ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γλώσσα Ειδικού Σκοπού & Εργαλείο Ανάπτυξης για Πειραματισμό σε Κινητές Ρομποτικές Διατάξεις IoT

Μιχαήλ Α. Τσιρούκης

Επιβλέπων: Ευστάθιος Χατζηευθυμιάδης, Αναπληρωτής Καθηγητής ΕΚΠΑ
Κωνσταντίνος Κολομβάτσος, Διδάκτωρ ΕΚΠΑ

ΑΘΗΝΑ

ΟΚΤΩΒΡΙΟΣ 2016

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γλώσσα Ειδικού Σκοπού & Εργαλείο Ανάπτυξης για Πειραματισμό σε Κινητές Ρομποτικές Διατάξεις IoT

Μιχαήλ Α. Τσιρούκης

Α.Μ. 1250

Επιβλέπων: Ευστάθιος Χατζηευθυμιάδης, Αναπληρωτής Καθηγητής ΕΚΠΑ
Κωνσταντίνος Κολομβάτσος, Διδάκτωρ ΕΚΠΑ

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:
Ευστάθιος Χατζηευθυμιάδης, Αναπληρωτής Καθηγητής ΕΚΠΑ
Μεράκος Λάζαρος, Καθηγητής ΕΚΠΑ

13 ΟΚΤΩΒΡΙΟΥ 2016

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια ιδιαίτερο ενδιαφέρον αποκτούν οι Γλώσσες Ειδικού Σκοπού (ΓΕΣ), λόγω της ευκολίας εκμάθησης τους από τους προγραμματιστές, η απλότητα συντήρησης των προγραμμάτων λόγω της αφαιρετικότητας των γλωσσών αυτών και της αύξησης της παραγωγικότητας. Βέβαια ένα πρόβλημα αποτελεί η υλοποίηση υποστηρικτικών εργαλείων για τις ΓΕΣ, το οποίο ουσιαστικά είναι και το κυριότερο πρόβλημα τους. Σε αυτό έρχεται να δώσει λύση η κοινότητα των προγραμματιστών, η οποία ασχολείται με την δημιουργία σύγχρονων αυτοματοποιημένων υποστηρικτικών εργαλείων για τις γλώσσες αυτές.

Στην παρούσα διπλωματική εργασία, υλοποιήθηκε μια ΓΕΣ για ερευνητές (Experiment Description Language - EDL) σε Διαδικτυακό περιβάλλον. Οι ερευνητές έχουν τη δυνατότητα μέσω της χρήσης ενός οπτικού συντάκτη και ενός συντάκτη κειμένου που προσφέρεται από την εφαρμογή, να δημιουργήσουν ένα σενάριο για τον απομακρυσμένο έλεγχο κινητών συσκευών και να εκτέλεσουν μια σειρά από πειράματα. Οι συσκευές αποτελούν αυτόνομα οχήματα που φέρουν διάφορα αισθητήρια (sensors) και καλούνται να εκτλέσουν ένα πλήθος ενεργειών.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Γλώσσες Ειδικού Σκοπού

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: αυτοματοποιημένα υποστηρικτικά εργαλεία, διαδικτυακή εφαρμογή, κινητό Διαδίκτυο των Πραγμάτων

ABSTRACT

Over the years the Domain Specific Languages (DSL) become more interesting due to their simplicity of their learning, their easy maintainance and for enhancing the productivity. The most common problem faced is the creation of supporting tools for the DSLs, which is overcome by the support of small community of programmers, who are occupied with the creation of modern automated supporting tools for these languages.

In the present Master Thesis, a DSL for experimenters (Experiment Description Language - EDL) has been developed in a web environment. The experimenters have the ability to create a script for remotely controlling mobile IoT devices and to execute a series of experiments, by using both visual and textual editors. The mobile IoT devices constitute autonomous vehicles that bear several sensors and they are programmed to execute many activities.

SUBJECT AREA: Domain Specific Languages

KEYWORDS: automated supporting tools, web application, mobile Internet of Things

Στη Λίτσα και το μικρό Αχιλλέα ...

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά τον κ. Ευστάθιο Χατζηευθυμιάδη, Αναπληρωτή Καθηγητή της Σχολής Πληροφορικής και Τηλεπικοινωνιών του ΕΚΠΑ, επιβλέποντα της συγκεκριμένης Διπλωματικής Εργασίας, για την επιλογή και ανάθεση του θέματος, την καθοδήγηση του, καθώς και για το συνεχές ενδιαφέρον που έδειξε σε όλη τη διάρκεια της εκπόνησης της Διπλωματικής Εργασίας μου. Επίσης, θα ήθελα να τον ευχαριστήσω ιδιαίτερα για την εμπιστοσύνη που μου έδειξε καθώς και τη δυνατότητα που μου προσέφερε να ενταχθώ στην ερευνητική ομάδα του p-comp, αποκομίζοντας έτσι πολύτιμες γνώσεις για την μετέπειτα μου πορεία. Επίσης, δεν θα μπορούσα να παραλείψω να εκφράσω τις ευχαριστίες μου στον Δρ. Κωνσταντίνο Κολομβάτσο, που χωρίς τη καταλυτική συμβολή του, τις πολύτιμες συμβουλές του και τη διαρκή αλληλεπίδρασή μας καθόλη την διάρκεια της ανάπτυξης του θέματος και της συγγραφής της Διπλωματικής Εργασίας, δεν θα ήταν εφικτή η ολοκλήρωσή της. Τέλος, ευχαριστώ πολύ τους γονείς μου, τη σύζυγό μου Λίτσα και το μικρό μου Αχιλλέα για τη συμπαράσταση και κατανόηση που έδειξαν καθ' όλη τη διάρκεια των σπουδών μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Η παρούσα διπλωματική εργασία εκπονήθηκε στα πλαίσια του Μεταπτυχιακού Προγράμματος Σπουδών στην κατεύθυνση «Υπολογιστικά Συστήματα» του Τμήματος Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών. Το αντικείμενο της εργασίας είναι η δημιουργία μιας γλώσσας ειδικού σκοπού για τη δημιουργία εφαρμογών για την ερευνητική περιοχή του κινητού Διαδικτύου των πραγμάτων (Internet of Things - IoT) μέσω Web. Οι ερευνητές έχουν τη δυνατότητα με τη χρήση των προσφερόμενων εργαλείων και μέσω ενός οπτικού συντάκτη και ενός συντάκτη κειμένου, να δημιουργήσουν αυτοτελείς και ολοκληρωμένες εφαρμογές για τη διαχείριση των IoT συσκευών.

Η διπλωματική εργασία διαρθρώνεται ως εξής. Στο πρώτο κεφάλαιο γίνεται μια γενική περιγραφή του αντικειμένου της εργασίας. Στο δεύτερο κεφάλαιο γίνεται μια εκτενής αναφορά στις γλώσσες ειδικού σκοπού (χαρακτηριστικά, βασικές κατηγορίες κλπ). Στο τρίτο κεφάλαιο αναλύονται οι έννοιες των μεταμοντέλων και γραμματικών, σύνταξης και μετασχηματισμών και αναφέρονται κάποια ευρέως χρησιμοποιούμενα περιβάλλοντα ανάπτυξης μοντέλων. Στο τέταρτο κεφάλαιο γίνεται αναλυτική επισκόπηση των διαδικτυακών εργαλείων δημιουργίας γλωσσών ειδικού σκοπού με τη χρήση του Xtext, των μεθόδων επικοινωνίας του προσφερόμενου εξυπηρέτη (server) και της εφαρμογής του πελάτη. Έπεται η μελέτη περίπτωσης για mobile IoT εφαρμογές, η αναλυτική περιγραφή της ανάπτυξης του οπτικού συντάκτη και του συντάκτη κειμένου. Επίσης δίνεται έμφαση στις επιπρόσθετες λειτουργίες που παρέχουν οι προσφερόμενοι συντάκτες όπως για παράδειγμα η επικύρωση περιεχομένων, η αναγνώριση και αντιμετώπιση λαθών καθώς και η παραγωγή κώδικα και αρχείων (απαραίτητα για την εκτέλεση της εφαρμογής στους κινητούς κόμβους). Στο τελευταίο κεφάλαιο δίδονται τα βήματα της δημιουργίας μιας γλώσσας ειδικού σκοπού με τη χρήση του εργαλείου Xtext.

1 ΕΙΣΑΓΩΓΗ

1.1 Γενικά

Στη σημερινή εποχή παρατηρείται αυξημένη χρήση των γλωσσών γενικού σκοπού για τη δημιουργία και συντήρηση εφαρμογών σε διάφορα ερευνητικά πεδία. Η σημερινή βιομηχανία λογισμικού αντιμετωπίζει μεγάλο πρόβλημα στην προσπάθεια της να υλοποιήσει μεγαλύτερα και πιο σύνθετα συστήματα λογισμικού σε μικρό χρονικό διάστημα και με το μικρότερο δυνατό κόστος. Για την υλοποίηση των εφαρμογών, οι προγραμματιστές συνήθως χρησιμοποιούν γλώσσες γενικού σκοπού, αντικειμενοστραφείς και υψηλού επιπέδου αφαιρετικότητας. Παραδείγματα τέτοιων γλωσσών προγραμματισμού αποτελούν η Java και η C++. Μολονότι οι γλώσσες αυτού του είδους προσδίδουν μεγάλη αξιοπιστία και υψηλή αφαιρετικότητα, πολλές φορές η συγγραφή πολύπλοκων προγραμμάτων δυσχαιρένει το έργο των προγραμματιστών κατά την ανάπτυξη και τη συντήρηση του κώδικα. Οι γλώσσες αυτού του είδους είναι πολύ δύσκολο να χρησιμοποιηθούν από τους τελικούς χρήστες, λόγω της πολυπλοκότητας της γλώσσας και της μη εξοικείωσης τους με τις διάφορες λεξικολογικές έννοιες της γλώσσας. Το πρόβλημα παίρνει τεράστιες διαστάσεις όταν οι γλώσσες αυτές χρησιμοποιούνται σε συγκεκριμένα πεδία όπως για παράδειγμα για τη συγγραφή και την εκτέλεση πειραμάτων στον τομέα του IoT. Για να αντιμετωπισθεί το πρόβλημα, οι εφαρμογές υλοποιούνται με γλώσσες ειδικού σκοπού που είναι απλοϊκές και περισσότερο κατανοητές στους ειδικούς του πεδίου αλλά και σε μη έμπειρους τελικούς χρήστες. Η αφαιρετικότητα της γλώσσας και οι ευνόητες έννοιες-ορολογίες που χρησιμοποιούνται από αυτή, την κάνουν περισσότερο προσιτή στη χρήση της από τους ειδικούς του πεδίου.

1.2 Αντικείμενο Εργασίας

Στην παρούσα διπλωματική εργασία αναπτύχθηκαν δύο συντάκτες (οπτικός, κειμένου) για τη δημιουργία εφαρμογών μέσω των οποίων θα υλοποιείται η διαχείριση και ο έλεγχος κινητών IoT συσκευών. Οι συντάκτες που αναπτύχθηκαν με τη χρήση μιας γλώσσας ειδικού σκοπού αποτελούν μια διαδικτυακή εφαρμογή που περιλαμβάνει ένα συντάκτη κειμένου, έναν οπτικό συντάκτη και ένα σύνολο από υποστηρικτικά εργαλεία. Με τη

χρησιμοποίηση της εφαρμογής, ο τελικός χρήστης μπορεί να συγγράψει εύκολα και γρήγορα πειραματικά σενάρια, για την εκτέλεση τους από τις IoT συσκευές. Οι τελικοί χρήστες-ερευνητές μπορούν να συγγράφουν πειραματικά σενάρια διαμέσου των προσφερόμενων συντακτών και τη χρήση έτοιμων προτύπων κώδικα που παρέχονται από την προτεινόμενη εφαρμογή. Επίσης, η εφαρμογή επιτρέπει την εισαγωγή και παραμετροποίηση των κόμβων, τον ορισμό των χαρακτηριστικών τους, τον καθορισμό της χρησιμοποίησης των αισθητηρίων ενός κόμβου κ.α.. Και οι δύο συντάκτες μπορούν να συγχρονιστούν μεταξύ τους. Ο συγχρονισμός είναι η διαδικασία κατά την οποία ο οπτικός συντάκτης της εφαρμογής ενημερώνεται για τις τρέχουσες αλλαγές που συμβαίνουν στο συτάκτη κειμένου και το αντίστροφο. Η σχέση του συγχρονισμού μεταξύ των συντακτών της εφαρμογής είναι αμφίδρομη και σκοπό έχει να περιλαμβάνουν τα ίδια δεδομένα την εκάστοτε χρονική στιγμή. Κύριο μέλημα είναι η ορθή συγγραφή των σεναρίων από τη μεριά του τελικού χρήστη, ο έλεγχος της ορθότητας των εντολών που πρόκειται να εκτελεστούν στους κινητούς κόμβους και ο καθορισμός της ‘συμπεριφοράς’ των κόμβων όσον αφορά τη λήψη-επεξεργασία των δεδομένων που λαμβάνονται μέσω των αισθητήριων.

2 Γλώσσες Ειδικού Σκοπού

Οι Γλώσσες Ειδικού Σκοπού (ΓΕΣ, Domain Specific Languages - DSLs) [1] είναι γλώσσες που ειδικεύονται σε ένα συγκεκριμένο πεδίο εφαρμογής. Οι DSLs διαφέρουν σε σχέση με τις Γλώσσες Γενικού Σκοπού (ΓΓΣ, General Purpose Languages - GPLs). Ένα παράδειγμα χρήσης μιας DSL στο πεδίο του Web (ιστοσελίδες), αποτελεί η χρήση των HTML και CSS. Οι ΓΕΣ έχουν μελετηθεί στο παρελθόν, αλλά ο όρος «γλώσσα σε συγκεκριμένους τομείς» έγινε πιο δημοφιλής λόγω της ανοδικής πορείας της σε συγκεκριμένους τομείς μοντελοποίησης. Στην Εικόνα 1 παρουσιάζονται κάποιες ευρέως χρησιμημοποιούμενες γλώσσες ειδικού σκοπού.

Domain Specific Languages	Πεδίο Χρήσης
VHDL, Verilog , systemc	γλώσσα περιγραφής υλικού (hardware)
ANTLR	γλώσσα παραγωγής parser
SQL	γλώσσα επερωτήσεων για σχεσιακές βάσεις

Εικόνα 1: Παράδειγμα γλωσσών ειδικού σκοπού.

Μερικές φορές η γραμμή ανάμεσα στις γλώσσες γενικού σκοπού και στις γλώσσες ειδικού σκοπού δεν είναι πάντα διακριτή. Μια γλώσσα μπορεί να έχει εξειδικευμένες λειτουργίες για έναν συγκεκριμένο τομέα, αλλά όμως, να εφαρμόζεται ευρύτερα. Αντιστρόφως, μια γλώσσα μπορεί να έχει γενικές λειτουργίες και ενώ αρχικά προορίζονταν για ένα ευρύ φάσμα εφαρμογών, αλλά τελικά να χρησιμοποιείται κυρίως σε ένα συγκεκριμένο τομέα. Ένα παράδειγμα αποτελεί η Perl. Η Perl αναπτύχθηκε αρχικά ως μια γλώσσα επεξεργασίας κειμένου παρόμοια με την AWK και τον προγραμματισμό φλοιού (shell script), αλλά, ως επί το πλείστον, χρησιμοποιήθηκε ως γλώσσα προγραμματισμού γενικού σκοπού. Αντίθετα, η PostScript [2] μπορεί να χρησιμοποιηθεί σε πολλά πεδία, όμως στην πράξη, χρησιμοποιείται κυρίως ως γλώσσα περιγραφής σελίδων.

2.1 Χαρακτηριστικά

Οι ΓΕΣ παρουσιάζουν πολλά ιδιαίτερα χαρακτηριστικά. Το κυριότερο χαρακτηριστικό είναι το πολύ καλά καθορισμένο και περιορισμένο πεδίο εφαρμογής. Η γλώσσα προορίζεται για την ανάπτυξη εφαρμογών με πολύ συγκεκριμένα χαρακτηριστικά και δυνατότητες που θα υιοθετηθούν από ένα πολύ αυστηρά καθορισμένο περιβάλλον. Για το σκοπό αυτό, η σύνταξη της γλώσσας χρησιμοποιεί τις κατάλληλες έννοιες και δομές του συγκεκριμένου πεδίου πιο εύκολα. Μια πολύ καλά σχεδιασμένη DSL μπορεί να είναι πολύ πιο εύκολο να χρησιμοποιηθεί σε αντίθεση με τη χρήση μιας παραδοσιακής βιβλιοθήκης. Είναι περισσότερο κατανοητές από τους ειδικούς του συγκεκριμένου πεδίου. Αυτό έχει ως αποτέλεσμα να βελτιώνεται η παραγωγικότητα του προγραμματιστή, η οποία είναι πάντα πολύτιμη. Ειδικότερα, μπορεί να βελτιώσουν την επικοινωνία με τους ειδικούς του χώρου, αναδεικνύοντας το ως ένα πολύτιμο εργαλείο , που σκοπό έχει να διεκπεραωθούν επιτυχώς όλες οι καταγεγραμμένες απαιτήσεις του πελάτη κατά την φάση της ανάπτυξης του λογισμικού. Δυστυχώς οι DSLs χρησιμοποιούνται από ένα μικρό αριθμό χρήστων και για ένα συγκεκριμένο πεδίο εφαρμογών. Αυτό είναι ένα από τα μειονεκτήματα των γλωσσών αυτών μιας και μετά από κάποιο χρονικό διάστημα, οι συγκεκριμένες γλώσσες παύουν να χρησιμοποιούνται και συνεπώς εγκαταλείπονται. Το σημαντικότερο όμως μειονέκτημα είναι η δυσκολία κατασκευής της ΓΕΣ, καθώς χρειάζεται πεπειραμένους προγραμματιστές που να γνωρίζουν τους κανόνες της δημιουργίας μιας ΓΕΣ. Αυτό σημαίνει ότι θα πρέπει να μπορούν να ορίσουν τους κανόνες της γραμματικής μιας ΓΕΣ, και τεχνικές που είναι συνυφασμένες με το πεδίο των «Μεταγλωτιστών».

2.2 Βασικές Κατηγορίες

Οι ΓΕΣ χωρίζονται σε δύο κατηγορίες [3].

- *Εσωτερική* (internal) ΓΕΣ. Οι γλώσσες αυτής της κατηγορίας χρησιμοποιούν τη σύνταξη μιας γενικής χρήσης γλώσσα προγραμματισμού (ΓΓΠ) και την επεκτείνουν ή την περιορίζουν ανάλογα, εισάγοντας ένα νέο τομέα με ειδικά γλωσσικά στοιχεία, όπως για παράδειγμα ειδικούς τύπους δεδομένων, ρουτίνες ή μακροεντολές [4,5]. Οι ΓΕΣ οι οποίες χρησιμοποιούν ένα μικρό σετ εντολών της ΓΓΠ , περιορίζουν την γλώσσα.

Αντίθετα χρησιμοποιώντας διάφορες τεχνικές η γλώσσα μπορεί να επεκταθεί με την εισαγωγή νέων λέξεων (keywords) στην ΓΓΠ.

- Εξωτερική (external) ΓΕΣ. Οι *εξωτερικές ΓΕΣ* εισαγάγουν μια εντελώς νέα σύνταξη και σημασιολογία, είναι πιο ευέλικτες και πιο εκφραστικές από ότι οι εσωτερικές γλώσσες. Ο σχεδιαστής αναλαμβάνει να αναπτύξει τη γλώσσα από την αρχή. Ενσωματώνει κατάλληλα δομές και έννοιες κάνοντας την περισσότερο κατανοητή σε ειδικούς του συγκεκριμένου πεδίου [4].

Όμως, υπάρχει ένα υψηλό κόστος για να πετύχει ο σχεδιασμός μιας ΓΕΣ. Από την έναρξη, το σχεδιασμό, τη συντήρηση μέχρι και την υλοποίησή της. Οι ΓΕΣ, όπως ακριβώς και ΓΓΣ, μπορούν να αναπαρασταθούν με την μορφή κειμένου ή με έναν οπτικό τρόπο. Πολύ συχνά, ωστόσο, βλέπουμε ότι υπάρχουν οπτικές γλώσσες προγραμματισμού, όπως για παράδειγμα το LabVIEW και η Simulink που αναπαραστούν οπτικά την περιγραφή των σεναρίων τους. Στην πραγματικότητα, η οπτική αναπαράσταση τους συχνά είναι ο πιο φυσικός τρόπος, μιας και απώτερος σκοπός είναι ένα πιο φιλικό και κατανοητό περιβάλλον στον ειδικό του πεδίου.

2.3 Περιγραφές Γλωσσών

Μια ΓΕΣ μπορεί να αναπαρασταθεί με μορφή κειμένου ή οπτικά. Η οπτική αναπαράσταση των δεδομένων είναι ο πιο φυσικός τρόπος για να περιγράψουμε σενάρια σε συγκεκριμένα πεδία εφαρμογής. Αυτό συνεπώς κάνει το εργαλείο ποιό εύχρηστο και φιλικό στη χρήση του ακόμη και από χρήστες με μικρή εμπειρία στο χώρο του συγκεκριμένου πεδίου και στον προγραμματισμό. Η αναπαράσταση μιας ΓΕΣ με μορφή κειμένου, απευθύνεται κυρίως σε χρήστες με μεγαλύτερη εμπειρία.

Το 1956 ο Νόαμ Τσόμσκι ταξινόμησε τις τυπικές γραμματικές σε ιεραρχία με κριτήριο τους τύπους των κανόνων παραγωγής τους. Η ιεραρχία Τσόμσκι [6] θεωρείται πολύ χρήσιμη στο πεδίο της επιστήμης υπολογιστών. Το επίπεδο ιεραρχίας των γραμματικών χαμηλώνει καθώς προχωράμε από τον τύπο 0 ως 3 σύμφωνα με την ιεραρχία του Τσόμσκι: Η διαφοροποίηση μεταξύ των τύπων έγκειται στο ότι διαδοχικά αυτοί έχουν κανόνες

παραγωγής αυξανόμενης αυστηρότητας, οπότε μπορούν να εκφράσουν λιγότερες τυπικές γλώσσες.

I. Γενικές Γραμματικές(General Grammars)

Η μορφή των κανόνων παραγωγής είναι: $\alpha \rightarrow \beta,\ \alpha \neq \epsilon$, όπου $\alpha,\beta \in (V \cup T)^*$

II. Γραμματικές Με Συμφραζόμενα (Context Sensitive Grammar)

Η μορφή των κανόνων παραγωγής είναι: $\alpha \rightarrow \beta,\ |\alpha| \leq |\beta|$, όπου $\alpha,\beta \in (V \cup T)^*$

III. Γραμματικές Χωρίς Συμφραζόμενα(Context Free Grammars)

Η μορφή των κανόνων παραγωγής είναι: $A \rightarrow \alpha$, όπου $A \in V$ και $\alpha \in (V \cup T)^*$

IV. Κανονικές Γραμματικές(Regular Grammars)

Η μορφή των κανόνων παραγωγής είναι: $A \rightarrow u,\ B \rightarrow Bu$, όπου $A, B \in V$

Είναι γνωστό ότι κάθε φυσική γλώσσα (natural language) παράγεται από μία γραμματική και υπόκειται στους κανόνες τους οποίους ορίζει η τελευταία. Αυτό ισχύει και για τις διάφορες γλώσσες προγραμματισμού οι οποίες αποτελούν τυπικές γλώσσες (formal languages) και έχουν αυστηρό συντακτικό ορισμό. Πρωταρχικές έννοιες στη θεωρία των τυπικών γλωσσών είναι τα σύμβολα (ως αντικείμενα) και η παράθεση ως πράξη. Τα σύμβολα και συγκεκριμένα ένα πεπερασμένο σύνολο αυτών συγκροτούν ένα αλφάβητο. Εάν Σ είναι ένα αλφάβητο τότε Σ* είναι το σύνολο όλων των συμβολοσειρών (strings) από το Σ, όπου με τον όρο string ή συμβολοσειρά εκφράζεται μία πεπερασμένου μήκους ακολουθία συμβόλων. Το μήκος μιας συμβολοσειράς w συμβολίζεται με |w| , ενώ ε είναι ο συμβολισμός για το κενό string. Η παράθεση των συμβολοσειρών x και y συμβολίζεται με xy.

