ΑΤΟΜΙΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΟΝΤΕΛΟΚΕΝΤΡΙΚΗ ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΩΝ ΓΙΑ INTERNET OF THINGS ΜΕ ΠΛΑΤΦΟΡΜΕΣ ΜΙΚΡΟΕΛΕΓΚΤΩΝ

ΚΥΡΙΑΚΟΣ ΚΥΡΙΑΚΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2017

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μοντελοκεντρική Ανάπτυξη Εφαρμογών Για Internet Of Things με Πλατφόρμες Μικροελεγκτών

Κυριάκος Κυριάκου

Επιβλέπουσα Καθηγήτρια
Γεωργία Καπιτσάκη

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου.

Μάιος 2017

Ευχαριστίες

Οφείλω να εκφράσω τις βαθύτατες μου ευχαριστίες στην επιβλέπουσα καθηγήτριά μου Δρ. Γεωργία Καπιτσάκη, για την σημαντική βοήθεια, υποστήριξη και καθοδήγηση που μου πρόσφερε, καθ' όλη την διάρκεια της εκπόνησης της συγκεκριμένης ατομικής διπλωματικής εργασίας. Η άψογη συνεργασία που είχαμε, οι συμβουλές και οι συστάσεις της, οδήγησαν στην ολοκλήρωση μιας αρκετά αξιόλογης δουλειάς.

Κλείνοντας, θα ήθελα να ευχαριστήσω τους δικούς μου ανθρώπους, γονείς, συγγενείς και φίλους, που με στήριζαν καθ' όλη την διάρκεια της πορείας των σπουδών μου στο Πανεπιστήμιο Κύπρου, καθώς και της εκπόνησης της συγκεκριμένης ατομικής διπλωματικής εργασίας.

Περίληψη

Βρισκόμαστε σε μια εποχή όπου οι συσκευές Internet of Things (IoT) άρχισαν να γίνονται πλέον ευρέως διαδεδομένες και απαραίτητες στην καθημερινότητα του ανθρώπου. Έξυπνα gadgets, έξυπνα αυτοκίνητα και έξυπνα σπίτια είναι μόνο μερικές από τις «καθημερινές» εφαρμογές που μπορούμε να τα συναντήσουμε.

Υπάρχουν πολλοί διαφορετικοί τρόποι, ανόμοιοι μεταξύ τους, με τους οποίους μπορεί ένας μηχανικός λογισμικού να παράξει εφαρμογές που αφορούν κάποια λύση σε μορφή IoT συσκευής. Διαφορετικότητα όμως υπάρχει και στο υλικό το οποίο μπορεί να χρησιμοποιήσει κάποιος, για να εφαρμόσει μια συγκεκριμένη λύση, εφόσον υπάρχει μια πληθώρα διαφορετικών πλατφόρμων μικροελεγκτών από διαφορετικούς κατασκευαστές. Η κάθε πλατφόρμα μικροελεγκτών έχει τα δικά της μοναδικά χαρακτηριστικά, μπορεί να προγραμματιστεί σε μια ή περισσότερες διαφορετικές γλώσσες που και αυτές με την σειρά τους απαιτούν συνήθως τη χρήση ενός συγκεκριμένου συνόλου βιβλιοθηκών, ειδικά κατασκευασμένων για την συγκεκριμένη πλατφόρμα.

Βέβαια, η παραγωγή μιας άρτιας λύσης απαιτεί επίσης την χρήση διαφόρων αισθητήρων ή εξαρτημάτων γενικότερα, τα οποία μπορούν να εμπλουτίσουν και να υποβοηθήσουν την λειτουργία μιας πλατφόρμας μικροελεγκτών. Αρκετά από αυτά όμως είναι επίσης ανόμοια μεταξύ τους αναλόγως της πλατφόρμας που είναι συμβατά.

Στόχος λοιπόν της παρούσας διπλωματικής εργασίας είναι ο καθορισμός μιας μεθοδολογίας μοντελοκεντρικής ανάπτυξης και η παραγωγή μέρους του κώδικα οποιασδήποτε λύσης μπορεί να υλοποιηθεί σε μικροελεγκτές Arduino ή Raspberry Pi, βάζοντας κατά νου το σύνολο των αισθητήρων και εξαρτημάτων που μπορεί να χρησιμοποιηθούν.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Σκοπός Ατομικής Διπλωματικής Εργασίας

Οι αισθητήρες μας δίνουν τη δυνατότητα συλλογής πληροφορικών από το περιβάλλον για την ανάπτυξη εφαρμογών με επίγνωση συγκειμένου (context-aware εφαρμογές). Παράλληλα η χρήση πλατφόρμων με μικροελεγκτές και είσοδο/έξοδο που διευκολύνουν τον προγραμματισμό και τη σύνδεση με διάφορα εξωτερικά κυκλώματα έχει διαδοθεί πολύ τα τελευταία χρόνια. Ο συνδυασμός των πλατφόρμων αυτών με αισθητήρες που υπάρχουν στο περιβάλλον ή σε εξωτερικές συσκευές έχει μεγάλο ενδιαφέρον για τη διευκόλυνση της ανάπτυξης εφαρμογών με επίγνωση συγκειμένου.

Σκοπός της παρούσας διπλωματικής αποτελεί η εξοικείωση και ο πειραματισμός με πλατφόρμες μικροελεγκτών (Arduino/Raspberry Pi) και η σχεδίαση μιας μεθοδολογίας για την ανάπτυξη εφαρμογών για τους μικροελεγκτές αυτούς μέσω μοντελοκεντρικής ανάπτυξης λογισμικού (model driven engineering - MDE). Στόχος να οριστούν τα κατάλληλα μοντέλα και μεταμοντέλα σε μια γλώσσα μοντελοποίησης (π.χ. UML, MOF) που θα υποστηρίζουν την ανάπτυξη βασικών εφαρμογών για τις πλατφόρμες αυτές. Επίσης να γίνεται η κατάλληλη παραγωγή κώδικα για τις εφαρμογές που σχεδιάζονται λαμβάνοντας υπόψη τα χαρακτηριστικά της κάθε

πλατφόρμας. Αν κριθεί απαραίτητο, θα ληφθούν υπόψη και άλλες αρχιτεκτονικές μικροελεγκτών από τις προαναφερόμενες μεταγενέστερα.

1.2 Μεθοδολογία που ακολουθήθηκε

Πίνακας 1.1

Φάση	**Περιγραφή**
	1ο ΕΞΑΜΗΝΟ
Μελέτη υποβάθρου	Εξοικείωση με τις πλατφόρμες μικροελεγκτών για IoT και με την μοντελοκεντρική ανάπτυξη. Έγινε αναζήτηση ενός συνόλου σχετικού υλικού και άλλων άρθρων τα οποία μελετήθηκαν. Ακολούθησε μελέτη για τα κύρια χαρακτηριστικά διαφόρων πλατφόρμων μικροελεγκτών. Έγινε μελέτη του εργαλείου openHAB για να δούμε κατά πόσο μπορεί αν χρησιμοποιηθεί αργότερα στο παρόν έργο. Επιπλέον, μελέτη έγινε και για την πιθανή χρήση της πλατφόρμας Blynk για iOS και Android επεκτάσεις.
Επιλογή πλατφόρμων μικροελεγκτών και εργαλείων που θα χρησιμοποιηθούν	Επιλέχτηκαν οι πλατφόρμες μικροελεγκτών Arduino & Raspberry Pi, ενώ συντάχθηκαν και συγκεκριμένες αναφορές για καθένα από αυτά.
Μελέτη περιβάλλοντων μοντελοποίησης	Έγινε μελέτη για διάφορα περιβάλλοντα μοντελοκεντρικής ανάπτυξης εφαρμογών (κυρίως τα Eclipse Papyrus και Eclipse Modeling Framework). Καταλήξαμε σε επιλογή του εργαλείου Eclipse Papyrus λόγω της χρήσης των UML έναντι του eCore.

Μοντελοποίηση IoT εφαρμογών	Δημιουργήθηκε ένα ενιαίο προφίλ μοντελοποίησης για τις εφαρμογές και τα εργαλεία που επιλέγηκαν. Λήφθηκαν υπόψη τα χαρακτηριστικά των IoT εφαρμογών γενικότερα, ώστε το μοντέλο να μπορεί να προσαρμοστεί σε μεταγενέστερα εργαλεία στο μέλλον.
Μελέτη περιβάλλοντων παραγωγής κώδικα	Έγινε εκτενής μελέτη σε διάφορα περιβάλλοντα και εργαλεία παραγωγής κώδικα από μοντέλα που αναπαριστώνται με δομές xml. Εντέλει, επιλέχτηκε το Acceleo σε συνδυασμό με το Eclipse Papyrus ως το εργαλείο παραγωγής κώδικα με την χρήση της Model Transformation Language (MTL).
Σχεδίαση και υλοποίηση μετασχηματισμών	Με τη χρήση του Acceleo εργαλείου, δημιουργήθηκε ένα σύνολο από κανόνες μετασχηματισμού που μπορούν να αντιστοιχίσουν τα μοντέλα σχεδιασμένα με UML σε κώδικα για την πλατφόρμα που θα επιθυμεί να ασχοληθεί ένας μηχανικός λογισμικού. Έχει ληφθεί υπόψη η ύπαρξη Platform Independent Models (PIM) και των Platform Specific Models (PSM) όπως αυτά ορίζονται στην μοντελοκεντρική ανάπτυξη εφαρμογών. Σημείωση πως η διαδικασία μετασχηματισμού έχει ως προϊόν την παραγωγή μόνο του σκελετού της IoT εφαρμογής και όχι ολόκληρου

	του λειτουργικού κώδικα.
2ο ΕΞΑΜΗΝΟ	
Σχεδίαση και υλοποίηση μετασχηματισμών (Συνέχεια)	Συνέχιση από προηγούμενο εξάμηνο του καθορισμού κανόνων μετασχηματισμού για τις πλατφόρμες μικροελεγκτών που έχουν επιλεγεί και για τις γλώσσες που μπορούν να παραχθούν σε αυτά και που επίσης έχουν καθοριστεί. Επιπλέον, προσθήκη χαρακτηριστικών σχολίων στον κώδικα που θα παραχθεί για να προσανατολίσει τον μελλοντικό μηχανικό λογισμικού να συμπληρώσει ή να τροποποιήσει τυχόν απαραίτητα πεδία, μεταβλητές ή άλλα κομμάτια κώδικα.
Eclipse plug-in	Δημιουργία ενός plug-in για το Eclipse IDE σχετικά με το μεταμοντέλο που έχει οριστεί σε προηγούμενη φάση.
Εφαρμογές δοκιμής	Επιλέχτηκαν κάποιες κύριες εφαρμογές που θα χρησιμοποιηθούν ως δοκιμές των εργαλείων που θα παραχθούν στην παρούσα ατομική διπλωματική εργασία μαζί με τα ανάλογα σενάρια.

