στα πλαίσια της οποίας καλούνται να λειτουργήσουν οι εφαρμογές, καθιστώντας τη χρήση τους κατά τη μετάβαση από το ένα πλαίσιο στο άλλο αν μη τι άλλο προβληματική.

Σε αυτό το αναδυόμενο υπολογιστικό περιβάλλον (πανταχού παρών), ιδιαίτερο ενδιαφέρον παρουσιάζει η ανάπτυξη σύγχρονου συνεργατικού λογισμικού. Στόχος αυτής της κατηγορίας του λογισμικού είναι η υποστήριξη της από κοινού εκτέλεσης καθηκόντων μεταξύ χρηστών που είναι φυσικά διεσπαρμένοι και μπορεί να αξιοποιούν διαφορετικές τερματικές συσκευές. Ένα εξειδικευμένο συστατικό λογισμικού που απαιτεί ιδιαίτερη προσοχή σε αυτό το πλαίσιο είναι η διεπαφή χρήστη-υπολογιστή. Η ανάπτυξη του λογισμικού μιας διεπαφής συνιστά εκ φύσεως απαιτητική διαδικασία μιας και εμπλέκει σειρά απαιτήσεων που έχουν να κάνουν με εύρος ζητημάτων όπως επιλογή εργαλειοθηκών διαδραστικών αντικειμένων, διασύνδεση με λειτουργικό κορμό, οργάνωσης μονάδων εισόδου / εξόδου, ευχρηστία, κλπ. Στη περίπτωση σύγχρονου συνεργατικού λογισμικού τα ζητήματα αυτά επεκτείνονται και περιλαμβάνουν έλεγχο ταυτοχρονισμού, διαμοιρασμό πληροφορίας, κοκ με αποτέλεσμα η ανάπτυξη τέτοιων διεπαφών να καθίσταται πολύπλοκη. Αυτό συμβαίνει διότι οι διαθέσιμες μέθοδοι μηχανικής διεπαφών, λόγω συγκεκριμένων υποθέσεων που κάνουν, επιβάλλουν περιορισμούς φορητότητας, συμβατότητας και εκτελεσιμότητας κατά μήκος ριζικά διαφορετικών πλαισίων χρήσης. Παραδείγματος χάριν, μια συνεργατική 2D εργαλειοθήκη που προορίζεται για χρήση σε περιβάλλον σταθερού υπολογιστή δεν διαθέτει την ευελιξία να προσαρμοστεί και να εκτελεστεί σε περιβάλλοντα κινητής συσκευής ή ενός browser και το αντίστροφο, κοκ..

Η μοντελο-κεντρική μηχανική διεπαφών [19] από την άλλη πλευρά προσφέρει ένα καλύτερο πλαίσιο για τη διαχείριση της ετερογένειας. Τα βασικά πλεονεκτήματα απορρέουν από την εστίαση στη χρήση αφηρημένης σημειολογίας και γλωσσών σήμανσης για την υποστήριξη του προσδιορισμού αφηρημένων αντικειμένων και της μετέπειτα διασύνδεσής τους με διαδραστικά λεξικά (interaction vocabularies). Συνηθίζεται ωστόσο τα λεξικά αυτά να θεωρούνται πρότυπα και οι παραγόμενες προδιαγραφές να είναι τέτοιες που να αξιοποιούν μόνο τα εγγενώς υποστηριζόμενα διαδραστικά αντικείμενα (native widgets) – όπως κουμπιά, ετικέτες, μενού επιλογής, κλπ., που είναι κοινά στις δημοφιλής εργαλειοθήκες. Ως εκ τούτου η εκφραστική τους επάρκεια περιορίζεται σε απλές διεπαφές τύπου διαλόγου ή form filling.

1.2 Στόχος πτυχιακής εργασίας

Η πτυχιακή εργασία εντάσσεται σε μια ευρύτερη δραστηριότητα που αποσκοπεί στον εμπλουτισμό, ενίσχυση και βελτίωση των δυνατοτήτων των μοντελο-κεντρικών μεθόδων μηχανικής διεπαφών έτσι ώστε να μπορούν να υποστηρίξουν σύνθετες διεπαφές που να προσαρμόζονται στις απαιτήσεις του περιβάλλοντος χρήσης και της πλατφόρμας / τερματικής συσκευής που αξιοποιεί ο χρήστης. Σ’ αυτό το πλαίσιο έχουν δρομολογηθεί προσπάθειες που γεφυρώνουν τα πλεονεκτήματα που απορρέουν από τη χρήση των επικρατέστερων μεθόδων ανάπτυξης [7][10][11][12].

Η παρούσα εργασία εδράζεται σε αυτές τις τεχνικές, αξιοποιεί βασικά τους επιτεύγματα και συμπληρώνει πτυχές αυτών που βρίσκονται σε εξέλιξη. Ως εκ τούτου κατά ένα τρόπο επικυρώνει επιμέρους θέματα και αποδεικνύει την εγκυρότητα και την επάρκεια τους για την ανάπτυξη πλήρως λειτουργικών συνεργατικών εφαρμογών. Ειδικότερα, η εργασία παρουσιάζει την υλοποίηση μιας περίπτωσης χρήσης του παιγνίου του ποδοσφαίρου – αυτής που αφορά την πλατφόρμα του web. Η πρόκληση συνίσταται στο γεγονός πως άκρως διαδραστικές εφαρμογές είναι εκ φύσεως πιο απαιτητικές απαιτώντας πολύ πιο εντατική επικοινωνία πληροφορίας μεταξύ των κατανεμημένων μερών. Ωστόσο, η προσέγγιση που πρόκειται να υιοθετηθεί, η οποία αποτελεί επέκταση της δημοφιλούς μοντελο-κεντρικής γλώσσας USiXML [20], μπορεί να καλύψει όλο το φάσμα εναλλακτικών πλατφορμών.

1.3 Οργάνωση της αναφοράς

Η αναφορά περιλαμβάνει πέντε κεφάλαια. Το κεφάλαιο 1 παρουσίασε συνοπτικά το ευρύτερο πεδίο και το στόχο της εργασίας. Το κεφάλαιο 2 παρουσιάζει την τρέχουσα τεχνολογική στάθμιση όσον αφορά σύγχρονες μεθόδους αμφίδρομης επικοινωνίας στο web, καθώς επίσης και τρόπους ανάπτυξης σύγχρονων συνεργατικών εφαρμογών συμπεριλαμβανομένων APIs, εργαλειοθηκών και μοντελο-κεντρικών προσεγγίσεων. Το κεφάλαιο 3 παρουσιάζει το σενάριο χρήσης και την προσέγγιση που υιοθετήθηκε για την διεκπεραίωση του σε περιβάλλον web. Το κεφάλαιο 4 παρουσιάζει την υλοποίηση του σεναρίου εξειδικεύει τα βασικά μοντέλα που αξιοποιήθηκαν. Τέλος, στο κεφάλαιο 5 παρουσιάζονται μια ποιοτική αξιολόγηση των επιδόσεων του συστήματος και οι μελλοντικές επεκτάσεις.

Τρέχουσα Τεχνολογική Στάθμιση – State of the Art

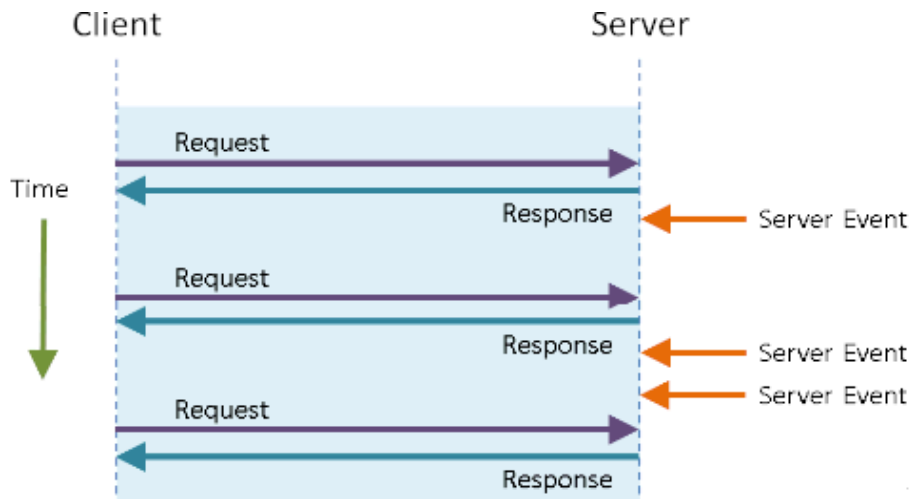
Κύριο ζήτημα και χαρακτηριστικό της ανάπτυξης σύγχρονων συνεργατικών εργαλειοθηκών και εφαρμογών στο web έχει να κάνει με τις μεθόδους υποστήριξης αμφίδρομης επικοινωνίας δεδομένων. Η προσέγγιση του ζητήματος δεν είναι το ίδιο σαφής και κατανοητή με αυτή των αντίστοιχων desktop-oriented, γεγονός που οφείλεται στη φύση της αρχιτεκτονικής του web. Υπό αυτή τη λογική, παρακάτω αναλύονται πρωτίστως οι κύριες τεχνικές-μέθοδοι για την υλοποίηση σύγχρονης επικοινωνίας δεδομένων. Ακολούθως αναλύονται πρώιμα ενδεικτικά APIs και εργαλειοθήκες για την ανάπτυξη σύγχρονων web-based groupware, καταλήγοντας με την περιγραφή σύγχρονων (modern) λύσεων συμπεριλαμβανομένων APIs και μεθόδων που εναπόκεινται στην υιοθέτηση της μοντελο-κεντρικής μηχανικής.

2.1 Σχήματα σύγχρονης επικοινωνίας πληροφορίας στο Web

Η υποστήριξη σύγχρονων μορφών επικοινωνίας πληροφορίας στα πλαίσια κατανεμημένων χωρικά κόμβων στη βάση της ευρύτερης πλατφόρμας του Web, συνιστά ιδιαίτερη πρόκληση σε αντιδιαστολή με τις αντίστοιχες desktop-oriented προσεγγίσεις. Κάτι τέτοιο είναι λογικό, φυσικό και επόμενο, δεδομένης της εγγενώς ασύγχρονης φύσης του web. Παρούσης ωστόσο της αναδυόμενης και ολοένα αυξανόμενης ανάγκης για υποστήριξη σύγχρονων συνεργατικών εφαρμογών στο web, επινοήθηκε σταδιακά πλήθος εναλλακτικών τακτικών αλλά και μετέπειτα ολοκληρωμένων τεχνικών λύσεων για την υποστήριξή τους. Αυτές παρουσιάζονται συνοπτικά στη συνέχεια.

2.1.1 Ajax Polling

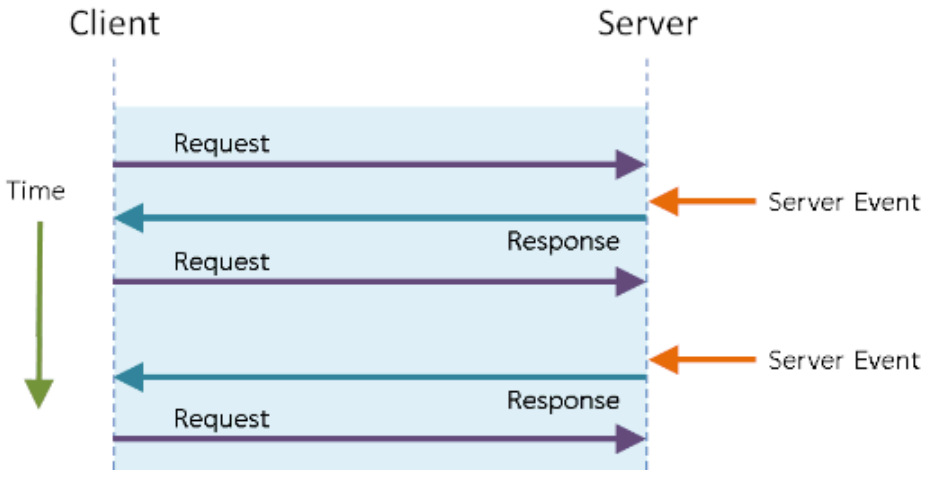
Το Polling είναι μια τεχνική όπου ο client στέλνει μέσω ajax συνεχόμενα αιτήματα (requests) προς τον server και ζητάει δεδομένα (βλ. Εικόνα 1). Αν υπάρχουν νέα δεδομένα τα επιστρέφει, αν όχι, επιστρέφει κενή απάντηση (response) ή ξανά την ίδια πληροφορία και έπειτα κλείνει την σύνδεση (connection). Σε συστήματα πραγματικού χρόνου (real-time), γίνεται εύκολα αντιληπτό ότι το μοντέλο αυτό δεν είναι τόσο αποδοτικό λόγω της μη προβλεψιμότητας της ενημέρωσης των δεδομένων στον server με αποτέλεσμα να μην υπάρχουν κενές απαντήσεις και άρα άσκοπη κίνηση στο δίκτυο.



Εικόνα 1 Polling

2.1.2 Ajax Long-polling

Μια παραλλαγή του polling είναι το Long-polling (βλ. Εικόνα 2) , το οποίο προσπαθεί να εξομαλύνει αδυναμίες του συμβατικού polling. Ο server δέχεται το αίτημα αλλά αν δεν έχει καινούρια δεδομένα να επιστρέψει κρατά την σύνδεση ανοικτή, με μέγιστο ένα συγκεκριμένο (προκαθορισμένο) χρονικό όριο, έως ότου υπάρξουν. Όταν συμβεί αυτό, στέλνει την απάντηση πίσω στον client, τερματίζεται η σύνδεση και ο client στέλνει εκ νέου αίτημα στον server. Αυτός ο κύκλος συνεχίζεται επ' αόριστον έως ότου τερματιστεί ρητώς. Όντως, η προσέγγιση αυτή είναι αποδοτικότερη και μειώνει την άσκοπη κίνηση στο δίκτυο. Αξίζει να σημειωθεί, ότι σε περιπτώσεις που υπάρχει μεγάλος όγκος μηνυμάτων που μεταφέρεται από και προς τον server, το long-polling δεν προσφέρει καμία βελτίωση στην απόδοση έναντι του παραδοσιακού polling.



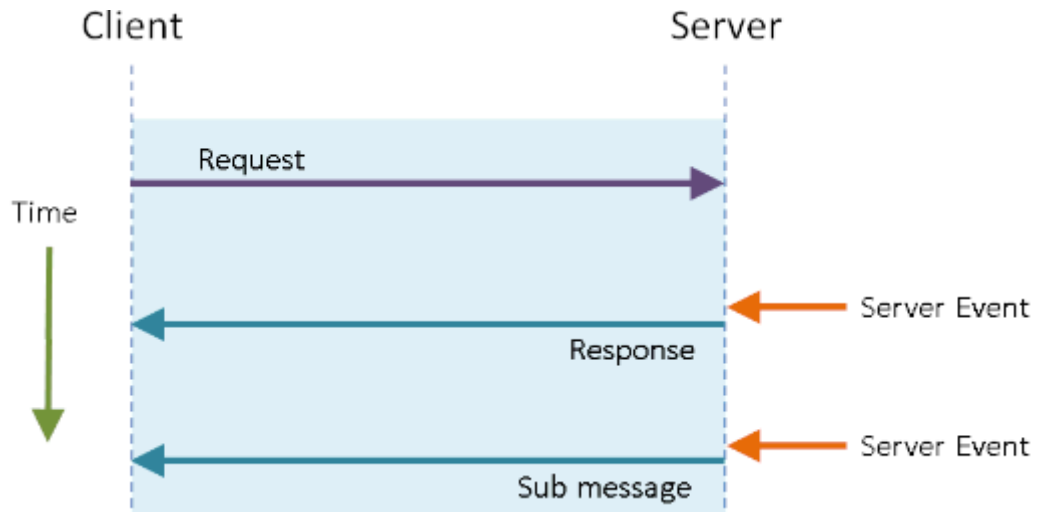
Εικόνα 2: Long polling

2.1.3 HTML5 – Server-Sent Events (SSE)

Το Server-Sent Events, ως τεχνολογία, προσφέρει μονόδρομη επικοινωνία από τον server προς τον client Αφού έχει σταλθεί το αρχικό αίτημα από τον client προς τον server, εγκαθιδρύεται ένα κανάλι επικοινωνίας μεταξύ τους, διαμέσου του οποίου ο δεύτερος στέλνει αυτοβούλως μηνύματα μόνο όταν υπάρξουν νέα δεδομένα (βλ. Εικόνα 4). Με αυτό τον τρόπο, δεν υπάρχει πλέον η ανάγκη για συνεχές polling. Επίσης, το SSE υποστηρίζει automatic reconnection. Όταν, για κάποιο λόγο, διακοπεί η σύνδεση, ο client προσπαθεί μετά από 3 δευτερόλεπτα (από default 3 δευτερόλεπτα, με δυνατότητα παραμετροποίησης) να ξανασυνδεθεί. Προς το παρόν, το SSE υποστηρίζεται, μέσω του EventSource API, από τους μοντέρνους browsers πλην των Internet Explorer και Microsoft Edge. Από πλευράς server δεν υπάρχουν ιδιαίτερες απαιτήσεις. Οι απαντήσεις είναι απλό κείμενο, αλλά έχουν μια ειδική μορφή (βλ. Εικόνα 3).Το μόνο που χρειάζεται να κάνει ο server για να στείλει ένα server-sent event είναι στο μήνυμα που θα σταλεί, να θέσει το Content-Type σε “text/event-stream”.

```
data: first line\n
data: second line\n\n
```

Εικόνα 3: Ένα Server-Sent event μήνυμα

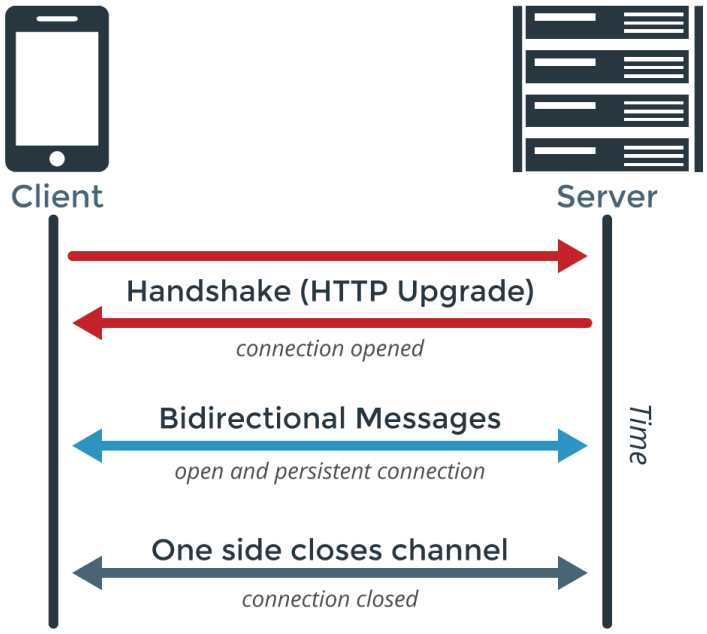


Εικόνα 4: Server-Sent events

2.1.4 HTML5 – WebSockets

Σε αντίθεση με τις προηγούμενες τεχνολογίες, το WebSockets είναι ένα σχετικά νέο πρωτόκολλο το οποίο προσφέρει πραγματική αμφίδρομη επικοινωνία μεταξύ client και server.

Όπως και με το SSE, εγκαθιδρύεται ένα κανάλι επικοινωνίας μεταξύ τους, μέσω του οποίου όμως αυτή την φορά μπορούν ανά πάσα στιγμή να στείλουν μηνύματα και οι δύο (βλ. Εικόνα 5).



Εικόνα 5: WebSockets

Το πρωτόκολλο αυτό ενδείκνυται για low-latency εφαρμογές, όπως για παράδειγμα multiplayer παιχνίδια ή άλλες εφαρμογές οι οποίες απαιτούν την αποστολή μεγάλου όγκου μηνυμάτων αλλά μικρού μεγέθους το καθένα. Επιπρόσθετα, σε περιπτώσεις διακοπής της σύνδεσης, ισχύουν τα εξής:

α) Αν ο server αποφασίσει να διακόψει την σύνδεση, τότε ο client λαμβάνει ένα CloseEvent και γίνεται trigger η onClose μέθοδος του websocket. Μέσω αυτής της μεθόδου μπορεί κάποιος, με χειροκίνητο τρόπο, να επιχειρήσει επανασύνδεση με το server (βλ Εικόνα 6),

```
webSocket.onclose = function(e) {
  console.log('Socket is closed. Reconnect will be attempted in 1 second.', e.reason);
  setTimeout(function() {
    reconnect();
  }, 1000)
};
```

Εικόνα 6: Ενδεικτική προσέγγιση για επανασύνδεση μέσω HTML5 WebSockets API

β) Αν για οποιοδήποτε άλλο λόγο "πέσει" η σύνδεση, τότε απαιτείται ειδική μέριμνα (μέσω custom code) για την επανασύνδεση.