Ορισμός Μια τυπική γραμματική G=(NT, T, P, S) είναι μια τετράδα που αποτελείται:

(i) από ένα αλφάβητο N από μη τερματικά σύμβολα (non-terminals)

(ii) από ένα αλφάβητο T από τερματικά σύμβολα (terminals) για το οποίο όμως ισχύει ότι $N \cap T=0$

(iii) από ένα πεπερασμένο σύνολο P από κανόνες παραγωγής, δηλαδή διατεταγμένα ζεύγη (α,β) όπου $\alpha,\beta \in (N \cup T)^*$ και $\alpha \neq \varepsilon$

(iv) από ένα αρχικό σύμβολο (ή αξίωμα) $S \in N$

Η υπάρχουσα σύμβαση όσον αφορά τους συμβολισμούς υποδεικνύει ότι τα μη τερματικά σύμβολα αναπαρίστανται με κεφαλαίους λατινικούς χαρακτήρες (A, B, C, D,…) ενώ τόσο τα τερματικά σύμβολα όσο και οι συμβολοσειρές που αυτά συγκροτούν συμβολίζονται με πεζούς λατινικούς χαρακτήρες. (Με γράμματα όπως a, b, c, d, … οι σταθερές και με z,y,x,w,u,v, .. οι προαναφερθείσες συμβολοσειρές). Τέλος με πεζούς ελληνικούς χαρακτήρες αναπαρίστανται όλα τα strings που περιέχουν τόσο τερματικά όσο και μη τερματικά σύμβολα (α, β, γ, δ, …$\in$(N$\cup$ T)*).

Σε μια Γραμματική χωρίς συμφραζόμενα, το αριστερό μέρος κάθε κανόνα παραγωγής περιέχει μόνο ένα μη–τελικό σύμβολο. Σε αυτήν τη περίπτωση, ένα μη-τελικό σύμβολο είναι μία συμβολοσειρά, η οποία αποτελείται από ένα ή περισσότερους χαρακτήρες. Σε μια Κανονική γραμματική, το αριστερό μέρος κάθε κανόνα παραγωγής περιέχει μόνο ένα μη–τελικό σύμβολο, αλλά και το δεξιό μέρος του κανόνα έχει περιορισμό: Επιτρέπεται να είναι κενό, ή να περιέχει μόνο ένα μη–τελικό σύμβολο, ή να περιέχει μόνο ένα τελικό σύμβολο ακολουθούμενο από ένα μη–τελικό σύμβολο.

Υπάρχουν μια σειρά από διαφορετικές προσεγγίσεις αναλυτών όπως οι "LR" και οι "LL" αναλυτές για γραμματικές χωρίς συμφραζόμενα που διαφέρουν στον τρόπο εισόδου και στο χειρισμό αντιστοίχισης γραμματικών κανόνων [7]. Οι LL αναλυτές ξεκινούν από το αρχικό σύμβολο στο αριστερό μέρος εφαρμόζοντας τους κανόνες παραγωγής στο δεξιό τμήμα του αλφαριθμητικού. Αντίθετα οι LR ξεκινούν από το αλφαριθμιτικό και καταλήγουν στην ρίζα του δέντρου.

Συνήθως οι γραμματικές ορίζονται χρησιμοποιώντας μια αναπαράσταση κειμένου που ονομάζεται Backus-Naur Form (BNF) [8]. Αποτελείται από ένα αριθμό κανόνων (παραγωγές), όπου ορισμένοι χαρακτήρες έχουν μια ορισμένη έννοια. Για παράδειγμα , ο χαρακτήρας ‘|’ αντιπροσωπεύει εναλλακτικές λύσεις. Η εκτεταμένη μορφή Backus-Naur (EBNF) [8] είναι μια τυποποίηση του BNF για την αναπαράσταση της γλώσσας προγραμματισμού. Ένα μικρό παράδειγμα για τον ορισμό των αριθμητικών εκφράσεων θα μπορούσε να μοιάζει με το παρακάτω τμήμα κώδικα που παρουσιάζεται στην Εικόνα 2.

```
assignment =  identifier "=" expr
     expr  =  number | binaryOpExpr
binaryOpExpr = expr binaryOp expr
  binaryOp  =  "+" | "–"
```

Εικόνα 2: Δείγμα (E)BNF γραμματικής

Σε αυτή την περίπτωση, η έκφραση (expr) Θα μπορούσε να είναι είτε ένας αριθμός ή ένα δυαδική έκφραση (binaryOpExpr). Μια δυαδική έκφραση αποτελείται από δύο εκφράσεις που χωρίζονται από ένα δυαδικό τελεστή (binaryOp). Binary τελεστές είναι το "+" και το "-" (ένα κάθε φορά).

Ορισμοί: Λεκτικός Αναλυτής και Συντακτικός Αναλυτής

Ο λεκτικός αναλυτής είναι η πρώτη φάση της μεταγλώττισης ενός προγράμματος. Σκοπός του είναι να διαβάσει μία είσοδο από χαρακτήρες και να παράξει μία σειρά από λεκτικές μονάδες (λεκτήματα - tokens) που θα χρησιμοποιηθούν στην συνέχεια για την συντακτική ανάλυση. Κατά την ανάγνωση των χαρακτήρων εισόδου, ο λεκτικός αναλυτής εκτελεί κάποιες επιπλέον λειτουργίες όπως για παράδειγμα

- η αφαίρεση σχολίων, κενών, νέων γραμμών και tabs
- η συσχέτιση λεκτικών μονάδων και λαθών με τον αριθμό γραμμής
- η υλοποίηση preprocessing macros

Ο λεκτικός αναλυτής υλοποιείται συνήθως ως υπορουτίνα του συντακτικού αναλυτή.

Μετά τον καθαρισμό της εισόδου από τα κενά και τους χαρακτήρες αλλαγής γραμμής, τα tabs κ.λπ. και τον χωρισμό των συμβόλων σε λεκτήματα, ο λεκτικός αναλυτής έχει τελειώσει την δουλειά του και οδηγεί την έξοδό του (δηλαδή το ρεύμα λεκτημάτων που φτιάχνει) στον συντακτικό αναλυτή. Δουλειά του συντακτικού αναλυτή είναι να αναγνωρίσει οποιαδήποτε συντακτικά λάθη υπάρχουν στον κώδικα και ίσως να ανανήψει από κάποια συχνά συναντούμενα λάθη, για να μπορεί να συνεχίσει να αναλύει το υπόλοιπο του προγράμματος. Κατά την ανάλυση αυτή, ο συντακτικός αναλυτής φτιάχνει ένα δέντρο με τα

λεκτήματα με τα οποία ο λεκτικός αναλυτής τον προμηθεύει, το οποίο αναπαριστά πλήρως το πρόγραμμα που έχει γράψει ο χρήστης, και ονομάζεται Συντακτικό Δέντρο (Syntax Tree). Υπάρχουν αρκετοί τρόποι αναπαράστασης των συντακτικών δέντρων σε ένα μεταγλωττιστή, οι κυριότεροι από τους οποίους είναι τα Αφηρημένα Συντακτικά Δέντρα (Abstract Syntax Trees - ASTs) [9] και οι Κατευθυνόμενοι Ακυκλικοί Γράφοι (Directed Acyclic Graphs - DAGs). Υπάρχουν δύο μέθοδοι γεμίσματος του συντακτικού δέντρου που χρησιμοποιούνται από τους μοντέρνους μεταγλωττιστές: ο “από πάνω προς τα κάτω” (top-down), στον οποίο το συντακτικό δέντρο φτιάχνεται ξεκινώντας από την ρίζα και καταλήγοντας στα φύλλα, και ο “από κάτω προς τα πάνω” (bottom-up), στον οποίο το συντακτικό δέντρο φτιάχνεται ξεκινώντας από τα φύλλα και καταλήγοντας στη ρίζα.

Στην σημασιολογική ανάλυση που αποτελεί και το 3^{ο} στάδιο της μεταγλώτισης ελέγχονται οι έννοιες που παράγουν οι εκφράσεις που παράγονται από το συντακτικό δέντρο.

2.4 Υποστηρικτικά Εργαλεία

Για τον επιτυχή σχεδιασμό και την υλοποίηση μιας ΓΕΣ θα πρέπει να χρησιμοποιηθούν τα κατάλληλα υποστηρικτικά εργαλεία. Τα εργαλεία αυτά, θα πρέπει να είναι φιλικά προς το χρήστη και να πληρούν τις προδιαγραφές της ΓΕΣ για την περαιτέρω ανάπτυξή της. Μια προδιαγραφή μπορεί να αποτελεί η πολυπλοκότητα της γραμματικής μιας ΓΕΣ. Τα πιο ευρέως διαδεδομένα υποστηρικτικά εργαλεία είναι:

- οι *συντάκτες* (editors): εργαλεία που επιτρέπουν στο χρήστη να δημιουργήσει ένα στιγμιότυπο της γλώσσας από την αρχή ή να τροποποιήσει ένα υπάρχων. Υπάρχουν δύο είδη συντακτών : οι συντάκτες κειμένου και οι οπτικοί συντάκτες.
- οι *αναλυτές* (analyzers): εργαλεία που αναλύουν ένα στιγμιότυπο γλώσσας και παράγουν κάποια κρίση/αναφορά για αυτό. Δημοφιλή παραδείγματα είναι εκτιμητές κόστους (cost estimators) και οι επικυρωτές μοντέλων (model verifiers).
- οι *μετατροπείς* (transformers): εργαλεία που μετατρέπουν ένα στιγμιότυπο της γλώσσας σε κάποια άλλη μορφή. Τέτοια εργαλεία είναι οι μεταγλωττιστές (compilers) [10], οι μετατροπείς μοντέλων (model transformers), εργαλεία αντίστροφης μηχανικής

(reverse engineering tools), γεννήτορες τεκμηρίωσης (documentation generators) και γεννήτορες κώδικα (code generators).

- οι *εκτελεστές* (executors): εργαλεία τα οποία εκτελούν ένα στιγμιότυπο της γλώσσας για να εξάγουν χρήσιμα συμπεράσματα σε σχέση με την ορθή εισαγωγή του κώδικα. Μερικά παραδείγματα αυτών είναι οι αποσφαλματωτές (debuggers), οι προσομοιωτές (simulators), και τα αυτοματοποιημένα εργαλεία ελέγχου (automated testers).

Συνήθως, τα παραπάνω εργαλεία δεν αρκούν για να αναπτυχθεί μια εφαρμογή γρήγορα και εύκολα. Οι περισσότερες εφαρμογές αναπτύσσονται ταυτόχρονα από μια οι περισσότερες ομάδες προγραμματιστών. Αυτό σημαίνει ότι θα πρέπει να υπάρχουν εργαλεία για τον έλεγχο των εκδόσεων (versioning control) των εφαρμογών. Πολλά ολοκληρωμένα περιβάλλοντα ανάπτυξης γλώσσων παρέχουν τέτοια είδους εργαλεία είτε ενσωματωμένα είτε ως εξωτερικά εργαλεία (π.χ git).

Επιπλέον, οι χρήστες των γλωσσών καθ’ όλη τη διαδικασία της ανάπτυξης μια εφαρμογής χρησιμοποιούν εργαλεία διαχείρισης αναφοράς σφαλμάτων (bug reporting). Δεν είναι λίγες οι φορές που όλα αυτά τα εργαλεία ενσωματώνονται σε ένα ολοκληρωμένο περιβάλλον ανάπτυξης (Integrated Development Environment - IDE) και προσφέρονται ως σύνολο.

3 Μοντελοποίηση Γλωσσών Ειδικού Σκοπού

Η ανάπτυξη στον τομέα της πληροφορικής και οι συνεχώς αυξανόμενες απαιτήσεις για ταχύτερη επεξεργασία μεγάλου όγκου πληροφοριών, δυσκολεύει όλο και περισσότερο το έργο των προγραμματιστών στην παραγωγή νέου λογισμικού. Το λογισμικό θα πρέπει να υλοποιηθεί σε σαφώς σε μικρό χρονικό διάστημα, να είναι ταυτόχρονα ποιοτικό και ικανό να καλύψει τις προαναφερθείσες ανάγκες. Αυτό φυσικά σημαίνει υψηλό κόστος για μια επιχείρηση και φυσικά λόγω του όγκου και της πολυπλοκότητας του κώδικα, το λογισμικό πολλές φορές δεν δύναται να υλοποιηθεί στον προβλεπόμενο χρόνο.

Τη λύση έρχεται να δώσει η *αρχιτεκτονική βασισμένη σε μοντέλα* (Model Driven Architecture - MDA). Ο όρος MDA προωθήθηκε από την OMG (Object management Group) [11] με κύριος στόχο τη φορητότητα, τη διαλειτουργικότητα και την επαναχρησιμοποίηση των μεταμοντέλων [12] για το σχεδιασμό μοντέρνων συστημάτων. Η MDA ορίζει τρία μοντέλα :

1. Ένα μοντέλο ανεξάρτητο υπολογισμών (Computation Independent Model) το οποίο είναι η άποψη ενός συστήματος που δεν δείχνει λεπτομέρειες της δομής των συστημάτων και έχει ένα λεξικό που χρησιμοποιείται για τον προσδιορισμό του συστήματος.
2. Ένα μοντέλο ανεξάρτητο της πλατφόρμας (Platform Independent Model- PIM) το οποίο είναι η άποψη ενός συστήματος που παρέχει ένα ιδιαίτερο τεχνικό προσδιορισμό του συστήματος. Το PIM περιγράφεται σε μορφή ανεξάρτητη της πλατφόρμας στο τέλος της φάσης σχεδίασης και μπορεί να χρησιμοποιηθεί από έναν διαφορετικό αριθμό πλατφορμών.
3. Ένα μοντέλο ειδικής πλατφόρμας (Platform Specific Model-PSM) που συνδυάζει τους προσδιορισμούς στο PIM με τις λεπτομέρειες που εξειδικεύουν πώς αυτό το σύστημα χρησιμοποιεί μια ιδιαίτερου τύπου πλατφόρμα.

Στην περιοχή της μηχανικής λογισμικού ένα *μοντέλο* είναι μια αφαίρεση ενός συστήματος λογισμικού ή μέρους αυτού και ένα *μεταμοντέλο* είναι άλλη αφαίρεση, ορίζοντας τις ιδιότητες του μοντέλου μόνο. Έτσι, όπως ένα πρόγραμμα υπολογιστή συμμορφώνεται με την γραμματική της γλώσσας προγραμματισμού στην οποία είναι γραμμένο, ένα μοντέλο

συμμορφώνεται με το μεταμοντέλο του. Οπωσδήποτε, ακόμη καιένα μεταμοντέλο είναι από μόνο του ένα μοντέλο.

Ορισμοί

- Ένα *μεταμεταμοντέλο* είναι ένα μοντέλο το οποίο είναι το μοντέλο της αναφοράς του (δηλ. συμμορφώνεται στον εαυτό του)
- Ένα *μεταμοντέλο* είναι ένα μοντέλο τέτοιο ώστε το μοντέλο αναφοράς του είναι ένα μεταμοντέλο.

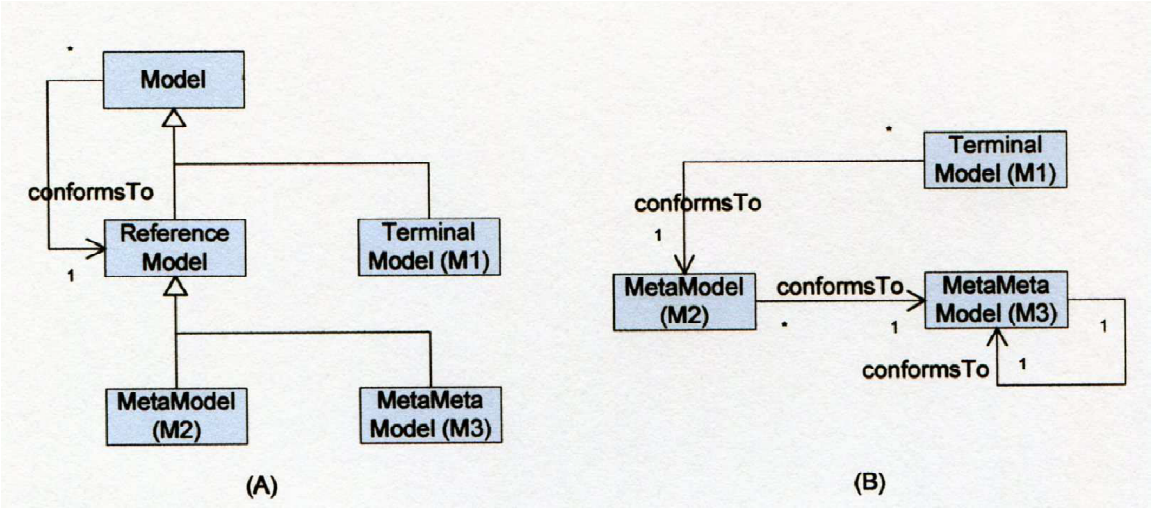
Ουσιαστικά, ο προγραμματιστής καλείται να αποκρύψει την πολυπλοκότητα του κώδικα και τις πλεονάζουσες πληροφορίες χρησιμοποιώντας υψηλότερα επίπεδα αφαίρεσης (abstraction level). Αυτό το πετυχαίνει χρησιμοποιώντας ένα νέο περιβάλλον ανάπτυξης που βασίζεται σε μετα-μοντέλα και γραμματικές.
Για τη δημιουργία του περιβάλλοντος ανάπτυξης χρειάζεται να υλοποιήθει ένα Μοντέλο Γλώσσας Ειδικού Σκοπού (Domain Specific Modeling Language - DSML), ένας γεννήτορας κώδικα για το συγκεκριμένο πεδίο (code generator) και ένα πλαίσιο εκτέλεσης (domain framework).

3.1 Μεταμοντέλα και Γραμματικές

Τα τελευταία χρόνια έχει γίνει μεγάλη προσπάθεια για τη γεφύρωση του χάσματος μεταξύ των γραμματικών και των μοντέλων. Νέες τεχνικές έχουν εισαχθεί για το μετασχηματισμό των γραμματικών σε μεταμοντέλα και αντιστρόφως. Ο μετασχηματισμός μοντέλων είναι η διαδικασία μετασχηματισμού ενός μοντέλου σε ένα άλλο μέσω ενός γεννήτορα κώδικα. Το παραγόμενο μεταμοντέλο έχει τις παρακάτω ιδιότητες:

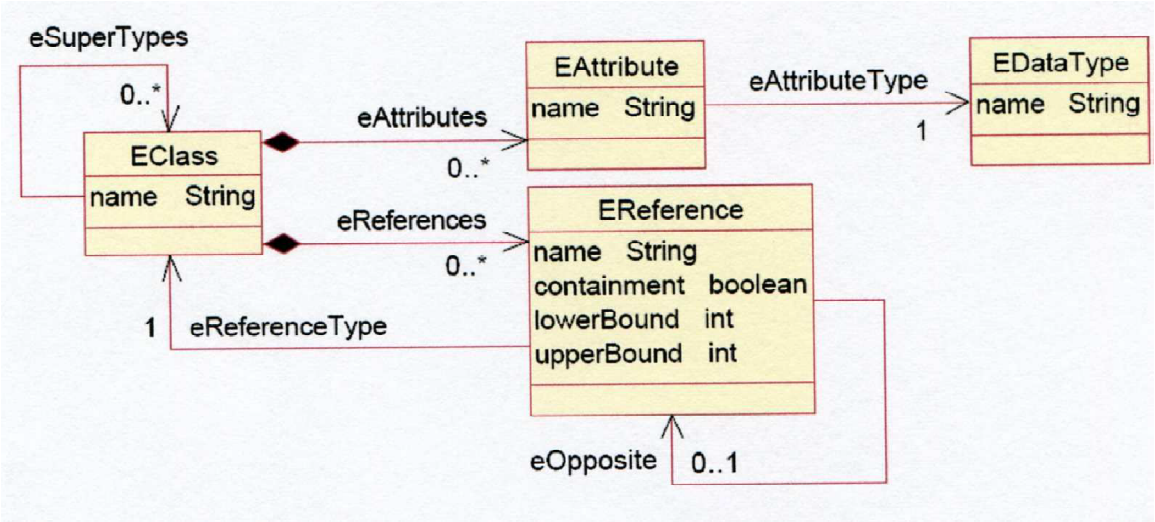
- αναπαριστά με απλότητα στον προγραμματιστή τις εισόδους και τις εξόδους του μοντέλου.
- αποκρύπτει πλεονάζοντα κώδικα που πρακτικά δεν χρειάζεται.
- απλουστεύει τη διαχείριση του κώδικα.

Για την διαδικασία του μετασχηματισμού, απαραίτητη είναι η ύπαρξη των μεταμοντέλων και μιας γλώσσας μετασχηματισμού που να έχει κατάλληλους κανόνες για το μετασχηματισμό ενός μοντέλου σε ένα άλλο. Στην Εικόνα 3 παρουσιάζεται η συσχέτιση μεταξύ ενός μεταμοντέλου, ενός μοντέλου αναφοράς, και του τερματικού μοντέλου.



Εικόνα 3: Μετασχηματισμός μοντέλων

Στην Εικόνα 4 παρουσιάζεται ένα στιγμιότυπο ενός Eclass και οι σχέσεις του με τα Attributes, References και DataType.



Εικόνα 4: Στιγμιότυπο ενός eclass και απεικόνιση των Attributes, References και DataType

3.1.1 Domain specific modeling

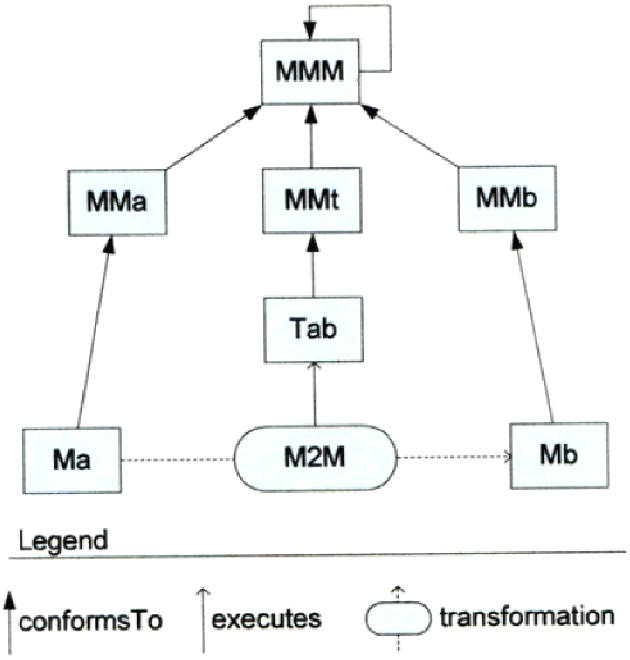
Το ειδικό πλαίσιο μοντελοποίησης (Domain Specific Modeling - DSM) [13] είναι μια μεθοδολογία παραγωγής λογισμικού. Εμπλέκει συστηματικά γλώσσες ειδικού σκοπού για να αναπαραστήσει πολλαπλές πτυχές ενός συστήματος. Επίσης περιλαμβάνει συχνά την ιδέα της δημιουργίας κώδικα απευθείας από τα μοντέλα των ΓΕΣ αυτοματοποιώντας τη δημιουργία του εκτελέσιμου κώδικα. Συνεπώς, με αυτό τον τρόπο αποφεύγεται η χειροκίνητη δημιουργία-συντήρηση του πηγαίου κώδικα, βελτιώνοντας σημαντικά την παραγωγικότητα των προγραμματιστών. Η αξιοπιστία της αυτόματης παραγωγής σε σύγκριση με τη χειροκίνητη κωδικοποίηση μειώνει επίσης τον αριθμό των ελαττωμάτων (bugs) που μπορεί να προκύψουν, βελτιώνοντας έτσι την ποιότητα των προγραμμάτων. Οι ΓΕΣ διαφέρουν από τις προηγούμενες προσπάθειες δημιουργίας κώδικα με σύγχρονα εργαλεία υποβοήθησης της διαδικασίας ανάπτυξης Λογισμικού υπολογιστών (Computer Aid Software Engineering - CASE) [14,15] της δεκαετίας του 1980 ή με τα εργαλεία Ενοποιημένης Γλώσσας Μοντελοποίησης (Unified Modeling Language - UML) [16] της δεκαετίας του 1990. Σε αυτά τα εργαλεία, οι γεννήτριες κώδικα και οι γλώσσες μοντελοποίησης υλοποιήθηκαν από τους κατασκευαστές των εργαλείων. Συνήθως, ένας μικρός σχετικά αριθμός πεπειραμένων προγραμματιστών του πεδίου, δημιουργούν μια γλώσσα μοντελοποίησης και τη γεννήτρια κώδικα, για να τα χρησιμοποιούν πλήθος προγραμματιστών για περαιτέρω ανάπτυξη. Έχοντας τη γλώσσα μοντελοποίησης και το γεννήτορα κώδικα που υλοποιήθηκε από έναν οργανισμό επιτρέπεται μια σφιχτή τακτοποίηση με τις ανάγκες του συγκεκριμένου πεδίου. Επίσης, μειώνεται ο χρόνος που απαιτείται για τους προγραμματιστές να μάθουν τη γλώσσα μοντελοποίησης, δεδομένου ότι μπορεί να χρησιμοποιεί οικείους όρους και έννοιες. Τέλος, δεδομένου ότι χρειάζεται να ληφθούν υπόψη οι απαιτήσεις ενός μόνο οργανισμού, είναι ευκολότερο για τη γλώσσα μοντελοποίησης να εξελίσσεται ανάλογα με τις αλλαγές στο πεδίο.