Κατασκευή μοντέλων προς αξιολόγηση	Δημιουργήθηκαν τα απαραίτητα διαγράμματα βάση μεταμοντέλων για τις 5 εφαρμογές και σενάρια που έχουν επιλεγεί.
Δοκιμή σεναρίων	Δοκιμή των μοντέλων που κατασκευάστηκαν στην προηγούμενη φάση ούτως ώστε να γίνει η παραγωγή του σκελετού κώδικα για κάθε ένα από τα επιλεγμένα σενάρια εφαρμογών. Καταγραφή των αποτελεσμάτων.
Αξιολόγηση / Εκτίμηση πειραμάτων	Αξιολόγηση και εκτίμηση των πειραματικών αποτελεσμάτων, συγκρίνοντας κατά πόσο είναι ορθή η παραγωγή κώδικα αλλά και πόσος κώδικας έχει παραχθεί σε σχέση με τον τελικό πλήρες υλοποιημένο και λειτουργικό κώδικα.
Συγγραφή Ατομικής Διπλωματικής Εργασίας	Συγγραφή του τελικού παραδοτέου της ατομικής διπλωματικής εργασίας με την συμπερίληψη όλης της απαραίτητης μελέτης και προόδου.

1.3 Δομή της ατομικής διπλωματικής εργασίας

Το κεφάλαιο 1 αναφέρεται στη γενική ιδέα της συγκεκριμένης ατομικής διπλωματικής εργασίας καθώς και στη μεθοδολογία η οποία ακολουθήθηκε προς διεξαγωγής της.

Το κεφάλαιο 2 μιλά για ό,τι αφορά τις συσκευές Internet of Things και την γενική τους φιλοσοφία. Αναφέρετε επίσης σε μια τύπου πρόβλεψη για το πώς φαίνεται να εξελίσσεται ο τομέας αυτός στο άμεσο μέλλον και πώς θα το επηρεάσει.

Το κεφάλαιο 3 εξηγεί το τί είναι μοντελοκεντρική μηχανική κάνοντας μια σαφή αναφορά σε αυτή, στον τρόπο που χρησιμοποιείται και στα πλεονεκτήματά της. Παρουσιάζει επίσης τον μεταμοντελισμό και τη μέθοδο μετατροπής του σε κώδικα μέσω της MTL.

Το κεφάλαιο 4 αναφέρει προηγούμενες δουλειές και μελέτες που έχουν γίνει γύρω από τη μοντελοκεντρική μηχανική γενικότερα αλλά και για την μοντελοκεντρική μηχανική για IoT εφαρμογές συγκεκριμένα.

Το κεφάλαιο 5 υποδεικνύει τα εργαλεία και τους συνολικούς στόχους που επικεντρώθηκε η συγκεκριμένη ατομική διπλωματική εργασία για επίτευξή της.

Το κεφάλαιο 6 περιλαμβάνει όλα τα εργαλεία που χρησιμοποιήθηκαν για την μελέτη, μοντελοποίηση και υλοποίηση των εργαλείων που έχουν παραχθεί για την συγκεκριμένη ατομική διπλωματική εργασία.

Το κεφάλαιο 7 παρουσιάζει όλα τα εργαλεία που έχουν υλοποιηθεί για την μοντελοποίηση IoT εφαρμογών για μικροελεγκτές Arduino & Raspberry Pi.

Το κεφάλαιο 8 δείχνει όλα τα παραδείγματα που έχουν συλλεχθεί για έλεγχο των εργαλείων που έχουν υλοποιηθεί. Τα παραδείγματα είναι κατά κύριο λόγο IoT εφαρμογές σε Arduino ή Raspberry Pi ή συνδυασμού των δύο όπου η κάθε μια έχει δικό της σκοπό και σενάριο χρήσης.

Το κεφάλαιο 9 αναφέρει μερικές από τις δοκιμές που έγιναν σε κάποιες από τις περιπτώσεις χρήσεις παρουσιάζοντας τα ανάλογα UML σχεδιαγράμματα και κώδικα που παράγεται μετά το πέρας της όλης διαδικασίας που έχει οριστεί.

Το κεφάλαιο 10 περιέχει όλα τα συμπεράσματα που εξήχθησαν μέσω της παρούσας ατομικής διπλωματικής εργασίας μαζί με κάποιες ενδεικτικές συμβουλές και προτάσεις για μελλοντικές βελτιώσεις.

Τέλος γίνεται αναφορά σε όλη τη βιβλιογραφία που έχει χρησιμοποιηθεί, ενώ υπάρχει και ένα επιπλέον παράρτημα με διάφορες οδηγίες χρήσης των εργαλείων και δειγμάτων κώδικα για διάφορα σημεία που αναφέρονται σε κάποια σημεία του κειμένου.

Κεφάλαιο 2

Internet of Things

2.1 Εισαγωγή

Ο όρος Διαδίκτυο των Πραγμάτων ή Internet of Things ή ως από κοινού γνωστό IoT αναφέρεται συνήθως σε ένα δίκτυο αποτελούμενο από αντικείμενα (όπως αισθητήρες και μικροελεγκτές/ενεργοποιητές) τα οποία μπορούν να συλλέξουν δεδομένα μέσω της έξυπνης αυτοματοποίησης και αυτορρύθμισής τους, βασισμένη σε φυσικά συμβάντα στον κόσμο, επιτρέποντας έτσι μια τύπου ευελιξία ώστε αυτά τα συστήματα να γίνουν ενεργοί συμμετέχοντες σε διάφορες δημόσιες, εμπορικές, επιστημονικές και προσωπικές διαδικασίες.

2.2 Η φιλοσοφία και οι προκλήσεις

Γενικότερα, φαίνεται πως δεν υπάρχει μια και μοναδική εκδοχή για το ποιος είναι ο κύριος ορισμός του όρου Internet of Things, αλλά όλες οι απαντήσεις που πάρθηκαν από διάφορους κατασκευαστές (έρευνα DZONE 2016), καταλήγουν στο συμπέρασμα ότι μιλάμε συνήθως για ένα δίκτυο από συνδεδεμένες συσκευές [1,4,6].

Υπάρχουν βέβαια δύο βασικές πεποιθήσεις περί του θέματος. Πρώτιστα, ότι αυτού του τύπου συσκευές πρέπει να μπορούν είτε να αισθανθούν, μετρήσουν, αντιδράσουν, διαδράσουν, ή να μεταφέρουν δεδομένα μεταξύ άλλων συσκευών και πλατφόρμων λογισμικού. Κατά δεύτερον, αυτές οι συσκευές πρέπει να μπορούν να

λειτουργήσουν χωρίς κάποια συνεχόμενη ανθρώπινη διάδραση, από τη στιγμή που ο όρος Internet of Things προϋποθέτει την παραγωγή συλλογών δεδομένων όσο πιο ανώδυνα γίνεται.

Η φροντίδα υγείας και η αυτοματοποίηση σπιτιού είναι δύο περιοχές όπου τα IoT είναι εξαιρετικά δημοφιλή στην παρούσα φάση. Προσωπικές φορητές συσκευές υγείας και ευεξίας (wearable fitness) μπορούν να βοηθήσουν καθημερινά τους χρήστες τους ενημερώνοντάς τους άμεσα για το κάθετι, κυρίως όσο αφορά την σωματική τους κατάσταση. Ενώ ακόμη συσκευές παρακολούθησης ασθενούς στα νοσοκομεία κάνουν αντίστοιχη δουλειά με το να παρακολουθούν και να ενημερώνουν τους αρμόδιους γιατρούς για την κατάσταση και την πρόοδο του ασθενούς που νοσηλεύεται σε μορφή στατιστικών. Στην περιοχή της αυτοματοποίησης σπιτιού, συσκευές όπως το Nest βοηθούν τους ενοίκους να σώσουν αρκετά χρήματα από την χρήση άλλων επιπλέον βοηθημάτων. Άλλα νέα προϊόντα όπως για παράδειγμα το Amazon Echo μπορούν να εγγυηθούν την ασφάλεια μιας οικείας ενόσω ο κάτοχός της βρίσκεται σε εκτός αυτής. Υπάρχει πληθώρα περιπτώσεων χρήσεων που εμπίπτουν στην κατηγορία των βιομηχανικών IoT (industrial IoT) η οποία επικεντρώνεται στην βελτίωση αποδοτικότητας της βιομηχανοποίησης και της κατανάλωσης ενέργειας.

Υπάρχουν αρκετές τεχνολογίες που κατά κόρον θεωρούνται να είναι υπεύθυνες για την ανάπτυξη στις τεχνολογίες IoT, όπως συσκευές χαμηλής κατανάλωσης ισχύος και πρωτοκόλλων μηνυμάτων, δίκτυα αισθητήρων και λύσεις αναλυτικών στοιχείων. Πολλές από τις διασυνδεδεμένες συσκευές που συγκροτούν το Internet of Things έχουν ως πλεονέκτημα τη συλλογή δεδομένων από ενσωματωμένους αισθητήρες. Αυτά τα δεδομένα βοηθούν χρήστες και άλλες συσκευές ούτως ώστε να παράγουν ανάλογες αποφάσεις και να διαχειριστούν εξωτερικά συμβάντα. Τεχνολογίες χαμηλής κατανάλωσης ισχύος και υπηρεσίες μηνυμάτων είναι τα βασικότερα για την διατήρηση της λειτουργίας των αισθητήρων αυτών για μεγαλύτερες χρονικές περιόδους χωρίς οποιαδήποτε χειροκίνητη επισκευή ή απώλεια δεδομένων που θα μπορούσαν να συλλεχθούν. Επειδή αυτοί οι αισθητήρες είναι ικανοί στο να συλλέγουν περισσότερα δεδομένα σε μακρύτερες χρονικές περιόδους, τα εργαλεία ανάλυσης αυτών των δεδομένων καθίστανται εξίσου σημαντικά σε ένα αποτελεσματικό σύστημα IoT λόγω του ότι καλούνται να θέσουν τα σημαντικά δεδομένα σε σωστή χρήση.

Ενόσω τα big data analytics είναι σημαντικά για την επιτυχία των IoT, πολλοί ειδικοί βλέπουν αρκετά περιθώρια βελτίωσης στον τομέα αυτό, καθώς οι βελτιωμένες τεχνικές και εργαλεία ανάλυσης μόνο θα βοηθήσουν τους χρήστες να γευτούν το μέγιστο δυνατό που μπορούν να τους προσφέρουν οι IoT συσκευές τους. Σχετικά αναλυτικά στοιχεία πραγματικού χρόνου θα επιτρέπουν στους χρήστες να παίρνουν στιγμιαίες αποφάσεις βασισμένες στα δεδομένα πραγματικού χρόνου, καθιστώντας έτσι τις τεχνολογίες IoT ακόμη πιο χρήσιμες και αποδοτικές. Πολλοί είναι οι ειδικοί που βλέπουν μια ευκαιρία ανάπτυξης του παρόντος «κατακερματισμού» των προτύπων και πρωτοκόλλων των IoT. Πολλοί βέβαια διαφωνούν στο γεγονός ότι εάν υπάρχει τρόπος για να προτυποποιήσεις επίσημα το τρόπο που λειτουργούν και επικοινωνούν οι συσκευές αυτές η μια με την άλλη, θα μπορούσε να ήταν ένα μεγάλο όφελος για τον χώρο ως σύνολο.