Πλέον, το WebSockets θεωρείται web standard και υποστηρίζεται, μέσω του WebSockets API, από όλους τους μοντέρνους web browsers και μεγάλο πλήθος servers (π.χ. NodeJS, Apache Tomcat, κοκ.). Τέλος, υπάρχουν υλοποιήσεις του WebSockets πρωτοκόλλου και σε άλλες γλώσσες, όπως Java, Python, C#, κ.α.

2.2 Μέθοδοι υλοποίησης σύγχρονων συνεργατικών εφαρμογών στο Web

Η ανάπτυξη συνεργατικού λογισμικού ήταν μια επίπονη και χρονοβόρα διαδικασία ακόμα και για τα πιο απλά συστήματα. Οι προγραμματιστές έπρεπε κάθε φορά και για κάθε σύστημα να μεριμνούν για τα εξής σύνθετα και αμιγώς συνεργατικής φύσης ζητήματα [1]:

- Επιλογή και υλοποίησή επιπέδου διαμοιρασμού πληροφορίας (Data sharing)
- Επιλογή τοπολογίας διασποράς τμημάτων εφαρμογής, δηλ. replicated, centralized και hybrid (Sharing Architecture)
- Διαχείριση συνόδων (Collaborative session management)
- Έλεγχος ταυτοχρονισμού (Concurrency control)
- Έλεγχος πρόσβασης (access control)
- Υποστήριξη ενημερότητας (awareness)
- Συγχρονισμός των κατανεμημένων διεπαφών (Feedthrough)
- Τεχνολογίες κατανεμημένης επικοινωνίας δεδομένων (π.χ. WebSockets, IDL-CORBA-RMI, κοκ.)
- Δημιουργία διαχειριστών συνεδρίας (session managers)

Για την εξομάλυνση της διευθέτησης των παραπάνω ζητημάτων προέκυψε γρήγορα η ανάγκη δημιουργίας APIs και εργαλειοθηκών τα οποία αυτοματοποιούν τις παραπάνω πτυχές. Ως εκ τούτου οι προγραμματιστές εστιάζουν σε υψηλότερου επιπέδου ζητήματα που αφορούν το application logic διευκολύνοντας την ανάπτυξή τους άρδην. Παρακάτω, παρουσιάζονται αντιπροσωπευτικές εργαλειοθήκες (toolkits) οι οποίες είχαν σκοπό να διευκολύνουν τη διαδικασία ανάπτυξης συνεργατικών συστημάτων.

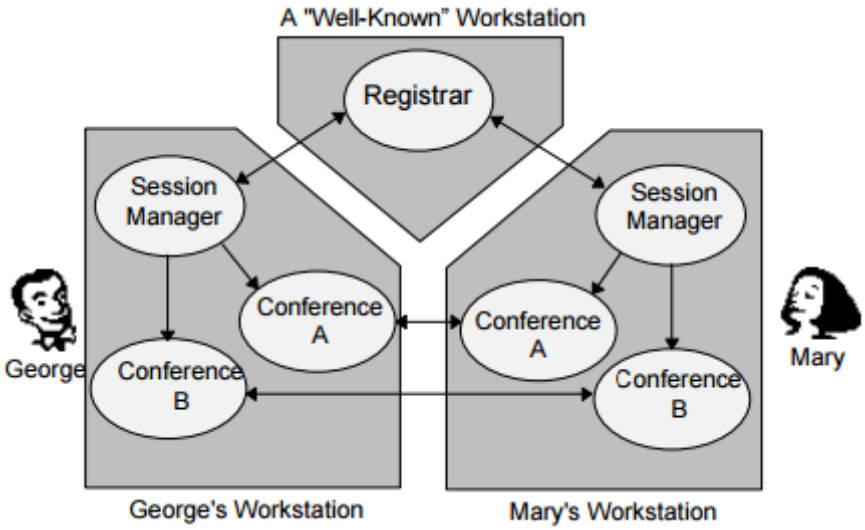
2.2.1 Πρώιμες 'web-oriented' συνεργατικές εργαλειοθήκες

2.2.1.1 GroupWeb

Το GroupWeb [2] είναι ένα σύστημα (συνεργατικός custom Browser) το οποίο αναπτύχθηκε στη βάση του GroupKit. Το GroupKit [1] ήταν ένα toolkit που δημιουργήθηκε για την ανάπτυξη γεωγραφικά διεσπαρμένων ή πρόσωπο με πρόσωπο πραγματικού χρόνου εφαρμογών συνδιάσκεψης. Προσέφερε προγραμματιστικές αφαιρέσεις οι όποιες έκαναν σχεδόν διάφανες τις

προαναφερθείσες δυσκολίες Αρχικά αναπτύχθηκε με τη γλώσσα προγραμματισμού C++ και το Interviews1 widget toolkit. Μετέπειτα, στην δεύτερη γενιά του, έγινε χρήση της γλώσσας Tcl και του Tk widget toolkit. Και οι δύο γενιές στόχευαν στο X Windows System και κατά συνέπεια συστήματα UNIX.

Το σύστημα διαχειριζόταν την δημιουργία, τον εντοπισμό, την διασύνδεση και την καταστροφή των κατανεμημένων διεργασιών (registrar, session managers, conferences), τον συγχρονισμό των εφαρμογών (conferences), όπως και την αυτόματη διαχείριση συνεδριών (session management). Ο συγχρονισμός των εφαρμογών γινόταν μέσω multicast Remote Procedure Calls. Επίσης, παρείχε κάποια στοιχεία ενημερότητας των άλλων χρηστών, όπως για παράδειγμα telepointers, πολυχρηστικά scrollbars, κ.α. Στηρίζεται σε μια υβριδική αρχιτεκτονική (semi-replicated architecture) και αποτελείται από τρεις διαφορετικές διεργασίες (βλ. Εικόνα 7): το Registrar, το Session Manager και το Conference.



Εικόνα 7: An example of GroupKit's runtime process model

Ο Registrar είναι η μόνη κεντρικοποιημένη διεργασία του συστήματος και δρα ως ο συνδετικός κρίκος ανάμεσα σε κάποιο σύνολο χρηστών των εφαρμογών. Επίσης, διατηρεί μια λίστα των conferences και των χρηστών που τα αποτελούν. Για κάθε ένα χρήστη, δημιουργείται (τοπικά) ένας Session Manager ο οποίος αμέσως συνδέεται με τον Registrar και παρέχει την δυνατότητα στον χρήστη να δημιουργήσει, να διαγράψει, να συμμετάσχει ή να αποχωρήσει από ένα conference. Ένα Conference Application είναι ένα GroupKit πρόγραμμα, παραδείγματος χάριν

[1] https://en.wikipedia.org/wiki/InterViews

ένας κοινόχρηστος επεξεργαστής κειμένου, το οποίο έχει ξεκινήσει ο χρήστης διαμέσου του Session Manager. Όταν κάποιος (άλλος) χρήστης δεχτεί μια πρόσκληση για να εισέλθει σε ένα conference, τότε το σύστημα δημιουργεί ένα αντίγραφο του conference αυτού στο δικό του μηχάνημα και τα δύο conferences συνδέονται μεταξύ τους μέσω ενός ομότιμου καναλιού επικοινωνίας (peer to peer).

Το GroupWeb επιτρέπει σε πολλά άτομα να περιηγηθούν και να μοιραστούν σελίδες (HTML) στο διαδίκτυο σε πραγματικό χρόνο. Κάποια από τα χαρακτηριστικά του είναι: slaving του εγγράφου για το συγχρονισμό των σελίδων, slaving της όψης για σύγχρονη κύλιση μέσα στην σελίδα, telepointers και ένα χαλαρό WYSIWIS (what you see is what I see) για να διαχειριστεί τυχόν διαφορετικότητες των viewports, multi-user scrollbars. Επίσης, γίνεται χρήση ενός συνεργατικού επεξεργαστή κειμένου ο οποίος επιτρέπει στους χρήστες να δημιουργήσουν και να επισυνάψουν σχόλια (annotations) σε σελίδες.

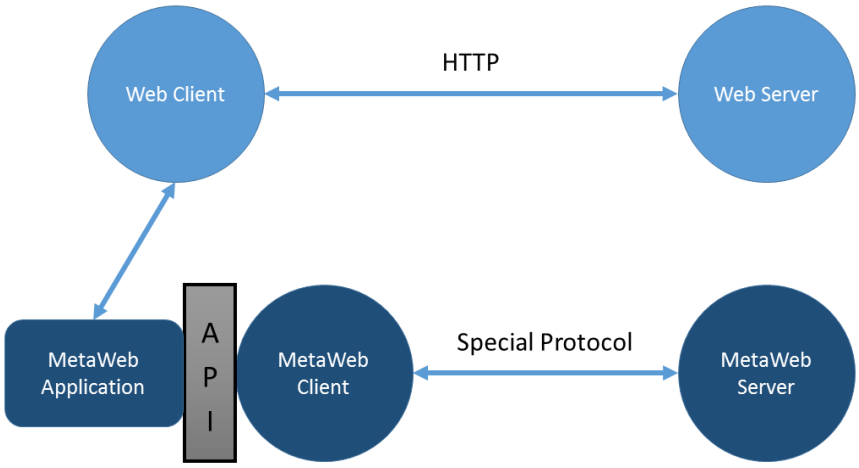
2.2.1.2 MetaWeb

Το MetaWeb [3] αποτελεί μια plugin-based προσέγγιση και εργαλειοθήκη για την ανάπτυξη σύγχρονων συνεργατικών web-based εφαρμογών και προσφέρει διαχείριση συνόδων, έλεγχο πρόσβασης, ανταλλαγή μηνυμάτων, ενημερότητα και επιλύει διάφορα ζητήματα που αναδύονται λόγω της φύσης των web-based εφαρμογών, για παράδειγμα: ανεξαρτησία πλατφόρμας και browser, διαχείριση του διαμοιραζόμενου μοντέλου και υποστήριξη για γενικού σκοπού εφαρμογές. Βασίζεται στην χρήση των Java Applets δεδομένου ότι την περίοδο της ανάπτυξής του δεν είχαν κάνει την εμφάνιση τους τεχνολογίες αμφίδρομης επικοινωνίας.

Για την υποστήριξη της συνεργασίας, το MetaWeb παρέχει τρεις υψηλού επιπέδου αφαιρέσεις: χρήστης (user), τοποθεσία (location), συνεδρία (session). Για κάθε μια από αυτές τις έννοιες, υπάρχει και ο αντίστοιχος τύπος αντικειμένου όπου οι εφαρμογές μπορούν να δημιουργήσουν instances αυτών. Ένα αντικείμενο χρήστη αντιπροσωπεύει μια εφαρμογή που είναι συνδεδεμένη στο MetaWeb. Το αντικείμενο αυτό δημιουργείται κατά την σύνδεση της εφαρμογής με τον MetaWeb server και χρησιμοποιείται για την αναγνώριση του χρήστη στο σύστημα. Τα αντικείμενα συνεδρίας διατηρούν τα τρέχοντα μέλη μιας συνεδρίας, μια λίστα με το ποιοι χρήστες μπορούν να γίνουν μέλη, όπως επίσης και μια περιγραφή της συνεδρίας. Επιπλέον λειτουργικότητα, όπως για παράδειγμα η πρόσκληση ή ο έλεγχος δαπέδου, είναι αποκλειστική

ευθύνη της εφαρμογής. Τα αντικείμενα τοποθεσίας συσχετίζονται με κανένα ή παραπάνω αναγνωριστικά συνεδριών που λαμβάνουν χώρα στην συγκεκριμένη τοποθεσία.

Το MetaWeb επεκτείνει την client/server αρχιτεκτονική του Web με το να προσθέσει ακριβώς δίπλα ένα επιπλέον σύστημα client/server το οποίο επικοινωνεί με ένα custom πρωτόκολλο. Το σύστημα αυτό αποτελείται από τέσσερις οντότητες: εφαρμογή, API, client και server (βλ. Εικόνα 8).



Εικόνα 8: Η αρχιτεκτονική του MetaWeb

Ο προγραμματιστής αναπτύσσει μια MetaWeb εφαρμογή διαμέσου της οποίας ο χρήστης αλληλοεπιδρά με το MetaWeb (μετάφραση δεδομένων εισόδου του χρήστη σε κλήσεις στο MetaWeb, εμφάνιση πληροφορίας από το MetaWeb στον χρήστη, κτλ.) μέσω ενός API το οποίο αποτελείται από κλάσεις και μεθόδους οι οποίες μεταβάλλουν την κατάσταση του μοντέλου αντικειμένων (χρήστες, τοποθεσίες, συνεδρίες) και αποκρύπτουν μεγάλο μέρος της πολυπλοκότητας από τον προγραμματιστή.

Ο MetaWeb client παρέχει δύο λειτουργίες. Πρώτον, δρα ως η αναπαράσταση του χρήστη στο MetaWeb και δεύτερον, διατηρεί τοπικά αντίγραφα των MetaWeb αντικειμένων που είναι αποθηκευμένα στον server. Ο server είναι το σημαντικότερο τμήμα της εργαλειοθήκης. Εδώ βρίσκονται όλα τα MetaWeb αντικείμενα, επιβάλλεται ο έλεγχος πρόσβασης, πραγματοποιείται η δήλωση ενδιαφέροντος (ποιος client ενδιαφέρεται να λάβει συγκεκριμένου είδους γεγονότα) και η διάδοση των γεγονότων (events), τα οποία λαμβάνονται σειριοποιημένα (object serialization) από έναν client και από κει αποστέλλονται σε όλους τους "ενδιαφερόμενους" clients.

2.2.1.3 GroupScape

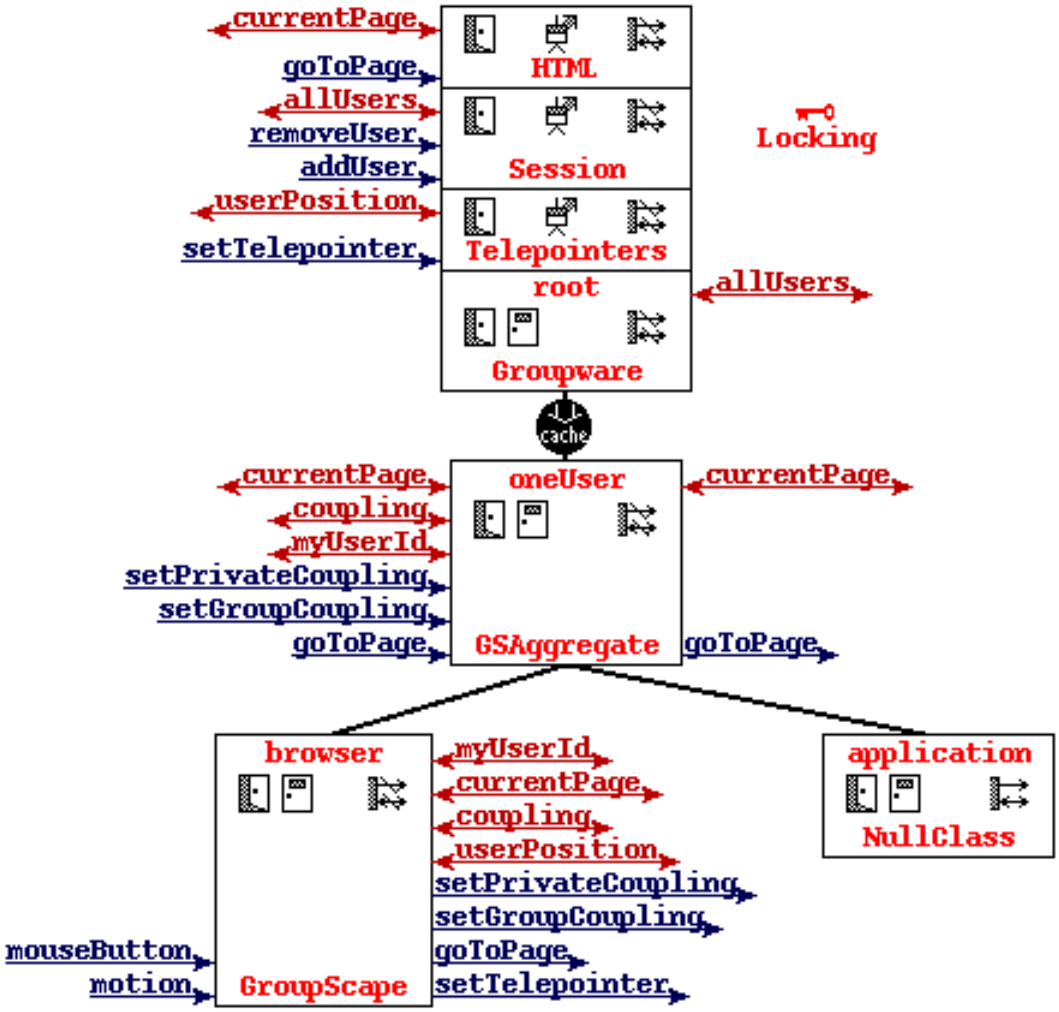
Το 1997 ο Nicholas Graham παρουσίασε μια προσέγγιση [4] όπου επέτρεπε την σύγχρονη συνεργασία ατόμων πάνω σε μια σελίδα στο web, και κατ' επέκταση την επικοινωνία μιας συνεργατικής εφαρμογής με ένα browser, χωρίς επιπρόσθετο προγραμματισμό. Η προσέγγιση αυτή είχε τα εξής τέσσερα χαρακτηριστικά:

- **Ομαδική περιήγηση στο διαδίκτυο**, δηλαδή όλοι οι συμμετέχοντες σε μια συνεδρία να βλέπουν την ίδια σελίδα
- **Περιήγηση βάση της εφαρμογής**, δηλαδή το να μπορεί η εφαρμογή να ελέγχει ποια σελίδα θα δείχνει ανά πάσα στιγμή ο browser
- **Ενσωματωμένες σύγχρονες όψεις**, δηλαδή εξαρτώμενα από την εφαρμογή κομμάτια μέσα στην σελίδα τα οποία θα ενημερώνονται σε πραγματικό χρόνο, χωρίς να χρειάζεται να γίνει refresh η σελίδα
- **Ενσωματωμένα γεγονότα εφαρμογής**, δηλαδή το να μπορεί μια σελίδα να ελέγχει την εφαρμογή μέσω application-specific γεγονότων

Για να επιτευχθούν τα παραπάνω έγινε: α) χρήση ενός design pattern βασιζόμενο στο MVC το οποίο επέτρεπε την εύκολη ενσωμάτωση κάποιου σύγχρονου συνεργατικού λογισμικού με έναν πολυχρηστικό browser, β) εισαγωγή δυο νέων HTML ετικετών (tags), για να καταστεί εφικτή η σύγχρονη συνεργασία, χωρίς επιπρόσθετο προγραμματισμό, γ) χρήση δηλωτικών περιορισμών (declative constraints) που επέτρεπαν την διασύνδεση μιας συνεργατικής εφαρμογής με μια ιστοσελίδα. Η χρήση του MVC design pattern επιτρέπει στον GroupScape browser να συνδεθεί στις διάφορες εφαρμογές χωρίς άμεση επικοινωνία. Αυτό σημαίνει, ενσωμάτωση του Groupscape με τις εφαρμογές, χωρίς επαναπρογραμματισμό είτε του browser είτε της εφαρμογής. Για να παραχθεί το επιθυμητό σύστημα, γίνεται χρήση ενός οπτικού περιβάλλοντος προγραμματισμού (visual programming environment), του ClockWorks, όπου δίνεται η δυνατότητα σε κάποιον να το σχεδιάσει (το σύστημα), με drag'n'drop.

Στην Εικόνα 9 φαίνεται η δομή ενός συστήματος στην πιο απλή μορφή του. Στην ουσία, ο σχεδιαστής όριζε το διαμοιραζόμενο μοντέλο (URL, θέση telepointers, πληροφορίες συνεδρίας, κτλ.) και επέλεγε τον τύπο της εφαρμογής. Η χρήση δύο επιπρόσθετων HTML ετικετών (tags) τις οποίες αντιλαμβάνεται μόνο ο GroupScape browser, κάνει εφικτή την ενημέρωση της

σελίδας βάση του διαμοιραζόμενου μοντέλου σε πραγματικό χρόνο δίχως ανανέωση (refresh), καθώς επίσης και την ενημέρωση του.