Οι συγκεκριμένες γλώσσες πεδίου μπορεί να καλύπτουν συνήθως μια σειρά από επίπεδα αφαίρεσης για ένα συγκεκριμένο πεδίο. Για παράδειγμα, μια γλώσσα μοντελοποίησης σε συγκεκριμένους τομείς για τα κινητά τηλέφωνα θα μπορούσε να επιτρέψει στους χρήστες να καθορίσουν αφαιρέσεις υψηλού επιπέδου για τη διεπαφή του χρήστη, καθώς και αφαιρέσεις σε χαμηλότερο επίπεδο για την αποθήκευση δεδομένων, όπως για παράδειγμα οι αριθμοί τηλεφώνων. Ομοίως, μια γλώσσα μοντελοποίησης σε συγκεκριμένους τομείς

των χρηματοπιστωτικών υπηρεσιών θα μπορούσε να επιτρέψει στους χρήστες να καθοριστούν αφαιρέσεις υψηλού επιπέδου για τους πελάτες, καθώς και αφαιρέσεις σε χαμηλότερο επίπεδο για μια εφαρμογή αλγόριθμων διαπραγμάτευσης για τις μετοχές και τα ομολόγα.

3.2 Σύνταξη και Μετασχηματισμοί

Μετά από τον ορισμό των μεταμοντέλων είναι δυνατό να ορίσουμε μετασχηματισμούς ενός μοντέλου σε κάποιο άλλο. 'Ενα γενικό σχήμα της διαδικασίας μετασχηματισμού ακολουθείται από τις δύο κυριότερες γλώσσες μετασχηματισμού την ATL (Atlas Transformation Language) και την QVT (Query-View Transformation). Για να γίνει περισσότερο κατανοητό η όλη διαδικασία του μετασχηματισμού, παρουσιάζεται ένα παράδειγμα στην Εικόνα 5. Στην κορυφή υπάρχει ένα κοινό μεταμεταμοντέλο (MMM) στο οποίο συμμορφώνονται δύο μεταμοντέλα (MMa και MMB). Ο στόχος της διαδικασίας μετασχηματισμού ή μοντέλο σε μοντέλο διαδικασίας (M2M) είναι να πάρει ένα μοντέλο Ma, το οποίο συμμορφώνεται προς το MMa, ως είσοδο (ή μοντέλο πηγή) και παράγει το Mb, το οποίο συμμορφώνεται προς το MMb ως έξοδο (ή μοντέλο στόχος). Εκτός από τα μοντέλα πηγή και στόχος, η διαδικασία εκτελεί ένα πρόγραμμα μετασχηματισμού, που ονομάζεται «Tab», και περιγράφει τη διαδικασία μετασχηματισμού ενός μοντέλου που συμμορφώνεται προς το MMa σε ένα μοντέλο που συμμορφώνεται σε ένα μεταμοντέλο MMb. Το πρόγραμμα μετασχηματισμού είναι από μόνο του ένα μοντέλο που συμμορφώνεται σε ένα μεταμοντέλο (MMt), το οποίο με τη σειρά του συμμορφώνεται στο μεταμεταμοντέλο (MMM). Κατ’ αυτόν τον τρόπο, όπως στην περίπτωση του συμβολισμού EBNF, το MMt ορίζει την αφαιρετική σύνταξη της γλώσσας μετασχηματισμού. Αμφότερες οι ATL και QVT μέθοδοι ορίζουν τις αφαιρετικές συντάξεις τους μέσω ενός τέτοιου μεταμοντέλου.



Εικόνα 5: Γενικό σχήμα ενός μοντέλου μετασχηματισμού.

Ακολούθως, παρουσιάζεται μια αξιολόγηση και επέκταση των διαθέσιμων προσεγγίσεων, όσον αφορά κάποιες από τις γλώσσες μετασχηματισμού μοντέλων. Οι μετασχηματισμοί μοντέλων είναι ένα κλειδί – προαπαιτούμενο για τη *Μηχανική Οδηγούμενη από Μοντέλα* και για αυτό παρουσιάζει μια ενεργή ερευνητική περιοχή. Πολλές γλώσσες μετασχηματισμού μοντέλων είναι διαθέσιμες, ενώ μπορούν να κατηγοριοποιηθούν σε διαφορετικές κατηγορίες. Μια κατηγορία μετασχηματισμού μοντέλων, θα πρέπει να είναι ικανή, ώστε να μπορεί να εκπληρώσει την δεδομένη διαδικασία από μια διαφορετική προσέγγιση. Υπάρχουν ταξινομήσεις των γλωσσών μετασχηματισμού μοντέλων που περιλαμβάνουν μια αναφορά των γενικών χαρακτηριστικών της γλώσσας που πρέπει να υποστηρίξουν. Είναι δυνατόν να συγκρίνουμε προσεγγίσεις μετασχηματισμού μοντέλων με σκοπό να βρούμε πόσο είναι διαθέσιμες για ένα δεδομένο πρόβλημα. Όπως στις κοινές προσεγγίσεις του αντικειμενοστραφούς προγραμματισμού (π.χ. Java), υπάρχουν μερικά ιδιαίτερα προβλήματα που εμφανίζονται επαναληπτικά. Για παράδειγμα, είναι συχνά αναγκαίο να μετασχηματίσουμε μια τιμή μιας ιδιότητας στο μοντέλο πηγή σε ένα αντικείμενο στο μοντέλο στόχος. Για τέτοιες περιπτώσεις, θα μπορούσαμε να θεωρήσουμε

ότι τέτοια προβλήματα λύνονται με ένα γενικό τρόπο. Επιπλέον, γενικές λύσεις μπορούν να επαναχρησιμοποιηθούν σε άλλους μετασχηματισμούς μοντέλων.

Άλλες γλώσσες μετασχηματισμού μοντέλων είναι οι SmartQVT, Kemeta και Triple Graph Grammas (TGG) (χρησιμοποιώντας το MOFLON ως εργαλείο μετασχηματισμού). Η SmartQVT είναι μια πλήρης εφαρμογή ανοικτού κώδικα Java της QVT. Οι ATL και SmartQVT είναι υβριδικές (ένα μείγμα δηλωτικής και επιτακτικής) γλώσσες. Επιπλέον, η SmartQVT υλοποιεί το «λειτουργικό» κομμάτι του προτύπου OMG QVT. Η Kermeta δρά ως παράδειγμα για μια «επιτακτική» γλώσσα, ενώ οι TTGs παρουσιάζουν μια δηλωτική προσέγγιση με μια γραφική σύνταξη.

3.3 Περιβάλλοντα Ανάπτυξης Μοντέλων

3.3.1 Microsoft DSL Tools

Η Microsoft την τελευταία δεκαετία ασχολείται με τα περιβάλλοντα ανάπτυξης μοντέλων [17]. Η πρώτη της προσπάθεια ξεκίνησε με το Visual Studio 2005 SDK 3.0, και έχει φτάσει έως σήμερα στην υλοποίηση έξι εκδόσεων εργαλείων ΓΕΣ (DSL tools), με την τελευταία έκδοση να περιλαμβάνεται στα εργαλεία του Visual Studio 2015**.** Τα microsoft DSL tools αποτελούν έναν συνδυασμό από πλαίσια (frameworks), γλώσσες, συντάκτες (editors), γεννήτορες (generators) και wizards που επιτρέπουν στους σχεδιαστές γλωσσών να καθορίσουν τις δικές τους γλώσσες μοντελοποίησης και εργαλεία. Χρησιμοποιώντας τα εργαλεία αυτά, ο χρήστης μπορεί να ορίσει με αρκετά διαισθητικό τρόπο τη δική του γραφική γλώσσα μοντελοποίησης ειδικού σκοπού (Visual Domain Specific Language - VDSL) καθώς και ένα βασικό εργαλείο για την επεξεργασία των στιγμιότυπων της γλώσσας αυτής. Επιπρόσθετα, για τη μετατροπή των δομών της γλώσσας μοντελοποίησης σε κάποιου είδους εκτελέσιμη μορφή κώδικα χρησιμοποιείται μια γλώσσα προτύπων (template language), που αποτελείται κυρίως από C# ή Visual Basic δομές. Ένα βασικό μειονέκτημα όλων των παραπάνω εργαλείων μοντελοποίησης είναι ότι είναι εμπορικά προιόντα και η χρήση τους επιβαρύνει αρκετά τον υποψήφιο αγοραστή.

3.3.2 MEtaGile

Το MEtaGile είναι ένα περιβάλλον το οποίο υποστηρίζει λύσεις για μοντελοποίηση ΓΕΣ και με το οποίο μπορούμε να δημιουργήσουμε νέους συντάκτες κειμένου ΓΕΣ ή μετα-μοντέλα που να προσαρμοσμόζονται σε ένα συγκεκριμένο πεδίο. Επίσης υποστηρίζει τη χρήση της παραγώμενης ΓΕΣ από τους χρήστες για τον ορισμό ενός στιγμιοτύπου του συστήματος (στιγμιότυπο ΓΕΣ ή μοντέλου). Το MEtaGile σε σχέση με τις υπάρχουσες τεχνολογίες περιβαλλόντων ανάπτυξης μοντέλων (π.χ. OAW [18], Microsoft DSL tools, GME, MetaEdit+), χειρίζεται ρεαλιστικά όλες τις πτυχές της ανάπτυξης. Μερικές από αυτές τις πτυχές της ανάπτυξης είναι: α) η ύπαρξη μοναδικού μοντέλου κειμένου για την είσοδο του, β) η ιεραρχική και οι γραφικές απόψεις για την τεκμηρίωση και την πλοήγηση, γ) ο εκ νέου ορισμός σε πραγματικό χρόνο των ΓΕΣ και των προτύπων, και δ) η αποτελεσματική διαχείριση των παραγόμενων αρχείων.

3.3.3 MetaEdit

Τα εργαλεία MetaEdit+ [19] κατασκευάστηκαν με τη βασική αρχή ότι όλα τα εργαλεία CASE θα είναι ουσιαστικά το ίδιο. Όλα τα αντικείμενα μπορούν να μπούν σε ένα διάγραμμα, να τροποποιηθούν οι ιδιότητές τους και να διασυνδέονται μεταξύ τους με σχέσεις (relationships). Τα εργαλεία του MetaEdit+ περιλαμβάνουν μια γενική συμπεριφορά για τα αντικείμενα και τις σχέσεις, και ένα διάγραμμα συντάκτη. Επίσης περιλαμβάνουν αντικείμενα, γραφικούς περιηγητές (Graph Browsers), και τις ιδιότητες των διαλόγων. Οι προγραμματιστές γλωσσών μοντέλοποίησης ειδικού σκοπού (Domain Specific Modeling Languages - DSML) χρειάζεται να καθορίσουν μόνο τη γλώσσα μοντελοποίησης όπως για παράδειγμα τη δημιουργία ενός νέου τύπου αντικειμένου, δίνοντάς του ένα όνομα και την επιλογή που έχουν οι τύποι ιδιοκτησίας. Ένας φορέας που βασίζεται σε συμβολικό συντάκτη μας επιτρέπει να ορίσουμε τα σύμβολα του αντικειμένου και τη σχέση τους ή την επαναχρησιμοποίηση υπαρκτών συμβόλων. Οι MetaEdit+ συντάκτες ακολουθούν μια καθορισμένη γλώσσα προτύπων.

Οι MetaEdit+ συντάκτες περιλαμβάνουν δυνατότητες εισαγωγής και εξαγωγής XML αρχείων, ένα API για τα δεδομένα και τον έλεγχο της πρόσβασης σε λειτουργίες MetaEdit+,

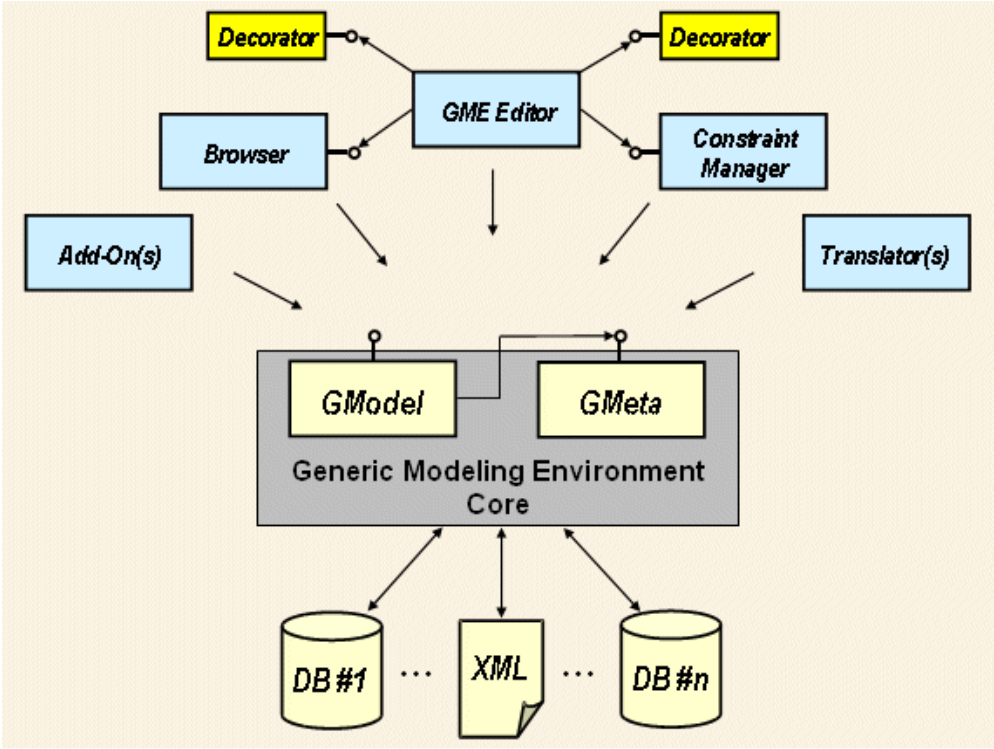
και ένα γεννήτορα κώδικα. Ο γεννήτορας κώδικα χρησιμοποιεί ΓΕΣ που επιτρέπει να καθορίσουμε τον τρόπο προσπέλασης μέσα από τα μοντέλα και την παραγωγή του περιεχομένου τους, μαζί με άλλο κείμενο. Αυτό κάνει ην υλοποίηση του γεννήτορα πολύ απλή. Χρειάζεται μόνο λίγες γραμμές κώδικα για τον ορισμό του γεννήτορα σε αντίθεση με τις κλασικές γλώσσες προγραμματισμού. Η υλοποίηση ενός γεννήτορα στις κλασικές γλώσσες προγραμματισμού απαιτεί πολύ περισσότερες γραμμές κώδικα με αποτέλεσμα η παραγωγικότητα του προγραμματιστή να μειώνεται δραματικά.

3.3.4 GME

Το Generic Modeling Environment [20] είναι μια ρυθμιζόμενη εργαλειοθήκη για τη δημιουργία μοντέλων και τη σύνθεση περιβάλλοντων για ΓΕΣ. Η διαμόρφωση επιτυγχάνεται μέσω μεταμοντέλων προσδιορίζοντας το παράδειγμα μοντελοποίησης (modeling paradigm) της περιοχής της εφαρμογής. Το παράδειγμα μοντελοποίησης περιέχει όλες τις συντακτικές, σημασιολογικές και την παρουσίαση πληροφοριών σχετικά με τον τομέα. Τέτοιες πληροφορίες είναι: α) οι έννοιες που θα χρησιμοποιηθούν για την κατασκευή μοντέλων, β) οι σχέσεις που μπορεί να υπάρχουν μεταξύ αυτών των εννοιών, γ) πώς μπορεί να οργανωθούν και να προβληθούν οι πληροφορίες από τη μοντελοποίηση των εννοιών, και δ) οι κανόνες που διέπουν την κατασκευή των μοντέλων. Το παράδειγμα μοντελοποίησης καθορίζει την οικογένεια των μοντέλων που μπορούν να δημιουργηθούν με τη χρήση του περιβάλλοντος μοντέλων. Η ΓΕΣ βασίζεται σε συμβολισμούς UML διαγράμματα κλάσης και σε αντικειμενοστραφής γλώσσες με περιορισμούς (Object Constraint Language - OCL) [21]. Τα μεταμοντέλα προσδιορίζουν το μοντέλο προσομοίωσης που χρησιμοποιείται, και δημιουργούν αυτόματα το στόχο σε συγκεκριμένους τομείς του περιβάλλοντος. Το παραγόμενο μοντέλο στη συνέχεια χρησιμοποιείται για την κατασκευή μοντέλων του τομέα που είναι αποθηκευμένα σε μια βάση δεδομένων ή σε XML μορφή. Αυτά τα μοντέλα χρησιμοποιούνται για να δημιουργηθούν αυτόματα οι εφαρμογές ή για να συνθέσουν στις εισόδους διάφορα εργαλεία ανάλυσης.

Η GME έχει μια σπονδυλωτή, επεκτάσιμη αρχιτεκτονική που χρησιμοποιεί την MS COM για την ολοκλήρωση. Η GME είναι εύκολα επεκτάσιμη: εξωτερικά συστατικά (components)

μπορεί να γραφούν σε οποιαδήποτε γλώσσα που υποστηρίζει την COM (C ++, Visual Basic, C #, Python, κ.λπ.). Η GME έχει πολλά προηγμένα χαρακτηριστικά. Ένα ενσωματωμένο διαχειριστή περιορισμών που επιβάλλει όλους τους περιορισμούς του χώρου. Η GME υποστηρίζει πολλαπλές πτυχές μοντελοποίησης. Παρέχει σύνθεση ενός μεταμοντέλου για την επαναχρησιμοποίηση του, συνδυάζοντας τις υπάρχουσες γλώσσες μοντελοποίησης και τις έννοιες της γλώσσας. Υποστηρίζει βιβλιοθήκες μοντέλων για την επαναχρησιμοποίηση αυτής σε επίπεδο μοντέλου. Όλες οι γλώσσες GME μοντελοποίησης υποστηρίζουν την κληρονομικότητα. Η οπτικοποίηση του μοντέλου είναι προσαρμόσιμη μέσω συγκεκριμένων διεπαφών. Στην Εικόνα 6 παρουσιάζεται η άποψη ενός GME.



Εικόνα 6: Άποψη ενός Generic Modeling Environment

3.3.5 EMF

Το EMF (Eclipse Modeling Framework) [22] είναι ένα πλαίσιο μοντελοποίησης (modeling framework) και γεννήτορας κώδικα για την κατασκευή εργαλείων και άλλων εφαρμογών που βασίζονται σε ένα δομημένο μοντέλο δεδομένων. Από μια προδιαγραφή του μοντέλου

που περιγράφεται με το XMI, το EMF παρέχει εργαλεία που κατά την διάρκεια της εκτέλεσης παράγει:

- κατά αντιστοιχία με το μοντέλο μια σειρά από κλάσεις σε Java,
- ένα σύνολο κατηγοριών προσαρμογέα που επιτρέπει την προβολή και την εντολή με βάση την επεξεργασία του μοντέλου,
- ένα βασικό πρόγραμμα συντάκτη.

Ο πυρήνας του EMF είναι ένα κοινό πρότυπο για τα μοντέλα δεδομένων όπου πολλές τεχνολογίες και πλαίσια το χρησιμοποιούν ως κύρια βάση. Αυτό εμπεριέχει λύσεις για εξυπηρετητές, επίμονα πλαίσια, πλαισίων διεπαφής χρήστη και υποστήριξη για μετασχηματισμούς.

Το EMF αποτελείται από τρία σημαντικά συστατικά:

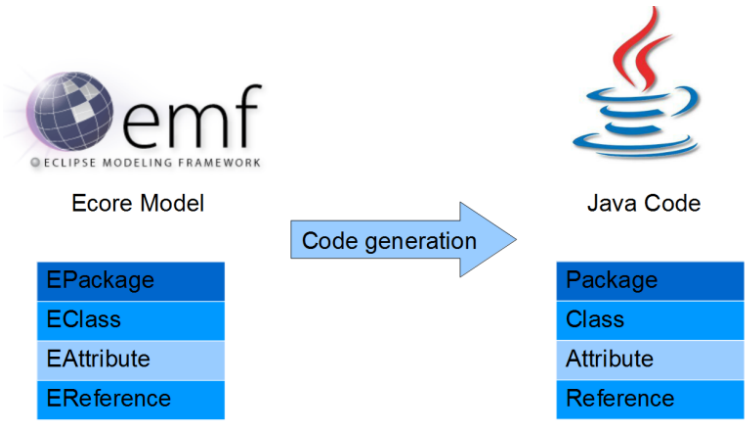
1. EMF - Ο πυρήνας του EMF πλαισίου εμπεριέχει ένα μετα-μοντέλο (Ecore), το οποίο έχει τις εξής ιδιότητες:
   - περιγράφει τα μοντέλα,
   - ελέγχει για τυχόν αλλαγές που συμβαίνουν σε ένα μοντέλο κατά τη διάρκεια της εκτέλεσης,
   - διαχειρίζεται αποτελεσματικά τα EMF αντικείμενα με τη χρήση ενός αποδοτικού API.
2. EMF.Edit - Το πλαίσιο EMF.Edit περιλαμβάνει επαναχρησιμοποιήσιμες γενικευμένες κλάσεις για τη δημιουργία ενός συντάκτη με βάση τα μοντέλα EMF.
3. EMF.Codegen - Ο γεννήτορας κώδικα του EMF μπορεί να παράξει όλα όσα χρειάζονται για να δημιουργήσουν ένα ολοκληρωμένο πρόγραμμα επεξεργασίας για ένα μοντέλο EMF. Περιλαμβάνει ένα γραφικό περιβάλλον από το οποίο μπορούμε να τροποποιήσουμε τα χαρακτηριστικά του γεννήτορα, αλλά και να εμπλέξουμε περισσότερες από ένα γεννήτορες κώδικα.

Ο γεννήτορας κώδικα υποστηρίζει τρία διαφορετικά επίπεδα

- Μοντέλο (Model) – παρέχει τις διεπαφές στην java και την υλοποίηση των κλάσεων για όλες τις κλάσεις του μοντέλου,
- Προσαρμογέας (Adapters) – δημιουργεί τις υλοποιήσεις των κλάσεων για όλες τις κλάσεις του μοντέλου,

- Συντάκτης (Editor) – παράγει έναν κατάλληλα δομημένο editor που συμμορφώνεται με τις προδιαγραφές των μοντέλων EMF της Eclipse.

Όλοι οι γεννήτορες μπορούν να εκτελεστούν είτε μέσω ενός γραφικού περιβάλλοντος χρήστη (Grafical User Interface - GUI) είτε μέσω της γραμμής εντολών και υποστηρίζουν αναπαραγωγή του κώδικα. Στην Εικόνα 7 παρουσιάζεται ένα Ecore μοντέλο το οποίο αποτελείται από ένα υποσύνολο διαγραμμάτων κλάσεων UML. Από το Ecore μοντέλο μπορούμε μέσω του γεννήτορα κώδικα να παραχθεί κώδικας σε Java.