Η μόνη κυρίαρχη ομοφωνία που επικρατεί μεταξύ των ειδικών είναι σε σχέση με τα εμπόδια και τα επίμαχα σημεία που υπάρχουν στις διαδικασίες ανάπτυξης και προσαρμογής των IoT συσκευών. Η ασφάλεια φαίνεται να είναι το κατά μακράν σημαντικό πρόβλημα στο μυαλό των ειδικών, δίνοντας έμφαση στη σημαντικότητα των δεδομένων και στην πιθανότητα υποκλοπής τους με την παράνομη πρόσβαση ανεπιθύμητων χρηστών σε αισθητήρες, άλλες συσκευές ή λογισμικό που συγκροτούν κάποια συσκευή IoT. Συμπληρώνοντας στο πρόβλημα της προτυποποίησης, αρκετοί υποστηρίζουν πως το ο συγκεκριμένος κατακερματισμός που υπάρχει στην παρούσα φάση, μπορεί να οδηγήσει σε προσαρμογή προβλημάτων καθώς υπάρχει η πιθανότητα οι καταναλωτές να προσπαθούν ανεπιτυχώς να συνδυάσουν συσκευές ή υλικό γενικότερα από διαφορετικούς κατασκευαστές.

Μια άλλη όψη του νομίσματος που είναι καλή να αναφερθεί, είναι το φαινόμενο όπου οι στρατηγικές κάποιων εταιρειών που προσπαθούν να γίνουν οι «κεντρικοί» παροχείς όλων των πιθανών λύσεων σε IoT, θα οδηγήσουν μόνο στην αύξηση του κατακερματισμού στην αγορά. Η ειρωνεία είναι ότι αρκετές από αυτές τις εταιρείες ή μη κερδοσκοπικές κοινοπραξίες συνήθως προσπαθούν να επιλύσουν προβλήματα προτυποποίησης που έχουν γίνει ήδη αντιληπτά προ πολλού χωρίς να προσδίδουν καμιά θετική εξέλιξη στο θέμα.

Οι φορητές συσκευές είναι προφανές η πιο διαδεδομένη μορφή χρήσης τεχνολογιών IoT μέχρι στιγμής. Η δεύτερη πιο διαδεδομένη επιλογή είναι οι

τεχνολογίες Bluetooth όπου είναι βασισμένες σε επικοινωνία και ανταλλαγή δεδομένων μεταξύ ραδιοκυμάτων και τις οποίες υιοθετούν αρκετές φορητές συσκευές, προσωπικοί υπολογιστές και αρκετά πλέον μοντέλα αυτοκινήτων. Εξίσου συχνά αναφερόμενες είναι οι τεχνολογίες αυτοματοποίησης σπιτιού όπου φαίνονται να αυξάνονται με την πάροδο του χρόνου.

Δυστυχώς μέχρι σήμερα υπάρχει ακόμη τεράστια διάκριση στις διαθέσιμες τεχνολογίες για IoT, καθώς δεν υπάρχει μια κυρίαρχη επιλογή η οποία να καθίσταται η πιο συνήθης ή η πιο αγαπημένη των κατασκευαστών και των χρηστών-καταναλωτών. Ενώ η αυτοματοποίηση σπιτιού, οι ιατρικές τεχνολογίες και οι έξυπνες εφαρμογές προσέλκυσαν το ενδιαφέρον των περισσότερων, αρκετές βιομηχανικές συσκευές IoT χρησιμοποιούν περιπτώσεις χρήσης που αναφέρονται στο πεδίο της γεωργίας, της βιομηχανοποίησης, διαχείρισης πόρων-πηγών κοινής ωφέλειας και των τεχνολογιών έξυπνων πόλεων.

2.3 Το μέλλον του Internet of Things

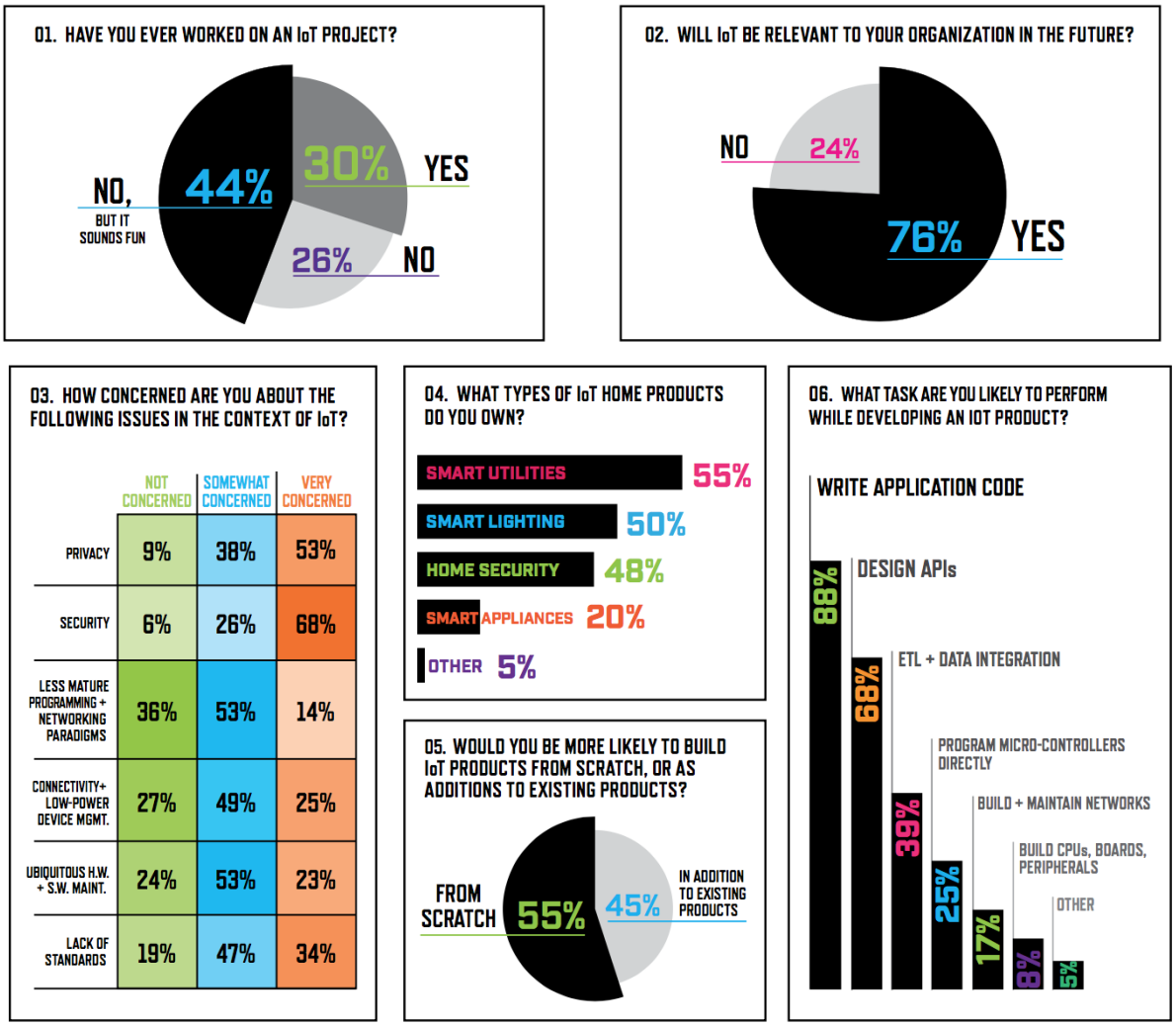
Όπως έχει αναφερθεί προηγουμένως υπάρχει μεγάλη ανομοιότητα και κατακερματισμός μεταξύ των υπαρχόντων τεχνολογιών καθώς επίσης υπάρχουν μεγάλα περιθώρια ανάπτυξης στο κάθε ένα από αυτά και σε διάφορους τομείς γύρω από τον χώρο των IoT. Συγκεκριμένα πρέπει να:

- Επιλυθούν και να διασφαλιστούν τα διάφορα ζητήματα ασφαλείας που υπάρχουν
- Βελτιωθεί η ακρίβεια και η κατανάλωση ενέργειας των διαφόρων αισθητήρων
- Βελτιωθεί η ανάπτυξη διαφόρων APIs που αφορούν την συνδεσιμότητα πολλαπλών συσκευών μαζί, ακόμα και από διαφορετικούς κατασκευαστές
- Βελτιστοποιηθούν οι τρόποι με τους οποίους οι άνθρωποι διαδρούν με τις τεχνολογίες IoT
- Γίνει καλύτερη κατηγοριοποίηση και συγχώνευση διαφόρων τεχνολογιών για μείωση του προβλήματος κατακερματισμού στον τομέα εφόσον υπάρχει μεγάλο ζήτημα

- Υλοποιηθούν τεχνολογίες ή μεθοδολογίες καθολικής ανάπτυξης ή χρήσης εφαρμογών IoT

Έτσι θα επιτύχουμε να αυξήσουμε την προσαρμογή σε μελλοντικά IoT οικοσυστήματα, βοηθώντας το ανθρώπινο είδος στην επίλυση διαφόρων προβλημάτων είτε στην καθημερινότητά τους, είτε στις διάφορες επιχειρήσεις τους.

Σύμφωνα με στατιστικά στοιχεία της έρευνας του DZONE [4] κατά το 2016, συλλέγοντας απόψεις ειδικών στον τομέα, φαίνονται ξεκάθαρα διάφορα από τα προβλήματα που έχουν καταγραφεί πιο πάνω:

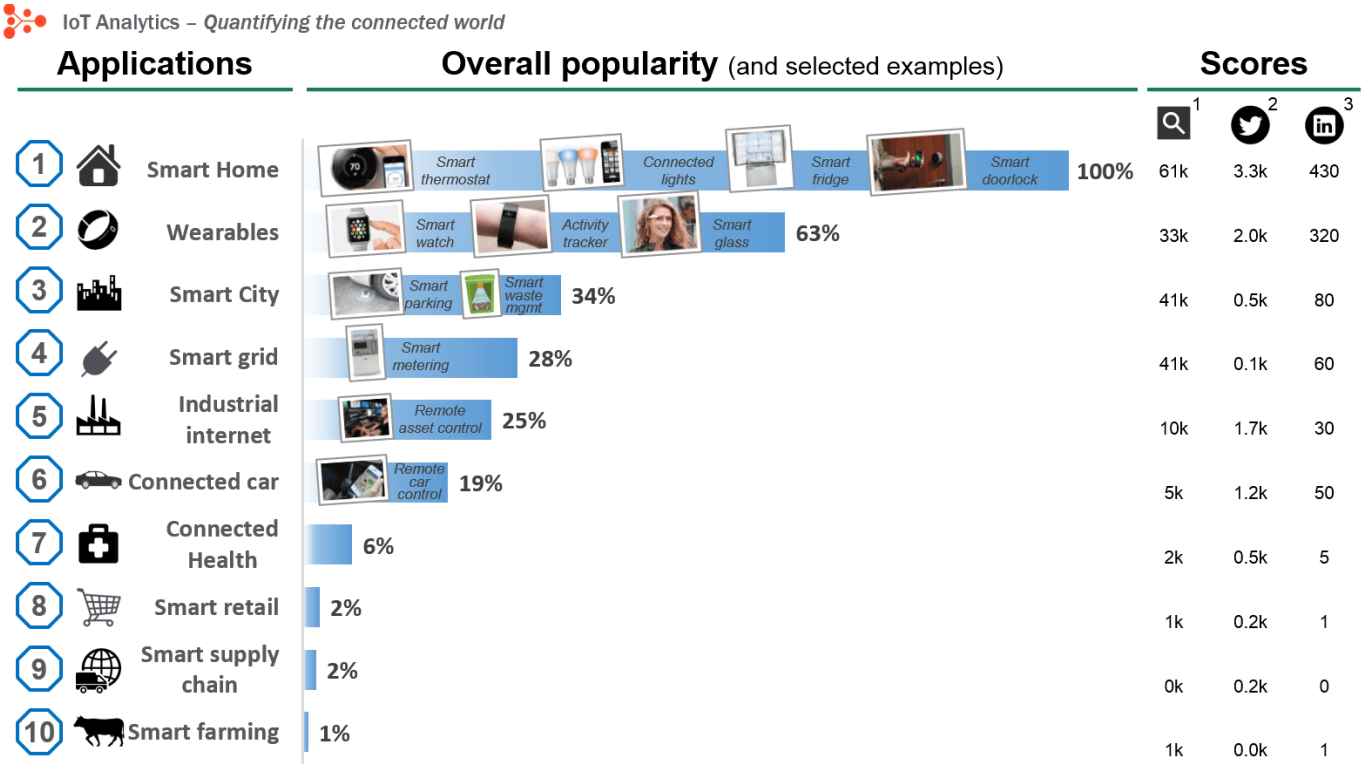


Σχήμα 2.1: [4] [DZONE 2016 Survey: Internet of Things, Volume III]

Παρατηρώντας τα στατιστικά στοιχεία μιας έρευνας που διεξήχθητε κατά το 2014 στα μέσα κοινωνικής δικτύωσης Twitter και Facebook, και σύμφωνα με τις

αναζητήσεις στη μηχανή αναζήτησης Google για καταγραφή της δημοτικότητας των τομέων στο IoT υπάρχουν αρκετοί τομείς στους οποίους διαφαίνεται αύξηση της δημοτικότητας μέσω των εφαρμογών τους.

Οι δημοφιλέστεροι τομείς είναι οι τεχνολογίες έξυπνων σπιτιών (smart home), προσωπικών φορητών συσκευών (wearables) και έξυπνων πόλεων (smart city). Στις τεχνολογίες έξυπνων σπιτιών συμπεριλαμβάνονται οι έξυπνοι θερμοστάτες, τα συστήματα διασυνδεδεμένων φώτων, έξυπνες καταψύξεις και έξυπνα συστήματα ασφαλείας. Στις προσωπικές φορητές συσκευές έχουμε την εμφάνιση διαφόρων έξυπνων ρολογιών (smartwatches), ιχνηλάτες δραστηριότητας (activity trackers) και έξυπνων γυαλιών (smart glasses). Τέλος, στις πρακτικές έξυπνων πόλεων συναντούμε τις διάφορες τεχνολογίες έξυπνου παρκαρίσματος, διαχείρισης της τροχαίας, κατανάλωσης ισχύος, μείωσης των ρύπων στα εργοστάσια και τεχνολογίες βελτίωσης υπηρεσιών κοινής ωφέλειας (πχ. νοσοκομεία).



1. Monthly worldwide Google searches for the application 2. Monthly Tweets containing the application name and #IOT 3. Monthly LinkedIn Posts that include the application name. All metrics valid for Q4/2014.
Sources: Google, Twitter, LinkedIn, IoT Analytics

Κεφάλαιο 3

Μοντελοκεντρική μηχανική

3.1 Εισαγωγή στη Μοντελοκεντρική μηχανική

Η μοντελοκεντρική μηχανική (Model-driven Engineering) ή μοντελοκεντρική ανάπτυξη (Model-driven Development) είναι μια υποσχόμενη προσέγγιση που στόχο έχει την απλούστευση της πολυπλοκότητας συγκεκριμένων πλατφόρμων και πεδίου γνώσης, την αντιμετώπιση της αδυναμίας τρίτων γλωσσών προγραμματισμού μέσω μιας μορφής απλούστευσης της πολυπλοκότητάς τους και την έκφραση διαφόρων εννοιών σε συγκεκριμένα πεδία μελέτης πιο αποτελεσματικά.

Αυτό επιτυγχάνεται διαμέσου διαφόρων πρακτικών που αποτελούν τις κατά κόρον πρακτικές που ορίζουν την ίδια την έννοια της Μοντελοκεντρικής μηχανικής. Οι κυριότερες πρακτικές είναι ο ορισμός Γλωσσών Μοντελοποίησης Πεδίου Μελέτης (Domain-Specific Modeling Languages – DSML) και Μηχανές-Γεννήτριες Μετασχηματισμού. Κατασκευάζοντας μια DSML συγκεκριμένα, βοηθά στη φορμαλοποίηση της δομής, της συμπεριφοράς και των απαιτήσεων μιας συγκεκριμένης εφαρμογής πεδίου ή ακόμη και πεδία ενδιάμεσων εφαρμογών. Ο ορισμό μιας τέτοιας γλώσσας γίνεται συνήθως χρησιμοποιώντας την έννοια των μεταμοντέλων. Δημιουργώντας λοιπόν το μεταμοντέλο, μπορούμε να ορίσουμε έτσι τα χαρακτηριστικά στοιχεία που πρέπει να φέρει μια εφαρμογή σε συγκεκριμένο

πεδίο μελέτης, καθώς επίσης τους διάφορους περιορισμούς και σχέσης που παρουσιάζονται. Έτσι οι κατασκευαστές μπορούν με μεγαλύτερη ευχέρεια να σχεδιάσουν μια εφαρμογή πεδίου βάση του μεταμοντέλου που έχει ορισθεί πρωτύτερα, χωρίς να χρειάζεται κάποια πιο ειδική εμπειρία όσο αφορά το σχεδιασμό εφαρμογών στο πεδίο αυτό. Με τον ορισμό Μηχανών-Γεννητριών Μετασχηματισμού, μπορούμε να αναλύσουμε συγκεκριμένες πτυχές των μοντέλων που προκύπτουν και να συνθέσουμε διάφορους τύπους αντικειμένων που προκύπτουν από αυτά για να παράξουμε κάποιο προϊόν για τον ίδιο τον κατασκευαστή. Όπως για παράδειγμα: κάποιο πηγαίο κώδικα, εισόδων προσομοίωσης, χρήσιμων αναπτυξιακών περιγραφών σε μορφή eXtensible Markup Language (XML), ή εναλλακτικές αναπαραστάσεις μοντέλων (Model-to-Model Transformation).

Τα μοντέλα είναι εξαιρετικά χρήσιμα και αφαιρετικά εργαλεία που συναθροίζουν και προσδίδουν εναλλακτικές προοπτικές. Η αξία τους βέβαια γίνεται ιδιαίτερα σημαντική όταν τα ίδια τα μοντέλα πλέον καθίστανται «χειροπιαστά αντικείμενα» που μπορούν να τύχουν επεξεργασίας. Δηλαδή, είτε να μπορούν να προσομοιωθούν, είτε να μετατραπούν σε κάποιο άλλο προϊόν που απορρέει από την πληροφορία που αντιπροσωπεύουν, είτε να ελεγχθούν και να επικυρωθούν κλπ.

Για την υποστήριξη όμως της Μοντελοκεντρικής Μηχανικής, των μοντέλων και των προϊόντων που απορρέουν από μέσου αυτής, χρειαζόμαστε διάφορα εργαλεία που υποβοηθούν στο κάθε τομέα της ανάλογα. Κάποια από αυτά είναι:

- Εργαλεία για έλεγχο/ενδυνάμωση της καλής διαμόρφωσης των απαραίτητων περιορισμών στα μοντέλα
- Εργαλεία υποστήριξης εργασίας με οντότητες μοντέλων
- Εργαλεία που προσφέρουν αντιστοίχιση μεταξύ μοντέλων
- Εργαλεία που υποστηρίζουν τις μοντελοκεντρικές δοκιμές (Model Driven Testing)
- Εφαρμογές πινάκων οργάνωσης (Dashboard Applications) για έλεγχο και διαχείριση
- Εργαλεία Ελέγχου Εκδόσεων (Version Control) και κατανεμημένης εργασίας
- Εργαλεία για διαχείριση της διαδικασίας εφαρμογής

Ένας σημαντικός παράγοντας που πρέπει να λάβουμε καλά υπόψιν για τα πιο πάνω εργαλεία, είναι πως πρέπει να διασφαλίσουμε ότι θα παραμείνουν ευέλικτα,

επεκτάσιμα και παραμετροποιήσιμα. Δυστυχώς υπάρχει ακόμη μεγάλη ανάγκη στην δημιουργία, συντήρηση και υποστήριξη τέτοιων εργαλείων ώστε να βοηθήσουν να διαδοθεί και να ασκηθεί περισσότερο η πρακτική της μοντελοκεντρικής ανάπτυξης εφαρμογών στον κόσμο των άμεσα ενδιαφερομένων κατασκευαστών. [7, 11]

3.2 Πλεονεκτήματα

Ο πιο κύριος ισχυρισμός αναφέρει πως με την αφαιρετική εικόνα που μπορούμε να αντλήσουμε από την λεπτομέρεια μιας συγκεκριμένης πλατφόρμας ή πεδίου, μπορούμε να υπογραμμίσουμε τα οφέλη της μοντελοκεντρικής μηχανικής τα οποία είναι αρκετά.