Εικόνα 9: An OO design pattern for coupling GroupScape to synchronous applications

2.2.1.4 CORK

Μια ακόμα προσέγγιση που αξιοποιεί τα Java applets είναι το CORK [5]. Ως στόχο έχει: α) την προσαρμογή μονοχρηστικών, μη συνεργατικών εφαρμογών σε συνεργατικές, β) την ελαχιστοποίηση του εύρους ζώνης με το να διαμοιράζει κάποιο από τα τρία υψηλότερα επίπεδα διεπαφής [6]: μοντέλου, όψης και αλληλεπιδραστικου αντικειμένου, αντί του παραθύρου ή της οθόνης τα οποία προϋποθέτουν μεγαλύτερο όγκο μηνυμάτων, γ) υποστήριξη για σύγχρονη και ασύγχρονη αλληλεπίδραση. Κατά τον προγραμματισμό των διεπαφών, ορίζονται ποια αντικείμενα θα διαμοιράζονται και θα παραμένουν συγχρονισμένα. Μεταβολές στην κατάσταση κάποιου αντικειμένου ανιχνεύονται από listener αντικείμενα, τα οποία αναλαμβάνουν να

προωθήσουν μέσω δικτύου την εκάστοτε μεταβολή -ένα change object το οποίο έχει υποστεί σειριοποίηση, στους απομακρυσμένους εταίρους. Έπειτα, το CORK εντοπίζει το αντικείμενο στο οποίο αντιστοιχεί η μεταβολή και την εφαρμόζει. Επιπρόσθετα, κάθε διαμοιραζόμενο αντικείμενο αντιστοιχίζεται με ένα permissions object, το οποίο επιτρέπει την λήψη ή την εφαρμογή (μέρος ή του συνόλου των) αλλαγών όπου έχουν ληφθεί από κάποιο απομακρυσμένο εταίρο. Με αυτό τον τρόπο, το σύστημα μπορεί να εφαρμόσει διαφορετικούς ρόλους για διαφορετικούς χρήστες. Για παράδειγμα, σε μια chat εφαρμογή, ενδέχεται να υπάρχουν επισκέπτες (guest) χρήστες οι οποίοι μπορούν μονάχα να διαβάσουν τα μηνύματα, κανονικοί χρήστες οι οποίοι μπορούν να διαβάσουν και να γράψουν μηνύματα, και διαχειριστές οι οποίοι μπορούν να διαβάσουν, γράψουν και επεξεργαστούν μηνύματα. Τέλος, παρέχεται υποστήριξη για latecomers αφού το σύστημα αποθηκεύει την πιο πρόσφατη κατάσταση των διαμοιραζόμενων αντικειμένων σε σχεσιακή βάση δεδομένων.

2.2.2 Σύγχρονα APIs για την υλοποίηση σύγχρονων μορφών επικοινωνίας στο Web

Τα APIs οικειοποιούνται μέρος των προαναφερθέντων τεχνικών και μεθόδων επικοινωνίας προσφέροντας σχετικά υψηλού επιπέδου APIs για την υλοποίηση χαμηλότερου επιπέδου τεχνικών ζητημάτων (π.χ. fallback functions, disconnection, session management, κοκ.). Παρακάτω αναλύονται τα κυριότερα-ενδεικτικότερα εξ αυτών.

2.2.2.1 Socket.io [2]

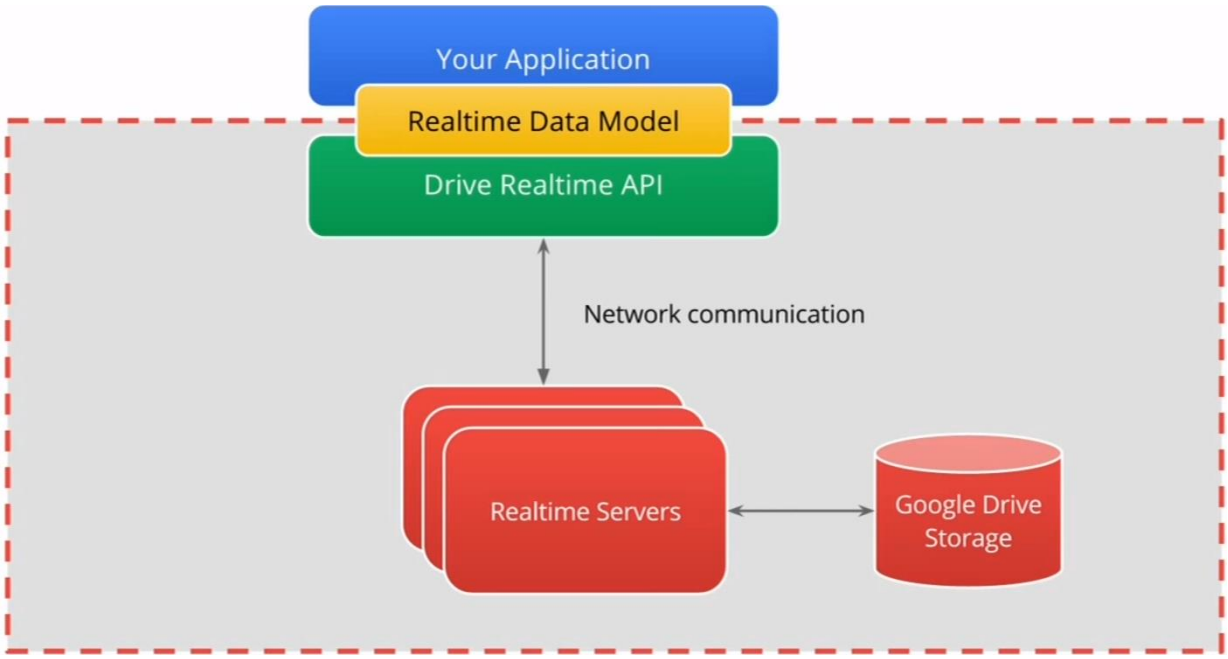
Το Socket.IO είναι μια JavaScript βιβλιοθήκη που επιτρέπει την ανάπτυξη πραγματικού χρόνου εφαρμογών διαδικτύου. Προσφέρει αμφίδρομη επικοινωνία μεταξύ client και server, καθώς αξιοποιεί διάφορες τεχνολογίες που την υποστηρίζουν (αμφίδρομη επικοινωνία) είτε εγγενώς (βλ. WebSockets, Flash Sockets) είτε την προσομοιάζουν (βλ. AJAX long-polling, Ajax multipart straming, Infinite-Frame) λειτουργώντας ως ένα επίπεδο αφαίρεσης πάνω από τα παραπάνω χρησιμοποιώντας ένα unified API. Εν εκτελέσει, η βιβλιοθήκη επιλέγει ποιο μέσο θα χρησιμοποιήσει λαμβάνοντας υπόψιν τις δυνατότητες του browser, έχοντας ως πρώτη επιλογή το WebSockets. Αν δεν υποστηρίζεται, καταφεύγει σε κάποια εφεδρικό. Αξιοσημείωτο χαρακτηριστικό, είναι η έννοια των Rooms and Namespaces το οποίο μπορεί να χρησιμοποιηθεί για να προσομοιάσει ένα Collaborative Session. Αυτό που ουσιαστικά συμβαίνει, είναι κάθε client επιλέγει σε ποιο room θέλει να κάνει subscribe ώστε να λαμβάνει ενημερώσεις που

[2] http://socket.io/

αφορούν μόνο το συγκεκριμένο room. Άλλες παρόμοιες, όχι τόσο γνωστές, βιβλιοθήκες είναι οι: Sock.js και Primus.

2.2.2.2 Google Real-time API [3]

Το Realtime API της Google (βλ. Εικόνα 10) είναι ένα cloud-based open API για συνεργατικές εφαρμογές και χρησιμοποιεί το Google Drive για την αποθήκευση των δεδομένων. Προσφέρει την δυνατότητα στον προγραμματιστή να ορίσει ένα διαμοιραζόμενο μοντέλο δεδομένων (shared data model), όπως maps, λίστες ή custom JavaScript αντικείμενα, το οποίο είναι κεντρικοποιημένο. Καθώς οι χρήστες εισέρχονται στην εφαρμογή αποκτούν ένα τοπικό, in-memory αντίγραφο του μοντέλου αυτού. Όταν κάποιος χρήστης μεταβάλει την κατάσταση του στο δικό του μηχάνημα, οι αλλαγές αμέσως αντικατοπτρίζονται και στους υπόλοιπους χρήστες.



Εικόνα 10: Αρχιτεκτονική Google Real-time API

Τέλος, το Google Realtime API ενδείκνυται για εφαρμογές όπου οι χρήστες δύνανται να επεξεργαστούν τα ίδια δεδομένα ταυτοχρόνως, παραδείγματος χάριν ένα επεξεργαστή κειμένου, και όχι με συγκεκριμένη σειρά όπως ένα επιτραπέζιο παιχνίδι (board game), ενδεχομένως.

[3] https://developers.google.com/google-apps/realtime/overview

2.2.2.3 Firebase Realtime Database [4]

Το Firebase Realtime Database είναι μια cross-platform NoSQL βάση δεδομένων (αποθηκεύει τα δεδομένα σε JSON format) που φιλοξενείται (hosted) στο cloud. Η βάση "διαμοιράζεται" στους clients, με αποτέλεσμα ότι αλλαγές συμβούν, να διαδίδονται στους υπόλοιπους clients σε πραγματικό χρόνο. Οι Firebase εφαρμογές έχουν offline πρόσβαση στα δεδομένα. Όταν η εφαρμογή επανέλθει στο δίκτυο, ότι αλλαγές έχουν συμβεί συγχρονίζονται αμέσως.

Εικόνα 11: Υποστήριξη ενημερώσεων κατά μήκος πολλαπλών πλατφορμών

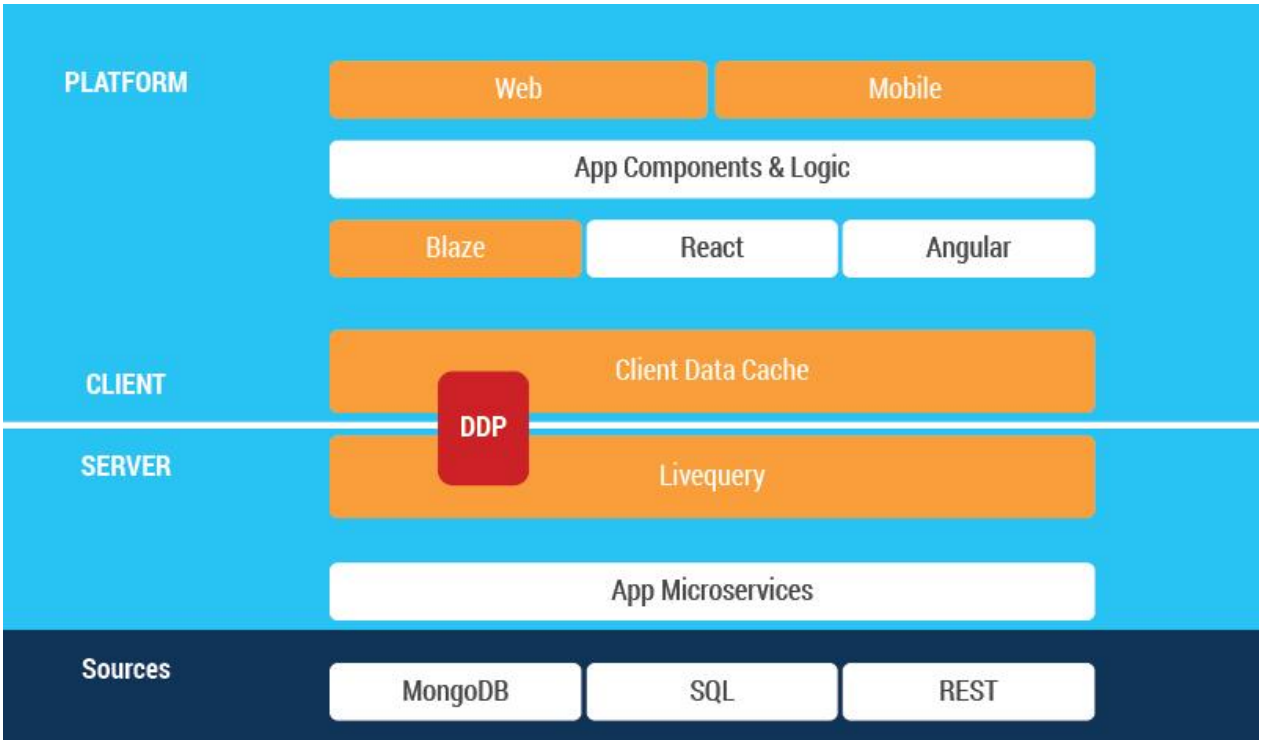
Υποστηρίζει εγγενώς τις εξής τρεις πλατφόρμες: Android, iOS και Web. Επίσης, μέσω του Rest API μπορούν να υποστηριχθούν και desktop εφαρμογές γραμμένες σε γλώσσες, όπως Java, Go, Python, κ.α.

2.2.2.4 Meteor [5]

Το Meteor είναι ένα δημοφιλές self-hosted web framework γραμμένο σε Node.js και σκοπό έχει την γρήγορη κατασκευή πραγματικού χρόνου cross-platform (Web, Android, iOS) εφαρμογών. Ως βάση δεδομένων χρησιμοποιείται η MongoDB. Χρησιμοποιεί διάφορα rendering systems (βλ. Blaze, React, Angular) για την γρηγορότερη κατασκευή διεπαφών και εξάλειψη επαναλήψιμου markup κώδικα (HTML), με προεπιλεγμένο το Blaze. Επίσης, για να επιτευχθεί ο συγχρονισμός των εφαρμογών που έχουν υλοποιηθεί χρησιμοποιώντας το συγκεκριμένο

[4] https://firebase.google.com/
[5] https://www.meteor.com/

framework, έχει σχεδιαστεί ένα ειδικό πρωτόκολλο το οποίο βασίζεται στο websockets πρωτόκολλο, το DDP (Distributed Data Protocol). Το DDP έχει ως κύρια επιλογή για transport το websockets αλλά αν δεν υποστηρίζεται από κάποιο client, καταφεύγει στο Long polling. Τέλος, μέσω του DDP διαδίδονται μηνύματα σε μορφή EJSON, μια επέκταση του JSON, η οποία επίσης έχει σχεδιαστεί στα πλαίσια του Meteor, και υποστηρίζει περισσότερους τύπους δεδομένων, όπως, Date, Binary και τύπους δεδομένων ορισμένους από το χρήστη (User-defined types).



Εικόνα 12: Meteor platform overview

2.2.2.5 CommonGround

Το CommonGround [8] είναι μια εργαλειοθήκη που προσπαθεί να κάνει ευκολότερη την ανάπτυξη σύγχρονου συνεργατικού λογισμικού στο Web χρησιμοποιώντας ευρέως γνωστές και εγγενώς υποστηριζόμενες τεχνολογίες. Είναι γραμμένο στο GWT[6] (Google Web Toolkit), πράγμα που σημαίνει πως σύγχρονο συνεργατικό λογισμικό βασιζόμενο στο AJAX μπορεί να γραφτεί από ένα μεγάλο πλήθος Java προγραμματιστών, και όχι μόνο από Web προγραμματιστές. Χρησιμοποιεί μια transactional αρχιτεκτονική αντιγράφων, η οποία έχει υλοποιηθεί χρησιμοποιώντας το ασύγχρονο RPC (remote procedure call) του GWT και την

[6] https://en.wikipedia.org/wiki/Google_Web_Toolkit

τεχνολογία AJAX Push, όπου στην συγκεκριμένη περίπτωση γίνεται χρήση του Long Polling. Java αντικείμενα που περιγράφουν τις αλλαγές (εν προκειμένω, γεγονότα) σειριοποιούνται, στέλνονται στον server, και αυτός τα προωθεί στους υπόλοιπους εταίρους. Ο ορισμός διαμοιραζόμενων αντικειμένων ανάμεσα στους χρήστες μιας συνεδρίας, πραγματοποιείται με την δημιουργία υποκλάσεων της κλάσης SharedModel. Οι υποκλάσεις της κλάσης SharedModel, προσδιορίζουν ρητώς ποια πεδία θα διαμοιράζονται. Επιπλέον, τα διαμοιραζόμενα αντικείμενα δύνανται να "συνδεθούν" με αντικείμενα της διεπαφής (checkboxes, textfields) μέσω Listeners αντικειμένων. Κατά συνέπεια, όταν η τιμή ενός διαμοιραζόμενου αντικειμένου μεταβληθεί από κάποιον απομακρυσμένο εταίρο, η διεπαφή θα ενημερωθεί κατάλληλα. Όσον αφορά το back-end κομμάτι, έχει υλοποιηθεί χρησιμοποιώντας την κλάση RemoteServiceServlet που παρέχεται από το GWT, και επιτρέπει την σειριοποίηση και απόσειριοποίηση των αντικειμένων, όσο και την κλήση της διαδικασιών (procedures). Τέλος, παρέχεται υποστήριξη για latecomers, αφού τα αντικείμενα (γεγονότα) που λαμβάνονται, αποθηκεύονται σε μια MySQL βάση δεδομένων πρώτού προωθηθούν προς τους υπόλοιπους εταίρους.

2.2.2.6 Towards Multi-Domain Collaboration toolkits

Στο [7] οι Bartel και Dewan, παρουσίασαν μια συνεργατική εργαλειοθήκη πολλαπλών πεδίων (multi-domain collaboration toolkit) η οποία αποκρύπτει την ετερογένεια των υποστηριζόμενων εργαλειοθηκών διεπαφών από τους προγραμματιστές και τους τελικούς χρήστες και επιτρέπει στις απομακρυσμένες διεπαφές να παραμένουν συγχρονισμένες μεταξύ τους. Η εργαλειοθήκη στοχεύει στα εξής πεδία: Desktop, Eclipse και Web, και για να τα υποστηρίξει, εκμεταλλεύεται τις παρακάτω μονοχρηστικές εργαλειοθήκες διεπαφών (single-user user-interface toolkits): AWT, Swing, SWT και GWT, οι οποίες είναι γραμμένες σε Java.

Το AWT και Swing υποστηρίζουν desktop widgets και browser plug-in widgets μέσα σε applets. Το SWT υποστηρίζει desktop widgets και plug-in Eclipse widgets. Το GWT υποστηρίζει plug-in browser widgets, καθώς ο πηγαίος κώδικας των εφαρμογών γράφεται σε Java, ο οποίος μετέπειτα μεταγλωττίζεται σε HTML και Javascript. Για να επιτευχθεί αυτή η ενοποίηση, το [7] δρα ως ένα abstraction layer πάνω από τις προαναφερθείσες εργαλειοθήκες και χρησιμοποιεί ένα κοινό API για την κατασκευή και τον συγχρονισμό των διεπαφών. Ποιο interface toolkit θα χρησιμοποιηθεί "στο παρασκήνιο" για την κατασκευή της διεπαφής, αποφασίζεται από το

σύστημα κατά τον χρόνο εκτέλεσης, αφού έχει εξακριβωθεί το μέσο όπου χρησιμοποιείται για να εκκινήσει η εφαρμογή.

Για παράδειγμα, αν χρησιμοποιηθεί ένας browser, θα γίνουν κλήσεις του GWT, κοκ. Επίσης, οι δημιουργοί αποφάσισαν να διαμοιράσουν το widget layer, κάτι που προσφέρει τον διαμοιρασμό των γεγονότων που εκπέμπουν τα αλληλεπιδραστικά αντικείμενα αφού έχουν "εντοπιστεί" από αντικείμενα listeners. Για κάθε αλλαγή που συμβαίνει στη διεπαφή κάποιου χρήστη, στέλνεται ένα γεγονός που την περιγράφει σε όλους τους άλλους χρήστες της ίδιας συνεδρίας. Το λαμβάνει ο εκάστοτε widget server, ο οποίος είναι υπεύθυνος να ενημερώσει την διεπαφή καταλλήλως. Τέλος, το σύστημα παρέχει υποστήριξη για χρήστες που εισέρχονται σε μεταγενέστερο χρόνο (latecomers) σε μια συνεδρία.

2.2.3 Μοντελο-κεντρική Μηχανική Διεπαφών

Η μοντελο-κεντρική μηχανική διεπαφών αποτελεί εναλλακτική των εργαλειοθηκο-κεντρικών προσεγγίσεων. Πλεονεκτήματα απορρέουν από την εναπόθεσή τους στην υιοθέτηση αφηρημένης σημειολογίας και markup γλωσσών με στόχο την περιγραφή αφηρημένων αντικειμένων και τη μετέπειτα διασύνδεσή του με λεξικά στόχους (target vocabularies).