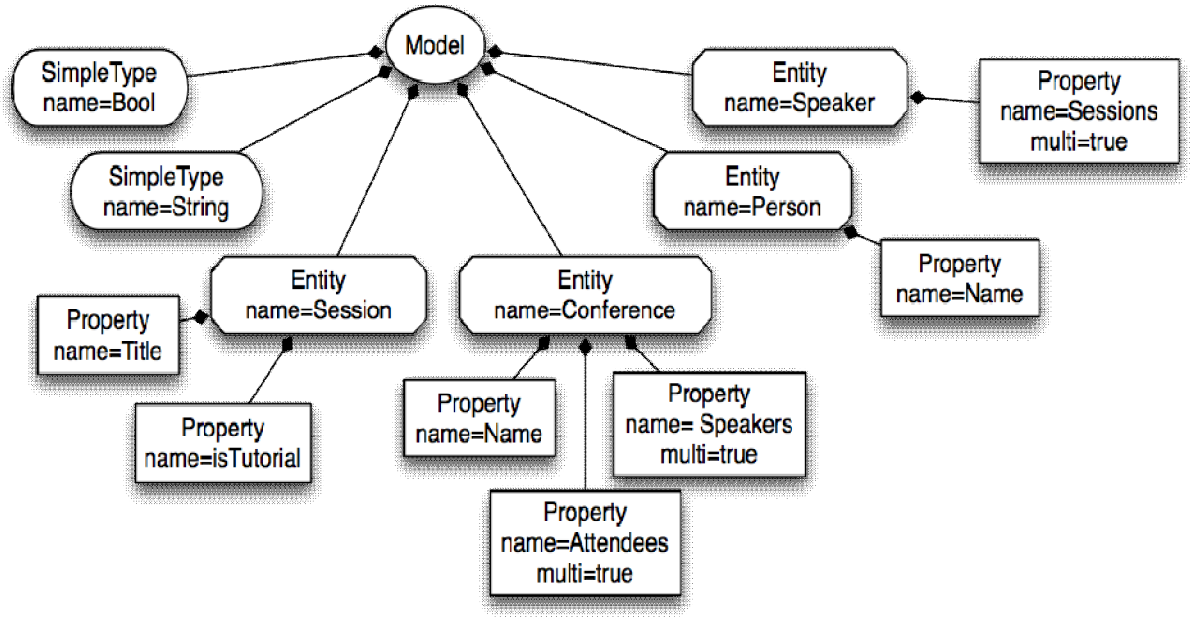


Εικόνα 7: Ecore to Java Code (Code generation)

3.3.6 Μοντέλο Ecore

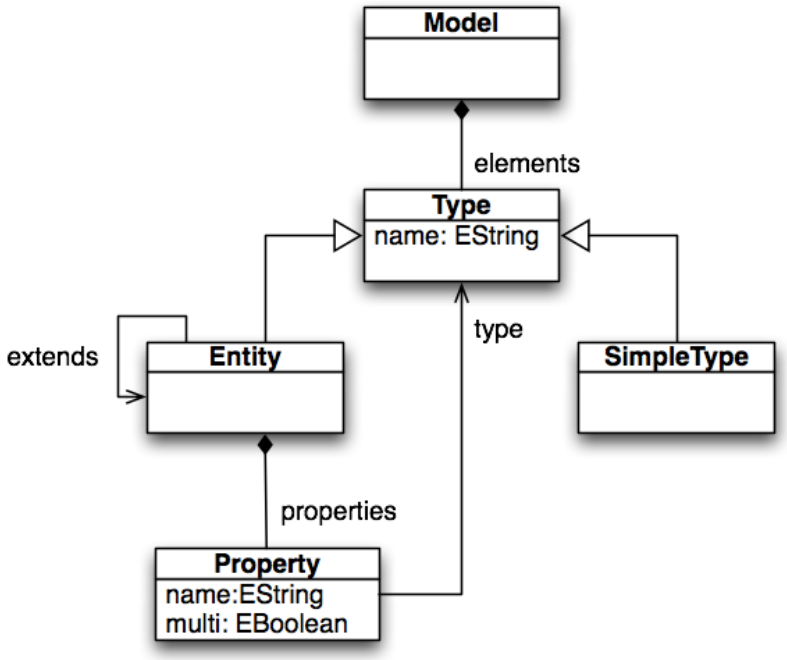
Το Xtext [23] είναι ένα πλαίσιο το οποίο χρησιμοποιείται για την ανάπτυξη ΓΓΣ και ΓΕΣ. Με το Xtext μπορούμε να ορίσουμε την γραμματική της γλώσσας για να κατασκευάσουμε μία ΓΕΣ. Το πλαίσιο αυτό αποτελεί μια πλήρη υποδομή, καθώς συμπεριλαμβάνει επίσης έναν αναλυτή, έναν συνδέτη (linker), έναν μηχανισμό ελέγχου τύπων (typechecker) και έναν μεταγλωτιστή. Το Xtext παρέχεται από τα Framework των Eclipse και IntelliJ IDEA, και υποστηρίζει την ανάπτυξη ΓΕΣ σε διαδικτυακές εφαρμογές. Το Xtext χρησιμοποιεί EMF μοντέλα, τα οποία αναλύουν το κείμενο και το αναπαριστούν στη μνήμη με τη μορφή γράφου. Αυτό το αντικείμενο του γράφου στη μνήμη ονομάζεται *αφηρημένο συντακτικό δέντρο* (Abstract Syntax Tree - AST). Επίσης μπορεί να το βρούμε και με άλλες ονομασίες

όπως *έγγραφο γράφημα αντικείμενο (DOM), σημασιολογικό μοντέλο,* ή απλά *μοντέλο.* Μια AST αναπαράσταση φαίνεται στο Εικόνα 8.



Εικόνα 8: AST αναπαράσταση

Το AST πρέπει να περιλαμβάνει όλες τις έννοιες του κειμένου του μοντέλου που δρά αφαιρετικά στο συντακτικό δέντρο. Επίσης το AST χρησιμοποιείται σε επόμενα στάδια για την επικύρωση, την μεταγλώτισση ή την διερμηνεία. Στο EMF, ένα μοντέλο αποτελείται από στιγμιότυπα *EObjects* τα οποία διασυνδέονται μεταξύ τους. Ένα EObject είναι ένα στιγμιότυπο ενός EClass. Ένα EPackage εμπεριέχει ένα σύνολο από *EClasses.* Στο Xtext, τα μετα-μοντέλα είτε δημιουργούνται από τη γραμματική είτε μπορεί να προκαθοριστούν από το χρήστη. Ένα παράδειγμα ενός μετα-μοντέλου φαίνεται στην Εικόνα 9.

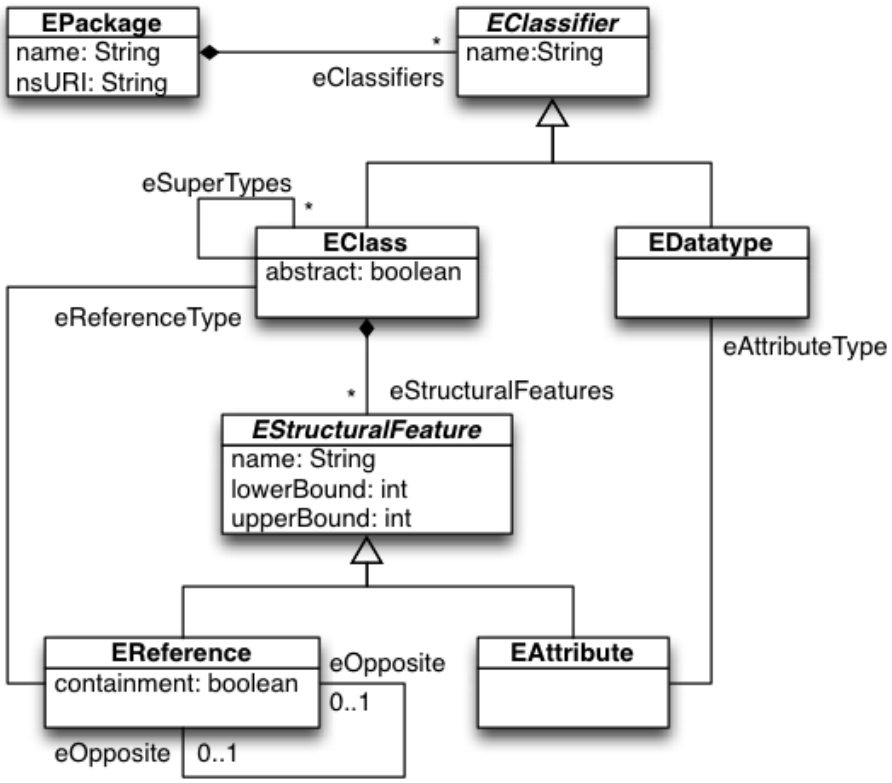


Εικόνα 9: Παράδειγμα ενός μετα-μοντέλου

Το μοντέλο της γλώσσας στο οποίο ορίζεται ένα μετα-μοντέλο ονομάζεται Ecore. Με άλλα λόγια, το μετα-μοντέλο είναι το μοντέλο Ecore της γλώσσας που δημιουργούμε. Το Ecore είναι ένα ουσιαστικό κομμάτι του EMF. Το μετα-μοντέλο ορίζει τους τύπους των σημασιολογικών κόμβων ως EClasses του Ecore. Τα EClasses εμφανίζονται ως κουτιά-θέσεις στο διάγραμμα μετά-μοντέλου. Έτσι στο παράδειγμά μας τα Model, Type, simpleType, Entity και Property είναι EClasses. Ένα EClass μπορεί να κληρονομήσει από άλλες EClasses. Πολλαπλή κληρονομικότητα επιτρέπεται, όμως φυσικά οι κυκλικές αναφορές απαγορεύονται.

Τα EClasses μπορεί να έχουν EAttributes ως απλές ιδιότητές τους. Αυτά εμφανίζονται στο εσωτερικό των κόμβων των EClasses. Το παράδειγμά μας περιέχει δύο EAttributes. Ένα EAttribute με την ονομασία «name» και ένα EAttribute με την ονομασία «isMulti». Το πεδίο που ορίζονται οι τιμές για ένα EAttribute ορίζεται από το EDataType (τύπος δεδομένων). Τα Ecore μοντέλα συνήθως εμπεριέχουν κάποια προκαθορισμένα EDataTypes, τα οποία αναφέρονται κυρίως σε πρωτόγονους τύπους δεδομένων (primmitive data types) της Java και σε κλάσεις όπως για παράδειγμα η κλάση String. Για να γίνει διάκριση από τους τύπους Java, τα EDataTypes χρησιμοποιούνε το πρόθεμα "E". Αυτό φαίνεται και στο παράδειγμά

του σχήματος, όπου στον κόμβο «Property» έχουμε το EString και το EBoolean. Τα EReferences χρησιμοποιούνται ως παραπομπές σε άλλα EClasses. Η σημαία συγκράτησης υποδηλώνει κατά πόσον ένα Ereference είναι μια αναφορά συγκράτησης ή μια παραπομπή. Στο διάγραμμα, οι αναφορές είναι τα άκρα και οι παραπομπές συγκράτησης σημειώνονται με ένα σχήμα διαμαντιού. Στο επίπεδο μοντέλο, κάθε στοιχείο μπορεί να έχει το πολύ ένα κουτάκι, δηλαδή ένα άλλο στοιχείο που αναφέρεται σε αυτό με αναφορά περιορισμού. Αυτό συνάγει μια δομή δέντρου για τα μοντέλα. Από την άλλη πλευρά, οι παραπομπές αναφέρονται σε στοιχεία που μπορεί να περιέχονται οπουδήποτε αλλού. Στο παράδειγμα αυτό, τα «elements» και τα «properties» είναι οι αναφορές συγκράτησης, ενώ το «type» και το «extends» είναι παραπομπές. Για λόγους αναγνωσιμότητας, παραλείπονται οι παραπομπές στο διάγραμμα μοντέλο του παραδείγματος. Να σημειωθεί ότι σε αντίθεση με άλλες γεννήτριες αναλυτή, το Xtext δημιουργεί ASTs με συνδεδεμένες παραπομπές. Τα EReferences στο Ecore ανήκουν πάντα σε ένα Eclass και η προσπέλαση γίνεται μονοσήμαντα από το «eclass» προς το «type». Η αμοιβαία συχέτιση μεταξύ δύο Eclasses πρέπει να μοντελοποιηθεί ως δύο αναφορές, που είναι ένα eOpposite μεταξύ τους και ανήκουν σε κάθε άκρο των ενώσεων. Η υπερκλάση των EAttributes και των EReferences είναι η κλάση EStructuralFeature και επιτρέπει να ορίσουμε ένα όνομα και μια πληθικότητα. Μπορούμε να ορίσουμε την πληθικότητα, θέτοντας ένα κατώτατο όριο και ένα ανώτατο όρια τιμής. Το ανώτατο όριο μπορεί να πάρει και την τιμή -1 που σημαίνει «απεριόριστη πληθικότητα». Ο EClassifier είναι ο κοινός υπερ-τύπος των EDataType και EClass. Ένα EPackage ενεργεί ως χώρος ονομάτων (namespaces) και ως περιέκτης των eClassifiers. Στην Εικόνα 10 συνοψίζονται οι πιο σημαντικές έννοιες του Ecore διαγράμματος:



Εικόνα 10: Παράδειγμα ecore diagram

4 Διαδικτυακά Εργαλεία

4.1 Η Προσέγγιση του Xtext

Το Xtext είναι μια ΓΕΣ που αποτελεί ένα τμήμα του eclipse modeling project. Ουσιαστικά αποτελεί μια ΓΕΣ που υλοποιεί άλλες ΓΕΣ γλώσσες και δημιουργήθηκε για να ξεπεραστούν τα προβλήματα που παρουσιάζονται στις εσωτερικές και στις εξωτερικές ΓΕΣ.

Το Xtext προσφέρει τα παρακάτω:

1. Αφαιρετικότητα της Γλώσσας (Abstraction): Για να υλοποιήσουμε μια εφαρμογή λογισμικού, θα πρέπει ουσιαστικά να τμηματοποιήσουμε τον κώδικα για να μπορούμε να τον χειριστούμε με ευκολία (divide and conquer). Γράφοντας μικρά κομμάτια κώδικα μπορούμε να χτίσουμε πολύ μεγαλύτερα κομμάτια, χρησιμοποιώντας αφηρημένες έννοιες-επίπεδα (layer abstraction). Για παράδειγμα αν θέλουμε να ορίσουμε μια μέθοδο, αρχικά της δίνουμε ένα όνομα. Η συγκεκριμένη συνάρτηση μπορεί να κληθεί πολλαπλές φορές από κάποια άλλη συνάρτηση, άλλες συναρτήσεις να καλέσουν την προηγούμενη συνάρτηση, κ.λπ. Βλέπουμε ότι η δομή μιας συνάρτησης αποτελεί ένα μικρό κομμάτι κώδικα (block of code) και τελικά όλες μαζί χτίζουν ένα πολύ μεγαλύτερο κομμάτι κώδικα που αποτελεί και το τελικό πρόγραμμα. Με το Xtext μπορούμε να χτίσουμε πολλά μικρά κομμάτια κώδικα και τελικά να διαχειριστούμε ποιό εύκολα το τελικό πρόγραμμα.

2. Καλύτερη διαχείριση κώδικα (reusability): Αντί να γράφουμε κώδικα με κάποια γλώσσα προγραμματισμού (π.χ. Java, C++) και να τον περάσουμε στην διαδικασία της σύνταξης (compilation) και της ανάπτυξης deployment, θα μπορούσαμε να γράψουμε κώδικα ο οποίος να είναι συμπεριφορικός (behavioural), να τροποποιείται κατά το χρόνο εκτέλεσης (runtime). Από μια ΓΕΣ μπορούμε πιο εύκολα να ορίσουμε τον κώδικα, λόγω της αφαίρεσης της γλώσσας, χωρίς να χρειάζεται ο μηχανικός λογισμικού να μπει σε τεχνικές λεπτομέρειες. Επίσης μια ΓΕΣ γλώσσα μπορεί να επαναχρησιμοποιηθεί αν είναι καταχωρημένη στη βιβλιοθήκη.

3. Κατανόηση των απαιτήσεων του πελάτη: Το λογισμικό το συναντάμε σε όλες τις βιομηχανίες, στο πεδίο του αυτοματισμού, σε διαφόρα άλλα επιστημονικά πεδία, κλπ. Είναι φανερό ότι στην καθημερινότητα μας έχει εισβάλει ο τομέας της πληροφορικής και χρειάζεται σε πολλά επιστημονικά και όχι μόνο πεδία να συμβάλλουν πολλοί στην

συγγραφή κώδικα για λογισμικό ή τουλάχιστον να μπορούν να ερμηνεύσουν και να πουν πως θα ήθελαν να αναπτυχθεί μία εφαρμογή στους ειδικούς του πεδίου. Όμως σε αυτό το σημείο υπάρχει ένα τεράστιο χάσμα επικοινωνίας με τους μηχανικούς λογισμικού. Στην προκειμένη περίπτωση οι μηχανικοί λογισμικού θα μπορούσαν να χρησιμοποιήσουνε μια ΓΕΣ που να περιγράφει τα πιο σημαντικά και λειτουργικά κομμάτια μέσω της ΓΕΣ, αποκρύπτοντας λεπτομέρειες που δεν καταλαβαίνει ο πελάτης. Με την ΓΕΣ ο πελάτης θα μπορούσε να κατανοήσει και να ερμηνεύσει τις προσδοκώμενες από τον πελάτη απαιτήσεις που πρέπει να έχει το λογισμικό. Με τη χρήση της κοινής αυτής διαλέκτου θα λυνόταν το πρόβλημα της επικοινωνίας. Αυτό θα έχει ως συνέπεια να μειωθεί ο χρόνος παραγωγής του προιόντος σημαντικά. Ένα παράδειγμα αποτελεί το λογισμικό για ασφαλιστικές εταιρείες όπου ο πελάτης θα πρέπει να θέσει κάποιες απαιτήσεις για τη δημιουργία του προϊόντος. Πολλές φορές οι μηχανικοί λογισμικού δεν μπορούν να συνεργαστούν και να καταλάβουν τις απαιτήσεις του πελάτη για την υλοποίηση της εφαρμογής.

3.Χρωματισμός Συντακτικών εννοιών (Syntax coloring): Το Xtext υποστηρίζει το χρωματισμό λέξεων κλειδιών στο συντάκτη που ταυτίζονται με τη γραμματική της γλώσσας. Υπάρχει η δυνατότητα επιλογής χρωμάτων από συγκεκριμένη κλάση που έχει παραχθεί από το Xtext. Προκαθορισμένο χρώμα είναι το κόκκινο.

4.Επικύρωση περιεχομένου (validation of content) : Κατά τη συγγραφή ενός σεναρίου, το περιεχόμενο του κείμενου ελέγχεται για συντακτικά και εννοιολογικά λάθη. Σε περίπτωση που υπάρχουν λάθη, ο επικυρωτής περιεχομένου μας υποδεικνύει την/τις γραμμή/ες που υπάρχουν λάθη με κόκκινο χρώμα και προβάλλει το αντίστοιχο μήνυμα λάθους. Το τελευταίο διευκολύνει στην ταχύτερη εύρεση του σφάλματος από τους χρήστες. Επιπλέον, αν το κείμενο έχει λάθη, το σενάριο δεν μπορεί να αποθηκευτεί στον αντίστοιχο αποθηκευτικό σώρο ή να εκτελεστεί.

5. Βοηθός περιεχομένου (Content assistant): Ο βοηθός περιεχομένου είναι μια από τις σημαντικότερες ιδιότητες που προσφέρει το περιβάλλον της Eclipse. Κατά τη διάρκεια της συγγραφής ενός σεναρίου, και με τη χρήση του συνδιασμού των πλήκτρων cntrl και space, το Xtextμπορεί να μας προτείνει μέσω αναδυόμενης λίστας επιλογές "λέξεων-κλειδιών" προς συμπλήρωση του κειμένου. Οι λέξεις που προβάλλονται στην αναδυόμενη λίστα είναι συνάρτηση της γραμματικής της γλώσσας μας.

6. Δένδο Διάρθρωσης (outline tree): Σε κάθε αλλαγή που πραγματοποιείται στη γλώσσα υλοποίησης, το περιβάλλον της Eclipse μας ενημερώνει απεικονίζοντας ταυτόχρονα τις αλλαγές του μοντέλου AST.

4.2 Ο Εξυπηρετητής

Η επικοινωνία μεταξύ πελάτη και εξυπηρετητή υλοποιείται με HTTP κλήσεις. Για την διαδικτυακή ολοκλήρωση (Web integration) το Xtext υποστηρίζει δύο καταστάσεις λειτουργίας όσον αφορά την αποστολή κειμένου προς τον εξυπηρετητή. Την πλήρη κατάσταση λειτουργίας (StateFul mode) και την μερική κατάσταση λειτουργίας (StateLess mode)

- **StateFul mode**: Για να ορίσουμε την κατάσταση ως StateFul, θα πρέπει αρχικά να θέσουμε την μεταβλητή sendFullText σε False state. Πλέον σε κάθε αλλαγή του κειμένου που συμβαίνει στο συντάκτη, αποστέλλεται ένα αίτημα ενημέρωσης προς τον εξυπηρετητή. Ο εξυπηρετητής λαμβάνει αντίγραφο του τροποποιημένου κείμενου. Στην συνέχεια οι υπηρεσίες του Xtext επιτελούν την ανάλυση και την επικύρωση του AST. Η απάντηση από τον εξυπηρετητή αποστέλλεται πίσω στην εφαρμογή. Το κείμενο που λαμβάνεται από τον εξυπηρετητή είναι πολύ μικρό, το οποίο σημαίνει ότι οι υπηρεσίες εκτελούνται πολύ γρήγορα. Αποτέλεσμα είναι να έχουμε βέλτιστους χρόνους απόκρισης.
- **Stateless mode**: Αυτή η κατάσταση είναι ενεργή μόνο όταν θέσουμε την μεταβλητή sendFullText σε κατάσταση true. Σε αυτή την περίπτωση δεν αποστέλλεται το τροποποιημένο κείμενο στον εξυπηρετητή μόνο, αλλά ολόκληρο το κείμενο του συντάκτη. Το κείμενο περνάει ολόκληρο από τον αναλυτή (parser) με αποτέλεσμα ο χρόνος απόκρισης να αυξηθεί αισθητά. Αυτήν την καθυστέρηση τη βιώνει ο χρήστης, αν θέλουμε να ανεβάσουμε τον πήχη της αξιοπιστίας.

Η επεξεργασία των δεδομένων και η διαχείριση του κύκλου ζωής της γλώσσας πραγματοποιείται από την κλάση XtextServlet. Η επικοινωνία αυτή επιτυγχάνεται με τη δημιουργία ενός εμβόλιμου κώδικα (injector code) της γλώσσας κατά την εκκίνηση του servlet και της καταχώρησης του στην κλάση IResourceServiceProvider.Registry.

4.3 Η Εφαρμογή του Πελάτη

Το Xtext υποστηρίζει τρεις βιβλιοθήκες σε JavaScript:

- Orion
- Ace
- CodeMirror

Στην Εικόνα 11 παρουσιάζονται οι τεχνολογίες με τις αντίστοιχες υπηρεσίες που υποστηρίζει η κάθε μια βιβλιοθήκη.