Θα προσπαθήσω να παραθέσω μερικά από τα οφέλη. Καταρχάς, η επικύρωση της ορθότητας των μοντέλων γίνεται ευκολότερη, όπως και η παραγωγή υλοποιήσεων σε διαφορετικές πλατφόρμες ενώσω βέβαια συμμορφώνονται στην ίδια συμπεριφορά και δομή που απαιτεί το σύστημα. Ακόμη, μπορεί να οριστεί πιο ξεκάθαρα η ενσωμάτωση και η διαλειτουργικότητα από σύστημα σε σύστημα μέσω ορισμών ανεξαρτήτου πλατφόρμας, οι οποίοι αργότερα θα αντιστοιχηθούν σε μηχανισμούς πλατφόρμων συγκεκριμένου πεδίου όπως αναφέραμε στην προηγούμενη υπό-ενότητα 3.1. [11]

Όπως διάφοροι αναφέρουν, άλλα οφέλη της μοντελοκεντρικής μηχανικής μπορούν να χαρακτηριστούν και τα εξής πιο κάτω:

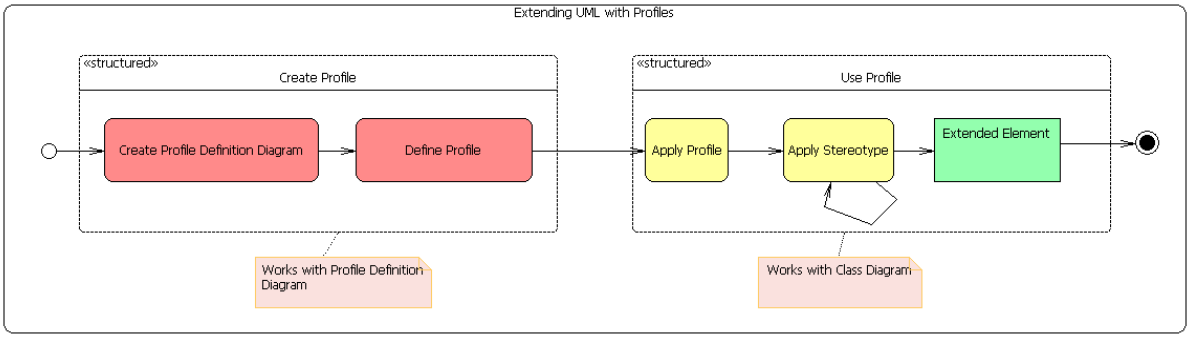
Η μοντελοκεντρική μηχανική

- Επιταχύνει την διαδικασία ανάπτυξης εφαρμογών σε συγκεκριμένο πεδίο.
- Χαρακτηρίζεται πιο γρήγορη πρακτική σε σύγκριση με της πατροπαράδοτες μεθόδους προγραμματισμού εφαρμογών.
- Ελαχιστοποιεί το κόστος ανάπτυξης αν είναι σωστά ορισμένη.
- Οδηγεί σε προϊόντα αυξημένης ποιότητας.
- Είναι λιγότερο επιρρεπής σε λάθη και σφάλματα.
- Οδηγεί σε επικύρωση με νόημα.
- Ενδυναμώνει τους γνώστες πεδίου.

- Επιτρέπει τους έμπειρους προγραμματιστές να επικεντρωθούν στις δύσκολες πτυχές.
- Γεφυρώνει το κενό μεταξύ ειδικών επιχειρησιακών διαδικασιών και ειδικών τεχνικής γνώσης.
- Παράγει λογισμικό το οποίο είναι λιγότερο ευαίσθητο σε αλλαγές των απαιτήσεων.
- Παράγει λογισμικό το οποίο είναι λιγότερο ευαίσθητο σε τεχνολογικές αλλαγές.
- Ενδυναμώνει την έννοια της αρχιτεκτονικής των εφαρμογών.
- Συλλέγει όλη την απαραίτητη γνώση για ένα συγκεκριμένο πεδίο μελέτης.
- Προσδίδει ενημερωμένη τεκμηρίωση.
- Επιτρέπει την εστίαση στις επιχειρησιακές διαδικασίες έναντι της εστίασης στη τεχνολογία που απαιτείται.

3.3 Εισαγωγή στα Μεταμοντέλα

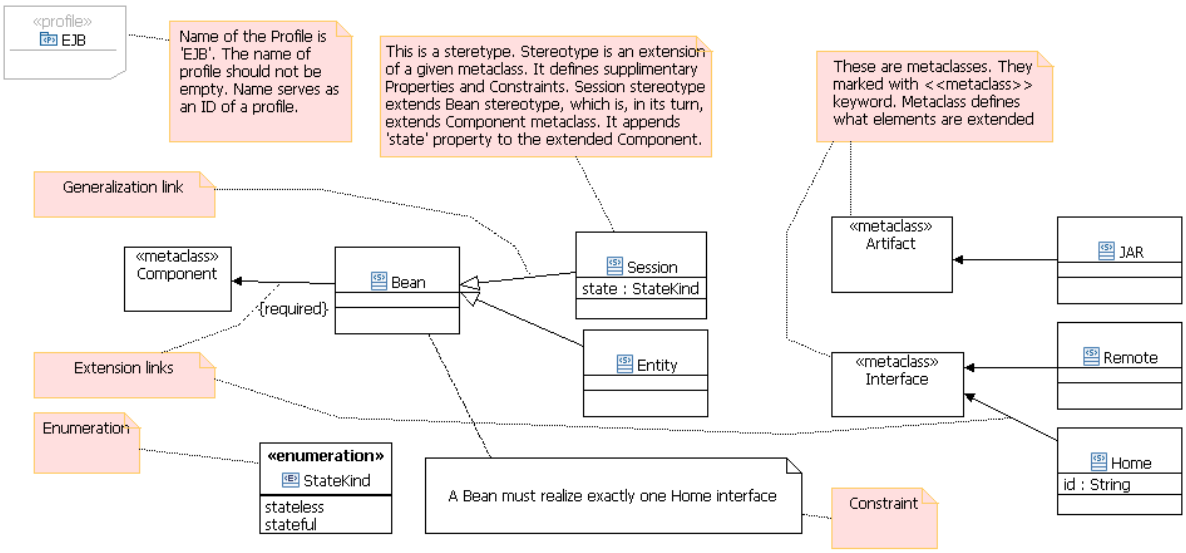
Ένα μεταμοντέλο (metamodel) είναι ουσιαστικά το μοντέλο-πατέρας για κάποιο παράγωγο μοντέλο αργότερα. Ένα μεταμοντέλο κατασκευάζεται κατά τη διάρκεια της μεταμοντελοποίησης, όπου αυτή προϋποθέτει την ανάλυση, την κατασκευή και την υλοποίηση διαφόρων πλαισίων, κανόνων, περιορισμών και εφαρμόσιμων και χρήσιμων θεωριών για μοντελοποίηση μιας λύσης για ένα σύνολο προβλημάτων κάτω από ένα συγκεκριμένο θέμα ή σε ένα συγκεκριμένο πεδίο γνώσης. Σε ένα μεταμοντέλο ορίζεται και παρουσιάζεται ξεκάθαρα η αναπαράσταση όλων αυτών των εννοιών και πώς αυτές είναι συνδεδεμένες μεταξύ τους. Μπορεί να παρουσιάζεται σε κάποια μορφή κειμένου ή ακόμα και γραφική μορφή μέσω διαγραμμάτων που υπόκεινται επίσης σε κάποια φιλοσοφία κανόνων.



Σχήμα 3.1: [14] [MDT-UML2 Tools How to Use UML Profiles]

Χρησιμοποιώντας ένα καλά ορισμένο μεταμοντέλο, μπορούμε να σχεδιάσουμε μοντέλα που ακολουθούν του κανόνες του μεταμοντέλου μεταφέροντας έτσι κομμάτι της γνώσης του πεδίου μελέτης της εφαρμογής/συστήματος που θέλουμε να υλοποιήσουμε αργότερα.

Ένα προφίλ μεταμοντέλου στη γλώσσα μοντελοποίησης Unified Modeling Language (UML) έκδοσης 2.1+, ορίζεται δηλώνοντας και αναπαριστώντας τις διάφορες έννοιες ως οντότητες στερεοτύπων ανάλογα με την εφαρμογή τους και συνδέοντάς τες με σχέσεις κληρονομικότητας ή επεκτασιμότητας μεταξύ των υπολοίπων στερεοτύπων που έχουμε δηλώσει. Ένα στερεότυπο (stereotype) μπορεί να μεταφέρει χαρακτηριστικές ιδιότητες του είδους του όπως ένα σύνολο μεταβλητών με συγκεκριμένους τύπους ή ένα σύνολο μεθόδων. Συνήθως μια συνάθροιση αυτών των στερεοτύπων ή ακόμη και ένα μοναδικό στερεότυπο, μπορεί να επεκτείνει ή να κληρονομήσει μια ήδη υπάρχουσα έννοια όπως για παράδειγμα μια μετακλάση συστατικού στοιχείου (component metaclass). Βλέπετε το πιο κάτω σχήμα:



Σχήμα 3.2: [14] [MDT-UML2 Tools How to Use UML Profiles]

3.4 Εισαγωγή στην Model Transformation Language (MTL)

Όπως έχουμε αναφέρει ξανά, ο καλύτερος τρόπος για να αντιμετωπιστεί η πολυπλοκότητα στην διαδικασία ανάπτυξης λογισμικού είναι μέσω της αφαιρετικότητας, της ανάλυσης του προβλήματος σε υπό-προβλήματα και του διαχωρισμού ή κατηγοριοποίησης των διαφόρων εννοιών. Έτσι, μέσω των αρχών που χαρακτηρίζουν τη μοντελοκεντρική μηχανική, φτάνουμε σε ένα αφαιρετικό επίπεδο που έχει συνήθως την μορφή καλά ορισμένων μοντέλων.

Αυτό βέβαια δημιουργεί ένα επιπρόσθετο πρόβλημα στην όλη διαδικασία. Το πρόβλημα της μετάφρασης και της μετατροπής αυτών των αφαιρετικών μοντέλων σε κάτι πιο απτό, με περισσότερο νόημα και σκοπούς χρήσης. Για να επιτύχουμε κάτι τέτοιο, συνήθως χρησιμοποιούμε μια γλώσσα σήμανσης. Η πιο ευρέως διαδεδομένη είναι η γλώσσα μετατροπής μοντέλων (Model Transformation Language - MTL). Μέσω της διαδικασίας αυτής προκύψει είτε παραγωγή πραγματικού πηγαίου κώδικα είτε μετατροπή του υπάρχοντος μοντέλου σε κάποιο άλλο παράγωγο που σκοπό συνήθως έχει να προσθέσει περισσότερο νόημα όσον αφορά την ίδια τη χρήση του.

Η εφαρμογή της συγκεκριμένης πρακτικής, ουσιαστικά αυτοματοποιεί την όλη διαδικασία στην μοντελοκεντρική μηχανική σε διάφορα σημεία, απομακρύνοντας το οποιοδήποτε περιττό βάρος σχετικά με διάφορες ενέργειες που καλείται κάποιος μηχανικός λογισμικού να πράξει για να φέρει εις πέρας το αποτέλεσμα που επιθυμεί. Τέτοιοι τομείς αποτελούν οι εξής:

- Το ραφινάρισμα των βημάτων που χρειάζονται για να περιγράψουν και να προσδιορίσουν κάποιο συστατικό στοιχείο λογισμικό.
- Η υλοποίηση μοντέλων αντίστροφης μηχανικής που υποβοηθούν στην καλύτερη κατανόηση των μηχανικών λογισμικού που μοντελοποιούν σενάρια μετατροπής συγκεκριμενοποιημένων μοντέλων σε αφαιρετικών.
- Η παραγωγή καινούριων αναπαραστάσεων από τα υπάρχοντα μοντέλα μπορεί να είναι αρκετά χρήσιμη για την επικέντρωση σε ιδιαίτερες έννοιες σε ένα σύστημα, όπου μη-σχετικές πληροφορίες φιλτράρονται από αυτό.
- Η εφαρμογή λογισμικών μοτίβων σε επίπεδο κλάσεων, είτε αυτά είναι αρχιτεκτονικά είτε σχεδιαστικά μοτίβα.