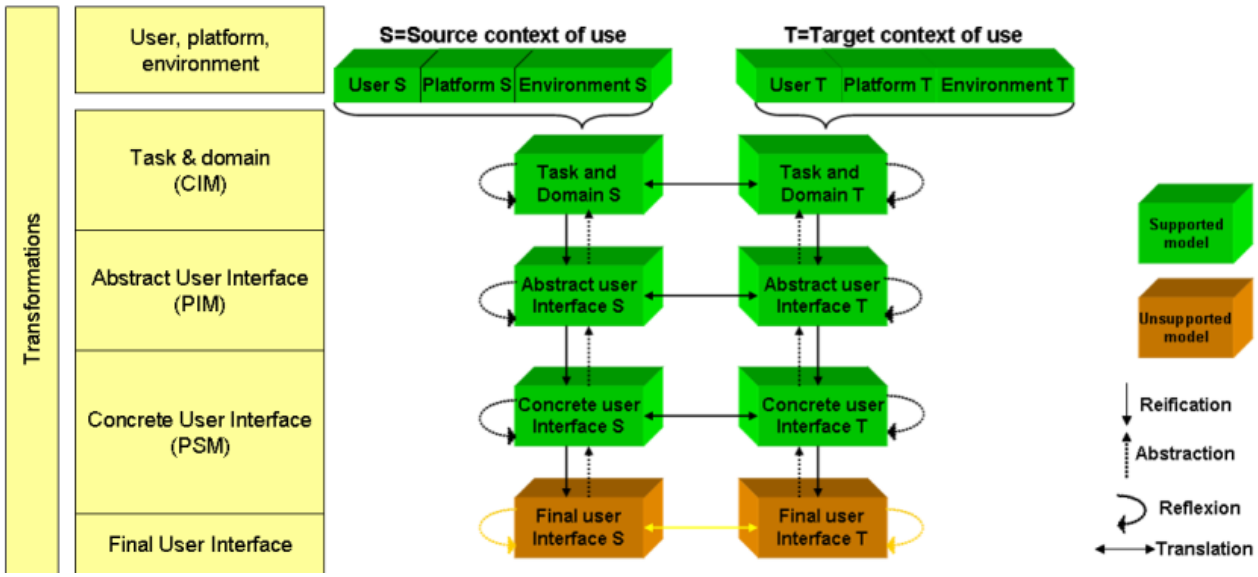
2.2.3.1 UsiXML/CRF (++extensions)

Η UsiXML [20] αποτελεί την υλοποίηση του Cameleon Reference Framework το οποίο προορίζεται για το ενοποιημένο specification και ανάπτυξη multi-context multi-modal αλληλεπιδραστικών συστημάτων. Ως context θεωρείται η τριπλέτα: 'πλατφόρμα', 'στερεότυπο χρήστη' και 'περιβάλλον εκτέλεσης'. Οι υποστηριζόμενες προδιαγραφές μπορεί να υποστηρίξουν ποικίλους συνδυασμούς των προηγούμενων στα πλαίσια της ίδιας μεθοδολογίας.

Αναλυτικότερα τα επίπεδα προσδιορισμού μιας διεπαφής ανάλογα με το επίπεδο αφαίρεσης στο οποίο στοχεύει κάθε φορά ο σχεδιαστής του συστήματος χωρίζονται στα ακόλουθα τρία: α) Tasks & Concepts, β) Abstract UI και γ) CUI (βλ. Εικόνα 13). Το Tasks and Concepts αφορά σε ένα 'implementation agnostic' specification μοντέλο το οποίο δίνει τη δυνατότητα προσδιορισμού ενός αλληλεπιδραστικού συστήματος υπό τη μορφή της ιεραρχικής αποσύνθεσης των καθηκόντων που αυτή προβλέπεται να υποστηρίζει. Το επόμενο επίπεδο, AUI, υποστηρίζει την περιγραφή της διεπαφής με τρόπο ανεξαρτήτου modality και πλατφόρμας (modality και platform independent). Τέλος το CUI αφορά στη δυνατότητα περιγραφής της διεπαφής με τρόπο

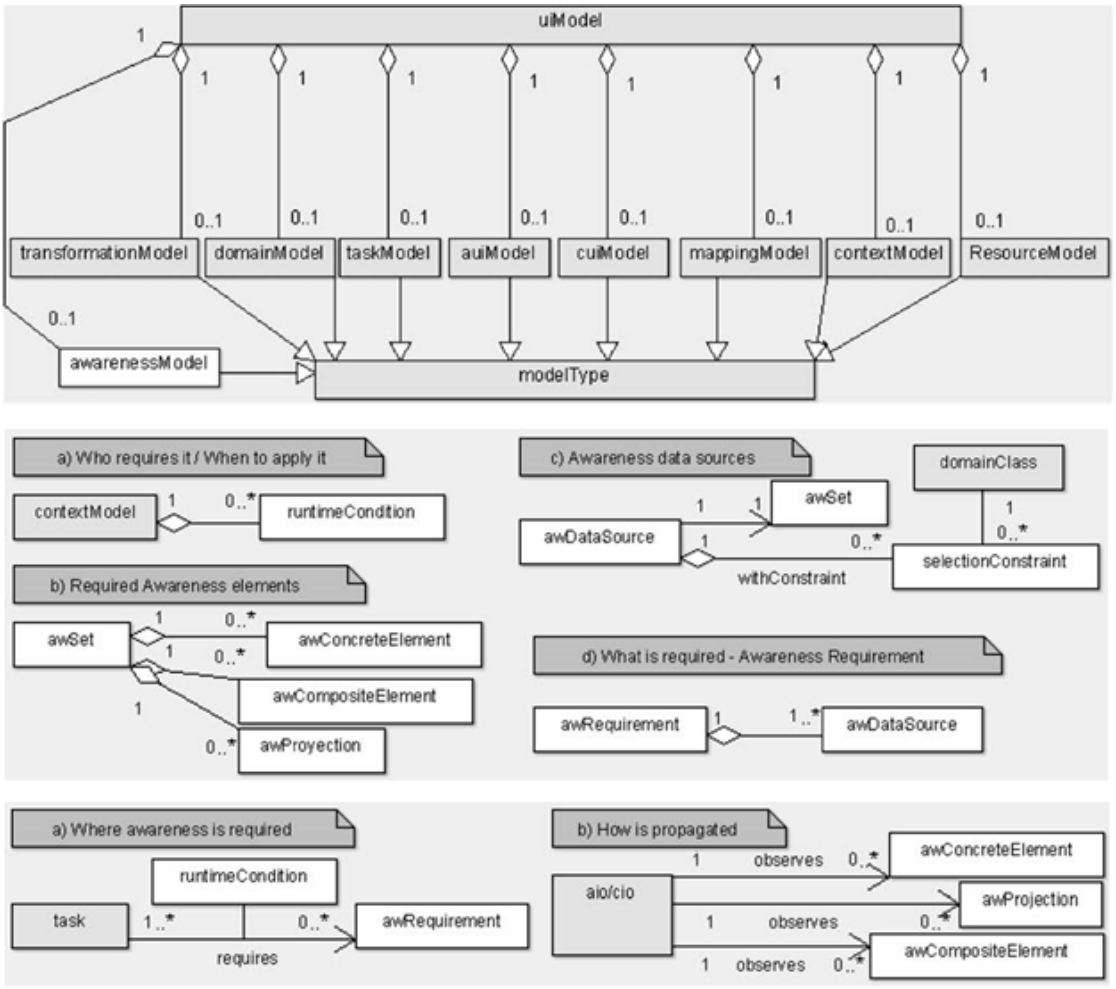
που να είναι ανεξάρτητος πλατφόρμας (platform independent) και διαδραστικής βιβλιοθήκης (vocabulary independent).

Μεταβάσεις μεταξύ των μοντέλων υποστηρίζονται προς κάθε κατεύθυνση διαμέσου μετασχηματισμών (transformations) οι οποίοι αποτελούν εγγενές χαρακτηριστικό των μοντελο-κεντρικών γλωσσών και υποστηρίζονται μέσω ενός mapping μοντέλου στα πλαίσια του οποίου ορίζονται εννοιολογικές συνδέσεις-συσχετίσεις μεταξύ των μοντέλων. Κάθε φορά που οι παραγόμενες προδιαγραφές είναι να παράξουν την τελική διεπαφή (Final UI - FUI), αυτό γίνεται μέσω ενδεδειγμένων τμημάτων λογισμικού (platform-specific renderers) οι οποίοι αντιστοιχίζουν την αφηρημένη σημειολογία με platform-specific αλληλεπιδραστικά λεξικά (ανάλογα με τη διαθεσιμότητά τους στην εκάστοτε πλατφόρμα).



Εικόνα 13: Cameleon Refrence Framework

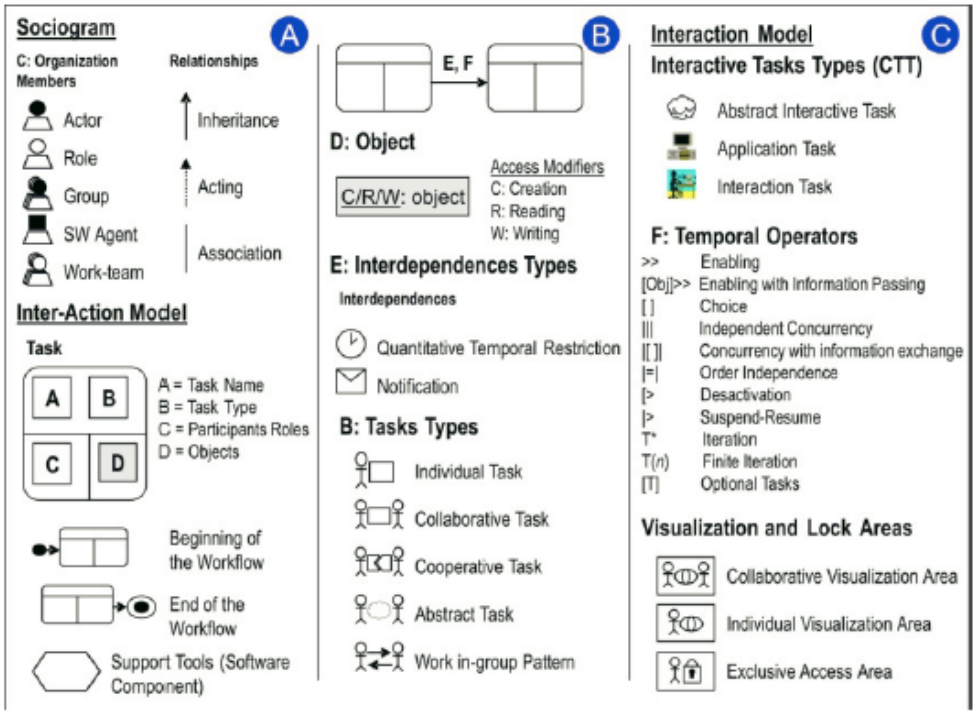
Στο [13] τεκμηριώνονται πρόσφατα αποτελέσματα επέκτασης της γλώσσας προς την κατεύθυνση της υποστήριξης ενημερότητας (awareness). Οι επεκτάσεις, υπό τη μορφή νέων μοντέλων, φαίνονται στην **Εικόνα *14***. Βέβαια οι προτεινόμενες επεκτάσεις αφορούν μόνο τα ανώτερα επίπεδα αφαίρεσης (CIM και PIM), ως εκ τούτου δεν προνοούν και δεν διευθετούν θέματα που αφορούν σειρά χαμηλότερου επιπέδου, αμιγώς συνεργατικών και μη ζητημάτων, όπως του επιπέδου και αρχιτεκτονικής διαμοιρασμού δεδομένου, της διαχείρισης συνόδων, του συγχρονισμού (feedthrough), των τεχνολογικών μεθόδων κατανεμημένης επικοινωνίας δεδομένων στο δίκτυο, της διαχείρισης υποστηριζόμενων εγγενώς και μη διαδραστικών αντικειμένων, κοκ.



Εικόνα 14: Επεκτάσεις UsiXML για την υποστήριξη Awareness

2.2.3.2 CIAN (Collaborative Interactive Applications Notation)

Η εν λόγω προσέγγιση [21] εστιάζει στα ανώτερα και μόνο επίπεδα προσδιορισμού ενός συνεργατικού συστήματος, χωρίς να υποστηρίζει ή να προνοεί για τη 'δημιουργία' της τελικής – λειτουργικής διεπαφής. Αναλυτικότερα προτείνει μια αφηρημένη σημειολογία (βλ. Εικόνα 15), πλήρως απαλλαγμένη από οποιαδήποτε αναφορά σε συσκευές (devices), υπολογιστικές οντότητες (platform independent), αλληλεπιδραστικές μεταφορές και κανάλια επικοινωνίας (modality independent), στα πλαίσια της οποίας ένα αλληλεπιδραστικό συνεργατικό σύστημα αναλύεται διαμέσου της ιεραρχικής αποσύνθεσης καθηκόντων (tasks).



Εικόνα 15: CIAN Notation

Παρά το γεγονός ότι τέτοιους είδους περιγραφές υποστηρίζονται και από άλλες μοντελο-κεντρικές γλώσσες, η διαφοροποίηση αυτής εστιάζει στην πρόβλεψη και στο ρητό διαχωρισμό μεταξύ των μεμονωμένων (‘individual’), των ‘cooperative’ και ‘collaborative’ tasks. Επίσης, εισάγει την έννοια των ρόλων καθιστώντας εφικτό το διαχωρισμό των tasks ανάλογα με τον-τους ρόλους που ενδείκνυνται για την εκτέλεσή τους. Στο [15] παρουσιάζεται η διασύνδεση της οικείας σημειολογίας με τη UML. Η εν λόγω σημειολογία προτείνεται στα πλαίσια μιας επίσης οικείας μεθοδολογίας που ονομάζεται CIAM. Τέλος παρέχει υποστήριξη, για το αλληλεπιδραστικό specification των μοντέλων, μέσω ενδεδειγμένων οικείων εργαλείων (CIAT) που έχουν αναπτυχθεί ως plugins του Eclipse. Σε κάθε περίπτωση τα εν λόγω ‘εργαλεία, δεν προνοούν ως τη διευθέτηση ζητημάτων που έχουν να κάνουν με την εν εκτέλεση υποστήριξη της εκτελέσιμης συνεργατικής διεπαφής-εφαρμογής. Ως εκ τούτου δεν καταπιάνονται με πτυχές που έχουν να κάνουν με το διαμοιρασμό δεδομένων, έλεγχο ταυτοχρονισμού, διαχείριση συνόδων, κοκ.

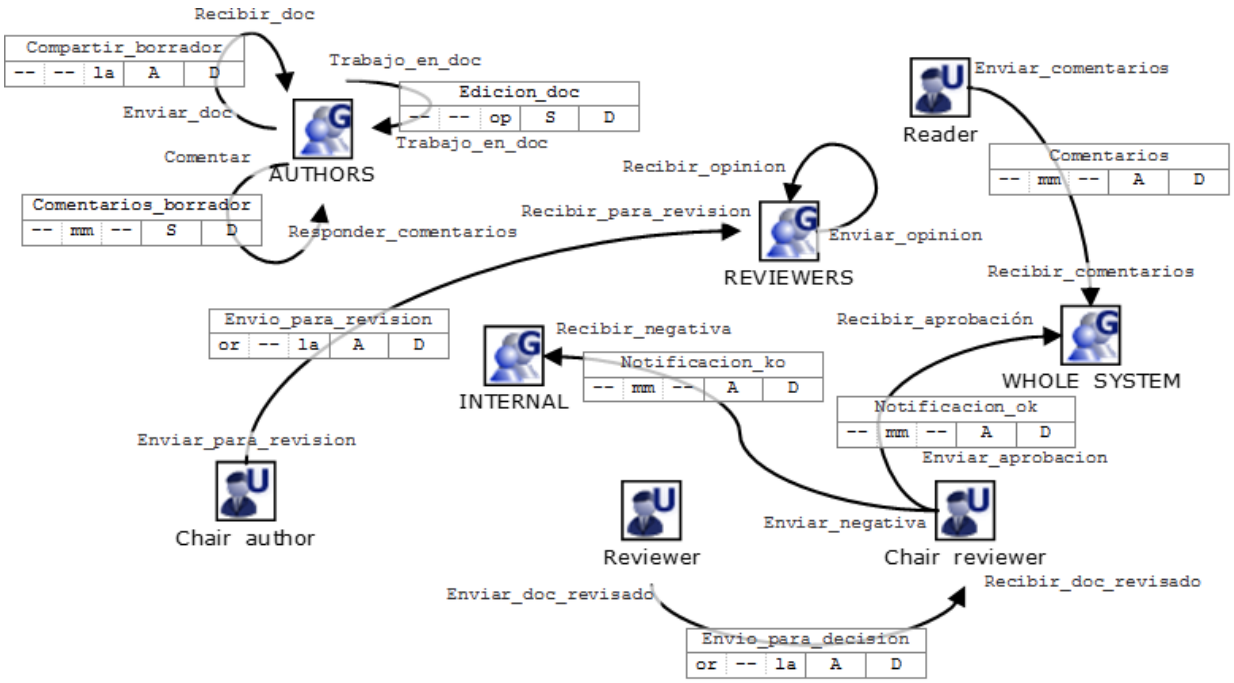
2.2.3.3 TOUCHE

Το TOUCHE [16] αποτελεί ακρωνύμιο του: ‘Task-Oriented and User-Centred Process Model for Developing Interfaces for Human-Computer-Human Environments’. Το TOUCHE εστιάζει στην

οργανωτική δομή (organizational structure) των χρηστών σε ένα CSCW σύστημα καθώς και στις συσχετίσεις μεταξύ αυτών. Στην Εικόνα 16 απεικονίζεται η σημειολογία μέσω της οποίας αναπαρίστανται οι υποστηριζόμενοι τύποι 'Actors'. Η σημειολογία αφορά στο αντίστοιχο επίπεδο CIM της UsiXML (βλ. Εικόνα 13). Ένα ενδεικτικό παράδειγμα επιστράτευσής τους στην πράξη φαίνεται στην Εικόνα 17.