	Orion	Ace	CodeMirror
Content assist	✓	✓	✓
Validation	✓	✓	✓
Syntax highlighting	✓	✓	✓
Semantic highlighting	✓		✓
Mark occurrences	✓	✓	✓
Hover information	✓		
Formatting	✓	✓	✓
Code generator	✓	✓	✓
Persistence	✓	✓	✓

Εικόνα 11: Services που υποστηρίζουν οι αντίστοιχες βιβλιοθήκες JavaScript

Αντίστοιχα για να χρησιμοποιήσουμε μια από τις τρεις βιβλιοθήκες μέσα στο Xtext, θα πρέπει να ενσωματώσουμε το αντίστοιχο διαδικτυακό αρχείο της Java (WebJar) στο Xtext και να παραμετροποιήσουμε αντίστοιχα την συνάρτηση require που εμφανίζεται στην κεντρική σελίδα. Ένα παράδειγμα για τη χρήση του Ace στο Xtext, παρατίθεται στην Εικόνα 12 .

```
require.config({
    paths: {
        "jquery": "webjars/jquery/<version>/jquery.min",
        "ace/ext/language_tools": "webjars/ace/<version>/src/ext-language_tools",
        "xtext/xtext-ace": "xtext/<version>/xtext-ace"
    }
});
require(["webjars/ace/<version>/src/ace"], function() {
    require(["xtext/xtext-ace"], function(xtext) {
        xtext.createEditor();
    });
});
```

Εικόνα 12: Στιγμιότυπο κώδικα για την ενσωμάτωση του WebJar

JavaScript API

Η βιβλιοθήκη της Javascript ενσωματώνει πολλές συναρτήσεις για να διαχειριστούμε και να προσαρμόσουμε το συντάκτη κατά βούληση. Οι πιο βασικές συναρτήσεις είναι:

- xtext.createEditor (options)**:** μπορούμε να δημιουργήσουμε ένα ή περισσότερα στιγμιότυπα του συντάκτη,
- xtext.createServices (editor, options)**:** αρχικοποιούμε τις υπηρεσίες του συντάκτη,
- xtext.removeServices (editor): διαγράφουμε τις υπηρεσίες του συντάκτη

Η παράμετρος options μπορεί να χρησιμοποιηθεί μόνο με τις συναρτήσεις createEditor και createServices. Το option πρέπει να δηλωθεί με τη χρήση ενός αντικειμένου που να περνάει μέσα από τις συναρτήσεις, όπως φαίνεται στο στιγμιότυπο του κώδικα που ακολουθεί στην Εικόνα 13.

```
xtext.createEditor({
    resourceId: "example.statemachine",
    syntaxDefinition: "xtext/ace-mode-statemachine"
});
```

Εικόνα 13: Δείγμα παραμετροποίηση της συνάρτηση createEditor

Λήψη του περιεχομένου του κειμένου

Η λήψη του περιεχομένου του κειμένου μπορεί να γίνει με δύο διαφορετικές τεχνικές. Μπορεί να φορτωθεί από τον εξυπηρετητή του Xtext, ή να παρέχεται μέσω της JavaScript. Σε κάθε περίπτωση, ο διακομιστής Xtext πρέπει να αναγνωρίσει τη γλώσσα που θα χρησιμοποιηθεί, ούτως ώστε να μπορεί να επεξεργαστεί οποιοδήποτε αίτημα. Μια γλώσσα συνήθως προσδιορίζεται με μια επέκταση αρχείου ή έναν τύπο περιεχομένου. Η επέκταση του αρχείου μπορεί να καθοριστεί με την επιλογή xtextLang, ενώ ο τύπος περιεχομένου δίνεται με την επιλογή ContentType.

Φόρτωση κειμένου από το διακομιστή του Xtext

Σε αυτήν την περίπτωση θα πρέπει να ορίσουμε μια τιμή στα options του resourceID. Η τιμή αυτή προσδιορίζει η επέκταση του αρχείου (file extension) της γλώσσας που πρόκειται να χρησιμοποιηθεί. Σε αυτήν την περίπτωση δεν ειναι απαραίτητο να προσδιοριστεί μια επιλογή για τη γλώσσα (xtextlang). Στην συνέχεια ο εξυπηρετητής φορτώνει την γλώσσα.

Φόρτωση κειμένου μέσω JavaScript

Το Api της JavaScript μας προσφέρει τη δυνατότητα να πάρουμε το περιεχόμενο του κειμένου χωρίς να χρησιμοποιήσουμε το επίμονο επίπεδο του εξυπηρετητή. Για παράδειγμα, το κείμενο μπορεί να αναπαραχθεί στον πελάτη μπορεί ή να το ζητηθεί μέσω μιας υπηρεσίας. Αν ο συντάκτης εμπεριέχει είδη κείμενο, τότε το κείμενο αυτό αποτελεί το περιεχόμενο του συντάκτη. Με τη χρήση του API μπορούμε να εισάγουμε κείμενο στο συντάκτη. Απαραίτητο είναι να θέσουμε την επιλογή **loadFromServer** σε **false**.

Καταστάσεις Λειτουργίας

Όπως έχουμε ήδη αναλύσει πιο πάνω, για την διαδικτυακή ολοκλήρωση το Xtext υποστηρίζει δύο καταστάσεις λειτουργίας όσον αφορά την αποστολή κειμένου προς τον εξυπηρετητή. Την πλήρη κατάσταση λειτουργίας (StateFul mode) και την μερική κατάσταση λειτουργίας (StateLess mode).

Επισήμανση Συντακτικού

Η JavaScript υποστηρίζει και στις τρεις βιβλιοθήκες του API την επισήμανση της σύνταξης στο συντάκτη. Η κλάση **WebIntegrationFragment** υποστηρίζει την επισήμανση της γραμματικής για συγκεκριμένες λέξεις κλειδιά ή για βασικά σύμβολα (tokens) (π.χ. αριθμοί,

σχόλια). Μπορούμε να επέμβουμε στην προκαθορισμένη επισήμανση του συντακτικού και να το τροποποιήσουμε (π.χ. να προσθέσουμε δικές μας λέξεις κλειδιά). Το μονοπάτι στο οποίο δηλώνεται η επισήμαση του συντακτικού είναι η επιλογή (option) **syntaxDefinition**. Αν δεν οριστεί το μονοπάτι, αυτομάτως δημιουργείται από την επιλογή **xtextLang**.

Εμπλοκή των Services

Οι λειτουργίες **createEditor** και **createServices** είναι υπηρεσίες που εμπλέκονται αυτόματα κατά τη δημιουργία ενός στιγμιότυπου του συντάκτη. Ωστόσο, όλες οι υπηρεσίες μπορούν επίσης να οριστούν με χρήση των αντίστοιχων συναρτήσεων της JavaScript. Αρκεί να χρησιμοποιήσουμε τις υπηρεσίες του Xtext για το DOM στοιχείο το οποίο διασυνδέεται με ένα στιγμιότυπο του συντάκτη. Ένα παράδειγμα χρήσης των υπηρεσιών φαίνεται στην Εικόνα 14. Το περιεχόμενο του συντάκτη αποθηκεύεται κατά την ενεργοποίηση του στοιχείου (element) button της html σελίδας που έχει id το "save-button".

```
1. var editor = xtext.createEditor();
2. $("#save-button").click(function() {
3.     editor.xtextServices.saveResource();
4. });
```

Εικόνα 14: Παράδειγμα εμπλοκής των υπηρεσιών

Όπως προαναφέρθηκε, υπάρχει ένα σύνολο συναρτήσεων που υποστηρίζει η κάθε μια από τις τρείς βιβλιοθήκες (Orion, Ace και CodeMirror) για να τροποποιήσουμε τις υπηρεσίες κατά βούληση. Οι συναρτήσεις θα πρέπει να εφαρμοστούν με μία παραμετροποίηση της επιλογής (option). Με αυτό τον τρόπο υπερκαλύπτουμε τα προκαθορισμένες επιλογές που είχαν οριστεί αρχικά με τη δημιουργία του συντάκτη. Ακολουθεί ο πίνακας 1 με τις σημαντικότερες συναρτήσεις.

Πίνακας 1: Πίνακας με τις σημαντικότερες συναρτήσεις των βιβλιοθηκών της JavaScript

Συνάρτηση (function)	Ενέργειες
getContentAssist()	Επιστρέφει τον βοηθό περιεχομένων
validate()	Επιστρέφει το αποτέλεσμα του επικυρωτή περιεχομένου. Το αντικείμενο που επιστρέφεται έχει την ιδιότητα issues, η

	οποία είναι ένας πίνακας από αντικείμενα που έχει τις παρακάτω ιδιότητες: description: μια περιγραφή που απεικονίζεται στο χρήστη. severity: εμφανίζει ένα από τα τρία παρακάτω 'error', 'warning', ή 'info'. line: η γραμμή στην οποία υπάρχει ένα συμβάν. column: η στήλη στην οποία υπάρχει κάποιο συμβάν. offset: η απόσταση του χαρακτήρα από την αρχή του κειμένου. Η αρχή του κειμένου είναι το 0. length: το μήκος της περιοχής του κειμένου που έχει δεχθεί επίδραση.
computeHighlighting()	Επιστρέφει το στυλ του δεδομένου του κειμένου.
getOccurrences()	Επιστρέφει το συμβάν ενός στοιχείου στη συγκεκριμένη θέση του κέρσορα.
getHoverInfo()	Επιστρέφει την πληροφορία στη συγκεκριμένη θεση που βρίσκεται ο κέρσορας.
format()	Επιστρέφει το προεπιλεγμενο κείμενο.
generate()	Επιστρέφει όλο το κείμενο που έχει δημιουργηθεί στον συντάκτη. Αν έχουν δημιουργηθεί περισσότερα από ένα κείμενα, τότε επιστρέφεται ένα αντικείμενο με την ιδιότητα "documents". Το αντικείμενο αυτό είναι ένας πίνακας από αντικείμενα που έχουν τις ιδιότητες name, contentType και content.
loadResource()	Φορτώνει το κείμενο στο συντάκτη.
saveResource()	Αποθηκεύει την τρέχουσα κατάσταση του κειμένου του συντάκτη.
revertResource()	Επιστρέφει το κείμενο σε μια προηγούμενη κατάσταση πριν την τελευταία αποθήκευση, και επιστρέφει ένα αντικείμενο με τις ιδιότητες fullText και dirty.
update()	Υπολογίζει τη διαφορά της τελευταίας έκδοσης του κειμένου με την τελευταία έκδοση που είναι αποθηκευμένη στον εξυπηρετητή. Αν υπάρχει διαφορά μεταξύ των δύο

	εκδόσεων, τότε αποστέλεται ένα αίτημα ενημέρωσης στον εξυπηρετητή για να ανανεώσει την έκδοσή του με τα νέα δεδομένα.

Ακολοθουθεί η λίστα με τις επιλογές (Options) στον Πίνακα 2

Πίνακας 2: Πίνακας με τις επιλογές options

Επιλογες (options)	Επεξήγηση
baseUrl	Το τμήμα της μονοπατιού που βρίσκεται η υπηρεσία του Xtext (προεπιλεγμένο: '/').
contentAssistCharTriggers	Χαρακτήρες που εμπλέκουν την υπηρεσία του βοηθού περιεχομένου κατά τη διάρκεια της πληκτρολόγησης κειμένου. Αυτή η επιλογή ειναι διαθέσιμη μόνο για τη χρήση του με τη βιβλιοθήκη Orion.
contentAssistExcludedStyles	Εξαίρεση του στυλ για χαρακτήρες που πυροδοτούνται.
contentType	Ο τύπος του περιεχομένου που περιλαμβάνεται στο αίτημα προς τον εξυπηρετητή του Xtext.
dirtyElement	Το στοιχείο το οποίο έχει γράψει στην κλάση τροποποιημένης κατάστασης (dirty status), ενώσο ο συντάκτης είχε ήδη τροποποιηθεί .
dirtyStatusClass	Ένα CSS όνομα κλάσης που γράφτηκε μέσα στην επιλογή **dirtyElement** όσο ο συντάκτης ήταν ήδη τροποποιημένος.
document	Το κείμενο που έχει οριστεί. Αν δεν έχει οριστεί τότε λαμβάνεται υπ'όψιν όλο το κείμενο ως επιλεγμένο.
enableContentAssistService	Ενεργοποίηση της υπηρεσίας του βοηθού περιεχομένου**.**
enableFormattingAction	Ενεργοποίηση του συντόμευσης του συνδυασμού των πλήκτρων

	(Ctrl+Shift+F/Cmd+Shift+F) για την πράξη της μορφοποίησης του κειμένου.
enableFormattingService	Ενεργοποίηση της υπηρεσίας μορφοποίησης του κειμένου.
enableGeneratorService	Ενεργοποίηση της υπηρεσίας του γεννήτορα κώδικα**.**
enableHoverService	Ενεργοποίηση της υπηρεσίας **Hover**. Όταν το ποντίκι αιωρείται πάνω από τμήμα του κειμένου απεικονίζεται μια πληροφορία στο χρήστη.
enableHighlightingService	Ενεργοποίηση επισήμανσης σε σημασιολογικές έννοιες του κειμένου.
enableOccurrencesService	Ενεργοποίηση επισήμανσης σε επιλεγμένο κείμενο.
enableSaveAction	Ενεργοποίηση της συντόμευσης του συνδυασμού των πλήκτρων (Ctrl+S/Cmd+S) για την αποθήκευση του κειμένου.
enableValidationService	Ενεργοποίηση της υπηρεσίας επικύρωσης περιεχομένου
loadFromServer	Φόρτωση του περιεχομένου του εξυπηρετητή στον συντάκτη. Όταν η επιλογή "resourceId¨ είναι set. διαφορετικά είναι false, και δεν μπορεί να χρησιμοποιηθεί αυτή η επιλογή.
mouseHoverDelay	Ο χρόνος σε msec για την απεικόνιση της πληροφορίας, μετά την αιώρηση του ποντικιού πάνω από τμήμα του κειμένου.
parent	Το γονικό στοιχείο στο οποίο δημιουργείται ο συντάκτης. Μπορεί να είναι ένα στοιχείο DOM ή ένα id ενός DOM στοιχείου (default : ”xtext-editor”).
parentClass	Αν δεν έχει δωθεί η επιλογή "parent" , τότε αυτομάτως η επιλογή χρησιμοποιείται για να βρεθούν στοιχεία που ταιριάζουν με το όνομα κλάσης του Xtext -editor.

resourceId	Ο προσδιοριστής του πόρου που εμφανίζεται στον συντάκτη .
selectionUpdateDelay	Ο χρόνος καθυστέρησης σε msec της εμπλοκής των υπηρεσιών μετά την τελευταία επιλογή του κειμένου.
sendFullText	Επιλογή για μαζική αποστολή του κειμένου στον εξυπηρετητή ή αποστολή μέρος του κειμένου που έχει τροποποιηθεί στο συντάκτη (update request). Αυτή η επιλογή ορίζει την κατάσταση λειτουργίας του εξυπηρετητή.
serviceUrl	Το URL του Xtext servlet.
showErrorDialogs	Αναδυόμενα μηνύματα (Popup dialogues) που εμφανίζονται στον συντάκτη στην περίπτωση που υπάρχουν λάθη στο κείμενο.
syntaxDefinition	Το μονοπάτι όπου ορίζεται προς ένα αρχείο JavaScript, για να ορίσουμε την επισήμανση στην σύνταξη.
textUpdateDelay	Ο χρόνος καθυστέρησης σε msec της κλήσης των υπηρεσιών μετά την τελευταία αλλαγή του κειμένου.
xtextLang	Το όνομα της γλώσσας.

5 Μελέτη Περίπτωσης

Στα πλαίσια αυτής της Διπλωματικής Εργασίας, αναπτύσσουμε ένα Διαδικτυακό περιβάλλον ανάπτυξης εφαρμογών με χρήση μιας ΓΕΣ για την απομακρυσμένη διαχείριση IoT ρομποτικών συσκευών (π.χ., Unmanned Aerial Vehicles - UAV, Unmanned Ground Vehicles - UGV, Unmanned Surface Vehicles - USV). Το περιβάλλον αναπτύχθηκε με τη χρήση των γλωσσών/εργαλείων OpenLayers, Xtext, JavaScript, Html και css. Με τη βοήθεια του Xtext αναπτύχθηκε μια πειραματική γλώσσα πεδίου (EDL – **E**xperiment **D**escription **L**anguage) [25], μέσω της οποίας ένας χρήστης έχει τη δυνατότητα να δημιουργήσει σενάρια πειραμάτων που πρόκειται να εκτελεστούν από τις αυτόνομες ρομποτικές συσκευές. Για το σκοπό αυτό δημιουργήθηκαν ένας συντάκτης κειμένου και ένας οπτικός συντάκτης. Στο συντάκτη κειμένου ο χρήστης μπορεί να συγγράψει ένα πείραμα με χρήση των εντολών που έχουν οριστεί με τη βοήθεια της EDL ενώ στον οπτικό συντάκτη ο χρήστης μπορεί να δημιουργήσει ένα σενάριο χρησιμοποιώντας ένα χάρτη πάνω στον οποίο μπορεί να ορίσει τις θέσεις και τα χαρακτηριστικά των κινητών κόμβων. Το OpenLayers χρησιμοποιήθηκε για την απεικόνιση των χαρτών στον οπτικό συντάκτη (Visual editor). Το OpenLayers είναι μια JavaScript βιβλιοθήκη ανοιχτού κώδικα που επιτρέπει την προβολή χαρτών σε περιηγητές του ιστού (web browsers). Παρέχει μια προγραμματιστική διεπαφή (API) για την υλοποίηση πλούσιων διαδικτυακών γεωγραφικών εφαρμογών όπως είναι το Google Maps και το Bing Maps. Η αρχική υλοποίηση της βιβλιοθήκης βασίστηκε στο Prototype JavaScript Framework.

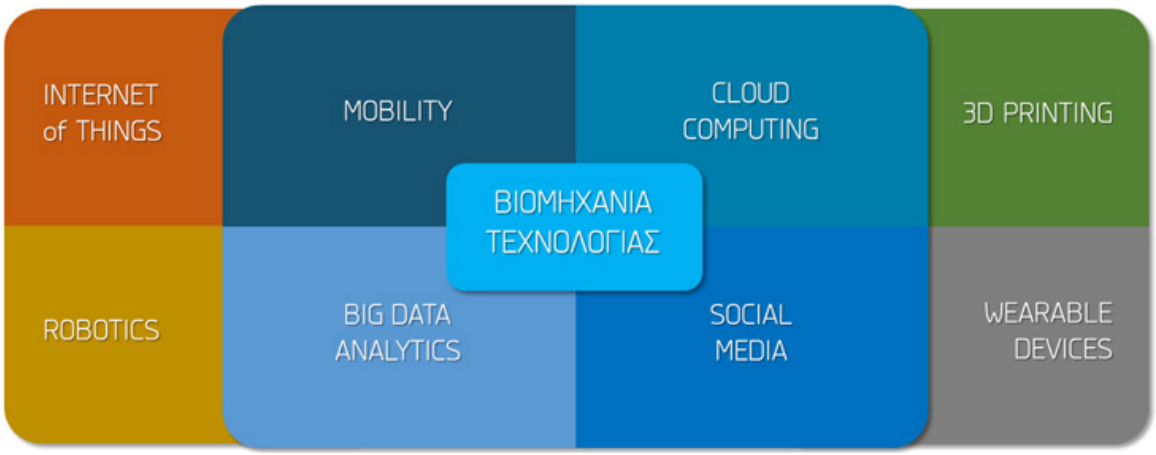
5.1 Mobile IoT Εφαρμογές

Το Διαδίκτυο των Πραγμάτων (ΔτΠ - Internet of Things - IoT) [26] είναι μία έννοια που αφορά τα αντικείμενα της καθημερινότητας μας, από βιομηχανικές μηχανές μέχρι και συσκευές που φοράει κάποιος (wearable devices) που χρησιμοποιούν ενσωματωμένους αισθητήρες για τη συλλογή και την επεξεργασία δεδομένων, την αποστολή πληροφοριών προς ομότιμες συσκευές ή σε κεντρικά συστήματα. Προυπόθεση είναι οι συσκευές να διασυνδέονται μέσω ενός δικτύου επικοινωνίας (wifi, ethernet, bluetooth). Για παράδειγμα,

ένα κτίριο που χρησιμοποιεί αισθητήρες (sensors) για την αυτόματη ρύθμιση της θέρμανσης ή του φωτισμού λειτουργεί με παραπλήσιο τρόπο. Άλλο παράδειγμα είναι όταν ένας εξοπλισμός παραγωγής προειδοποιεί το προσωπικό συντήρησης για μία επικείμενη βλάβη. Στην Εικόνα 15 παρουσιάζονται συσκευές της καθημερινότητας που χρησιμοποιούνται σε IoT εφαρμογές.

Εικόνα 15: IoT εφαρμογές

Το ΔτΠ επινοήθηκε στα τέλη της δεκαετίας του 1990 από τον επιχειρηματία Kevin Ashton. Ο Ashton [27], ο οποίος είναι ένας από τους ιδρυτές του Auto-ID Center στο MIT, ήταν μέρος μιας ομάδας που ανακάλυψε τον τρόπο να συνδέσει τα αντικείμενα με το Διαδίκτυο μέσω μιας ετικέτας RFID. Για πρώτη φορά χρησιμοποιήθηκε η φράση “Internet of Things” το 1999 και έκτοτε ο όρος καθιερώθηκε παγκοσμίως. Στην Εικόνα 16 παρουσιάζεται η άποψη της Βιομηχανίας της Τεχνολογίας σήμερα.



Εικόνα 16: Άποψη της Βιομηχανίας Τεχνολογίας

Το ΔτΠ βασίζεται στην τεχνολογία M2M, δηλαδή στην επικοινωνία μεταξύ μηχανών καθώς και στην τεχνολογία αναγνώρισης των αντικειμένων μέσω ραδιοσυχνοτήτων (RFID). Το ΔτΠ μπορεί να χρησιμοποιηθεί σε όλους του τομείς της καθημερινής μας ζωής. Το κινητό (mobile) ΔτΠ είναι φορητές έξυπνες συσκευές, που φέρουν διάφορα αισθητήρια, ικανοποιητική για τα σημερινά δεδομένα αυτονομία ενέργειας, μικρή επεξεργαστική ισχύ και ενσωματόνουν διάφορες τεχνολογίες επικοινωνίας (BT, Wi-Fi, GPS). Τα ΚΔτΠ συλλέγουν, επεξεργάζονται και ανταλλάσουν πολλά δεδομένα με ομότιμους κόμβους ή με κεντρικά συστήματα. Καινοτόμες εφαρμογές θα μπορούσαν να κατασκευαστούν για τα ΚΔτΠ, με απώτερο σκοπό την διευκόλυνση της ζωής των ανθρώπων. Οι εφαρμογές για ΚΔτΠ είναι πάρα πολλές. Μερικές από τις εφαρμογές θα μπορούσαν να είναι ή λήψη δεδομένων σε πραγματικό χρόνο κατά την εφαρμογή πειραμάτων σε διάφορους ερευνητικούς τομείς. Στον Πίνακα 3 παρουσιάζεται η χρήση των ΚΔτΠ σε διάφορους τομείς

Πίνακας 3: Η χρήση των ΚΔτΠ σε διάφορους τομείς

Τομέας	Πεδία
Smart Cities	Smart Parking Structural health Smartphone Detection Traffic Congestion Smart Lighting

	Waste Management
Smart Environment	Forest Fire Detection Air Pollution Earthquake Early Detection
Smart Water	Potable water monitoring Chemical leakage detection in rivers River Floods
Smart Metering	Smart Grid Silos Stock Calculation
Security & Emergencies	Perimeter Access Control Radiation Levels Explosive and Hazardous Gases
Industrial Control	M2M Applications Temperature Monitoring Vehicle Auto-diagnosis
Domotic & Home Automation	Energy and Water Use Remote Control Appliances Intrusion Detection Systems Art and Goods Preservation
Retail	Supply Chain Control NFC Payment Intelligent Shopping Applications Smart Product Management
Logistics	Quality of Shipment Conditions Item Location Storage Incompatibility Detection Fleet Tracking
Smart Agriculture	Wine Quality Enhancing Green Houses Meteorological Station Network

Smart Animal Farming	Hydroponics Offspring Care Animal Tracking
eHealth	Fall Detection Medical Fridges Sportsmen Care Patients Surveillance

Μέχρι το τέλος του 2016 θα υπάρχουν έξι δισεκατομμύρια συσκευές με δυνατότητες συνδεσιμότητας, ενώ, μέχρι το τέλος του 2020, ο αριθμός αυτός θα φτάσει τα 25 δισεκατομμύρια. Αυτό σημαίνει ότι κάθε άνθρωπος στον πλανήτη θα διαθέτει κατά μέσο όρο τρεις συσκευές που μπορούν να υποστηρίξουν το ΔτΠ. Το ίντερνετ του μέλλοντος είναι το διαδίκτυο των αντικειμένων που όπως εκτιμάται θα φέρει επανάσταση στην ανθρώπινη αλληλεπίδραση με το περιβάλλον, στον ίδιο μάλιστα βαθμό που το Διαδίκτυο προκάλεσε επανάσταση στον τρόπο ζωής ανθρώπων και επιχειρήσεων.