- Ο επαναπροσδιορισμός μοντέλων που προσφέρει ένα καλύτερο τρόπο ανακαθορισμού των υπαρχόντων μοντέλων σε πιο δομημένες και καλά ορισμένες μορφές, απομακρύνοντας την πολυπλοκότητα και την δυσκολία της επισκευής των μοντέλων μέσω μιας συσσώρευσης πολλών αλλαγών που πρέπει να γίνουν σε αυτά.

Πολλές από τις πιο πάνω ενέργειες μπορούν να εκτελεσθούν ως αυτοματοποιημένες διαδικασίες, από τις οποίες μία ή περισσότερες πηγές μοντέλων ως είσοδο, μπορούν να μετατραπούν σε μια ή περισσότερες επιθυμητές μορφές μοντέλων ως έξοδο. Νοείται πως όλα αυτά γίνονται ακολουθώντας συγκεκριμένων κανόνων μετατροπής που έχουν ορισθεί ανάλογα από την εκάστοτε γλώσσα μετατροπής μοντέλων που έχει επιλεγεί. Η όλη διαδικασία που έχει περιγραφεί μέχρι στιγμής, είναι από κοινού γνωστή ως η διαδικασία μετατροπής μοντέλων (model transformation process). [9, 12]

3.5 Βασικές Αρχές Μετασχηματισμού

Για την υλοποίηση ή επιλογή μιας γλώσσας μετατροπής μοντέλων, είναι πρέπον να ακολουθηθούν κάποιοι άγραφοι κανόνες, οι οποίοι προϋποθέτουν ότι η συγκεκριμένη γλώσσα θα είναι εύχρηστη και αποτελεσματική. Υπάρχουν αρκετοί άγραφοι κανόνες ή πρακτικές οι οποίες προτείνονται στο να ακολουθούνται σε τέτοιες περιστάσεις και τα οποία αναλύονται στην πορεία.

Είναι επιθυμητή η χρήση μιας γλώσσας μετατροπής μοντέλων που θα μοιάζει με την σύνταξη ή την φιλοσοφία άλλων ευρέως διαδεδομένων γλωσσών προγραμματισμού όπως για παράδειγμα η Java, η C#, η VB κλπ. Παρόλο που μια τέτοια γλώσσα είναι συνήθως μια γλώσσα σήμανσης, αν η μορφή της σύνταξης και η φιλοσοφία διαχείρισης που θα ακολουθεί μοιάζει με μια γνωστή γλώσσα προγραμματισμού όπου ο μηχανικός λογισμικού έχει εις γνώση του, τότε δεν θα χρειάζεται επιπλέον κόπος για εκμάθηση και χρήση της. Ένας ενδιαφερόμενος μηχανικός λογισμικού λοιπόν μπορεί να χρειαστεί ελάχιστη ή και καθόλου επιπλέον εκπαίδευση με σκοπό να μπορεί να γράφει μετατροπείς μοντέλων στη γλώσσα αυτή.

Παρόλο που η δυνατότητα άμεσης πρόσβασης και διαχείρισης των μοντέλων μέσω μιας γλώσσας μετατροπής μοντέλων αποτελεί μεγάλο πλεονέκτημα, ωστόσο υπάρχει και η αντίθετη, αρνητική πλευρά του νομίσματος. Παρόλο που οι γλώσσες

προγραμματισμού συνήθως χαρακτηρίζονται ως γλώσσες «γενικής χρήσεως», οι γλώσσες σήμανσης δυσκολεύονται στο να αποκτήσουν την ιδιότητα αυτή. Τουλάχιστον στο παρόν στάδιο οι γλώσσες σήμανσης που υπάρχουν για μετατροπή μοντέλων έχουν έλλειψη της κατάλληλης και πρέπουσας αφαιρετικότητας «υψηλού επιπέδου» για προσδιορισμό των επιθυμητών μετατροπέων. Ως αποτέλεσμα, η κωδικοποίηση των μετατροπέων αυτών μπορεί να γίνει εξαιρετικά χρονοβόρα, δυσκίνητη και οι αλγόριθμοι μετατροπής μπορεί να είναι δύσκολο να υλοποιηθούν ή να συντηρηθούν.

Μια άλλη επιθυμητή ιδιότητα μιας γλώσσας μετατροπής μοντέλων είναι η ύπαρξη υποστήριξης της γλώσσας για συγκεκριμένα πεδία γνώσης. Έχοντας έτσι ως πρωταρχικό στόχο την ευκολία περιγραφής κανόνων μετατροπής μοντέλων στη γλώσσα, αυτό αποτελεί ένα από τα σημαντικότερα και υποσχόμενα χαρακτηριστικά των γλωσσών μετατροπής μοντέλων. Μια κοινή μορφή στην οποία συναντούμε την ιδιότητα αυτή, είναι η οπτική αναπαράσταση από κάποιες γλώσσες για καθορισμό και εκτέλεση των μετασχηματισμών των μοντέλων που έχουν κατασκευαστεί. Αυτές οι γλώσσες είναι κατά κύριο λόγο είτε δηλωτικές, είτε διαδικαστικές, ή απλά ακολουθώντας ένα συνδυασμό των δύο. Ο γενικός στόχος-ιδέα είναι να μπορούν να παραχθούν οτιδήποτε είδους μοντέλα ως αποτέλεσμα της διαδικασίας του μετασχηματισμού.

Ένα πρόβλημα που θα ήταν καλό να αποφευχθεί είναι η ύπαρξη περιορισμού στην εκφραστικότητα μιας γλώσσας μετατροπής μοντέλων. Αν και όπως έχουμε δει μέχρι στιγμής, προσπαθούμε να πλησιάσουμε τις έννοιες των κανονικών γλωσσών προγραμματισμού, σε αρκετές περιπτώσεις οι γλώσσες σήμανσης περιορίζονται από την εκφραστικότητα και τις δυνατότητες διαχείρισης των ίδιων των μοντέλων, πράγμα που δυσκολεύει την όλη διαδικασία επιλογής κριτηρίων μετατροπής μοντέλων την κάνει αρκετά περίπλοκη.

Μια άλλη σωστή τεχνική είναι να συμπεριφερόμαστε στα μοντέλα μας ως γράφους. Για παράδειγμα, στα μοντέλα UML, έχει γίνει αρκετή δουλειά όσο αφορά τις γραμματικές γράφων και τα συστήματα μετασχηματισμών γράφων. Οι μετασχηματισμοί γράφων γίνονται υπαρκτοί με την εφαρμογή κανόνων μετασχηματισμών, οι οποίοι είναι επανεγγράψιμοι κανόνες για γράφους.

Η δυνατότητα εξέφρασης μιας λογικής συνέπειας μέσα από την γλώσσα είναι επίσης ένα επιθυμητό χαρακτηριστικό. Οι έλεγχοι, οι βρόγχοι επανάληψης και άλλα

χαρακτηριστικά, όπως και σε μια κανονική γλώσσα προγραμματισμού, υποβοηθούν ένα μηχανικό λογισμικού να θέσει κανόνες με μια πιο μεγάλη λογική σημασιολογία και να παράξει ένα πιο στρωτό και αρθρωτό αποτέλεσμα. Δυστυχώς κάτι τέτοιο στις μέρες μας ακόμη δεν υφίσταται.

Επανερχόμενοι στο ζήτημα της εκφραστικότητας, τονίζεται η δύναμη της ιδιότητας αυτής από αρκετούς ειδικούς. Ακόμη και αν περιορίσουμε την γλώσσα για υποστήριξη μόνο UML-σε-UML μετατροπών, πάλι θα χρειαζόμαστε μια πλήρως εκφραστική γλώσσα για την επίτευξή τους. Επειδή υπάρχουν σχεδόν άπειροι τρόποι που ένα μοντέλο UML μπορεί αν χρησιμοποιηθεί, είναι αδύνατο να προβλέψουμε τα είδη ανάλυσης και στρατηγικών που απαιτούνται για τις μετατροπές από UML-σε-UML γενικότερα. Έτσι θα ήταν πρέπον η γλώσσα μετατροπής να ήταν πιθανόν Turing-complete.

Υπάρχει ένα σύνολο παραγόντων που είναι καλό να διευθετηθούν και να ισορροπηθούν μέσα σε μια γλώσσα μετατροπής μοντέλων. Θα πρέπει να είναι εύκολη στην κατανόηση, όσο γίνεται πιο ακριβής και ξεκάθαρη, συνοπτική και εύκολη στη μετατροπή αλλά και πλήρης.

Μια δηλωτική γλώσσα προσφέρει συνήθως απεριόριστη ερμηνεία, την οποία μπορεί κάποιος να εκμεταλλευτεί για να διατυπώσει μια επιθυμητή προδιαγραφή μέσω της χρήσης του συνόλου των μηχανισμών που αυτή προσδίδει. Οι δηλωτικές γλώσσες είναι τυπικά πιο συνοπτικές απ’ ότι μια συγκρίσιμη επιτακτική γλώσσα. Παρολαυτά, υπάρχει ένας αντικρουόμενος συμβιβασμός μεταξύ συνοπτικότητας και κατανόησης. Για παράδειγμα, εάν μια προδιαγραφή περιέχει πάρα πολλές πολύπλοκες έννοιες, μπορεί να είναι δυσκολότερο στην κατανόηση, σε αντίθεση με μια περισσότερο ρητή, όμως αρκετά μεγάλη σε περιεχόμενο, προδιαγραφή. Το κλειδί για σχεδιασμό μιας γλώσσας μετατροπής είναι ένα σύνολο από σημαντικές και αφαιρετικές πρακτικές οι οποίες είναι κυρίως διαισθητικές και καλύπτουν ένα μεγαλύτερο εύρος πιθανών περιπτώσεων.

Πολλοί από τους κανόνες που αντιστοιχούν μια πηγή από στοιχεία μοντέλων σε άλλα στοιχεία μοντέλων μπορούν να οριστούν με ένα παρόμοιο τρόπο με αυτόν του τρόπου της ανθρώπινης φυσικής επικοινωνίας.

Η προσβασιμότητα και η αποδοχή μιας γλώσσας εξαρτάται επίσης από τη μορφή της. Όπως για παράδειγμα, στα μοντέλα UML, ελκυστικό χαρακτηριστικό αποτελεί η χρήση της μορφής γραφικών αναπαραστάσεων των μοντέλων με

σχήματα και άλλα γραφικά στοιχεία όπου το καθένα χωριστά αντιστοιχεί σε συγκεκριμένη σημασιολογία.

Πιο σύνθετες πρακτικές που θα ήταν καλό να ακολουθούνται είναι η σύνθεση υπάρχουσων περιγραφών μετασχηματισμών. Με την συνάθροισή τους μπορούν να κατασκευαστούν καινούρια και σύνθετα στοιχεία, παρά να κατασκευάσουμε καινούρια εξαρχής χρησιμοποιώντας μόνο τις απαραίτητες βασικές αρχές.