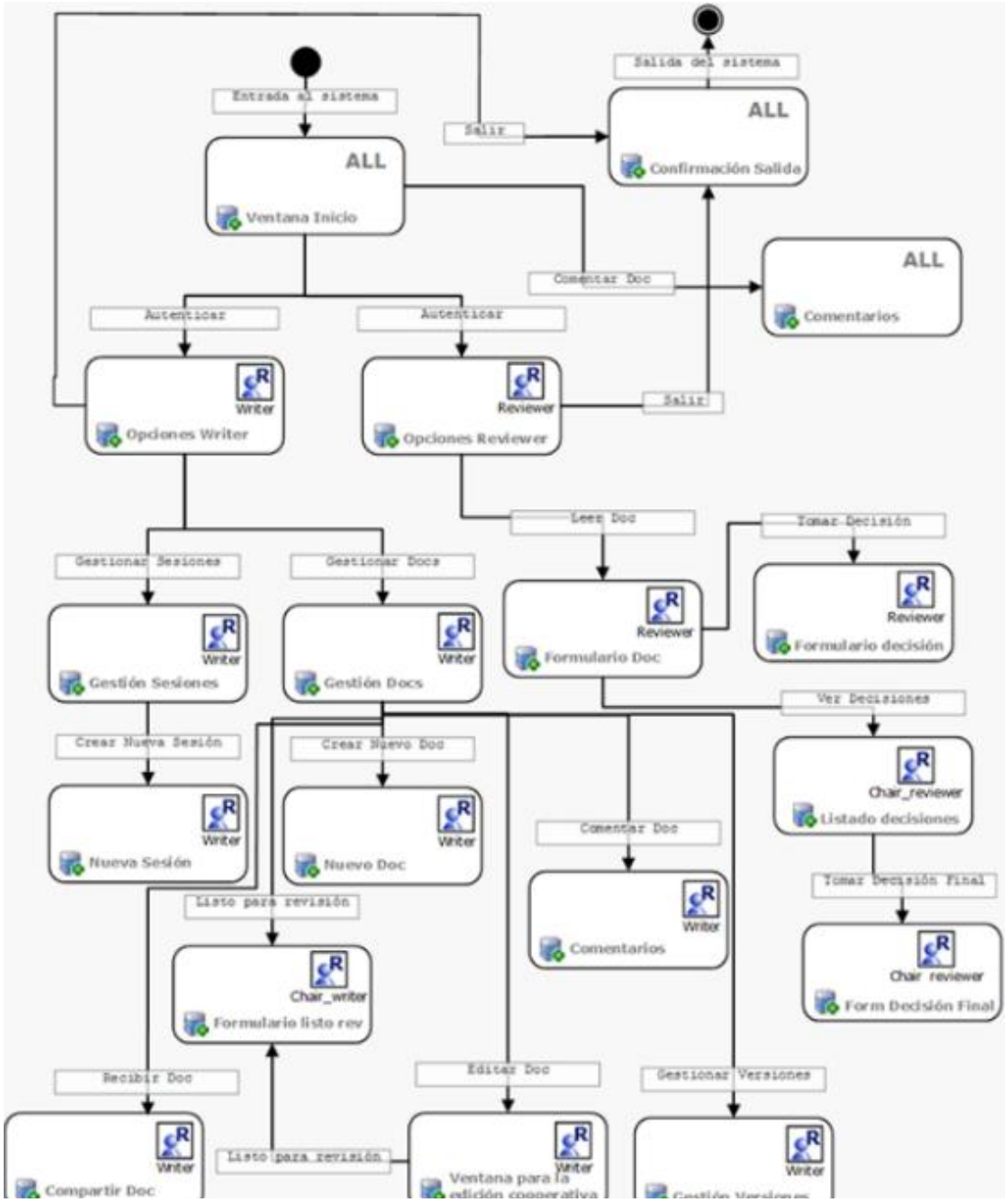
Organizational Item	Description	Notation
Actor	An actor is one or several persons or another external system that interacts with the system. Actors directly interact within the collaborative system to accomplish individual tasks, but they can also interact with each other, through the system, to perform cooperative tasks.	 ACTOR 1
Group	A group is a set of individual or collective actors which play roles. Such a set of actors needs to interact together and to collaborate in order to achieve a common objective.	 GROUP 1
Individual	An individual item is a unique actor which plays a role.	 Individual 1
User	A user is a human individual item who interacts with the system.	 User 1
Agent	An agent is a non human individual item who interacts with the system	 Agent 1
Role	A role is defined by means of the set of tasks which an actor performs. An actor is what it is due to the roles it plays.	 ROLE 1

Εικόνα 16: Υποστηριζόμενοι τύποι Actors στο TOUCHE

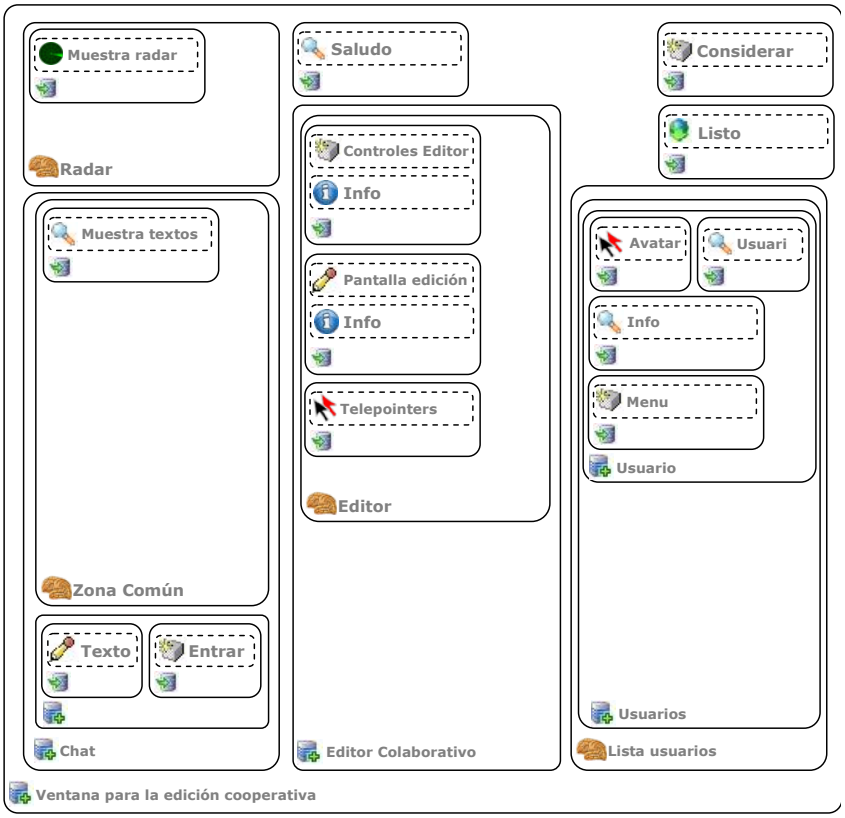


Εικόνα 17: Παράδειγμα CIM επιπέδου αφαίρεσης στο TOUCHE

Το TOUCHE επίσης επαυξάνει τα μοντέλα AUI και CUI ώστε να μπορούν να επιδεχθούν στα semantics τους πληροφορία που αφορά τόσο τους Actors του CIM επιπέδου όσο επίσης συνεργατικά καθήκοντα αλλά και την ανάγκη υποστήριξης ενημερότητας στα πλαίσια αυτών. Στα πλαίσια του προηγούμενου παραδείγματος της Εικόνα 17, το AUI και CUI θα αποτυπώνονταν όπως φαίνεται στις εικόνες Εικόνα 18 και Εικόνα 19 αντίστοιχα.

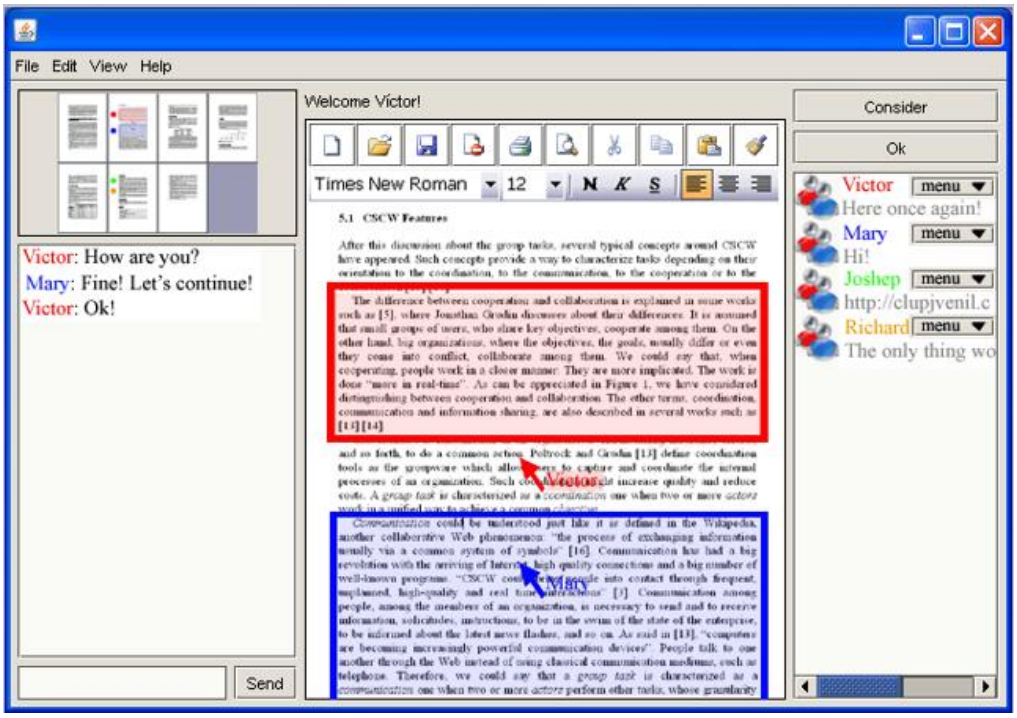


Εικόνα 18: Παράδειγμα στιγμιότυπου AUI μοντέλου του TOUCHE



Εικόνα 19: Παράδειγμα στιγμιότυπου CUI μοντέλου του TOUCHE

Τα παραπάνω διερμηνευόμενα κατάλληλα παράγουν την εκτελέσιμη διεπαφή της Εικόνα 20.

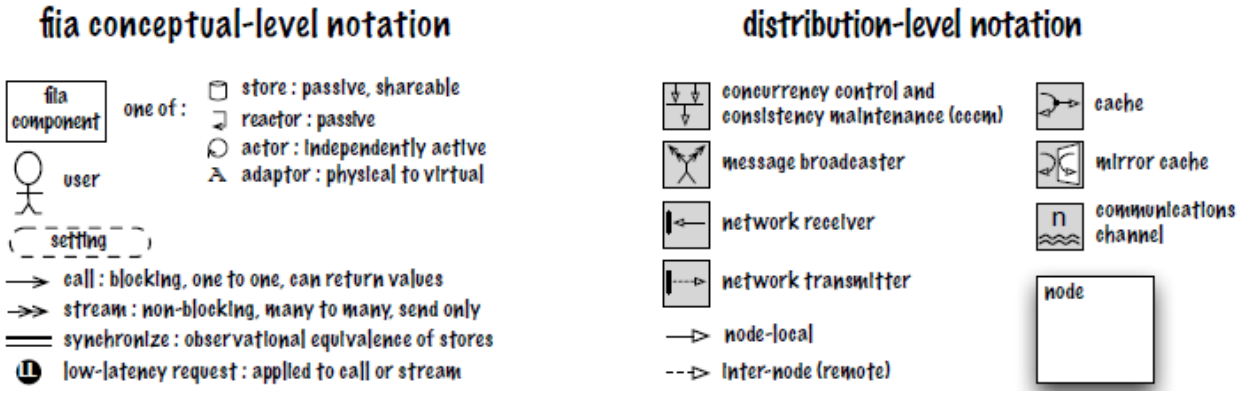


Εικόνα 20: Παράδειγμα τελικής συνεργατικής διεπαφής του TOUCHE

Το TOUCHE είναι μια από τις ελάχιστες εξαιρέσεις που υποστηρίζουν τη γέννηση της τελικής διεπαφής. Ωστόσο αυτό γίνεται μέσω της απευθείας διασύνδεσης αντικειμένων της αφηρημένης σημειολογίας που οικειοποιείται, με ένα συγκεκριμένο υποκείμενο groupware toolkit. Αυτό συνεπάγεται πως τα οφέλη υιοθέτησης της μοντελοκεντρικής μηχανικής υποβιβάζονται από πλευράς ενημερότητας-πλαισίου (context-awareness) στο επίπεδο των συνεργατικών εργαλειοθηκών.

2.2.3.4 Fiia

Το Fiia [14] είναι μια μοντελο-κεντρική user-centered προσέγγιση για το σχεδιασμό ετερογενών, προσαρμοστικών συνεργατικών συστημάτων σε πολύ υψηλό επίπεδο. Ο σχεδιαστής κάνει χρήση του Fiia notation (σημειογραφία) (βλ. Εικόνα 21, αριστερό μέρος) για να περιγράψει την αλληλεπίδραση και επικοινωνία των διάφορων μερών του συστήματος μεταξύ τους και να ορίσει το διαμοιραζόμενο μοντέλο. Χαρακτηριστικό γνώρισμα του Fiia αποτελεί το γεγονός πως προνοεί για συνεργατικής φύσης ζητήματα που αφορούν την κατανεμημένη επικοινωνία πληροφορίας. Αυτό γίνεται μέσω αφηρημένης σημειολογίας η οποία φαίνεται στο δεξί τμήμα της Εικόνα 21.



Εικόνα 21: The Fiia notation. Developers use the conceptual notation. The Fiia.Net toolkit generates implementations using the distribution notation.

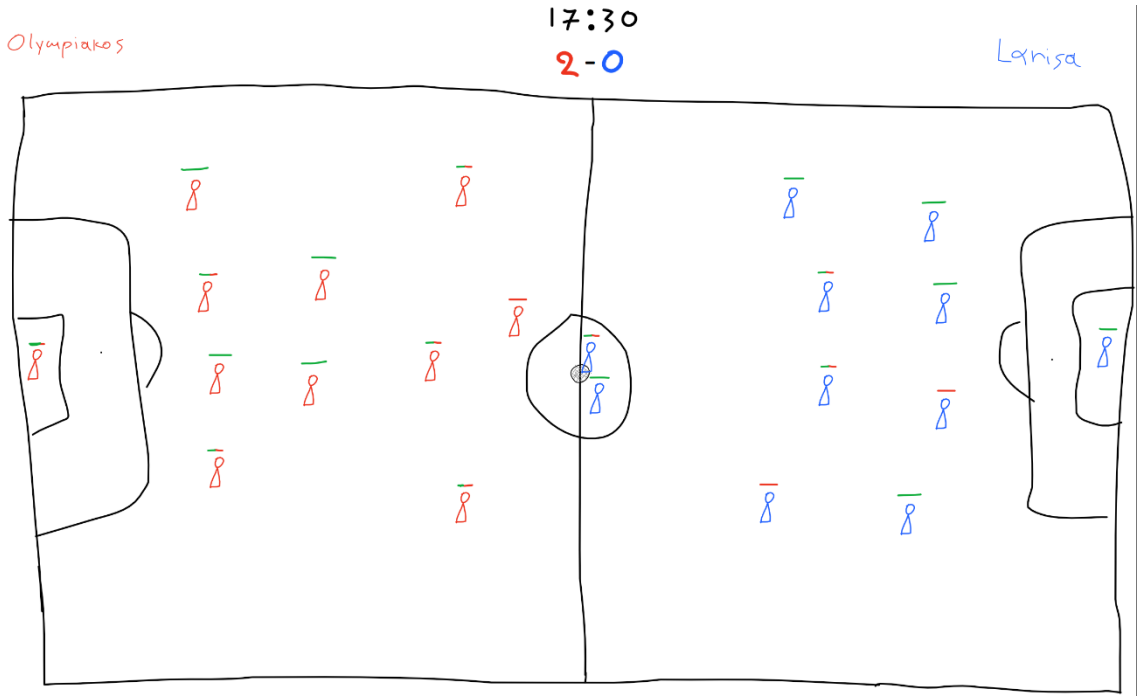
Οι υποστηριζόμενες προδιαγραφές μεταγλωττίζονται σε πηγαίο κώδικα αποτελούμενος από κλήσεις του API της .NET (C#) καθώς και ενός οικείου API που υλοποιεί ορισμένα αμιγώς συνεργατικά ζητήματα. Αυτή η εναπόθεση στη .Net και στην υποκείμενη οικεία συνεργατική 'βιβλιοθήκη' εγείρει τα ίδια ακριβώς σε φύση ζητήματα και περιορισμούς, δεδομένης της ανάγκης πανταχού παρούσα υποστήριξη, που εγείρουν όλες οι εργαλειοθηκο-κεντρικές προσεγγίσεις.

2.3 Αποτίμηση

Οι διαθέσιμες μέθοδοι για την ανάπτυξη σύγχρονου συνεργατικού λογισμικού κατά μήκος ετερογενών πλαισίων χρήσης παρουσιάζουν συγκεκριμένες αδυναμίες καθιστώντας τη χρήση τους αν μη τι άλλο προβληματική. Από τη μια πλευρά ο εργαλειοθηκο-κεντρικός προγραμματισμός, λόγω υποθέσεων σε σχέση με τις κλάσεις αντικειμένων και το διάλογο αποτρέπει τη χρήση μιας εργαλειοθήκης κατά μήκος ριζικά διαφορετικών πλαισίων χρήσης. Παραδείγματος χάριν, μια συνεργατική εργαλειοθήκη που προορίζεται για χρήση σε περιβάλλον σταθερού υπολογιστή είναι αδύνατο να εκτελεστεί στα πλαίσια μιας κινητής συσκευής ή ενός browser και το αντίστροφο, κοκ.. Απεναντίας, η μοντελο-κεντρική μηχανική διεπαφών προσφέρει ένα καλύτερο πλαίσιο για τη διαχείριση της ετερογένειας. Πλεονεκτήματα απορρέουν από την εστίαση στη χρήση αφηρημένης σημειολογίας και γλωσσών σήμανσης για την υποστήριξη του προσδιορισμού αφηρημένων αντικειμένων και της μετέπειτα διασύνδεσής τους με λεξικά στόχο. Για την απρόσκοπτη εκτέλεση των παραγόμενων προδιαγραφών τους ωστόσο επιλέγουν συνειδητά την υποστήριξη μόνο εγγενώς υποστηριζόμενων διαδραστικών αντικειμένων που απαντώνται κατά μήκος των πιο δημοφιλών εργαλειοθηκών. Ως εκ τούτου η εκφραστική τους επάρκεια περιορίζεται σε form-based διεπαφές. Επίσης, όσες αφορούν την υποστήριξη σύγχρονου συνεργατικού λογισμικού, δεν προνοούν ως προς τη γέννηση της τελικής διεπαφής. Όσες εξαιρέσεις (ελάχιστες) το κάνουν, το κάνουν διαμέσου direct κλήσεων σε ένα προκαθορισμένο groupware toolkit, καταλήγοντας να έχουν τις ίδιες αδυναμίες με τις αντίστοιχες εργαλειοθηκο-κεντρικές προσεγγίσεις.

Σενάριο χρήσης & προσέγγιση

Το σενάριο που θα προσεγγίσουμε αφορά ένα multi-platform multi-player σύγχρονο (synchronous) παίγνιο ποδοσφαίρου. Το γενικό σενάριο αναφοράς υποθέτει ότι οι χρήστες του παιγνίου μπορεί να εμπλακούν χρησιμοποιώντας διαφορετικές τερματικές συσκευές και πλατφόρμες. Παρόλα αυτά, η εστίαση της παρούσας πτυχιακής αφορά στην υποστήριξη χρηστών που δρουν διαμέσου ενός web-based setting.



Εικόνα 22: SoccerField mockup

3.1 Βασικές παραδοχές, απαιτήσεις και αντικείμενα πεδίου

Το παιχνίδι υιοθετεί τους συμβατικούς κανόνες του γνωστού παιγνίου. Επομένως, απαιτείται αναπαράσταση βασικών αντικειμένων πεδίου όπως συνοψίζονται παρακάτω.

3.1.1 Χώρος ποδοσφαίρου

Ο χώρος του ποδοσφαίρου ή αλλιώς ‘Field’, αφορά ένα domain-specific container που ενδείκνυται για τη φιλοξενία (hosting), των παιχτών και της μπάλας. Πιο συγκεκριμένα αφορά σε ένα γραφικό δισδιάστατο χώρο, που ως εκ τούτου διακρίνεται από πλάτος και ύψος, ο οποίος οριοθετεί το επιτρεπτό χωρικά εύρος κινήσεων των αντικειμένων που φιλοξενεί (παίχτες και μπάλα).

3.1.2 Παίκτες

Οι παίχτες αφορούν αντικείμενα τα οποία χειρίζονται οι εν δυνάμει χρήστες του παιγνίου, φιλοξενούμενα στα πλαίσια του ‘χώρου ποδοσφαίρου’, και τα οποία φέρουν συγκεκριμένα χαρακτηριστικά. Το χρώμα υποδεικνύει την πλευρά στην οποία ανήκουν (δηλ. ομάδα), ενώ φέρουν και αντοχή η οποία υποδεικνύεται μέσω ξεχωριστής γραφικής ‘μπάρας’ στα πλαίσια κάθε παίχτη. Κάθε παίχτης μπορεί να κινηθεί προς όλες τις κατευθύνσεις του καρτεσιανού επιπέδου αναλόγως της κατεύθυνσης που υποδεικνύει ο χρήστης μέσω της αλληλεπίδρασή του με τα φυσικά κουμπιά της συσκευής εισόδου του (πληκτρολογίου). Επιπλέον, των βασικών κινήσεων κάθε παίχτης δύναται, όπως είναι φυσικό, να σουτάρει ή να δώσει πάσα, προς την κατεύθυνση που επιθυμεί, ρυθμίζοντας τη δύναμη αυτών (των ενεργειών) βάσει του χρόνου που το ενδεδειγμένο πλήκτρο παραμένει πατημένο από το χρήστη.

3.1.3 Η μπάλα

Η μπάλα αφορά σε ένα γραφικό δισδιάστατο αντικείμενο (δηλ. πλάτος και ύψος), το οποίο μπορεί να λάβει συγκεκριμένη ταχύτητα και κατεύθυνση κίνησης. Αυτή μπορεί να ταυτίζεται με την κίνηση του παίχτη που την έχει στην κατοχή του, ή με την κατεύθυνση και ταχύτητα της πάσας ή σουτ που έχει καθορίσει.