5.2 Γλώσσα Ειδικού Σκοπού για Εφαρμογές του ΚΔτΠ

Η EDL είναι μια ΓΕΣ για τη δημιουργία πειραματικών σεναριών στον τομέα του ΚΔτΠ. Η EDL έχει σχεδιαστεί με στόχο να βοηθήσει τους ειδικούς του χώρου (domain experts) και τους μη έμπειρους χρήστες (experimenters), στην αποτελεσματική συγγραφή σεναρίων πειραματισμού. Ο κύριος στόχος της EDL είναι η παροχή υψηλού επιπέδου αφαίρεσης που προστατεύει τους ερευνητές από την πολυπλοκότητα της υποκείμενης πλατφόρμας και των διαθέσιμων συσκευών. Η EDL παρέχει στοιχεία για το χειρισμό των απαιτήσεων σε πόρους/διαμόρφωση, τη θέση/πληροφορίες τοπολογίας, περιγραφή των εργασιών, κλπ και η σύνταξη της είναι απλή και συνδυάζει ορισμένα κοινά χαρακτηριστικά των γλωσσών προγραμματισμού XML. Η EDL είναι χτισμένη με τη βοήθεια της σουίτας Xtext. Στην Εικόνα 17 παρουσιάζεται ένα μικρό μέρος της EDL γραμματικής.

```
grammar eu.rawfie.edlWeb.Edl with org.eclipse.xtext.common.Terminals

generate edl "http://www.rawfie.eu/edlWeb/Edl"

Experiment:
        'Experiment'
            metadata = MetadataSection
            (requirements=RequirementsSection)?
            (declarations=DeclarationsSection)?
            execution=ExecutionSection
        '~Experiment'
;

/********** Metadata Section **********/
MetadataSection:
    'Metadata'
         met+=Metadata
    '~Metadata';

Metadata:
    'Name' name = ID
    (experimentVersion=Version)?
    (experimentDescritpion=Description)?
    (experimentDate=Date)?;

Version:
    'Version' ver=VER;

Description:
    'Description' name=ID;

Date:
    'Date' dat=DAT;

/********** Requirements Section **********/
RequirementsSection:
    'Requirements'
        ('Nodes' noNodes=NUMBER)?
        ('Testbed' testbed=ID)?
        ('Location('location=Coordinates')')?
        ('Duration' duration=NUMBER)?
        ('MinDistance' minDistance=NUMBER)?
        ('MaxDistance' maxDistance=NUMBER)?
    '~Requirements'
;

Coordinates:
    coordX=CVER ', 'coordY=CVER;

/********** Declarations Section **********/
```

Εικόνα 17: Δείγμα EDL γραμματικής

Ένα EDL πείραμα αποτελείται από τα εξής μέρη:

• **Metadata section** - τμήμα μεταδεδομένων. Περιέχει γενικές πληροφορίες που σχετίζονται με κάθε πείραμα, όπως το όνομα, την ημερομηνία, κλπ. Η πληροφορία της περιγραφής ενός πειράματος είναι σημαντική, ώστε να γίνεται καλύτερη διαχείριση των διαθέσιμων πειραμάτων.

• ***Requirements section -*** τμήμα απαιτήσεων. Περιέχει πληροφορίες που σχετίζονται με τις απαιτήσεις του κάθε πειράματος όσον αφορά τα δεδομένα της πλατφόρμας δοκιμών (Testbed), τη θέση, τη διάρκεια ή την απόσταση που οι κόμβοι πρέπει να καλύψουν κατά την εκτέλεση ενός πειράματος. Ο ερευντητής θα πρέπει επιπλέον σε αυτό το τμήμα να καθορίσει τον αριθμό των κόμβων που θα εμπλακούν στο πείραμα, έτσι ώστε η πλατφόρμα να γνωρίζει τις ανάγκες του υλικού (UxV) που θα χρησιμοποιηθούν.

• ***Declarations section***. - τμήμα δηλώσεων. Αφορά τις απαραίτητες δηλώσεις όπως οι σταθερές και οι μεταβλητές που θα χρησιμοποιηθούν για την αποθήκευση των δεδομένων κατά τη διάρκεια της εκτέλεσης του πειράματος. Οι δηλώσεις των σταθερών και των μεταβλητών είναι το βασικό στοιχείο της διασύνδεσης του τμήματος με τα δεδομένα που ανακτώνται από UxVs.

• **Execution section** - τμήμα εκτέλεσης. Περιλαμβάνει εντολές που σχετίζονται με τη διαχείριση της λογικής του κάθε πειράματος. Η EDL προσφέρει δηλώσεις για τους κόμβους ή ομάδες κόμβων. Κάθε πτυχή των κόμβων / συμπεριφορά των ομάδων μπορεί να πραγματοποιηθεί με συγκεκριμένη ορολογία στο τμήμα εκτέλεσης. Επιπλέον, μια σειρά από καταστάσεις είναι αφιερωμένες:

- στη διαχείριση των σημείων των διαδρομών που εκτελούν οι κόμβοι,
- στη διαχείριση της γραμμής του χρόνου (π.χ., διαδοχική ή παράλληλη εκτέλεση),
- στη διαχείριση του συντονισμού των κόμβων,
- στη διαχείριση του ελέγχου των κόμβων (π.χ., ενεργοποίηση / απενεργοποίηση των αισθητήρων),
- στη διαχείριση των δεδομένων σε κάθε κόμβο,
- στη διαχείριση της επικοινωνίας (π.χ. αλλαγή στις διεπαφές του δικτύου).

Θα πρέπει να σημειωθεί ότι εντολές που προέρχεται από ‘τυπικές’ γλώσσες προγραμματισμού, περιλαμβάνονται επίσης στην EDL. Ως εκ τούτου, υπάρχουν οι αναθέσεις, οι δηλώσεις επιλογής (δηλαδή, if statement) και οι επαναλήψεις (δηλ., for, while). Στην Εικόνα 18, παρουσιάζεται ένα μικρό μέρος ενός σεναρίου EDL που συνδέεται με τον ορισμό της συμπεριφοράς ενός κόμβου. Η εντολή ‘Route’ του κόμβου εμπεριέχει έναν αριθμό σημείων διαδρομής και δηλώνονται με την ακολουθία των εντολών ‘WP’. Κάθε

σημείο διαδρομής απαρτίζεται από τέσσερις αριθμούς: το χρόνο, και τις συνταταγμένες x, y και z. Για παράδειγμα, η εντολή WP <3, 50, 22, 15> επιβάλλει στον κόμβο 1, στην 3η χρονική μονάδα, να μετατοπιστεί στη θέση (x=50, y= 22), στο ύψος / βάθος 15 μετρικές μονάδες.

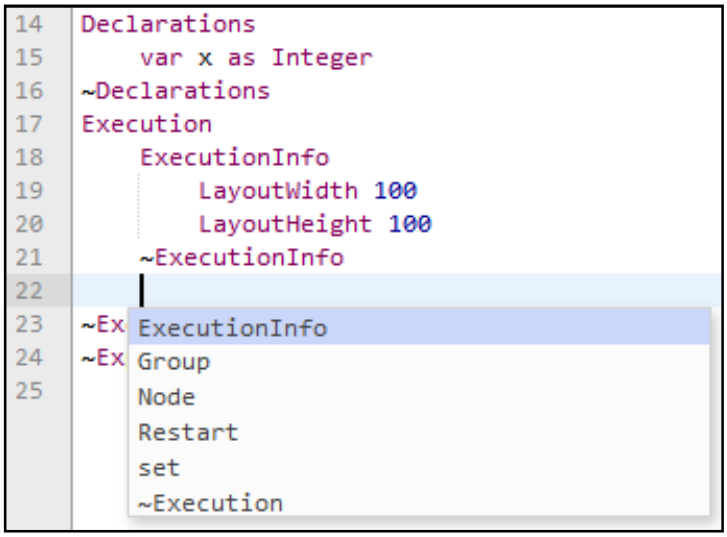
```
Node
        ID node1
        Route[
                WP<1, 10, 12, 12>
                WP<3, 50, 22, 15>
                WP<15, 84, 42, 15>
                WP<18, 36, 22, 15>
                ]
        DataManagement
                Time 14 Algorithm average(history = 10)
        ~DataManagement
        NodeCommunication
                NIC WiFi
        ~NodeCommunication
        DataManagement
                Time 25 Algorithm average(history = 5)
        ~DataManagement
~Node
```

Εικόνα 18: Παράδειγμα συγγραφής σεναρίου EDL για έναν κόμβο

5.3 Ανάπτυξη Συντάκτη Κειμένου για ΚΔτΠ Εφαρμογές

Οι δύο συντάκτες παρέχονται ως μια διαδικτυακή εφαρμογή σε μια κοινή διεπαφή που χωρίζεται σε δύο μέρη. Είναι υπεύθυνοι για την παροχή των απαραίτητων λειτουργιών στους ερευνητές για τη δημιουργία, την ενημέρωση και την επικύρωση των πειραμάτων. Ο συντάκτης κειμένου προσφέρει ένα πλούσιο μενού επιλογών στους ερευνητές για την εισαγωγή προκαθορισμένων μοτίβων κώδικα, και πολλές αυτοματοποιημένες διαδικασίες για τον έλεγχο της ορθής συγγραφής σεναρίων. Κατά την συγγραφή ενός σεναρίου, οι λέξεις-κλειδιά επισημαίνονται με διαφορετικό χρώμα. Μερικές από τις παρεχόμενες λειτουργίες του συντάκτη κειμένου είναι: (i) ο χρωματισμός της σύνταξης (ii) η βοήθεια περιεχομένου (iii) η επικύρωση του περιεχομένου του συντάκτη (iv) η αυτόματη

συμπλήρωση τμημάτων κώδικα (ν) ο έλεγχος των σφαλμάτων. Επίσης παρέχονται μια σειρά από πρόσθετα εργαλεία για τη συντακτική και τη σημασιολογική επικύρωση. Ο συντάκτης του κειμένου δίνει πρόσβαση στις EDL έννοιες, μέσω των οποίων ορίζεται ένα πείραμα. Οι ερευνητές έχουν τη δυνατότητα να καθορίσουν τις διαδρομές των κινητών κόμβων και άλλες σχετικές πληροφορίες απευθείας πάνω στο χάρτη του οπτικού συντάκτη ή να εισάγουν μέσω του πληκτρολογίου τα δεδομένα της διαδρομής των κόμβων απευθείας στο συντάκτη κειμένου. Στην Εικόνα 19, βλέπουμε ένα στιγμιότυπο του συντάκτη κειμένου, όπου εμφανίζεται μια αναδυόμενη λίστα των πιθανών επερχόμενων εντολών που μπορούν να επιλεχθούν για την εισαγωγή τους σε ένα πείραμα.



Εικόνα 19: Άποψη αναδυόμενης λίστας επερχόμενων εντολών

5.4 Ανάπτυξη Γραφικού Περιβάλλοντος για ΚΔτΠ Εφαρμογές

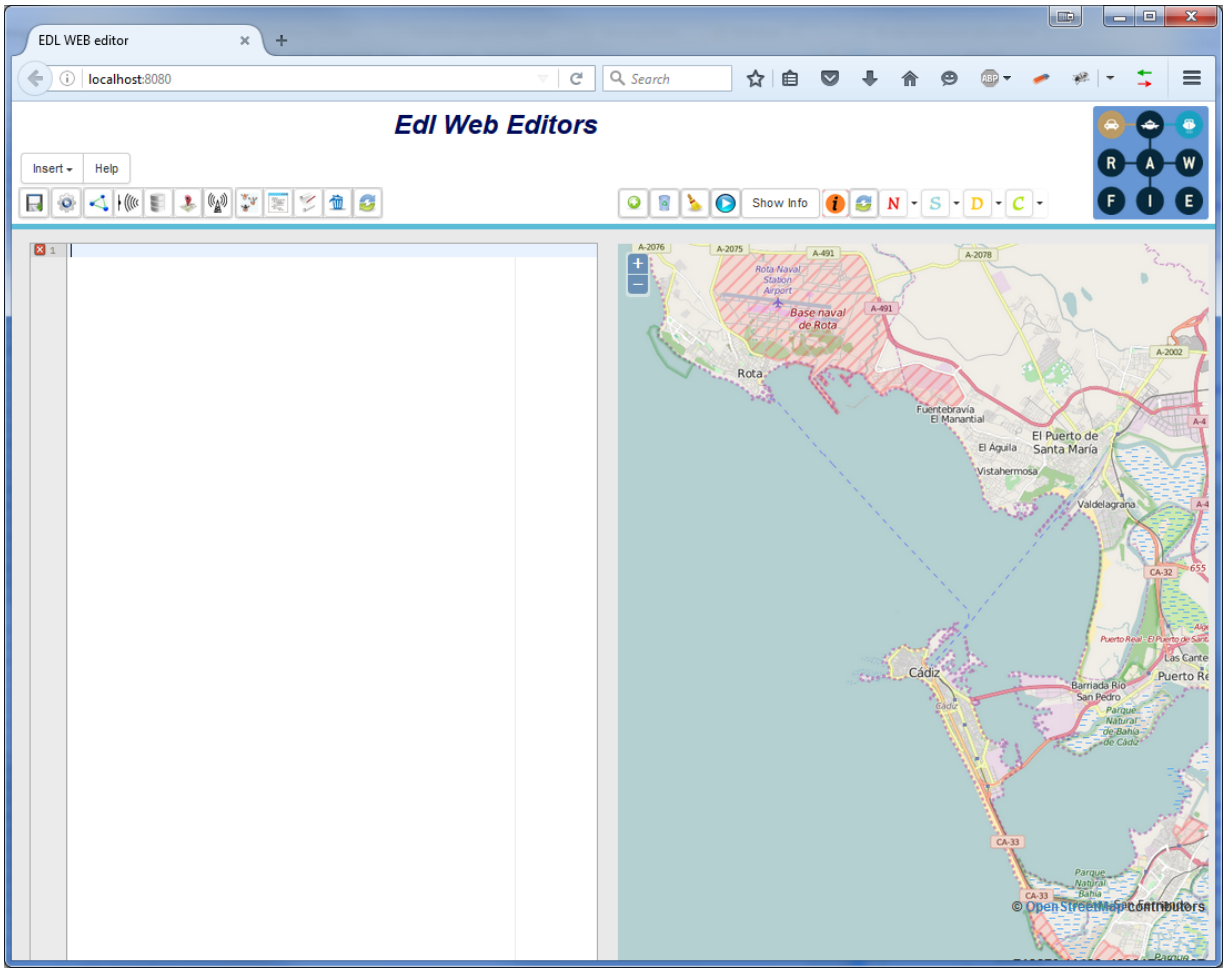
Ο οπτικός συντάκτης είναι ένα ισχυρό εργαλείο για τη δημιουργία πειραμάτων. Ο κύριος στόχος του είναι να παρέχει στο χρήστη ένα φιλικό περιβάλλον που απλοποιεί τη δημιουργία ενός πειράματος με την υιοθέτηση «τυπικών» λειτουργιών GUI (π.χ. ενέργειες του ποντικιού). Οι ερευνητές έχουν τη δυνατότητα να καθορίσουν τη βασική συμπεριφορά των κόμβων (π.χ., ορισμός σημείων αναφοράς) απευθείας πάνω στο χάρτη. Ένα σύνολο εργαλείων, με τη μορφή των κουμπιών, είναι στη διάθεση των ερευνητών. Κάθε κουμπί

επιτελεί μια συγκεκριμένη λειτουργία δηλαδή, διαχείριση των κόμβων (π.χ., προσθήκη, διαγραφή), ο ορισμός της συμπεριφοράς των κόμβων (π.χ., ενεργοποίηση των αισθητήρων, ορισμός αλγορίθμων διαχείρισης δεδομένων), ενώ με τη χρήση του ποντικιού, οι ερευνητές μπορούν να ορίσουν τη διαδρομή του κάθε κόμβου στο χάρτη. Κάθε κόμβος απεικονίζεται στο χάρτη με ένα διαφορετικό χρώμα για να αποφευχθεί η σύγχυση στις περιπτώσεις που ένα πείραμα περιλαμβάνει πολλαπλούς κόμβους.

Επιπλέον, ο οπτικός συντάκτης δίνει την ευκαιρία στους ερευνητές να δηλώσουν για κάθε χρονική στιγμή του πειράματος τις δράσεις και τις κινήσεις που θα επιτελέσει ο κάθε κόμβος. Θα πρέπει να σημειωθεί ότι και οι δύο συντάκτες συγχρονίζονται κατά βούληση μέσα από το GUI, ενώ τα μηνύματα λάθους και οι προειδοποιήσεις παρουσιάζονται στην περιοχή επεξεργασίας κειμένου.

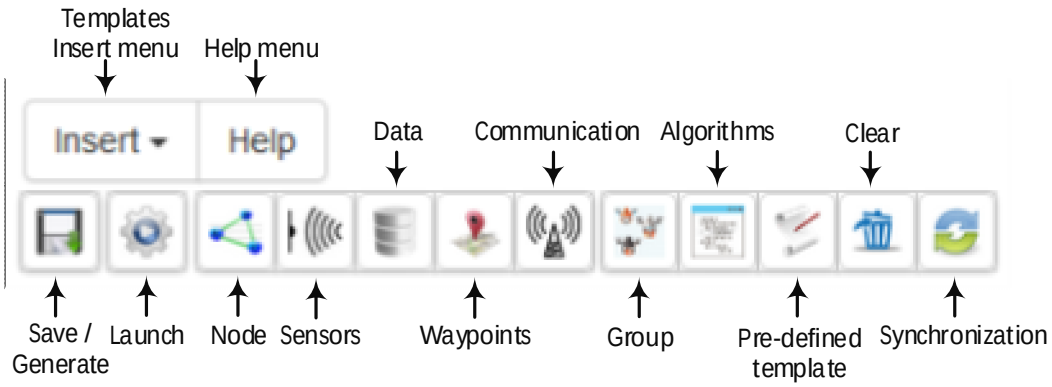
5.5 EDL Web Editors - Η web εφαρμογή για ΚΔτΠ

Στην παράγραφο αυτή παρουσιάζεται ολοκληρωμένο το περιβάλλον της διαδικτυακής εφαρμογής με τους δύο συντάκτες. Στην Εικόνα 20 απεικονίζεται η ολοκληρωμένη άποψη της εφαρμογής.



Εικόνα 20: Άποψη του περιβάλλοντος της εφαρμογής EDL Web Editors

Στο αριστερό μέρος βρίσκεται ο συντάκτης κειμένου. Πάνω από τον συντάκτη κειμένου υπάρχει ένα σύνολο λειτουργιών με την μορφή κουμπιών και αναδυόμενων λιστών. Με τη χρήση των λειτουργιών αυτών, ο ερευνητής δύναται να ολοκληρώσει τη συγγραφή ενός σεναρίου σε σύντομο χρονικό διάστημα. Στην Εικόνα 21 παρουσιάζονται η εργαλειοθήκη και τα κουμπιά του συντάκτη κειμένου



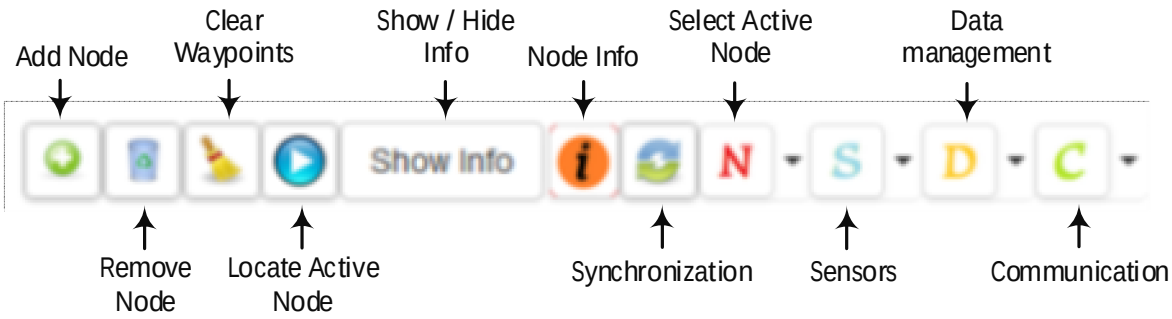
Εικόνα 21: Η εργαλειοθήκη και τα κουμπιά του συντάκτη κειμένου

Οι λειτουργίες που προσφέρει ο συντάκτης κειμένου αναλυτικά είναι:

- **Insert** – εισαγωγή ενός μοτίβου κώδικα (code template) στο συντάκτη (π.χ. τμήμα απαιτήσεων, τμήμα μεταδεδομένων, τμήμα δηλώσεων, τμήμα εκτέλεσης)
- **Help** - μενού βοήθειας π.χ. τρόπος χρήση της εργαλειοθήκης, τρόπος συγγραφής ενός έγκυρου σεναρίου, κλπ
- **Save/Genarate -** μετατροπή του σεναρίου σε json αρχείο και αποθήκευση του json στη βάση δεδομένων
- **Launch -** επιλογή και εκτέλεση ενός αποθηκευμένου αρχείου json από τη βάση δεδομένων **Node** - εισαγωγή ενός μοτίβου κώδικα για τους κόμβους
- **Sensors –** εισαγωγή ενός μοτίβου κώδικα για τα αισθητήρια
- **Data -** εισαγωγή ενός μοτίβου κώδικα για τα δεδομένα
- **Waypoints -** εισαγωγή ενός μοτίβου κώδικα για τις διαδρομές
- **Communication -** εισαγωγή ενός μοτίβου κώδικα για την επικοινωνία ανάμεσα στους κόμβους
- **Group -** εισαγωγή ενός μοτίβου κώδικα για τη διαχείριση ομάδων κόμβων
- **Algorithms -** εισαγωγή ενός μοτίβου κώδικα για τους αλγόριθμους που προσφέρει η εφαρμογή για την εκτέλεση τους σε κάθε κόμβο

- **Pre-defined template -** εισαγωγή ενός προκαθορισμένου δείγματος σεναρίου, το οποίο λειτουργεί ως παράδειγμα ορθής συγγραφής σεναρίων για τους χρήστες της εφαρμογής
- **Clear -** καθαρισμός του σεναρίου
- **Synchronization –** συγχρονισμός των δεδομένων μεταξύ του συντάκτη κειμένου και του οπτικού συντάκτη

Στο δεξιό μέρος της εφαρμογής βρίσκεται ο οπτικός συντάκτης. Πάνω από τον οπτικό συντάκτη υπάρχει ένα σύνολο λειτουργιών με τη μορφή κουμπιών και αναδυόμενων λιστών το οποίο προυσιάζεται στην Εικόνα 22.