Υποθέτοντας ότι οι μετατροπείς που έχουμε δημιουργήσει ερμηνεύονται σωστά από τα ανάλογα εργαλεία που χρησιμοποιούμε θα πρέπει να αναμένουμε να παραχθεί από αυτά ένα αποτέλεσμα άρτιο, ορθό και με νόημα. Το συγκεκριμένο χαρακτηριστικό είναι ένα από τα κατά κόρον κομβικά σημεία στην διαδικασία μετατροπής.

Ακολούθως, παρατίθενται κάποια επιπρόσθετα προτεινόμενα επιθυμητά χαρακτηριστικά όσο αφορά τον μετασχηματισμό μοντέλων. Είναι προτεινόμενο λοιπόν για μια γλώσσα μετατροπής μοντέλων που υποστηρίζει τη μοντελοκεντρική μηχανική λογισμικού, να:

- είναι εκτελέσιμη
- είναι υλοποιήσιμη με έναν αποτελεσματικό τρόπο
- είναι πλήρως εκφραστική και ξεκάθαρη για μετασχηματισμούς που μπορούν να τροποποιήσουν υπάρχοντα μοντέλα (όσο αφορά την πρόσθεση, την αλλαγή ή την αφαίρεση στοιχείων μοντέλου) ως επίσης και να δημιουργούν εντελώς καινούρια μοντέλα
- διευκολύνει την παραγωγικότητα των μηχανικών λογισμικού, με ακρίβεια, συνέπεια και ξεκάθαρες περιγραφές:
    - η γλώσσα θα ήταν πρέπον να διαφοροποιεί ξεκάθαρα την περιγραφή των κανόνων του πηγαίου μοντέλου που επιλέγονται από αυτούς που παράγουν το επιθυμητό μοντέλο
    - η γλώσσα θα ήταν πρέπον να προσφέρει γραφικές κατασκευές σε περιπτώσεις όπου οι έννοιες αναπαρίστανται με περισσότερη συνέπεια και διαισθητικότητα στις μορφές γραφικών αναπαραστάσεων σε σύγκριση με τις αναπαραστάσεις σε μορφή κειμένου

- η γλώσσα θα ήταν πρέπον να είναι δηλωτική κάνοντας ρητές οποιεσδήποτε έννοιες ή μηχανισμούς που μπορούν διαισθητικά να ερμηνευθούν από το περιεχόμενο

- προσδίδει σημασία στον συνδυασμό των μετασχηματισμών για δημιουργία άλλων σύνθετων, προσφέροντας τουλάχιστον μια γκάμα από λειτουργίες ακολουθιών, επιλογών βάση υποθέσεων και επανάληψη των μετασχηματισμών
- προσδίδουν σημασία στον καθορισμό των συνθηκών κάτω από τις οποίες ο μετασχηματισμός επιτρέπεται να εκτελεσθεί.

Εν κατακλείδι, υπάρχουν τεράστια περιθώρια βελτίωσης των ήδη υπάρχουσων τεχνολογιών και πρακτικών τον τομέα αυτό ή ακόμη και δημιουργία καινούριων. Ως ένα από τα σημαντικότερα χαρακτηριστικά της μοντελοκεντρικής μηχανικής, η διαδικασία μετατροπής μοντέλων δεν πρέπει να κατακερματίζεται σε μια πληθώρα διαφορετικών τεχνολογιών και μη-κοινών πρακτικών που δυσκολεύουν στην εξιδανίκευση και στην πρακτική της φιλοσοφίας γύρω από την μοντελοκεντρική μηχανική. Μόνο με συλλογική συνάθροιση των εννοιών και του καλύτερου καθορισμού των επόμενων βημάτων θα μπορούμε να πούμε πως έχουμε με επιτυχία κατορθώσει να δημιουργήσουμε ένα πλαίσιο στο οποίο τόσο ειδικευμένοι όσο και ανειδίκευτοι μηχανικοί λογισμικού στο χώρο, θα μπορούν να το χρησιμοποιούν χωρίς καμιά απολύτως δυσκολία και λαμβάνοντας πάντα το πρωταρχικό τους επιθυμητό αποτέλεσμα ως παράγωγο. [12]

Κεφάλαιο 4

Προηγούμενες εργασίες

4.1 Εισαγωγή

Όσο αφορά την μοντελοκεντρική μηχανική ή μοντελοκεντρική ανάπτυξη λογισμικού υπάρχει μια πληθώρα άρθρων και σχετικών ερευνών που αναφέρουν κυρίως τη σημασία και την ιδέα γύρω από αυτήν ή τις πρακτικές και τα εργαλεία τα οποία έχουν υλοποιηθεί ή προτείνονται για την ομαλή περάτωσή της. Υπάρχει δε μεγάλη ανομοιότητα μεταξύ των πρακτικών και των εργαλείων που χρησιμοποιούνται, πράγμα που δυσκολεύει την εκμάθηση και την υιοθέτηση σωστών συνηθειών στην ευρεία κοινότητα των μηχανικών λογισμικού. Σε αρκετές περιπτώσεις τα εργαλεία αυτά είναι είτε υποανάπτυκτα είτε όχι καλά ορισμένα ακόμη, καθώς υπάρχουν περιθώρια στη βελτίωσή τους.

Όσο αφορά το Internet of Things, ως ο πλέον καινούριος τομέας στον χώρο της πληροφορικής δεν υπάρχει αρκετή ουσιαστική έρευνα και υλικό πέραν από απλές αναφορές στην ιδέα του, σε διάφορα στατιστικά στοιχεία και κάποιες ανόμοιες πρακτικές που υλοποιούνται από διάφορους χωρίς κάποια κοινή συνιστώσα.

4.2 Ενδεικτικές προηγούμενες εργασίες

4.2.1 HEADS FP7

Η ιδέα του έργου HEADS στοχεύει στην εκμετάλευση της μοντελοκεντρικής μηχανικής λογισμικού και στην παραγωγή προγραμματιστικών τεχνικών για να προσφέρει μια καινούρια προσέγγιση γύρω από την μηχανική λογισμικού (software engineering) η οποία θα επιτρέπει την προχωρημένη εκμετάλλευση πιο ευρέως φάσματος ποικιλομορφίας και εξειδίκευσης της μελλοντικής υπολογιστικής συνέχειας.

4.2.2 MPM4CPS

Τα πραγματικά πολύπλοκα, σχεδιασμένα συστήματα δράσης COST, επίσης γνωστά ως Cyber Physical Systems (CPS), είναι αναδυόμενες τεχνολογίες που ενσωματώνουν τις φυσικές πτυχές, το λογισμικό και τις πτυχές του διαδικτύου. Μέχρι σήμερα, δεν υπάρχει θεωρητική ενοποίηση ούτε συστηματικές μέθοδοι σχεδιασμού, τεχνικές και εργαλεία για τέτοια συστήματα. Οι μεμονωμένοι (μηχανολογικοί, ηλεκτρικοί, δικτυακοί ή λογισμικού) κλάδοι μηχανικής προσφέρουν μόνο μερικές λύσεις. Η Μοντελοποίηση Πολλαπλών Παραδειγμάτων (MPM) προτείνει να μοντελοποιηθεί κάθε μέρος και πτυχή ενός συστήματος με σαφήνεια, στο πιο κατάλληλο επίπεδο αφαίρεσης, χρησιμοποιώντας τον καταλληλότερο φορμαλισμό.

Οι μηχανισμοί των γλωσσών μοντελοποίησης, συμπεριλαμβανομένου του μετασχηματισμού μοντέλων, και η μελέτη της σημασιολογίας τους, χρησιμοποιούνται για την υλοποίηση του MPM. Το MPM θεωρείται ως μια αποτελεσματική απάντηση στις προκλήσεις του σχεδιασμού του CPS. Αυτή η δράση COST προωθεί την ανταλλαγή ιδρυμάτων, τεχνικών και εργαλείων και παρέχει εκπαιδευτικούς πόρους τόσο στον ακαδημαϊκό χώρο όσο και στον κλάδο. Αυτό επιτυγχάνεται με τη συγκέντρωση και τη διάδοση γνώσεων και πειραμάτων σχετικά με τα προβλήματα CPS και τις λύσεις MPM.

4.2.3 MetamoFAB

Ένα ερευνητικό πρόγραμμα το οποίο διεξήχθη από το Γερμανικό ομοσπονδιακό υπουργείο έρευνας και εκπαίδευσης κατά το 2014-2017. Στοχεύει στην ανάπτυξη μοντέλων, μεθόδων και εργαλείων που θα βοηθήσουν τις κατασκευαστικές εταιρείες να κάνουν τη μετάβαση σε έξυπνα, δικτυωμένα εργοστάσια με την εφαρμογή της CPS για την αύξηση της παραγωγικότητας στα υπάρχοντα εργοστάσιά τους.

Οι βάσεις για αυτό το μετασχηματισμό περιλαμβάνουν τη χαρτογράφηση του οράματος ενός δικτυωμένου εργοστασίου με το CPS χρησιμοποιώντας τρεις περιπτώσεις χρήσης - "Τεχνολογία αυτοματισμού κατασκευής", "Παραγωγή ημιαγωγών" και "Κατασκευή εξαρτημάτων ηλεκτρικής μηχανικής". Έχοντας ως καταληκτικό στόχο, οι λύσεις να αποδειχθούν σε πραγματικές συνθήκες σε διάφορες τοποθεσίες συνεργατών του κλάδου.

Κεφάλαιο 5

Επιλογές Στόχων

5.1 Πλατφόρμες Μικροελεγκτών

Μια πλατφόρμα μικροελεγκτών είναι ουσιαστικά μια μικρογραφία υλοποίησης ενός υπολογιστικού συστήματος με ανάλογες προοπτικές και ικανότητες. Συνήθως χρησιμοποιούνται είτε ως το κύριο συστατικό μιας υλοποίησης IoT εφαρμογής, είτε ως αισθητήρες-αναλυτές δεδομένων, είτε ακόμη και ως ολοκληρωμένα υπολογιστικά συστήματα όπως ένας ηλεκτρονικός υπολογιστής.

Σε αυτού του είδους πλατφόρμες περιέχεται συνήθως τουλάχιστον ένας μικροεπεξεργαστής χαμηλής καταναλωτικής ισχύος μαζί με διάφορους εφαρμοσμένους μικροελεγκτές, αισθητήρες, εισδοχές εισόδου – εξόδου (I/O ports) και τροφοδοσίας ισχύος. Μπορούμε να χαρακτηρίσουμε δηλαδή τις πλατφόρμες αυτές ως μια μικρογραφία της παραδοσιακής μητρικής κάρτας σε ένα υπολογιστή, η οποία φιλοξενεί όλα τα απαραίτητα για την υποστήριξη ενός συγκεκριμένου σκοπού και συνόλου λειτουργιών.