3.1.4 Ρόλοι χρηστών

Επιπλέον αυτών, οι χρήστες δύναται να εμπλακούν είτε ως ενεργά “μέλη” των δύο πλευρών είτε ως παρατηρητές. Στην τελευταία περίπτωση (δηλ. ρόλος τύπου observer) η διαφοροποίηση σε σχέση με των υπολοίπων είναι πως οι τοπικοί χρήστες δεν δύναται να τροποποιήσουν κατ’ οποιονδήποτε τρόπο τη διαμοιραζόμενη κατάσταση του παιγνίου. Σε κάθε περίπτωση οι κατανεμημένοι χωρικά χρήστες πρέπει να είναι ενήμεροι της τρέχουσας διαμοιραζόμενης κατάστασης του παιγνίου.

3.2 Προσέγγιση

Η προσέγγιση ικανοποίησης των βασικών απαιτήσεων που αναφέρθηκαν στο προηγούμενο κεφάλαιο εδράζεται στην αξιοποίηση καινοτόμων δομών στη βάση της μοντελοκεντρικής μηχανικής διεπαφών. Πλεονεκτήματα απορρέουν από τη χρήση μοντέλων τα οποία εστιάζουν σε διαφορετικά επίπεδα αφαίρεσης κατά τη φάση της συνολικής διαδικασίας σχεδίασης του αλληλεπιδραστικού συστήματος και κυρίως της διεπαφής. Πιο συγκεκριμένα βασίζεται σε μια

επέκταση της μοντελοκεντρικής γλώσσας UsiXML, με στόχο α) την υποστήριξη πανταχού παρόντων προδιαγραφών διεπαφών που συναθροίζουν εκτός των συμβατικών, καινοφανή, μη εγγενώς υποστηριζόμενα από τις δημοφιλείς εργαλειοθήκες διαδραστικά αντικείμενα, καθώς επίσης β) την υποστήριξη σύγχρονων συνεργατικών σχημάτων συνεργασίας κατά μήκος ριζικά ετερογενών πλαισίων χρήσης. Η εν λόγω επεκτάσεις έχουν τεκμηριωθεί σε σειρά άρθρων συμπεριλαμβανομένων των [10][11][12].

Τα σημαντικότερα μοντέλα της γλώσσας, όντας και αυτά που αξιοποιήθηκαν στην εν λόγω εργασία διαχωρίζονται ως ακολούθως.

3.2.1 Προδιαγραφές ανεξαρτήτου Υλοποίησης (Implementation-agnostic Instantiation Schemes)

Η υποστήριξη οικείων και μη-εγγενώς υποστηριζόμενων αντικειμένων βασίστηκε σε μια σειρά εξειδικευμένων μοντέλων: τα Widget Specification Language (WSL), Concrete User Interface (CUI), και το Widget Resource [17][18]. Αναλυτικότερα, το Widget Specification Language (WSL) [12], δίνει τη δυνατότητα κωδικοποίησης του API των υποστηριζόμενων εναλλακτικών υπολογιστικών αναπαραστάσεων ενός ‘αφηρημένου’ αντικειμένου ανά πλατφόρμα. Με τον όρο ‘αφηρημένο’ αποδίδεται ο ρητός διαχωρισμός του ρόλου ενός αντικειμένου από τις εναλλακτικές αναπαραστάσεις του ανά πλατφόρμα στόχο ή/και προφίλ χρήστη. Το CUI μοντέλο, με τη σειρά του, περιγράφει την ιεραρχική αποσύνθεση της πολυμορφικής διεπαφής με την εκφραστική επάρκεια να περιορίζεται στο σύνολο των διαθέσιμων αφηρημένων αντικειμένων που έχουν κωδικοποιηθεί διαμέσου του WSL. Η περιγραφή αυτή είναι ανεξάρτητη πλατφόρμας και υπολογιστικής συσκευής. Στο Widget Resource μοντέλο γίνονται αναφορές στα αφηρημένα αντικείμενα του CUI μοντέλου και προσδιορίζονται τιμές (pre-instantiation configuration) στα πλαίσια της αρχικής συναρμολόγησης-προδιαγραφής της διεπαφής. Τέλος, το Behaviour μοντέλο, μοντελοποιεί τις δυναμικές πτυχές της διεπαφής υπό τη μορφή Finite State Machines.

3.2.2 Μοντέλα για την υποστήριξη σχημάτων σύγχρονης συνεργασίας

Τα μοντέλα τα οποία επινοήθηκαν και εισήχθησαν ως επεκτάσεις της UsiXML για την υποστήριξη σύγχρονων συνεργατικών σχημάτων συνεργασίας κατά μήκος ριζικά ετερογενών πλαισίων χρήσης, είναι τα: Abstraction, Consistency, και Session [17]. Κύρια λειτουργία του Abstraction μοντέλου είναι η υποστήριξη του διαμοιρασμού των δεδομένων μεταξύ των

κατανεμημένων και μη διεπαφών. Συγκεκριμένα, ο σχεδιαστής αποφασίζει ποια είναι τα κοινά χαρακτηριστικά ανάμεσα στις διεπαφές προς συγχρονισμό και ορίζει abstraction objects τα οποία τις συγχρονίζουν. Στο Consistency μοντέλο ορίζονται συσχετίσεις ανάμεσα σε συγκεκριμένα τμήματα μιας διεπαφής και συγκεκριμένων αφηρημένων ιδιοτήτων του Abstraction μοντέλου. Με το που πραγματοποιηθεί μια αλλαγή σε μια αφηρημένη ιδιότητα (του διαμοιραζόμενου μοντέλου) λόγω της αλληλεπίδρασης κάποιου απομακρυσμένου χρήστη, αυτομάτως το σύστημα ενημερώνει (feedthrough) τα τμήματα της διεπαφής (τοπικά) που τις αντιστοιχούν. Τέλος, το Session μοντέλο αφορά τον ορισμό σύγχρονων συνόδων και τον κύκλος ζωής τους.

3.2.3 Περιβάλλοντα Εκτέλεσης

Για τη διερμήνευση και γενικώς την εν εκτελέσει αξιοποίηση της πληροφορίας που κωδικοποιείται διαμέσου των παραπάνω μοντέλων, κρίνεται αναγκαία η ύπαρξη (υποστήριξη) γενικευμένων δομών εκτέλεσης. Γενικευμένων υπό την έννοια ότι αυτούσια χωρίς αλλαγές στον κώδικα τους μπορεί να υποστηρίξουν όλο το εύρος και φάσμα στιγμιότυπων των παραπάνω μοντέλων.

3.2.3.1 Platform Server

Ο Platform Server είναι ένα πολύ-λειτουργικό λογισμικό το οποίο τρέχει στην πλευρά του client, και αναλαμβάνει σειρά καθηκόντων. Αυτά είναι: α) φόρτωση του specification (μοντέλα) της διεπαφής που σχετίζεται με το collaboration session στο οποίο επιθυμεί να εμπλακεί ο χρήστης, β) διερμήνευση της αφηρημένης διεπαφής και σε συνδυασμό με πληροφορία που αφορά το τρέχων context αποφασίζει ποια widgets πρέπει να επιστρατευθούν, στα πλαίσια της διαδικασίας πολυμορφικής στιγμιοτυποποίησης (polymorphic instantiation process), τα οποία εν συνεχεία φορτώνει (μαζί με τις εξαρτήσεις τους) από το widget archives repository, γ) εν εκτελέσει δημιουργία και διαχείριση των widgets, δ) διερμήνευση, δημιουργία (με βάση το προδιαγεγραμμένο state τους) και επισύναψη των FSM αντικειμένων στα αντίστοιχα widgets, ε) εν εκτελέσει διαχείριση της πολυμορφικής διεπαφής, ζ) δημιουργία και διαχείριση του κοινόχρηστου μοντέλου (abstraction model), στα πλαίσια της οποίας (διαχείρισης) διατηρεί το τοπικό αντίγραφο (δηλ. replica) του Abstraction μοντέλου συγχρονισμένο (synchronized) ανά πάσα χρονική στιγμή, καθώς επίσης διερμηνεύει τα σχετικά μοντέλα (Consistency, Widget

Resource και Abstraction) και ενημερώνει την τοπική διεπαφή σε περίπτωση αλλαγής του διαμοιραζόμενου μοντέλου (Feedthrough) από κάποιον απομακρυσμένο χρήστη

3.2.3.2 Server Side Framework

Το Server Side Framework (SSF) διαχειρίζεται συνόδους, υλοποιεί ένα μηχανισμό ειδοποιήσεων (λήψη και προώθηση των αλλαγών που έχουν πραγματοποιηθεί στα διαμοιραζόμενα αντικείμενα) και διατηρεί ένα κοινόχρηστο repository με όλα τα διαθέσιμα widget archives.

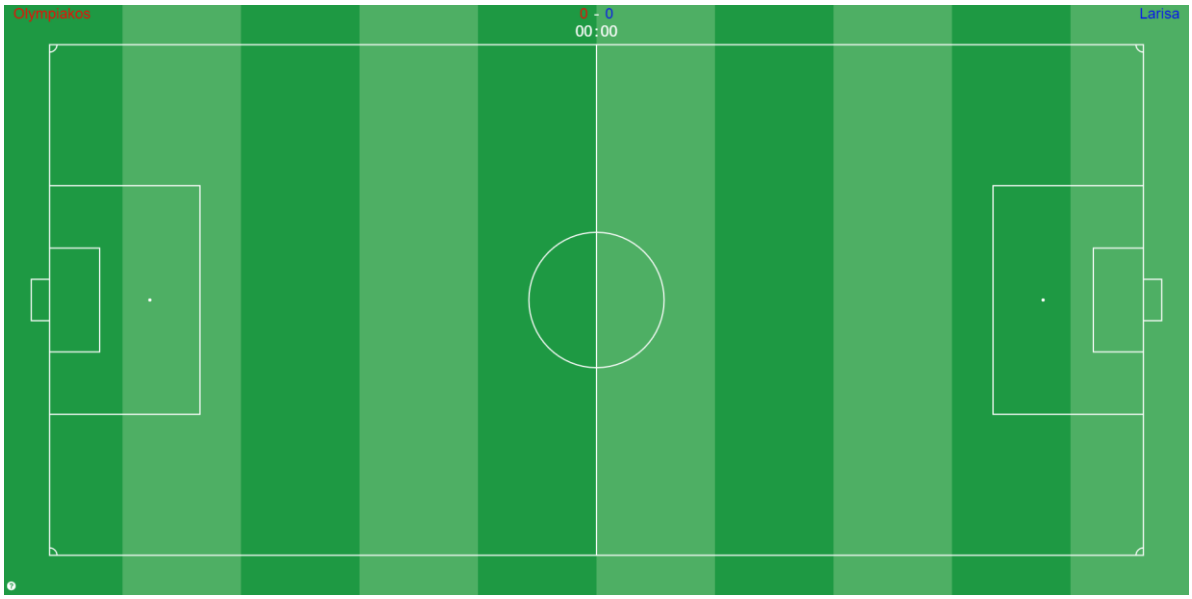
Υλοποίηση του σεναρίου για το web

Για την υλοποίηση του σεναρίου χρήσης του ποδοσφαίρου στο web, έπρεπε κατ' αρχάς (δεδομένου ότι δεν υπάρχουν διαθέσιμες) να υλοποιηθούν μέσω χαμηλού επιπέδου εντατικό προγραμματισμό οι υλοποιήσεις για την web πλατφόρμα που αφορούν τα συστατικά του γηπέδου (βλ. Γήπεδο, Παίκτης, Μπάλα). Αυτό έγινε με χρήση αμιγώς web-oriented τεχνολογιών (HTML5, JavaScript και CSS). Ακολούθως έγινε εξ αρχής ανάπτυξη του κατάλληλου platform server έτσι ώστε να υποστηριχθεί η διαδραστική απόδοση του παιγνίου.

4.1 Αλληλεπιδραστικά Αντικείμενα Ποδοσφαίρου

Αν και μιλάμε για σύγχρονο κατανεμημένο συνεργατικό λογισμικό, στην συγκεκριμένη περίπτωση έγινε μια συμβατική υλοποίηση των αλληλεπιδραστικών αντικειμένων του ποδοσφαίρου. Με τον όρο "συμβατική" εννοούμε ότι δεν έχει γίνει χρήση κώδικα που έχει να κάνει με συνεργατικότητα (collaboration), επομένως τα ίδια αντικείμενα θα μπορούσαν να χρησιμοποιηθούν για να αναπτυχθεί ένα συμβατικό multiplayer παιχνίδι ποδοσφαίρου όπου θα παιζόταν από δύο παίκτες ταυτόχρονα στο ίδιο μηχάνημα.

Το Γήπεδο (βλ. Εικόνα 23) έχει υλοποιηθεί με δύο Canvas, στοιβαγμένα το ένα πάνω στο άλλο. Στο κατώτερο, ζωγραφίζεται ο αγωνιστικός χώρος: χόρτο και γραμμές, ενώ στο ανώτερο ζωγραφίζονται οι παίκτες και οι μπάλα.



Εικόνα 23: Αναπαράσταση Γηπέδου

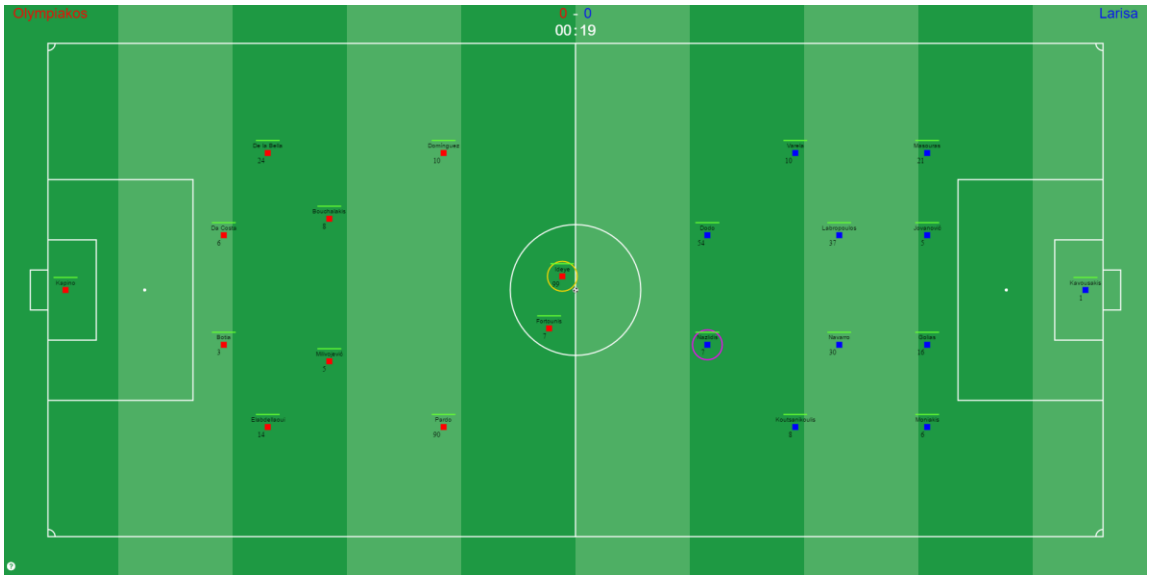
Από τη στιγμή που χρησιμοποιείται το Canvas, το παιχνίδι ακολουθεί μια Animation λογική ("animation loop" – clear, update, draw) για να ζωγραφίζονται τα διάφορα αντικείμενα. Ο παραπάνω διαχωρισμός (σε δύο Canvas) πραγματοποιήθηκε για να εξοικονομηθεί επεξεργαστική ισχύ, αφού ο αγωνιστικός χώρος δεν έχει νόημα να ζωγραφίζεται ξανά και ξανά, παρά μόνο όταν αλλάξει το μέγεθος του browser, ενώ οι παίκτες και οι μπάλα κινούνται συνεχώς, άρα το animation loop πρέπει να ενεργοποιείται πιο συχνά. Στο animation loop του Canvas των παικτών και της μπάλας, εσωκλείεται και ο κύριος αλγόριθμος του παιχνιδιού που καθορίζει πότε κάποιος παίκτης παίρνει στην κατοχή του την μπάλα, πότε μπαίνει γκολ, πότε βγαίνει η μπάλα εκτός ορίων, η επιτάχυνση και επιβράδυνση της μπάλας, κτλ καθώς και τις οριακές συνθήκες που σχετίζονται με το γκολ (όπου η μπάλα επιστρέφει αυτόματα στο κέντρο του γηπέδου και το παιχνίδι συνεχίζεται) ή την περίπτωση που η μπάλα βγει εκτός ορίων (οπότε και σταματάει στο σημείο που είναι και δεν μπορεί να μετακινηθεί ενώ ο χρήστης πρέπει να πιέσει το πλήκτρο "R" (recover) για να επαναφέρει την μπάλα στο κέντρο του γηπέδου και να συνεχιστεί ο αγώνας). Επιπρόσθετα, έχουν γίνει attach και διάφοροι listeners που υλοποιούν τα controls του παιχνιδιού. Τέλος, χρησιμοποιούνται μερικά επικαλύπτοντα Div που παρουσιάζουν πληροφορίες όπως τα ονόματα των ομάδων, το σκορ και το τρέχον λεπτό του αγώνα.

Ο Παίκτης αναπαρίσταται ως ένα τετράγωνο (βλ. Εικόνα 24), στο χρώμα της ομάδας που αντιστοιχεί.. Μέσω του δημόσιου API του, μπορεί να παραμετροποιηθεί ως προς το όνομα, τον αριθμό, την ταχύτητα, την αντοχή, κ.α. Καθώς κινείται κάποιος παίκτης, η αντοχή του μειώνεται. Ο χρήστης επιδέχεται δυνατότητας γραφικού κατοπτρισμού της κατάστασης της αντοχής του. Αυτό υποστηρίζεται μέσω μιας μπάρας που βρίσκεται ακριβώς πάνω από κάθε παίκτη. Η εν λόγω δυνατότητα μπορεί να (απ-) ενεργοποιείται κατά το δοκούν.

Εικόνα 24: Αναπαράσταση Παίκτη

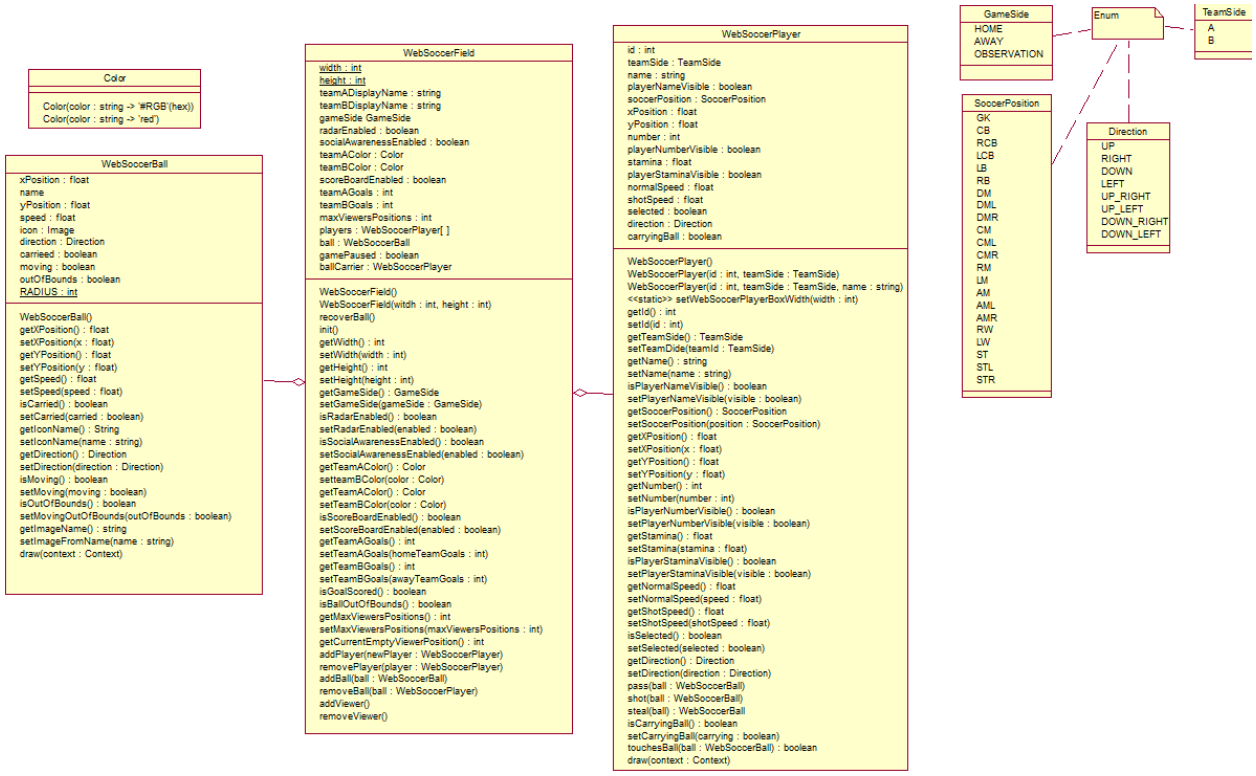
Εικόνα 25: Αναπαράσταση Μπάλας

Τέλος, η Μπάλα αναπαρίσταται με το εικονίδιο που εμφανίζεται στην Εικόνα 25. Στην Εικόνα 26, παρουσιάζεται ενδεικτικά η εν εκτελέσει αποτύπωση του παιγνίου του ποδοσφαίρου.