Εικόνα 22: Η εργαλειοθήκη και τα κουμπιά του visual editor

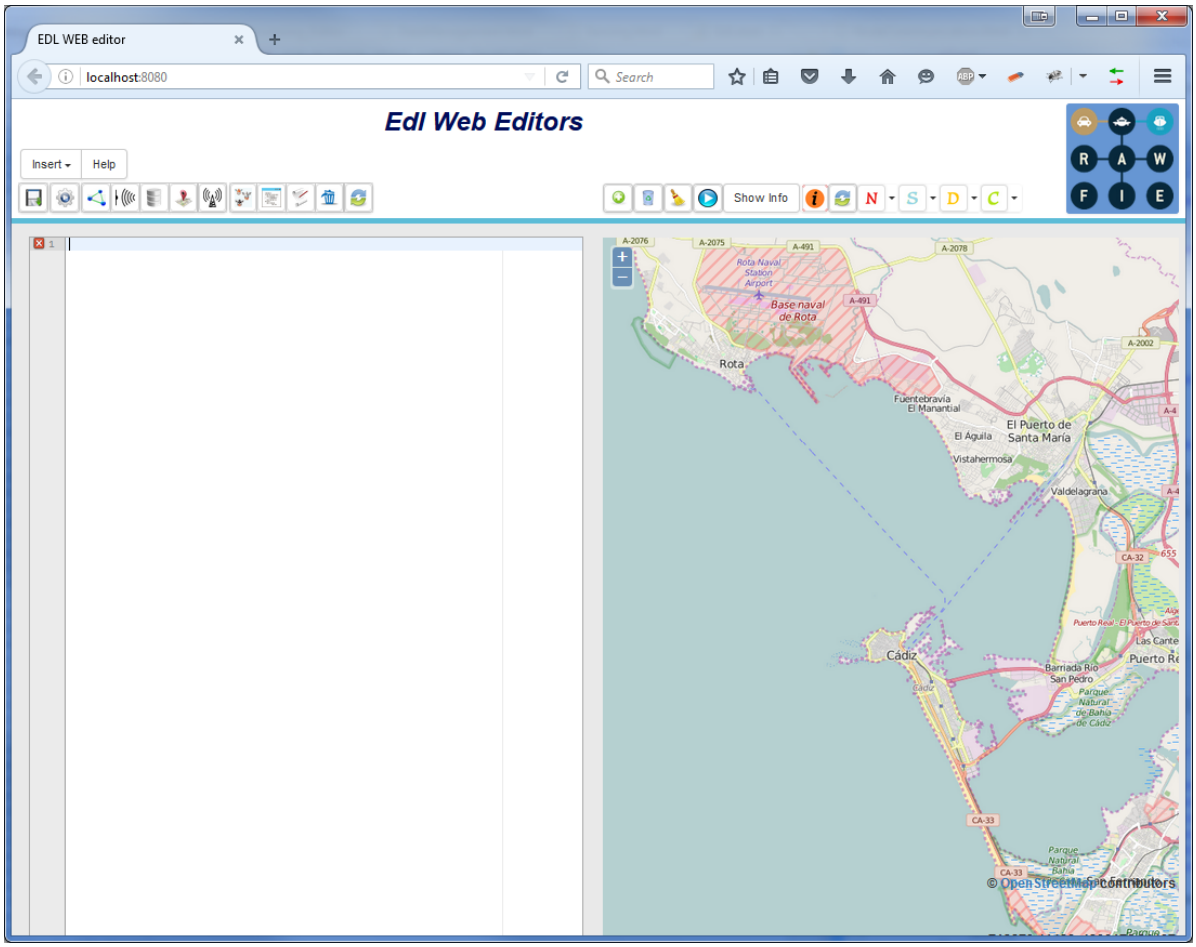
Οι λειτουργίες που προσφέρει ο οπτικός συντάκτης αναλυτικά είναι:

- **Add Node -** Πρόσθεση ενός επιπλέον κόμβου στο χάρτη
- **Remove Node** - Διαγραφή του τρέχοντος ενεργού κόμβου από το χάρτη
- **Clear WayPoints -** Καθαρισμός όλων των σημείων διαδρομής του τρέχοντος ενεργού κόμβου από το χάρτη
- **Locate Active Node -** Εμφάνιση του ενεργού κόμβου, με στιγμιαίο χρωματισμό των σημείων διαδρομής που ανήκουν στο ενεργό κόμβο
- **Show / Hide Info -** Προβολή/απόκρυψη πληροφοριών των σημείων διαδρομής (π.χ. id, γεωγραφικό πλάτος ,γεωγραφικό μήκος)

- **Node Info -** Επιλογή για εμφάνιση επιπλέον πληροφοριών των σημείων διαδρομής
- **Synchronization -** Συγχρονισμός των δεδομένων μεταξύ του συντάκτη κειμένου και του οπτικού συντάκτη
- **Select Active Node -** Επιλογή ενός εκ των κόμβων από την αναδυόμενη λίστα. Ο κόμβος που θα επιλεχθεί γίνεται ενεργός και οποιοδήποτε ενέργεια που εκτελούμε με το ποντίκι αφορά το συγκεκριμένο και μόνο κόμβο.
- **Sensors -** επιλογή των αισθητηρίων που θα χρησιμοποιηθούν από κάθε κόμβο
- **Data Management -** επιλογή των αλγορίθμων που θα χρησιμοποιήσει ο κάθε κόμβος
- **Communication** - επιλογή των τεχνολογιών επικοινωνίας που θα χρησιμοποιήσει ο κάθε κόμβος

5.6 Συγχρονισμός με ένα Συντάκτη Γραμμής

Για να είναι ίδια τα δεδομένα του συντάκτη κειμένου και του οπτικού συντάκτη απαιτείται να υπάρχει συγχρονισμός. Αυτό σημαίνει ότι όταν ένας ερευνητής γράφει ένα σενάριο στο συντάκτη κειμένου, αυτόματα θα πρέπει τα δεδομένα που εισάγονται στο συντάκτη να μετασχηματιστούν με κάποιο τρόπο σε δεδομένα και στην συνέχεια να απεικονιστούν με την αντίστοιχη μορφή στον οπτικό συντάκτη. Σαφώς θα πρέπει να ισχύει και το αντίστροφο, δηλαδή ο χρήστης της εφαρμογής να εισάγει δεδομένα με την βοήθεια του ποντικιού/πληκτρολογίου στον οπτικό συντάκτη και να αναπαράγεται το σενάριο στο συντάκτη κειμένου. Σημαντικός παράγοντας επιτυχούς συγχρονισμού αποτελεί η συντακτική και η σημασιολογική ορθότητα των δεδομένων κυρίως από την μεριά του συντάκτη κειμένου. Στην Εικόνα 23 παρουσιάζεται το γραφικό περιβάλλον της διαδικτυακής εφαρμογής.



Εικόνα 23: Άποψη της διαδικτυακής εφαρμογής

Για το συγχρονισμό μεταξύ των δύο συντακτών, υλοποιήθηκε ένας αμφίδρομος ημιαυτόματος μηχανισμός. Υλοποιήθηκαν δύο HTML κουμπιά, ένα από τη μεριά του συντάκτη κειμένου και ένα από την μεριά του οπτικού συντάκτη που επιλέγει ο χρήστης ανάλογα με την φορά του συγχρονισμού που θέλει να επιτύχει.

Ο αλγόριθμος που υλοποιήθηκε επεξεργάζεται τις συμβολοσειρές του περιεχομένου του συντάκτη κειμένου και ελέγχει για συγκεκριμένες λεξεις κλειδιά .

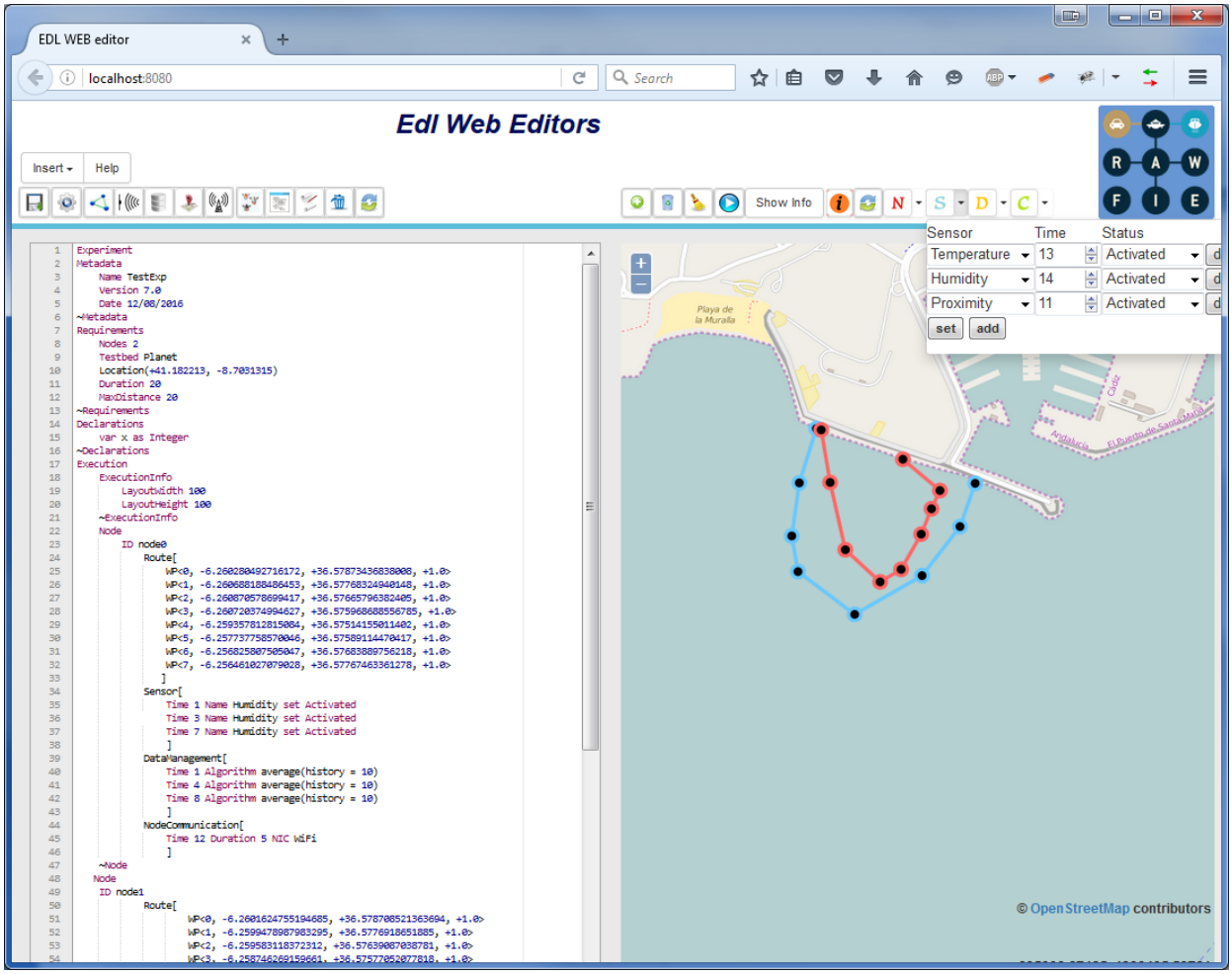
```
    //----- START of DataManagement ------- //
    var DataManagementStart;
    var DataManagementRest;
    var DataManagementEnd;
    var DataManagementSet;
    var DatamanagmentSubSet;

    if ( my_array[i].indexOf("DataManagement")!=-1){
        DataManagementStart= my_array[i].indexOf("DataManagement");
        DataManagementRest = my_array[i].slice(DataManagementStart,-1);
        DataManagementEnd = DataManagementRest.search("]");

        var add = DataManagementEnd+DataManagementStart
        DataManagementSet =  my_array[i].slice(DataManagementStart,add);


        DatamanagmentSubSet=DataManagementSet.split("Time");
        for (var j=1; j<DatamanagmentSubSet.length;j++){

            var AlgoStartPosition= DatamanagmentSubSet[j].indexOf("Algorithm");
            var AlgoTime = DatamanagmentSubSet[j].slice(1,AlgoStartPosition);
            var AlgoText =  DatamanagmentSubSet[j].slice(AlgoStartPosition,-1);
            var AlgoType = AlgoText.replace("Algorithm","");

            if ((vectorDataManagement[my_current_Node]==null)||(vectorDataManagement

                var newArray= new Array();
                   newEmptyObject= new Object();
                   newArray[0]=newEmptyObject;
                   vectorDataManagement.push(newArray);
            }

            if(vectorDataManagement[my_current_Node]==undefined){

            }

            vectorDataManagement[my_current_Node][j-1]={ time:AlgoTime.trim(), algor
        }
    }
   //----- END of DataManagement ------------- //
```

Εικόνα 24: Στιγμιότυπο κώδικα για την επεξεργασίας αλφαριθμητικών του πεδίου Data managment

Στην Εικόνα 24 παρατίθεται στιγμιότυπο του κώδικα που αφορά την επεξεργασία αλφαριθμητικών του πεδίου Data managment. Κατά την εκτέλεση του κώδικα, ελέγχεται αν υπάρχει στο αλφαριθμιτικό ή λέξη "DataManagement". Στην περίπτωση που υπάρχει, ο κώδικας του "if statement" εκτελείται επιτυχώς. Στην συνέχεια της εκτέλεσης του αλγορίθμου, πραγματοποιείται η αναζήτηση του υποσυνόλου των λέξεων κλειδιών "Time" και "Algorithm" και η διαδικασία ολοκληρώνεται με την επεξεργασία των δεδομένων. Στην περίπτωση που υπάρχουν λάθη στην εισαγωγή των δεδομένων από την μεριά του χρήστη, ο συγχρονισμός δεν πραγματοποιείται και ενημερώνει το χρήστη για τα αντίστοιχα σφάλματα.

Στην περίπτωση που τελικά η διαδικασία του επεξεργασίας του αλφαριθμητικού ολοκληρωθεί σωστά, προχωράει στη διαδικασία ενημέρωσης του πίνακα "Datamanagemnet" του αντίστοιχου κόμβου. Τα δεδομένα πλέον μετά το συγχρονισμό είναι διαθέσιμα στον οπτικό συντάκτη.

Απο την μεριά του οπτικού συντάκτη ο χρήστης μπορεί να εισάγει όσους κόμβους επιθυμεί. Για κάθε κόμβο μπορούμε να δημιουργήσουμε πολλά σημεία μιας διαδρομής. Κάθε σημείο διαδρομής έχει συγκεκριμένες ιδιότητες, όπως για παράδειγμα ID, γεωγραφικές συντεταγμένες, χρόνος. Ο οπτικός συντάκτης συμπεριλαμβάνει τρεις αναδυόμενες λίστες (DataManagment, Sensing, NodeCommunication). Σε κάθε κόμβο μπορούμε να εισάγουμε νέα δεδομένα σε καθεμία απο τις αναδυόμενες λίστες ή και να διαγράψουμε δεδομένα από αυτές. Ένα στιγμιότυπο της εισαγωγής δεδομένων στη λίστα Sensing παρουσιάζεται στην Εικόνα 25.



Εικόνα 25: Εισαγωγής δεδομένων στη λίστα Sensing

Στο συγκεκριμένο στιγμιότυπο παρατηρούμε ότι ο χρήστης εισαγάγει στη λίστα του Sensing ένα τρίτο αισθητήριο (π.χ. Proximity), το οποίο έχει οριστεί να είναι ενεργό (Activated) στη χρονική μονάδα 11 (Time=11). Όταν ο χρήστης επιλέξει να προστεθεί (add) στη λίστα, τα δεδομένα αποθηκεύονται στον αντίστοιχο πίνακα vectorControlSensors του πρώτου κόμβου (στον αλγόριθμό μας, τα ids των κόμβων δημιουργούνται αυτόματα ξεκινώντας την αρίθμηση από το 1, π.χ. node1). Η διαδικασία που εκτελείται για το συχρονισμό του οπτικού συντάκτη είναι παρόμοια διαδικασία με αυτήν που αναλύσαμε πιό πάνω. Στην Εικόνα 26 παρουσιάζεται ένα στιγμιότυπο του αλγόριθμου αποθήκευσης των δεδομένων για έναν κόμβο.

```
for (var k = 0; k < vectorControlSensors.length; k++) {
            var sensor="" ;
            NodeSensors[k]="";
            if (vectorControlSensors[k]!=""){
                if ((vectorControlSensors[k][0].time!=undefined)||(vectorControlSensors[k][0]
                        for(var i = 0; i < vectorControlSensors[k].length; i++){
                        sensor =
                                    ' Time ' + vectorControlSensors[k][i].time
                                +   ' Name ' + vectorControlSensors[k][i].sensor
                                +   ' set '+ vectorControlSensors[k][i].status
                                + '\r\n'
                                ;
                        NodeSensors[k] = NodeSensors[k] + sensor ;
                        }
                    }
                }
            else{
                NodeSensors[k]="";
            }

}
```

Εικόνα 26: Αλγόριθμος αποθήκευσης δεδομένων

Ποιό αναλυτικά και βάση του παραδείγματος, ο αλγόριθμος έχει την παρακάτω λογική. Για κάθε κόμβο που υπάρχει, ελέγχεται διαδοχικά ο πίνακας VectorControlSensors. Αν ο πίνακας VectorControlSensors του εκάστοτε ενεργού κόμβου περιέχει τουλάχιστον μια εγγραφή, τότε για κάθε εγγραφή δημιουργείται η κατάλληλη συμβολοσειρά .

5.7 Επικύρωση Περιεχομένων

Η στατική ανάλυση των δεδομένων μιας γλώσσας ΓΕΣ παίζει πολύ σημαντικό ρόλο κατά την υλοποίηση της γλώσσας και κρίνεται απαραίτητη η ύπαρξη της. Είναι, θα λέγαμε, το απαραίτητο συστατικό μιας γλώσσας ΓΕΣ, και πρέπει να αποτελεί αναπόσπαστο κομμάτι αυτής. Ο χρήστης της γλώσσας, κατά τη διάρκεια της συγγραφής ενός σεναρίου, έχει τη δυνατότητα να λαμβάνει έγκαιρα πληροφορίες από τον αναλυτή της γλώσσας. Με αυτό τον τρόπο ανατροφοδοτείται άμεσα με πληροφορίες για τυχόν συντακτικά ή εννοιολογικά λάθη που μπορεί να συμβούν. Στην Εικόνα 27 παρουσιάζεται ένα στιγμιότυπο προβολής μηνύματος λάθους στο συντάκτη κειμένου.

```
1  Experiment
2  Metadata
3      Name TestExp
4      Version 7.0
5      Date 12/08/2016
6  ~Metadata
7  Requirements
8      Nodes 3
   You have declared 3 nodes but you give instructions for 2 nodes!
   The number of nodes in the requirements section is greater than the number of nodes in the execution part!
11     Duration 20
12     MaxDistance 20
13 ~Requirements
14 Declarations
15     var x as Integer
16 ~Declarations
17 Execution
18     ExecutionInfo
19         LayoutWidth 100
20         LayoutHeight 100
21     ~ExecutionInfo
22     Node
23         ID node0
24             Route[
25                 WP<0, -6.260280492716172, +36.57873436838008, +1.0>
26                 WP<1, -6.260688188486453, +36.57768324940148, +1.0>
27                 WP<2, -6.260870578699417, +36.57665796382405, +1.0>
28                 WP<3, -6.260720374994627, +36.575968688556785, +1.0>
29                 WP<4, -6.259357812815084, +36.57514155011402, +1.0>
30                 WP<5, -6.257737758570046, +36.57589114470417, +1.0>
31                 WP<6, -6.256825807505047, +36.57683889756218, +1.0>
32                 WP<7, -6.256461027079028, +36.57767463361278, +1.0>
33                ]
34             Sensor[
35                 Time 1 Name Humidity set Activated
36                 Time 3 Name Humidity set Activated
37                 Time 7 Name Humidity set Activated
38                 ]
39             DataManagement[
40                 Time 1 Algorithm average(history = 10)
41                 Time 4 Algorithm average(history = 10)
42                 Time 8 Algorithm average(history = 10)
43                 ]
44             NodeCommunication[
45                 Time 12 Duration 5 NIC WiFi
46                 ]
47     ~Node
48    Node
49     ID node1
```

Εικόνα 27: Στιγμιότυπο προβολής μηνύματος λάθους στο συντάκτη κειμένου κατά την επικύρωση περιεχομένων

Ο επικυρωτής της EDL πραγματοποιεί το συντακτικό και σημασιολογικό έλεγχο στο σενάριο. Η επικύρωση των περιεχομένων πραγματοποιείται πάνω στο Ecore μοντέλο. Τα βήματα που ακολουθούνται για την επικύρωση είναι τα ακόλουθα. Αρχικά ο αναλυτής του Xtext ελέγχει για συντακτικά λάθη στο Ecore μοντέλο. Αν υπάρχουν λάθη ή προειδοποιήσεις τα προβάλει στο συντάκτη με κόκκινο χρώμα.

Σε περίπτωση που το σενάριο συμφωνεί με την γραμματική της γλώσσας EDL, τοτέ και μόνο τότε μπορεί να προχωρήσει η εκτέλεση του πειράματος. Επίσης γίνεται έλεγχος για σημασιολογικά λάθη που μπορεί να παρεμποδίσουν την εκτέλεση του πειράματος. Στο Xtext υπάρχουν δύο τρόποι επικύρωσης. Η αυτόματη επικύρωση και η προσαρμοσμένη από το χρήστη επικύρωση.

Αυτόματη επικύρωση

Η αυτόματη επικύρωση παρέχεται από το Xtext, και φροντίζει για τη συντακτική επικύρωση (Syntactical Validation), την επικύρωση των διασταυρούμενων αναφορών (Cross-reference Validation) και την επικύρωση διακριτής σύνταξης (Concrete Syntax Validation) των απαιτήσεων μιας εφαρμογής [28].

- Στη συντακτική επικύρωση ο αναλυτής ελέγχει το ρεύμα εισόδου ενός κειμένου και το επικυρώνει αυτόματα. Τα μηνύματα λάθους παράγονται αυτόματα και υποδεικνύονται συνήθως με κόκκινο χρώμα στο συντάκτη. Σε περίπτωση που ο χρήστης θέλει να αναπροσαρμόσει τα μηνύματα εξόδου κατά βούληση, το Xtext παρέχει αυτήν τη δυνατότητα με τη χρησιμοποίηση της κλάσης **ISyntaxErrorMessageProvider.**
- Ο έλεγχος για μη συνδεσιμότητα διασταυρούμενων αναφορών μπορεί να πραγματοποιηθεί γενικά μέσω του συνδέτη (lazy linker). Όμως για να εξασφασλιστεί το γεγονός ότι όλοι οι σύνδεσμοι είναι επικαιροποιημένοι, θα πρέπει να προσπελαστεί όλο το μοντέλο, ούτως ώστε να επιλυθούν όλα τα EMF proxies. Η επικαιροποίηση πραγματοποιείται αυτόματα από το συντάκτη. Για οποιαδήποτε μη επίλυση της λίστας των διασταυρωμένων αναφορών (cross-reference list), γίνεται αναφορά μηνύματος στο παράθυρο προβολής μηνυμάτων λαθών-προειδοποιήσεων της εφαρμογής (errors-warnings messages window). Ένας άλλος τρόπος για να λάβουμε τα μηνύματα λαθών ή τα μηνύματα

προειδοποιήσεων είναι να χρησιμοποιήσουμε τις δύο παρεχόμενες συναρτήσεις JavaScript που είναι η **Resource.getErrors()** και η **Resource.getWarnings()** αντίστοιχα.