Εικόνα 26: Εν εκτελέσει αποτύπωση του παιγνίου του ποδοσφαίρου

Στην Εικόνα 27 παρουσιάζεται το διάγραμμα κλάσεων που φανερώνει τεχνικές λεπτομέρειες της υλοποίησης του WebSoccer.



Εικόνα 27: Διάγραμμα κλάσεων Web Soccer

4.2 Ενσωμάτωση στο MBUI process

Αφού έχει ολοκληρωθεί η ανάπτυξη των αλληλεπιδραστικών αντικειμένων από τον προγραμματιστή, αυτά πρέπει να κωδικοποιηθούν καταλλήλως με την WSL (βλ. Εικόνα 28), να πακεταριστούν μαζί με τις εξαρτήσεις τους (π.χ. άλλα αρχεία js, css, εικόνες) σε ένα συμπιεσμένο αρχείο με κατάληξη "wdr" (π.χ. "AbstractSoccerPlayer.wdr") και να εναποθετηθούν σε ένα κοινό widget repository στο SSF.

```
<?xml version="1.0" encoding="UTF-8"?>
<widget id = "abstract_soccer_player" name = "AbstractSoccerPlayer" type="domainSpecificComponent">
    <display_name>Player Artifact</display_name>
    ...
    <abstract_properties>
        <property id="abs_prop_1" name="id" type="string" isPrimitive="true"/>
        <property id="abs_prop_2" name="name" type="string" isPrimitive="true"/>
        ...
    </abstract_properties>
    <instances>
        <instance id="abstract_player_inst_1" name="SwingSoccerPlayer" ...>
        <instance id="abstract_player_inst_2" name="AndroidSoccerPlayer" ...>
        <instance id="abstract_player_inst_3" name="NVTSoccerPlayer" ...>
        <instance id="abstract_player_inst_4" name="WebSoccerPlayer" ...>
            <display_name>Web Soccer Player</display_name>
            <dependencies>
                <software_library ver="1.0" url="/AbstractSoccerPlayer.wdr/dependencies/WebSoccerPlayer/Direction.js"/>
                <software_library ver="1.0" url="/AbstractSoccerPlayer.wdr/dependencies/WebSoccerPlayer/WebSoccerPlayer.js"/>
                <software_library ver="1.0" url="/AbstractSoccerPlayer.wdr/dependencies/WebSoccerPlayer/WebSoccerPlayerFSM.js"/>
                ...
            </dependencies>
            ...
            <polymorphic_properties>
                <property id="inst_4_prop_1" name="number" type="int" isPrimitive="true"/>
                <property id="inst_4_prop_2" name="playerNameVisible" type="boolean" isPrimitive="true"/>
                ...
            </polymorphic_properties>
            ...
            <mappings>
                <mapping id="inst_4_mapping_1">
                    <mappingProperty id="abs_prop_1" isMandatory="false">
                        <utilized in="method" asType="argument" classId="inst_4_cl_1" partId="inst_4_cl_1_method_1"/>
                        <utilized in="method" asType="return" classId="inst_4_cl_1" partId="inst_4_cl_1_method_2"/>
                    </mappingProperty>
                </mapping>
                ...
                <mapping id="inst_4_mapping_9">
                    <mappingProperty id="inst_4_prop_1" isMandatory="false">
                        <utilized in="method" asType="argument" classId="inst_4_cl_1" partId="inst_4_cl_1_method_17"/>
                        <utilized in="method" asType="return" classId="inst_4_cl_1" partId="inst_4_cl_1_method_18"/>
                    </mappingProperty>
                </mapping>
                ...
            </mappings>
            <instance_api>
                <class id="inst_4_cl_1" name="WebSoccerPlayer" path="WebSoccerPlayer" isNative="false">
                    <constructor id="inst_4_cl_1_constructor_1" isDefault="true"/>
                    <method id="inst_4_cl_1_method_1" name="setId" type="mutator" returnType="void">
                        <argument name="id" type="int"/>
                    </method>
                    <method id="inst_4_cl_1_method_2" name="getId" type="accessor" returnType="int"/>
                    ...
                    <method id="inst_4_cl_1_method_17" name="setNumber" type="mutator" returnType="void">
                        <argument name="number" type="int"/>
                    </method>
                    <method id="inst_4_cl_1_method_18" name="getNumber" type="accessor" returnType="int"/>
                    ...
                </class>
                ...
            </instance_api>
        </instance>
    </instances>
</widget>
```

Εικόνα 28: Τμήμα WSL "AbstractSoccerPlayer"

Έπειτα, σχεδιάζεται η πολυμορφική διεπαφή ως μια ιεραρχία αφηρημένων CUI στοιχείων (βλ. Εικόνα 29), το εύρος και οι ιδιότητες των οποίων αντλούνται από το σύνολο των διαθέσιμων widgets που βρίσκονται στο κοινόχρηστο repository, και δημιουργείται το WidgetResource μοντέλο (βλ. Εικόνα 30) μέσω του οποίου ορίζονται αρχικές τιμές για τα πολυμορφικά αντικείμενα.

```
<cuiModel name="cui-model">
    ...
    <AbstractContainer id="abstract_container_1" widgetId="abstract_container" >
        <AbstractContainer id="abstract_container_2" widgetId="abstract_container" >
            <AbstractContainer id="abstract_container_3" widgetId="abstract_container" >
                <AbstractSoccerField id="abstract_soccer_field_1" widgetId="abstract_soccer_field">
                    <AbstractSoccerBall id="abstract_soccer_ball_1" widgetId="abstract_soccer_ball"></AbstractSoccerBall>
                    <AbstractSoccerPlayer id="abstract_soccer_player_1" widgetId="abstract_soccer_player"></AbstractSoccerPlayer>
                    <AbstractSoccerPlayer id="abstract_soccer_player_2" widgetId="abstract_soccer_player"></AbstractSoccerPlayer>
                    <AbstractSoccerPlayer id="abstract_soccer_player_3" widgetId="abstract_soccer_player"></AbstractSoccerPlayer>
                    ...
                </AbstractSoccerField>
            </AbstractContainer>
            <AbstractContainer id="abstract_container_4" widgetId="abstract_container" >
                <AbstractSoccerLivescore id="abstract_soccer_livescore_1" widgetId="abstract_soccer_livescore"/>
            </AbstractContainer>
        </AbstractContainer>
    </AbstractContainer>
</cuiModel>
```

Εικόνα 29: Τμήμα CUI μοντέλου

```
<widgetResourceModel>
    ...
    <widgetResource id="wdg_rsc_3" cioId="abstract_soccer_field_1" widgetId="abstract_soccer_field">
        <property id="wr_3_abs_prop_1" wslPropId="abs_prop_1" value="120" />
        <property id="wr_3_abs_prop_2" wslPropId="abs_prop_2" value="90"/>
        <property id="wr_3_abs_prop_3" wslPropId="abs_prop_3" value=""/>
        <instance id="wr_3_inst_1" wslInstId="abstract_soccer_field_inst_1" name="SwingSoccerField">
            <property id="wr_3_inst_1_prop_3" wslPropId="inst_1_prop_3" value="true" />
            <property id="wr_3_inst_1_prop_4" wslPropId="inst_1_prop_4" value="true"/>
        </instance>
        <instance id="wr_3_inst_2" wslInstId="abstract_soccer_field_inst_2" name="AndroidSoccerField">
            <property id="wr_3_inst_2_prop_4" wslPropId="inst_2_prop_4" value="true" />
            <property id="wr_3_inst_2_prop_5" wslPropId="inst_2_prop_5" value="true"/>
        </instance>
        <instance id="wr_3_inst_4" wslInstId="abstract_soccer_field_inst_4" name="WebSoccerField">
            ...
            <property id="wr_3_inst_4_prop_3" wslPropId="inst_4_prop_3" name="teamAColor">
                <property id="wr_3_inst_4_prop_4" wslPropId="inst_4_prop_4" name="colorString" value="#FF0000"/>
            </property>
            <property id="wr_3_inst_4_prop_5" wslPropId="inst_4_prop_5" name="teamBColor">
                <property id="wr_3_inst_4_prop_6" wslPropId="inst_4_prop_6" name="colorString" value="#0000FF"/>
            </property>
            <property id="wr_3_inst_4_prop_7" wslPropId="inst_4_prop_7" name="scoreBoardEnabled" value="true"/>
            <property id="wr_3_inst_4_prop_9" wslPropId="inst_4_prop_9" name="teamAGoals" value="0"/>
            <property id="wr_3_inst_4_prop_10" wslPropId="inst_4_prop_10" name="teamBGoals" value="0"/>
            ...
        </instance>
    </widgetResource>
    <widgetResource id="wdg_rsc_4" cioId="abstract_soccer_ball_1" widgetId="abstract_soccer_ball">
        <property id="wr_4_abs_prop_1" wslPropId="abs_prop_1" name="xPosition" value="60"/>
        <property id="wr_4_abs_prop_2" wslPropId="abs_prop_2" name="yPosition" value="45"/>
        <instance id="wr_4_inst_1" wslInstId="abstract_soccer_ball_inst_1" name="SwingSoccerBall">
        <instance id="wr_4_inst_2" wslInstId="abstract_soccer_ball_inst_2" name="AndroidSoccerBall">
        <instance id="wr_4_inst_4" wslInstId="abstract_soccer_ball_inst_4" name="WebSoccerBall">
            <property id="wr_4_inst_4_prop_3" wslPropId="inst_4_prop_1" name="imageName" value="ball_10x10.png"/>
            <property id="wr_4_inst_4_prop_4" wslPropId="inst_4_prop_2" name="direction" value="1"/>
            <property id="wr_4_inst_4_prop_5" wslPropId="inst_4_prop_3" name="carried" value="false"/>
            <property id="wr_4_inst_4_prop_6" wslPropId="inst_4_prop_4" name="outOfBounds" value="false"/>
        </instance>
        <instance id="wr_4_inst_5" wslInstId="abstract_soccer_ball_inst_5" name="WebSimpleSoccerBall">
            <property id="wr_4_inst_5_prop_4" wslPropId="inst_5_prop_2" name="direction" value="1"/>
            <property id="wr_4_inst_5_prop_5" wslPropId="inst_5_prop_3" name="carried" value="false"/>
            <property id="wr_4_inst_5_prop_6" wslPropId="inst_5_prop_4" name="outOfBounds" value="false"/>
        </instance>
    </widgetResource>
    <widgetResource id="wdg_rsc_5" cioId="abstract_soccer_player_1" widgetId="abstract_soccer_player">
        <property id="wr_5_abs_prop_1" wslPropId="abs_prop_1" value="1" />
        <property id="wr_5_abs_prop_2" wslPropId="abs_prop_2" value="Roberto"/>
        <property id="wr_5_abs_prop_3" wslPropId="abs_prop_3" value="1"/>
        ...
        <instance id="wr_5_inst_1" wslInstId="abstract_player_inst_1" name="SwingSoccerPlayer"/>
        <instance id="wr_5_inst_2" wslInstId="abstract_player_inst_2" name="AndroidSoccerPlayer"/>
        <instance id="wr_5_inst_4" wslInstId="abstract_player_inst_4" name="WebSoccerPlayer">
            <property id="wr_5_inst_4_prop_1" wslPropId="inst_4_prop_1" name="number" value="16"/>
            <property id="wr_5_inst_4_prop_2" wslPropId="inst_4_prop_2" name="playerNameVisible" value="true"/>
            <property id="wr_5_inst_4_prop_3" wslPropId="inst_4_prop_3" name="playerStaminaVisible" value="true"/>
            ...
        </instance>
    </widgetResource>
    ...
</widgetResourceModel>
```

Εικόνα 30: Τμήμα Widget Resource μοντέλου

Ακολούθως, περιγράφεται το Abstraction μοντέλο, αφού πρώτα ο σχεδιαστής έχει επιλέξει τα κοινά semantics με βάση τα οποία επιθυμείται ο συγχρονισμός μεταξύ των (ριζικά ετερογενών ή όχι) διαδραστικών αντικειμένων. Στην Εικόνα 31 απεικονίζεται τμήμα του Abstraction μοντέλου στο οποίο φαίνεται ένα διαμοιραζόμενο αντικείμενο της κλάσης "SoccerPlayer".

```
<abstractionModel>
  ...
  <abstractionClass id="abs_class_3" name="SoccerPlayer" isDistributed="true" syncIntervalMillis="0">
    <abstractionProperty id="abs_3_property_1" name="id" dataType="int" groupAware="false" mutatorId="abs_3_method_1" accessorId="abs_3_method_2"/>
    <abstractionProperty id="abs_3_property_2" name="xPosition" dataType="float" groupAware="false" mutatorId="abs_3_method_3" accessorId="abs_3_method_4"/>
    <abstractionProperty id="abs_3_property_3" name="yPosition" dataType="float" groupAware="false" mutatorId="abs_3_method_5" accessorId="abs_3_method_6"/>
    ...
    <abstractionMethod id="abs_3_method_1" name="setId" returnType="void" type="mutator">
      <abstractionArgument id="abs_3_method_1_arg_1" name="id" dataType="int"/>
    </abstractionMethod>
    <abstractionMethod id="abs_3_method_2" name="getId" returnType="int" type="accessor"/>
    <abstractionMethod id="abs_3_method_3" name="setXPosition" returnType="void" type="mutator">
      <abstractionArgument id="abs_3_method_3_arg_1" name="xPosition" dataType="int"/>
    </abstractionMethod>
    <abstractionMethod id="abs_3_method_4" name="getXPosition" returnType="int" type="accessor"/>
    <abstractionMethod id="abs_3_method_5" name="setYPosition" returnType="void" type="mutator">
      <abstractionArgument id="abs_3_method_5_arg_1" name="yPosition" dataType="int"/>
    </abstractionMethod>
    <abstractionMethod id="abs_3_method_6" name="getYPosition" returnType="int" type="accessor"/>
    <abstractionMethod id="abs_3_method_7" name="setStamina" returnType="void" type="mutator">
      <abstractionArgument id="abs_3_method_7_arg_1" name="stamina" dataType="int"/>
    </abstractionMethod>
    ...
  </abstractionClass>
  ...
  <abstractionObject id="abs_soccerplayer_1" name="SoccerPlayer_Instance" replicaId="10002" isDistributed="true" syncIntervalMillis="0">
    <!-- Properties Instances -->
    <abstractionPropertyObject id="abs_soccerplayer_1_prop_1" name="id">
      <enumeratedValue value="" />
    </abstractionPropertyObject>
    <abstractionPropertyObject id="abs_soccerplayer_1_prop_2" name="xPosition">
      <enumeratedValue value="" />
    </abstractionPropertyObject>
    <abstractionPropertyObject id="abs_soccerplayer_1_prop_3" name="yPosition">
      <enumeratedValue value="" />
    </abstractionPropertyObject>
    ...
  </abstractionObject>
  ...
  <instantiation id="abs_instantiation_3" name="abs_instantiation_3">
    <source sourceId="abs_soccerplayer_1" />
    <target targetId="abs_class_3" />
  </instantiation>
  ...
</abstractionModel>
```

Εικόνα 31: Τμήμα Abstraction μοντέλου

Εν συνεχεία, ορίζεται το Consistency μοντέλο (βλ. Εικόνα 32). Για κάθε μια από τις ιδιότητες των διαμοιραζόμενων αντικειμένων που έχουν δηλωθεί στο Abstraction μοντέλο, ορίζεται ποια τμήματα της διεπαφής θα επηρεάζονται ως αποτέλεσμα της αλλαγής τους από έναν απομακρυσμένο χρήστη.

```
<consistencyModel>
    ...
    <!-- SoccerPlayer (1) [@id] -->
    <consistencyBlock id="consistency_block_8" name="consistency_block_8">
      <consistentItem id="consistent_8_item_1" wrmPropertyId="wr_5_abs_prop_1" wrmInstanceId="" wrmId="wdg_rsc_5"/>
    </consistencyBlock>
    <!-- SoccerPlayer (1) [@xPosition] -->
    <consistencyBlock id="consistency_block_9" name="consistency_block_9">
      <consistentItem id="consistent_9_item_1" wrmPropertyId="wr_5_abs_prop_5" wrmInstanceId="" wrmId="wdg_rsc_5"/>
    </consistencyBlock>
    <!-- SoccerPlayer (1) [@yPosition] -->
    <consistencyBlock id="consistency_block_10" name="consistency_block_10">
      <consistentItem id="consistent_10_item_1" wrmPropertyId="wr_5_abs_prop_6" wrmInstanceId="" wrmId="wdg_rsc_5"/>
    </consistencyBlock>
    ...
</consistencyModel>
```

Εικόνα 32: Τμήμα Consistency μοντέλου

Τέλος, αφού έχουν πραγματοποιηθεί τα προηγούμενα βήματα, το project γίνεται deploy στο SSF.

4.3 Υλοποίηση Web Platform Server

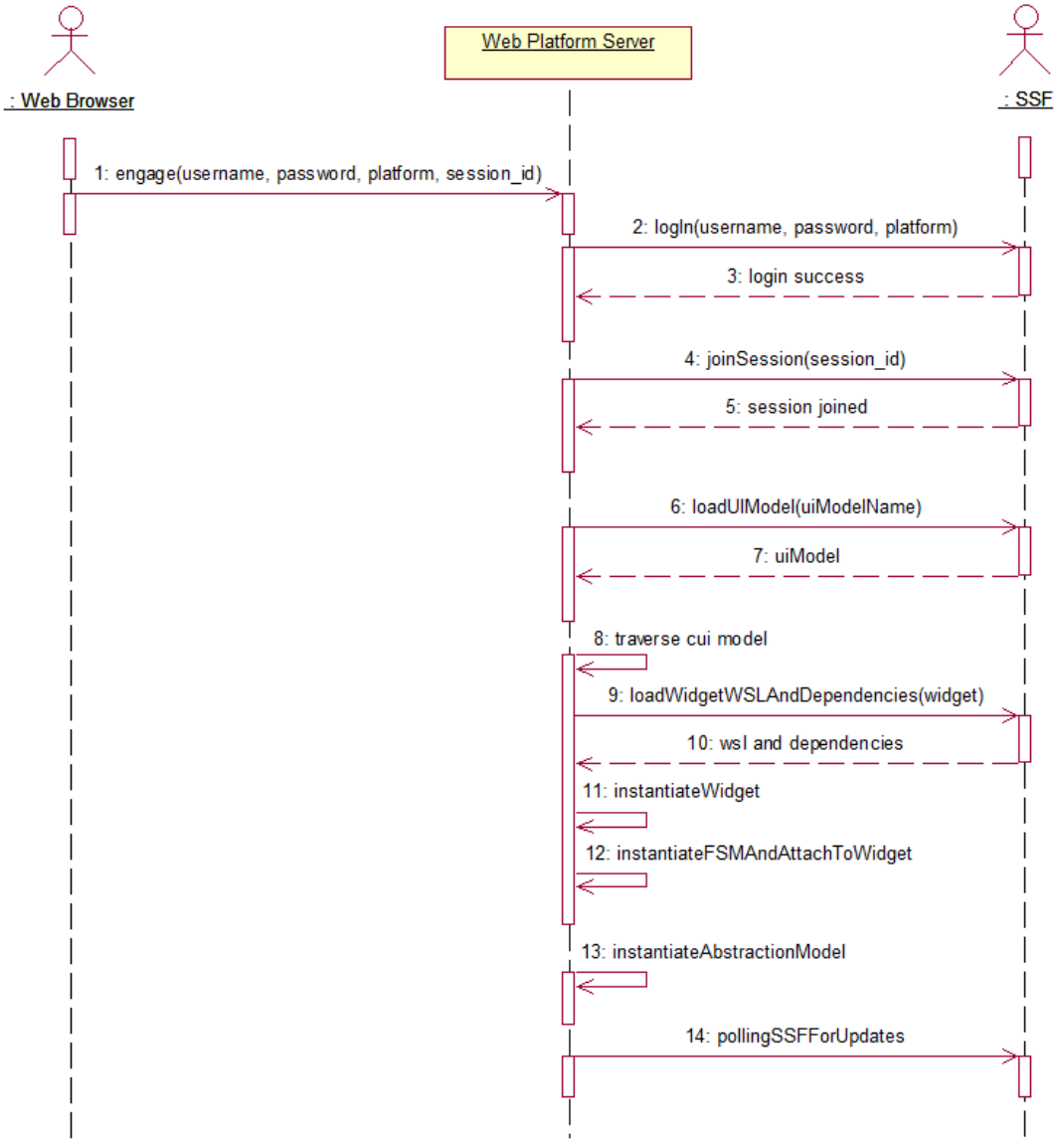
Στα πλαίσια αυτής της πτυχιακής, πέρα της υλοποίησης των διαδραστικών αντικειμένων του ποδοσφαίρου, υλοποιήθηκε και ένας Platform Server (PS), ενδεδειγμένος για χρήση στο Web. Δεδομένου ότι ο PS είναι λογισμικό το οποίο τρέχει στην πλευρά του client, δηλαδή στον browser, η υλοποίηση έγινε εξ ολοκλήρου στην γλώσσα προγραμματισμού JavaScript. Παρακάτω, δίδεται μια σύντομη περιγραφή της υλοποίησης αυτού.