- Το Xtext επίσης παρέχει την κλάση **IConcreteSyntaxValidator** που χρησιμοποιείται για την επικύρωση όλων των περιορισμών που ορίζει η γραμματική της γλώσσας. Σε αυτή την περίπτωση το μοντέλο θα πρέπει να απεικονίζεται στην μορφή κειμένου. Αυτό σημαίνει ότι το EMF μοντέλο θα πρέπει να αναπαρασταθεί σε κείμενο. Η αναπαράσταση του μοντέλου σε κείμενο στο Xtext υλοποιείται με τα παρακάτω βήματα

    1. Επικύρωση των σημασιολογικών εννοιών του μοντέλου

    2. Έλεγχος αν τα στοιχεία του μοντέλου ταιριάζουν με τους κανόνες της οριζόμενης γραμματικής και δημιουργία μιάς ροής συμβόλων

    3. Συσχέτιση σχολίων με σημασιολογικά αντικείμενα

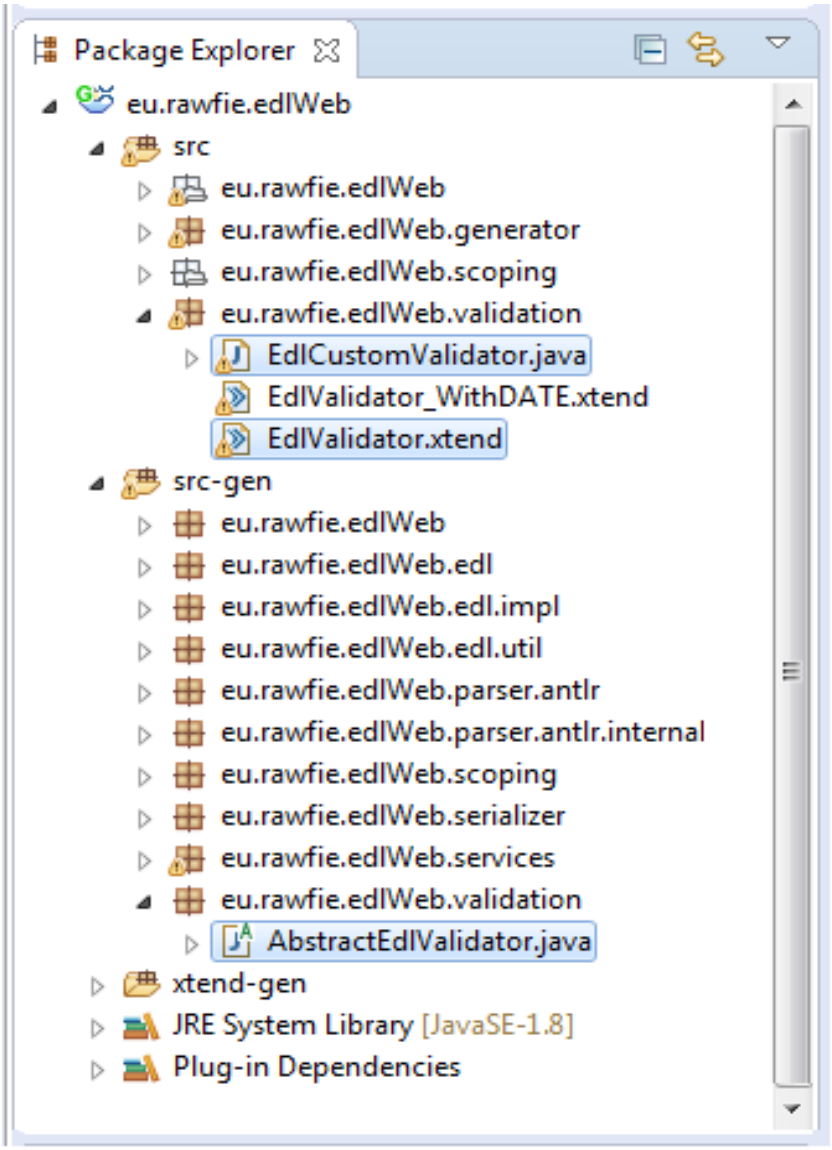
    4. Συσχέτιση των κόμβων του μοντέλου με την ροή των συμβόλων

    5.’Ελεγχος κενών χαρακτήρων και κενών γραμμών μέσα στη ροή των συμβόλων

    6. Προσθήκη ή αντικατάσταση κενών χαρακτήρων με τη χρήση ενός διαμορφωτή (formatter)

<u>Προσαρμοσμένη επικύρωση</u>

Ο γεννήτορας της γλώσσας Xtext μας παρέχει τη δυνατότητα της προσαρμοσμένης επικύρωσης με τη χρήση δύο κλάσεων Java. Η πρώτη είναι μια αφηρημένης κλάση που επεκτείνει τη χρήση της βιβλιοθήκης της κλάσης **AbstractDeclarativeValidator** και βρίσκεται στο φάκελο src-gen/eu.rawfie.edlWeb.validation/. Σε αυτή την κλάση καταχωρούνται τα EPackages, στα οποία έχουν τεθεί κάποιοι περιορισμοί από τον επικυρωτή. Στην Εικόνα 28 φαίνεται η οργάνωση των αρχείων EdlCustomValidator.java και EdlValidator.xtend, που περιλαμβάνονται στο πακέτο (package) eu.rawfie.edlWeb.Validation.



Εικόνα 28: Άποψη των αρχείων προσαρμοσμένης επικύρωσης

Η δεύτερη είναι μια υποκατηγορία της αφηρημένης κλάσης **AbstractEdlValidator** που βρίσκεται στο φάκελο src/eu.rawfie.edlWeb.validation. Εδώ μπορούμε να ορίσουμε εμείς τους δικούς μας περιορισμούς με δηλωτικό τρόπο στις αντίστοιχες κλάσεις όπως για παράδειγμα τα αρχεία EdlCustomValidator.java και EdlValidator.xtend. Με χρήση των ελεγκτικών μηχανισμών (checking mechanisms) αντίστοιχα μπορούμε να εισάγουμε δικούς μας περιορισμούς. Αυτό μας γλιτώνει από την εξαντλητική χρήση των if-else δομών για την κάλυψη όλων των πιθανών περιπτώσεων ή για την επέκταση του EMF μοντέλου. Οι κλήσεις (invocation) με τη

χρήση των ελεγκτικών μηχανισμών πάνω σε οποιαδήποτε μέθοδο, εφαρμόζονται αυτόματα κάθε φορά που γίνεται μια επικύρωση. Οι ελεγκτικοί μηχανισμοί παίρνουν επιπρόσθετα μια παράμετρο που αφορά τον τρόπο λειτουργίας τους:

- Fast check: ο έλεγχος θα εκτελείται κάθε φορά που ένα αρχείο έχει τροποποιηθεί,
- *Normal check :* ο έλεγχος θα εκτελείται κατά την αποθήκευση του αρχείου,
- *Expensive check:* ο έλεγχος θα εκτελείται όταν επικυρωθεί ρητά το αρχείο μέσω της επιλογή του μενού.

5.8 Αναγνώριση / Αντιμετώπιση Λαθών

Η αναγνώριση και η αντιμετώπιση λαθών αποτελεί ένα πολύ σημαντικό προνόμιο της γλώσσας. Ο πειραματιστής κατά τη διάρκεια της συγγραφής ενός σεναρίου, ενημερώνεται για την ύπαρξη συντακτικών ή εννοιολογικών λαθών. Αυτό έχει ως αποτέλεσμα την αύξηση της παραγωγικότητας του τελικού χρήστη. Αυτή η διαδικασία της ταχείας ανατροφοδότησης πληροφοριών προς το χρήστη, υλοποιείται από τις δύο κλάσεις Validator.xtend και CustomValidator.xtend. Ένα στιγμιότυποπο της κλάσης Validator.xtend, παρουσιάζεται στην Εικόνα 29. Στο αρχείο Validator.xtend έχουν δημιουργηθεί ελεγκτικοί μηχανισμοί για την επικύρωση του περιεχομένου του συντάκτη κειμένου. Κατά τη διάρκεια της συγγραφής ενός πειράματος, ο επικυρωτής περιεχομένων ελέγχει τα περιεχόμενα για τυχόν συντακτικά και σημασιολογικά λάθη. Στην συνέχεια ο επικυρωτής περιεχομένων διαβάζει τα προηγούμενα δύο αρχεία, και ελέγχει αν τα αρχεία εμπεριέχουν ελεγκτικούς μηχανισμούς για κάθε λεξικολογική μονάδα. Στην περίπτωση που ένας ελεγκτικός μηχανισμός σχετίζεται με μία λεξικολογική μονάδα, ο συγκεκριμένος ελεγκτικός μηχανισμός εκτελείται αυτόματα. Για παράδειγμα στο στιγμιότυπο της Εικόνας 29 παρατηρούμε ότι στο αρχείο Validator.xtend υπάρχει ένα μηχανισμός ελέγχου για to execution section. Σε αυτόν το μηχανισμό γίνεται έλεγχος για τυχόν δήλωση αρνητικού χρόνου χρόνου κατά την δήλωση του «WP». Στην περίπτωση που ο τελικός χρήστης ορίσει αρνητικό χρόνο, για παράδειγμα WP<-2, 32.4343, 45.99549, 12>, ο επικυρωτής περιεχομένων θα επιστρέψει στον editor το μήνυμα: «The time defined in the way point is negative! Please insert a positive number»

```
// checkPostiveDistance
@Check
    def checkPostiveDistance(RequirementsSection reqs) {
    if (Double.parseDouble(reqs.minDistance) > Double.parseDouble(reqs.maxDistance)  )
        error("The experiment min distance cannot be less than max distance!", reqs,
                Literals.REQUIREMENTS_SECTION__MIN_DISTANCE, 101);
}


// checkTestbed_existance
@Check
    def checkTestbed(RequirementsSection reqs) {
    testbed_selected=reqs.testbed;
    if (!edlV.checkTestbed(reqs.testbed))
        error("This testbed does not exist! Available Testbeds: " + edlV.getTestbeds(), reqs,
                Literals.REQUIREMENTS_SECTION__TESTBED, 101);
}


/****************  EXECUTION SECTION ****************************/

@Check
def checkWayPointTime(WayPoint wp) {
    if (wp.time < 0)
        error("The time defined in the way point is negative! Please insert a positive number.", wp,
                Literals.WAY_POINT__TIME, 101);
}

@Check
    def checkWayPointTimeEqualsZero(WayPoint exe) {
    var waypoint =  (exe.time);
    if ((waypoint) < 0)
        error("The WP time should be greater than zero.", exe,
                Literals.WAY_POINT__TIME, 101);
}
```

Εικόνα 29: Ελεγκτικοί μηχανισμοί του αρχείου Validator.xtend

Το αρχείο CustomValidator.xtend περιλαμβάνει όλους τους μηχαμισμούς ελέγχου για την επικοινωνία με τη βάση δεδομένων. Στην βάση δεδομένων αποθηκεύονται όλα τα σενάρια των ερευνητών, ο έλεγχος της ορθής ημερομηνίας, κλπ. Στην Εικόνα 30 περιγράφεται ένα στιγμιότυπο ελεγκτικού μηχανισμού για τον έλεγχο της ονομασίας των αισθητηρίων που μπορεί να έχει ένας κόμβος. Στην περίπτωση που ο τελικός χρήστης πληκτρολογήσει μια λανθασμένη ονομασία αισθητηρίου, που δεν υπάρχει καταχωρημένη στη βάση δεδομένων, ο επικυρωτής περιεχομένου επιστρέφει στον συτάκτη το μήνυμα λάθους.

```
@Check
public String getAlgorithms() throws SQLException {
    connectionOnDB();
    ResultSet res = connection.createStatement().executeQuery("select algorithm_name from public.algorithms;");
    ArrayList<String> tbs = new ArrayList<String>();
    /*****************************************************/
    //fill the list with the available algorithms
    while (res.next()) {
        tbs.add(res.getString(1));
    }

    String[] testbeds_list = new String[tbs.size()];
    for (int i = 0; i < tbs.size(); i ++)
        testbeds_list[i] = tbs.get(i);

    String algorithms = "";

    for (int i = 0; i < testbeds_list.length; i ++){
        algorithms = algorithms + testbeds_list[i].toLowerCase() + "; ";
    }
    connection.close();
    return algorithms;
}


@Check
public boolean checkSensors(String sensor_user) throws SQLException {

    connectionOnDB();
    ResultSet res = connection.createStatement().executeQuery("select sensor_name from public.sensor;");

    ArrayList<String> sensor_type = new ArrayList<String>();
    /*****************************************************/
    //fill the list with the available sensor type
    while (res.next()) {
        sensor_type.add(res.getString(1));
    }

    /*****************************************************/
    boolean match = false;
    for (int i = 0; i < sensor_type.size(); i ++){

        if (sensor_user.equals(sensor_type.get(i))){
            match = true;
            break;
        }
    }
    connection.close();
    return match;

}
```

Εικόνα 30: Checking μηχανιμοί του Αρχείου CustomValidator.xtend

5.9 Παραγωγή Κώδικα / Αρχείων

Ο γεννήτορας κώδικα είναι ένα εργαλείο το οποίο χρησιμοποιείται για τη μετάφραση και την παραγωγή αρχείων. Η είσοδος του τροφοδοτείται με το σενάριο που παράγει ο ερευνητής μέσω της εφαρμογής και στην έξοδο παράγονται τα αντίστοιχα αρχεία εξόδου. Τα αρχεία εξόδου περιλαμβάνουν εντολές και δεδομένα που θα χρησιμοποιήθουν για την τροφοδοσία των εισόδων των αντίστοιχων UxV συσκευών. Το Xtext έρχεται ενσωματωμένο με έναν γεννήτορα κώδικα (code Generator). Η δημιουργία του γεννήτορα κώδικα βασίζεται στο Xtend. Για τη δημιουργία του τελικού αρχείου ο γεννήτορας κώδικα

χρησιμοποιεί ένα σύνολο από αρχεία τύπου xtend και κλάσεις της Java. Οι κλάσεις Java μπορούν να χρησιμοποιηθούν από τον προγραμματιστή για να επεκτείνουν την διαλειτουργικότητα της ΓΕΣ.

6 Συμπεράσματα και Μελλοντικές Προεκτάσεις

6.1 Σύνοψη

Στην παρούσα διπλωματική εργασία αναπτύχθηκε μια εφαρμογή για τη διαχείριση και τον έλεγχο απομακρυσμένων κινητών συσκευών στο πεδίο του ΔτΠ. Η εφαρμογή που αναπτύχθηκε με τη χρήση μιας γλώσσας ειδικού σκοπού είναι μια web εφαρμογή που περιλαμβάνει ένα συντάκτη κειμένου, ένα οπτικό συντάκτη και ένα σύνολο από υποστηρικτικά εργαλεία. Με τα υποστηρικτικά εργαλεία ο τελικός χρήστης μπορεί να συγγράψει εύκολα και γρήγορα πειραματικά σενάρια, για την εκτέλεση τους από τις ΔτΠ συσκευές. Ο σκοπός είναι η εκτέλεση πειραμάτων και η λήψη-επεξεργασία των δεδομένων από τα αισθητήρια των αυτόνομων ΚΔτΠ συσκευών. Οι εφαρμογές για κινητά ΔτΠ παίζουν σημαντικό ρόλο στην ανάπτυξη εργαλείων και υπηρεσιών που συμβάλλουν στην καλύτερη ποιότητα της ζωής του ανθρώπου. Η χρήση ΓΕΣ τα τελευταία χρόνια φαίνεται να έχει κερδίσει έδαφος σε σχέση με τις γλώσσες ΓΓΣ. Αυτό ωθεί μια μεγάλη κοινότητα προγραμματιστών να ασχοληθεί με την ανάπτυξη υποστηρικτικών εργαλείων για ΓΕΣ. Είναι ίσως το μεγαλύτερο πρόβλημα για μια ΓΕΣ το να υπάρχει ένα περιβάλλον με υποστηρικτικά εργαλεία που να βελτιώνουν την ποιότητα του κώδικα και την αποδοτικότητα του τελικού χρήστη. Οι επικυρωτές περιεχομένου επίσης παίζουν σημαντικό ρόλο καθώς στην περίπτωση συντακτικών ή εννοιολογικών λαθών, ο τελικός χρήστης ενημερώνεται άμεσα μέσω μηνυμάτων λαθών από την εφαρμογή.

6.2 Μελλοντικές Επεκτάσεις

Η διπλωματική εργασία.έχει ως κύριο σκοπό την ανάδειξη των δυνατοτήτων των ΓΕΣ με την χρήση ενός γραφικού περιβάλλοντος. Κατά την ανάπτυξη της εφαρμογής, μεγαλύτερη βαρύτητα δόθηκε στην υλοποίηση και την ανάπτυξη ενός αλληλεπιδραστικού γραφικού περιβάλλοντος, που να πληροί όλες τις προαναφερθείσες λειτουργικές δυνατότητες. Οι πιό σημαντικές δυνατότητες που υλοποιήθηκαν είναι ο συγχρονισμός μεταξύ του οπτικού συντάκτη και του συντάκη κειμένου, η επικύρωση των περιεχομένων και η εγκυρότητα αυτών, η καλή και αξιόπιστη λειτουργικότητα της εφαρμογής. Η ανάπτυξη της εφαρμογής σε κατανεμημένα συστήματα αποτελεί μια πρόκληση και μελλοντικό στόχο, για την

επέκταση της διαλειτουργικότητας της εφαρμογής στο ευρύτερο δίκτυο. Στις μέρες μας επικρατεί η τάση ολοένα και περισσότερες εφαρμογές να μεταφέρονται στο υπολογιστικό νέφος (cloud computiong) με την μορφή διαδικτυακών υπηρεσιών. Είναι ίσως μια καλή ευκαιρία για να δοκιμαστεί και να εξελιχθεί ανάλογα η εφαρμογή, υποστηρίζοντας παράλληλα μεγάλο αριθμό ερευνητών. Σίγουρα για να επιτευχθεί το προηγούμενο θα πρέπει να αναπτυχθούν αυστηρότεροι ελεγκτικοί μηχανισμοί πρόσβασης για την ασφάλεια τόσο της ίδιας της εφαρμογής, όσο και για την ασφάλεια των δεδομένων που θα λαμβάνει ο κάθε ερευνητής από την εφαρμογή αντίστοιχα. Η συμπεριφορά της εφαρμογής σε εταιρογενή υπολογιστικά συστήματα νέφους, αποτελεί μια μελλοντική επέκταση του συστήματος. Η ετερογένεια των υπολογιστικών συστημάτων μπορεί να σημαίνει διαφορετικές διαδικτυακές υποδομές για την αποστολή και την λήψη των δεδομένων, ανομοιογένεια υλικού μεταξυ των συστημάτων κλπ. Οι προηγούμενοι παράγοντες μπορούν να επιρρεάσουν την ομαλή, αδιάλλειπτη και την αξιόπιστη λειτουργία της εφαρμογής.

ΟΡΟΛΟΓΙΑ

Αγγλική	Ελληνική
abstraction	αφαιρετικότητα
Abstract Syntax Tree	αφηρημένο συντακτικό δέντρο
analyzers	αναλυτές
checking mechanism	ελεγκτικός μηχανισμός
client	πελάτης
code generators	γεννήτορες κώδικα
Context free grammars	γραμματικές χωρίς συμφραζόμενα
Context Sensitive Grammar	γραμματικές με συμφραζόμενα
Computation Independent Model	μοντέλο ανεξάρτητο υπολογισμών
Concrete Syntax Validation	επικύρωση διακριτής σύνταξης
cross-reference list	λίστα διασταυρούμενων αναφορών
Cross-reference Validation	επικύρωση διασταυρούμενων αναφορών
debuggers	αποσφαλματωτές
Directed Acyclic Graphs	Κατευθυνόμενοι Ακυκλικοί Γράφοι
documentation generators	γεννήτορες τεκμηρίωσης
domain specific language	γλώσσες ειδικού σκοπού
editors	συντάκτης
errors-warnings messages window	παράθυρο προβολής μηνυμάτων λαθών-προειδοποιήσεων
external	εξωτερικός
Experiment Description Language	Γλώσσα περιγραφής πειραμάτων

extended Backus-Naur Form	εκτεταμένη μορφή Backus-Naur
General Grammars	γενικές γραμματικές
Graph Browsers	γραφικοί περιηγητές
Integrated Development Environment	ολοκληρωμένο περιβάλλον ανάπτυξης
internal	εσωτερικός
Internet of Things	Διαδίκτυο των Πραγμάτων - ΔτΠ
invocation	κλήση
metamodel	μεταμοντέλο
model transformers	μετασχηματιμός μοντέλων
natural language	φυσική γλώσσα
package	πακέτο
parser	αναλυτής
Platform Independent Model	μοντέλο ανεξάρτητο της πλατφόρμας
Platform Specific Model	μοντέλο ειδικής πλατφόρμας
Regular Grammars	Κανονικές Γραμματικές
Syntactical Validation	συντακτική επικύρωση
sensors	αισθητήρια
script	σενάριο
server	εξυπηρετητής
simulators	προσομειωτές
StateFul mode	με υποστήριξη καταστάσεων
Stateless mode	χωρίς υποστήριξη καταστάσεων

strings	Συμβολοσειρές
textual editor	συντάκτης κειμένου
tokens	λεκτικές μονάδες
file extension	επέκταση του αρχείου
formatter	διαμορφωτή
formal languages	τυπικές γλώσσες
visual editor	οπτικός συντάκτης

ΠΙΝΑΚΑΣ ΣΥΝΤΜΗΣΕΩΝ – ΑΡΚΤΙΚΟΛΕΞΩΝ

Σύντμηση	Περιγραφή
ANTLR	ANother Tool for Language Recognition
AST	Abstract Syntax Tree
DSM	Domain Specific Modeling
DSL	Domain Specific Language
EBNF	Extended Backus-Naur Form
EDL	Experiment Description Language
EMF	Eclipse Modeling Framework
EMP	Eclipse Modeling Project
DAGs	Directed Acyclic Graphs
GPL	Generic Purpose Language
IDE	Integrated Development Environment
IoT	Internet of Things
MDE	Model Driven Architecture
MWE	Model Workflow Engine
OMG	Object Management Group
UML	Unified Modeling Language
XMI	XML Metadata Interchange

XSD	XML Schema Definition
ΓΕΣ	Γλώσσες Ειδικού Σκοπού
ΓΓΣ	Γλώσσες Γενικού Σκοπού

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

1. *"Domain-specific language"*, Wikipedia, https://en.wikipedia.org/wiki/Domain-specific_language.
2. Adobe Systems Incorporated. PostScript LANGUAGE REFERENCE. 3rd ed.: Addison-Wesley Publishing Company; 1985–1999.
3. M. Fowler, *"Language Workbenches: The Killer-App for Domain Specific Languages?"*, 2005, http://martinfowler.com/articles/languageWorkbench.html
4. J. Acero*, "Internal vs. External DSL",* 2012. http://javieracero.com/blog/internal-vs-external-dsl
5. M. Fowler, *"InternalDslStyle"*, 2006., http://martinfowler.com/bliki/InternalDslStyle.html
6. G. Jäger, J. Rogers, *"Formal language theory: refining the Chomsky hierarchy",* Philosophical Transactions B R Soc Lond B Biol Sci., The Royal Society Publishing, 367, 2012, p. 1956-1970.
7. Lars Marius Garshol, *"BNF and EBNF: What are they and how do they work?",* 2008. http://www.garshol.priv.no/download/text/bnf.html
8. Josh Haberman, *"LL and LR in Context: Why Parsing Tools Are Hard",* 2013. http://blog.reverberate.org/2013/09/ll-and-lr-in-context-why-parsing-tools.html .
9. Thomas Kuhn, Olivier Thomann, *"Abstract Syntax Tree",* 2006. http://www.eclipse.org/articles/Article-JavaCodeManipulation_AST/
10. Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, *"Compilers Principles, Techniques, & Tools"*, 2nd ed., Pearson Education, Inc., 2007
11. Object Management Group, http://www.omg.org/
12. Anneke Kleppe, *"Software Language Engineering",* Pearson Education; 2008.
13. *"Domain Specific Modelling"*, 2004, http://www.dsmforum.org/why.html
14. ORACLE, JD Edwards World CASE Guide, Release A9.2, 2nd ed.: Oracle; 2010.
15. Tutorialspoint http://www.tutorialspoint.com/software_engineering/case_tools_overview.htm .
16. *"Unified Modeling Language",* http://www.uml.org/
17. Microsoft, *"Microsoft Developer Network",* https://msdn.microsoft.com/en-us/library/bb126327.aspx
18. Eclipse, *"OpenArchitectureWare",* https://www.eclipse.org/gmt/oaw/
19. MetaCase; 2016, http://www.metacase.com/mep/ .

20. Akos Ledeczi, Miklos Maroti, Arpad Bakay, Gabor Karsai, Jason Garrett, Charles Thomason, Greg Nordstrom, Jonathan Sprinkle and Peter Volgyesi, Copyright 2001 IEEE. In WISP; 2001. p. 1-6.
21. Jordi Cabot and Martin Gogolla, *"Object Constraint Language (OCL):a Definitive Guide",* http://modeling-languages.com/wp-content/uploads/2012/03/OCLChapter.pdf
22. Eclipse, *"Eclipse Modeling Framework (EMF)",* https://eclipse.org/modeling/emf/
23. Eclipse, Xtext*, "Language Engineering For Everyone!",* https://eclipse.org/Xtext/
24. OpenLayers 3.http://openlayers.org/
25. Sourceforge.net, *"Experiment Description Language (EDL)",* http://edl.sourceforge.net/
26. Feng Xia, Laurence T. Yang, Lizhe Wang and Alexey Vinel, *"Internet of Things",* International Journal of Communication Systems, 2012; 25: p. 1101–1102.
27. Kevin Ashton, *"That 'Internet of Things' Thing"*, RFID Journal, 2009, p. 1.
28. Eclipse*, "Language Implementation",* https://eclipse.org/Xtext/documentation/303_runtime_concepts.html