Όλα τα μοντέλα (WSL, CUI, Abstraction, Consistency, κτλ.) που χρησιμοποιούνται στην προσέγγιση μας, καθώς και οι απαντήσεις που λαμβάνει ο PS από το SSF, είναι κωδικοποιημένα σε μορφή XML. Για τον λόγο αυτό, λήφθηκε η σχεδιαστική απόφαση να χρησιμοποιηθεί το XPath API, για την διερμήνευση τους. Μέσω του XPath API, μπορούμε να εκτελέσουμε σύνθετα ερωτήματα σε XML μοντέλα και να λάβουμε την επιθυμητή πληροφορία.

Αρχικά, ο PS ζητάει μέσω AJAX το specification της διεπαφής που σχετίζεται με το collaboration session στο οποίο θέλει να εμπλακεί ο χρήστης. Έπειτα, αφού πρώτα έχει αποφασιστεί ποιο context θα χρησιμοποιηθεί, ο PS διατρέχει αναδρομικά την περιγραφή της αφηρημένης διεπαφής (CUI μοντέλο), αποφασίζει ποια πολυμορφικά αντικείμενα θα επιστρατευθούν για την αρχική συναρμολόγηση της διεπαφής και ζητάει από το SSF (επίσης μέσω AJAX) τα WSL μοντέλα που περιγράφουν το δημόσιο API τους και τις εξαρτήσεις τους. Στη συνέχεια, αφού έχει γίνει γνωστό το API των αντικειμένων (όνομα κλάσης, ονόματα μεθόδων getters-setters), ο PS κάνοντας χρήση Reflection, δημιουργεί εν εκτελέσει τα αντικείμενα και τους προσδίδει αρχικές τιμές, βάση του Widget Resource μοντέλου. Μετέπειτα, διερμηνεύοντας το Behaviour μοντέλο, δημιουργεί (μέσω Reflection) για κάθε widget ένα FSM αντικείμενο και το επισυνάπτει πάνω του. Τα FSM αντικείμενα, σε επίπεδο κώδικα, έχουν γραφτεί βάση της βιβλιοθήκης "scxml.js"7, η οποία αποτελεί υλοποίηση της γλώσσας μηχανών καταστάσεων SCXML8. Έπειτα, δημιουργεί ένα τοπικό αντίγραφο του κοινόχρηστου μοντέλου, βάση του Abstraction μοντέλου όπου πρέπει να παραμένει συγχρονισμένο ανά πάσα στιγμή κατά την διάρκεια μιας συνεδρίας. Για να επιτευχθεί αυτό, όταν συμβεί μια αλλαγή σε ένα

[7] https://github.com/jbeard4/SCION
[8] https://www.w3.org/TR/scxml/

widget της τοπικής διεπαφής, αυτή εντοπίζεται από κάποιον listener του FSM αντικειμένου που του αντιστοιχεί και διαμέσου μιας ειδικής κλήσης, ενημερώνεται πρώτα το Abstraction μοντέλο και έπειτα το SSF με την πιο πρόσφατη τιμή. Στην περίπτωση που ο PS λάβει μια ενημέρωση (λόγω της αλληλεπίδρασης κάποιου εταίρου στην δική του τοπική διεπαφή) που αφορά το κοινόχρηστο μοντέλο, ενημερώνει άμεσα το τοπικό αντίγραφο του κοινόχρηστου μοντέλου και έπειτα, μέσω διασυνδέσεων που υπάρχουν ανάμεσα στα μοντέλα Consistency, Abstraction και Widget Resource, ενημερώνει την τοπική διεπαφή (feedthrough). Στην Εικόνα 33 παρουσιάζεται το διάγραμμα ακολουθίας που φανερώνει τη ροή βημάτων του συστήματος.

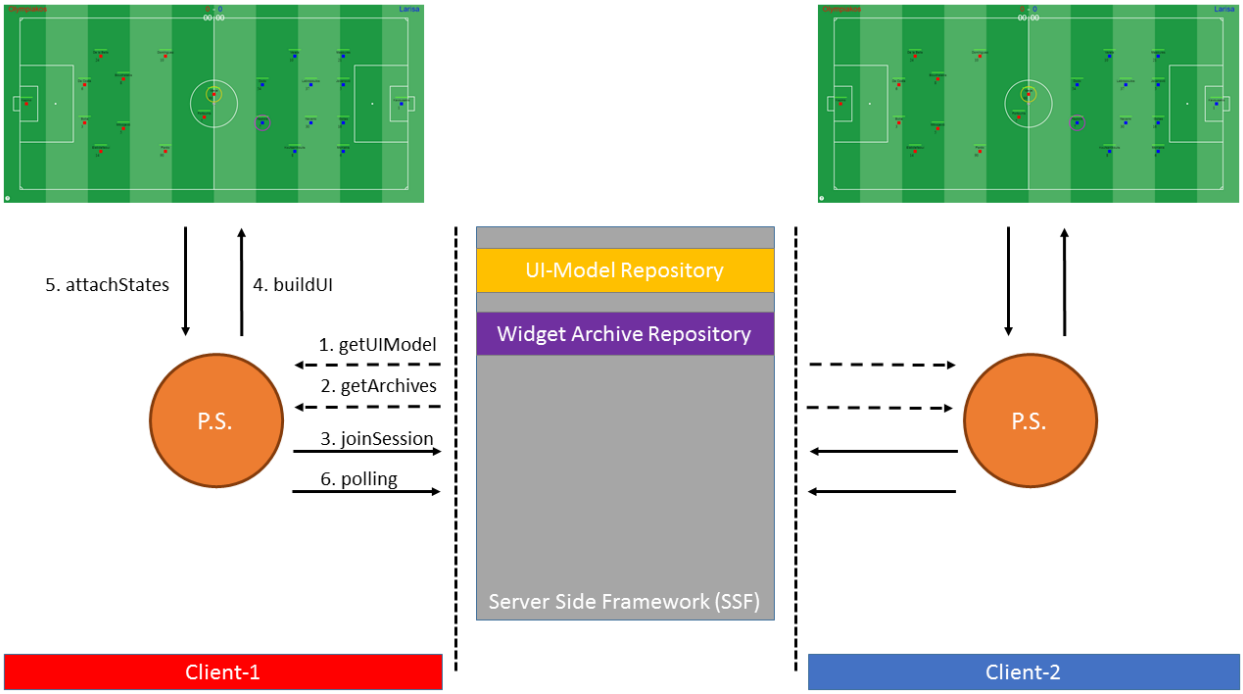


Εικόνα 33: Διάγραμμα ακολουθίας συστήματος

Να σημειωθεί, πως για να εντοπίσει ο PS αλλαγές του κοινόχρηστου μοντέλου που έχουν σταλθεί στο SSF, εκτελεί Ajax Polling κάθε 5ms. Τέλος, ο PS δύναται να μεταβεί, εν εκτελέσει, σε ένα διαφορετικό context. Επομένως, είναι σε θέση να συναρμολογεί και αποσυναρμολογεί, κατά το δοκούν, την εκάστοτε διεπαφή.

4.4 Περιγραφή της εν εκτελέσει ροής βημάτων για την υποστήριξη του σεναρίου χρήσης του soccer

Πλέον, το μόνο που μένει να κάνει ο χρήστης για να συμμετάσχει σε μια συνεδρία του παιγνίου, είναι να επισκεφθεί με τον browser ένα σύνδεσμο που λαμβάνει μέσω πρόσκλησης. Η σελίδα που επισκέπτεται ο χρήστης περιέχει όλα τα απαραίτητα JavaScript αρχεία και κλήσεις για να γίνει engage ο PS και να ξεκινήσει η συναρμολόγηση της διεπαφής. Στην Εικόνα 34 φαίνεται η αρχιτεκτονική του συστήματος.



Εικόνα 34: Αρχιτεκτονική συστήματος

Αρχικά, ο PS διασχίζει αναδρομικά το CUI μοντέλο και για κάθε widget που συναντάει, δημιουργεί τα διαφορετικά πολυμορφικά στιγμιότυπα που υποδεικνύει το Widget Resource μοντέλο, δίνοντας τους ταυτόχρονα αρχικές τιμές. Εν συνεχεία, δημιουργούνται τα FSM αντικείμενα βάση του Behaviour μοντέλου και συνδέονται με κάθε ένα διαδραστικό αντικείμενο. Έπειτα, δημιουργείται το διαμοιραζόμενο μοντέλο δεδομένων κατ' υπόδειξη του

Abstraction μοντέλου. Σε αυτό το σημείο, ο PS επιχειρεί την σύνδεση (log in) στην συνεδρία που έχει υποδείξει ο χρήστης προηγουμένως, χρησιμοποιώντας τα στοιχεία του (όνομα χρήστη, κωδικός χρήστη). Μετά την είσοδο του χρήστη στην συνεδρία, ο PS συμβουλεύεται το Mapping μοντέλο για να δημιουργήσει την αρχική διεπαφή. Τέλος, ξεκινάει μια διαδικασία συνεχούς παρακολούθησης της διεπαφής, ώστε αυτή να παραμένει συγχρονισμένη ανά πάσα στιγμή.

Κατακλείδα & συμπεράσματα

Σ' αυτό το κεφαλαίο θα συνοψίσουμε την συνολικότερη εμπειρία του συγγραφέα κατά τη διάρκεια υλοποίησης της συγκεκριμένης πτυχιακής εργασίας. Ειδικότερα, θα αναφερθούμε επιγραμματικά σε ad hoc αξιολόγηση της υλοποίησης καθώς και σε μελλοντικές επεκτάσεις που απορρέουν από την υπάρχουσα εφαρμογή.

5.1 Ποιοτική αξιολόγηση επιδόσεων

Κατά το στάδιο της πιλοτικής αξιολόγησης της εφαρμογής, εξετάστηκε το ερώτημα αν και κατά πόσο μπορούν οι παίχτες να εμπλακούν επιτυχώς και αποτελεσματικά στα πλαίσια του παιγνίου. Στόχος ήταν η αποτίμηση των σχεδιαστικών επιλογών σε επίπεδο Platform Server και η ικανότητα της υποστήριξης απαιτητικών διαδραστικών εφαρμογών όπως το παίγνιο. Για το λόγο αυτό, το πείραμα που οργανώθηκε αφορούσε σε πέντε συνολικά χρήστες, δύο (2) εκ των οποίων ήταν οι χρήστες των ομάδων και τρείς (3) παρατηρητές.

Για τις ανάγκες του πειράματος χρησιμοποιήθηκαν δύο (2) desktop μηχανήματα με επεξεργαστή τεσσάρων πυρήνων στα 3.4 GHz και 8 GB μνήμη RAM, δύο (2) notebooks με επεξεργαστή τεσσάρων πυρήνων στα 2.4 GHz και 8 GB μνήμη RAM, καθώς και ένα (1) Windows Tablet με επεξεργαστή Intel Atom τεσσάρων πυρήνων στα 1.4 GHz και 2 GB μνήμη RAM. Το πείραμα έτρεξε σε δίκτυο LAN και έγινε χρήση του Google Chrome browser.

Στην αρχική υλοποίηση παρατηρήθηκαν καθυστερήσεις οι οποίες οφείλονταν στο γεγονός ότι κάθε αφηρημένη (abstraction) κλάση επικοινωνούσε απευθείας με το SSF μέσω ατομικών καναλιών επικοινωνίας. Η προσέγγιση αυτή εισήγαγε σημαντικές καθυστερήσεις στο σύστημα, οι οποίες καθιστούσαν προβληματική την αλληλεπίδραση του χρήστη με την εφαρμογή. Προκειμένου να αντιμετωπιστεί το πρόβλημα ακολουθήθηκε μια νέα προσέγγιση η οποία δρομολογούσε όλη την επικοινωνία όλων των abstraction κλάσεων με το SSF διαμέσου ενός ενιαίου serial bus. Αυτομάτως, το σύστημα λειτούργησε αποδοτικότερα. Αν και υπήρχαν ελάχιστες καθυστερήσεις, αυτές δεν γίνονταν αντιληπτές από τους τελικούς χρήστες με αποτέλεσμα την απρόσκοπτη συνεργασία μεταξύ τους. Εν τέλει, το σενάριο χρήσης μπόρεσε να υποστηριχθεί αποτελεσματικά, χωρίς να δημιουργεί οποιαδήποτε θέματα από πλευράς καθυστερήσεων ή/και ευχρηστίας. Όπως σημειώθηκε, οι όποιες καθυστερήσεις ήταν κάτω του αντιληπτού (<2msec) και ως εκ τούτου δεν αναφέρθηκαν παρόμοια θέματα.

5.2 Συμπεράσματα και μελλοντικές επεκτάσεις

Παρότι η τελική εφαρμογή εξυπηρέτησε απολύτως το σκοπό της και επέτρεψε την υλοποίηση και εκτέλεση του παιγνίου σε περιβάλλον web, ανέδειξε σειρά ζητημάτων που θα μπορούσαν να μελετηθούν περαιτέρω μια πληρέστερη διερεύνηση των δυνατοτήτων της μοντελο-κεντρικής μηχανικής στο πεδίο των διεπαφών χρήστη-υπολογιστή. Μια πιθανή επέκταση που κρίνεται απολύτως απαραίτητη είναι η εκτέλεση του πειράματος με εμπλεκόμενους χρήστες που αξιοποιούν εντελώς διαφορετικές πλατφόρμες (π.χ. συσκευές android). Μια τέτοια περίπτωση ωστόσο απαιτεί εξειδικευμένο platform server για τη συσκευή android και επομένως δεν θα μπορούσε να μελετηθεί στο πλαίσιο της παρούσας πτυχιακής. Ένα δεύτερο χρήσιμο πείραμα θα ήταν η αξιολόγηση των αισθητικών χαρακτηριστικών της εφαρμογής όπως η διαδραστική απόδοση του σθένους του παίκτη (που θα ενίσχυε την αντίληψη περί των ενδεδειγμένων επόμενων κινήσεων), η ενημέρωση του παίκτη σχετικά με τον περίγυρο του (π.χ., αν απειλείται από αντίπαλο παίκτη, αν πλησιάζει οριακά σημεία του γηπέδου), κλπ.

Βιβλιογραφία

[1] Mark Roseman and Saul Greenberg. 1996. Building real-time groupware with GroupKit, a groupware toolkit. ACM Trans. Comput.-Hum. Interact. 3, 1 (March 1996), 66-106. DOI=http://dx.doi.org/10.1145/226159.226162

[2] Saul Greenberg and Mark Roseman. 1996. GroupWeb: a WWW browser as real time groupware. In Conference Companion on Human Factors in Computing Systems (CHI '96), Michael J. Tauber (Ed.). ACM, New York, NY, USA, 271-272. DOI=http://dx.doi.org/10.1145/257089.257317

[3] Trevor, Jonathan, Thomas Koch, and Gerd Woetzel. "MetaWeb: bringing synchronous groupware to the world wide web." Proceedings of the Fifth European Conference on Computer Supported Cooperative Work. Springer Netherlands, 1997.

[4] Graham, TC Nicholas. "Groupscape: Integrating synchronous groupware and the world wide web." Human-Computer Interaction INTERACT'97. Springer US, 1997.

[5] Isenhour, Philip L., Mary Beth Rosson, and John M. Carroll. "Supporting interactive collaboration on the Web with CORK." Interacting with Computers 13.6 (2001): 655-676.

[6] Dewan, Prasun. "Architectures for collaborative applications." Computer Supported Co-operative Work 7 (1999): 169-193.

[7] Jacob W. Bartel and Prasun Dewan. 2012. Towards multi-domain collaborative toolkits. In Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work (CSCW '12). ACM, New York, NY, USA, 1297-1306. DOI=http://dx.doi.org/10.1145/2145204.2145398

[8] Weigang Wang. 2008. Powermeeting on common ground: web based synchronous groupware with rich user experience. In Proceedings of the hypertext 2008 workshop on Collaboration and collective intelligence (WebScience '08). ACM, New York, NY, USA, 35-39. DOI=http://dx.doi.org/10.1145/1379157.1379166

[9] Brian de Alwis, Carl Gutwin, and Saul Greenberg. 2009. GT/SD: performance and simplicity in a groupware toolkit. In Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems (EICS '09). ACM, New York, NY, USA, 265-274. DOI=10.1145/1570433.1570483 http://doi.acm.org/10.1145/1570433.1570483

[10] Vellis, G., Kotsalis, D., Akoumianakis, D., & Vanderdonckt, J. (2011). Towards a new generation of MBUI engineering methods: Supporting polymorphic instantiation in synchronous collaborative and ubiquitous environments. UsiXML 2011, 86.

[11] Vellis, G., Kotsalis, D., Akoumianakis, D. and Vanderdonckt, J., 2012, October. Model-based engineering of multi-platform, synchronous and collaborative UIs-extending UsiXML for polymorphic user interface specification. In Informatics (PCI), 2012 16th Panhellenic Conference on (pp. 339-344). IEEE.

[12] 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. DOI: http://dx.doi.org/10.1145/2645791.2645825

[13] Figueroa-Martinez, J., López-Jaquero, V., Vela, F. L. G., & González, P. (2013). Enriching UsiXML language to support awareness requirements. Science of Computer Programming, 78(11), 2259-2267.

[14] Christopher Wolfe, T.C. Nicholas Graham, W. Greg Phillips, and Banani Roy. 2009. Fiia: user-centered development of adaptive groupware systems. In Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems (EICS '09). ACM, New York, NY, USA, 275-284. DOI=10.1145/1570433.1570484 http://doi.acm.org/10.1145/1570433.1570484

[15] Giraldo, W. J., Molina, A. I., Ortega, M., & Collazos, C. A. (2008). Integrating Groupware Notations with UML. In Engineering Interactive Systems (pp. 142-149). Springer Berlin Heidelberg.

[16] Penichet, Victor M. Ruiz, Maria Dolores Lozano, José A. Gallud, Ricardo Tesoriero, María Luisa Rodríguez, José Luis Garrido, Manuel Noguera, and María Visitación Hurtado. "Extending and Supporting Featured User Interface Models for the Development of Groupware Applications." J. UCS 14, no. 19 (2008): 3053-3070.

[17] Vellis, G. 2015. Model-based development of ubiquitous synchronous and collaborative UIs. Master's thesis. Technological Education Institution of Crete, Hellas. http://nefeli.lib.teicrete.gr/browse/stef/epp/2015/VellisGeorgios/attached-document-1449133794-631276-15771/VellisGeorgios2015.pdf .

[18] Kotsalis, D. Model-based UI Engineering of Advanced and Creativity-based UIs. Master's thesis. Technological Education Institution of Crete, Hellas. http://nefeli.lib.teicrete.gr/browse/stef/epp/2015/KotsalisDimitrios/attached-document-1449129283-284614-13040/KotsalisDimitrios2015.pdf

[19] https://www.w3.org/TR/mbui-intro

[20] Limbourg, Quentin, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, Murielle Florins, and Daniela Trevisan. "Usixml: A user interface description language for context-sensitive user interfaces." In Proceedings of the ACM AVI'2004 Workshop" Developing User Interfaces with XML: Advances on User Interface Description Languages"(Gallipoli, pp. 55-62. 2004.

[21] Molina, Ana I., Miguel A. Redondo, Manuel Ortega, and Ulrich Hoppe. "CIAM: A Methodology for the Development of Groupware User Interfaces." J. UCS 14, no. 9 (2008): 1435-1446.