Ένας μικροελεγκτής (microcontroller) ή αλλιώς MCU, είναι μια μικρή μονάδα υπολογιστή σε ένα ενσωματωμένο κύκλωμα. Μια μοντέρνα ετοιμολογία είναι το λεγόμενο «σύστημα σε ολοκληρωμένο κύκλωμα» (system on a chip). Ένας μικροελεγκτής περιλαμβάνει ένα ή περισσότερα CPUs (processor cores) μαζί με μερική μνήμη και προγραμματιζόμενα περιφερειακά εισόδου – εξόδου. Σχεδιάστηκαν με σκοπό την ενσωματωμένη εφαρμογή τους, σε αντίθεση με τους προσωπικούς

υπολογιστές ή άλλου είδους εφαρμογών γενικής χρήσεως που αποτελούνται από άλλα διακριτά επιμέρους ολοκληρωμένα κυκλώματα. Όπως έχουμε αναφέρει πιο πάνω οι συγκεκριμένοι μικροελεγκτές μπορούν να συμπεριληφθούν μαζί με άλλα για να συγκροτήσουν μια πλατφόρμα από ένα σύνολο μικροελεγκτών που σε συνεργασία μπορούν να ελέγξουν, επεξεργαστούν, αναλύσουν ή να παράξουν επιθυμητά αποτελέσματα σχετικά με κάποιο σκοπό που εμπίπτει σε κάποιο συγκεκριμένο πεδίο γνώσης.

Κάποια χαρακτηριστικά και δημοφιλή παραδείγματα πλατφόρμων μικροελεγκτών [13] είναι τα διαφόρων εκδόσεων Arduino(s), Raspberry Pi(s) και Intel Galileo(s). Στην παρούσα διπλωματική εργασία καταπιανόμαστε μόνο με τα Arduino(s) [3] και τα Raspberry Pi(s) [10].

Υπάρχουν διαφόρου είδους εκδόσεων πλατφόρμων Arduino για ανάλογα μεγάλα ή μικρά έργα ενσωματωμένων εφαρμογών με τα ανάλογα χαρακτηριστικά και κόστος. Μια πλατφόρμα Arduino εμπεριέχει ένα σύνολο προγραμματιζόμενων μικροελεγκτών τους οποίους κάποιος μηχανικός λογισμικού μπορεί να προγραμματίσει όπως αυτός επιθυμεί μέσω της χρήσης του Arduino Software IDE. Το συγκεκριμένο IDE περιέχει ένα σύνολο από πρότυπες βιβλιοθήκες που αφορούν την σωστή διαχείριση των πόρων του Arduino και προσδίδουν την απαραίτητη λειτουργικότητα σε μια εφαρμογή του. Σε αυτό υπάρχουν διάφορες θύρες (ports) όπου μπορεί κάποιος να το συνδέσει με κάποια διαδικτυακή πηγή, με διάφορους αισθητήρες εισόδου, με συστατικά εξαρτημάτων εξόδου κλπ.

Όπως τα Arduino(s) έτσι και τα Raspberry Pi(s) μπορούν να βρεθούν υπό πολλές μορφές εκδόσεων εκεί έξω. Είναι πλατφόρμες που κατά κύριο λόγο μπορούν να δεχθούν μικροελεγκτές κάποιας συγκεκριμένης μορφής, αλλά θα τα χαρακτήριζε κανείς πιο ολοκληρωμένα κυκλώματα συστημάτων εφόσον περιέχουν ένα πιο προηγμένο σε ταχύτητα και κατανάλωση ισχύος μικροεπεξεργαστή, αρκετές θύρες εισόδου – εξόδου όπως και του Arduino, απλά σε μια πιο ευρεία γκάμα. Οι πλατφόρμες αυτές κοντεύουν κατά πολύ την έννοια του προσωπικού υπολογιστή και είναι αρκετά συγκρίσιμα με αυτούς. Καθώς μπορούν άνετα να συμπεριληφθούν τόσο σε ενσωματωμένες εφαρμογές, όσο και σε κανονικές που μπορούν να σταθούν από μόνες τους (stand-alone applications).

Οι συγκεκριμένες επιλογές έγιναν λόγω της ραγδαίας αύξησης της δημοτικότητας των συγκεκριμένων δύο πλατφόρμων κατά τα τελευταία χρόνια,

καθώς επίσης και της αύξησης των χρήσεων τους σε πραγματικά και υλοποιήσιμα έργα.

5.2 Γλώσσες Προγραμματισμού

Ο κατασκευαστής της κάθε πλατφόρμας μικροελεγκτών ορίζει την αντίστοιχη γλώσσα προγραμματισμού με την οποία κάποιος μηχανικός λογισμικού μπορεί να προγραμματίσει σε αυτήν. Συνήθως γίνεται επιλογή μιας συγκεκριμένης γλώσσας προγραμματισμού, αν και τα τελευταία χρόνια έχουμε παρατηρήσει πως αυτό αρχίζει πλέον να ξεθωριάζει καθώς σιγά σιγά ορίζονται περισσότερες από μία.

Για παράδειγμα στις μοντέρνες πλατφόρμες μικροελεγκτών Raspberry Pi υπάρχει μια πληθώρα γλωσσών προγραμματισμού με την οποία μπορεί κάποιος να υλοποιήσει μια εφαρμογή σε αυτό. Αυτό συμβαίνει διότι το Raspberry Pi μπορεί να παραμετροποιηθεί εύκολα ούτως ώστε να φιλοξενεί ολοκληρωμένα λειτουργικά συστήματα όπως UNIX, Linux, Windows for IoT κλπ. στα οποία μπορούμε όχι μόνο να εγκαταστήσουμε το οτιδήποτε αλλά και να υλοποιήσουμε μοναδικές προγραμματιστικές ιδέες όπως θα μπορούσαμε να κάνουμε και σε ένα κανονικό υπολογιστή. Οι πιο συνήθης γλώσσες προγραμματισμού που χρησιμοποιούνται σε μια πληθώρα έργων σχετικών με το IoT είναι η Python και η Java.

Αντιθέτως οι πλατφόρμες μικροελεγκτών Arduino μπορούν να προγραμματιστούν μόνο με μια και μοναδική γλώσσα προγραμματισμού, την C Wiring οποία είναι μια παραλλαγή της γνωστή γλώσσας προγραμματισμού C με κάποιες επιμέρους τροποποιήσεις από τον κατασκευαστή για να εξυπηρετήσει τους σκοπούς του συνόλου που περιλαμβάνεται στην πλατφόρμα.

Για τους σκοπούς της συγκεκριμένης ατομικής διπλωματικής εργασίας έχουν επιλεγεί οι γλώσσες προγραμματισμού C Wiring για τις πλατφόρμες μικροελεγκτών Arduino και η γλώσσα προγραμματισμού Java για τις πλατφόρμες μικροελεγκτών Raspberry Pi.

5.3 Αισθητήρες και εξωτερικά εξαρτήματα

Υπάρχει μια πληθώρα από αισθητήρες και εξωτερικά εξαρτήματα που μπορούν να προσαρμοστούν στις διάφορες πλατφόρμες μικροελεγκτών ή εάν είναι ήδη συμβατά, να συμπεριληφθούν κατευθείαν σε μια υλοποίηση IoT εφαρμογής. Χωρίζονται σε δύο κύριες κατηγορίες για είσοδο ή έξοδο και σε δύο υποκατηγορίες που καθορίζουν εάν είναι αναλογικού ή ψηφιακού τύπου.

Για σκοπούς της παρούσας διπλωματικής εργασίας έχουν επιλεγεί μόνο μερικοί ενδεικτικοί αισθητήρες και εξωτερικά εξαρτήματα εισόδου – εξόδου. Οι αισθητήρες και εξαρτήματα που έχουν επιλεγεί αποσκοπούν στην γενίκευση και μοντελοποίηση της αφαιρετικότητας τους σε μορφή μεταμοντέλου για άλλου παρόμοιου τύπου συστατικά στοιχεία, που ακολουθούν την δική τους δομή, απαιτήσεις και κανόνες με τους οποίους λειτουργούν μέσα σε μια συνάθροιση. Ακολουθεί λίστα με τους επιλεγμένους γενικευμένους αισθητήρες και εξαρτήματα:

Είσοδοι

- Touch Switch
- Button
- Joystick
- Sensor
    - I/R Detection
    - PIR Motion Detection
    - Analog Sound
    - Temperature Measurement

Έξοδοι

- Console
- Display
    - LED
        - Mini 3 Digit LED

Κεφάλαιο 6

Εργαλεία που χρησιμοποιήθηκαν

6.1 Εισαγωγή

Όπως έχουμε αναφέρει ξανά, για την ομαλή εφαρμογή της ιδέας γύρω από την μοντελοκεντρική μηχανική χρειάζεται μια πληθώρα απαραίτητων εργαλείων που το κάθε ένα αφορά διαφορετικές πτυχές της. Κατά την διάρκεια της συγκεκριμένης ατομικής διπλωματικής εργασίας, έχουν μελετηθεί αρκετά εργαλεία από τα οποία έχουν επιλεγεί κάποια βάση κάποιων κριτηρίων. Τα κριτήρια αυτά ήταν συνήθως κατά κύριο λόγο θέματα συμβατότητας και υποστήριξης διαλειτουργικότητας μεταξύ των εργαλείων αλλά και σωστή αξιοποίηση το παραγώγων τους. Ακολούθως, παρατίθενται τα επιλεχθέντα εργαλεία μαζί με μια μικρή περιγραφή για τον σκοπό και τις ιδιότητες του καθενός.

6.2 Eclipse Modelling Framework (EMF)

Το Eclipse Modelling Framework είναι μια υλοποίηση ενός πλαισίου μοντελοποίησης βασισμένο στο γνωστό Eclipse IDE. Σε αυτό εμπεριέχονται υποδομές για μοντελοποίηση και παραγωγή κώδικα για κτίσιμο εργαλείων και άλλων εφαρμογών βασισμένων σε ένα δομημένο μοντέλο δεδομένων. Από μια

προδιαγραφή μοντέλου που αναπαρίσταται σε μορφή XMI, το EMF παρέχει εργαλεία για παραγωγή κώδικα είτε σε Java, είτε σε C++ γλώσσας προγραμματισμού, δημιουργώντας έτσι τις απαραίτητες κλάσεις, κλάσεων προσαρμογέων (adapters classes) που επιτρέπουν την προβολή και την επεξεργασία βάση εντολών σε ένα μοντέλο αλλά και ένα βασικό επεξεργαστή (editor) για αυτά. Τα μοντέλα μπορούν να εκφραστούν με τη χρήση Java, UML, XML εγγράφων ή άλλων εργαλείων μοντελοποίησης, που αργότερα θα εισαχθούν στο EMF. Το πιο βασικό του πλεονέκτημα είναι ότι το EMF ως σύνολο εργαλείων προσδίδει ένα θεμέλιο για διαλειτουργικότητα μεταξύ άλλων εργαλείων ή εφαρμογών που είναι βασισμένα ή κατασκευασμένα στο EMF. Είναι πρέπον να σημειώσουμε πως το EMF βασίζεται στο μεταμοντέλο του Ecore για να επιτρέπει την εκφραστικότητα σε άλλες μορφές και είδη μοντέλων που μπορούν να κατασκευαστούν μέσω αυτού.

6.3 Papyrus (Eclipse Plug-in)

[5] Το Papyrus είναι ένα Eclipse Plug-in βασισμένο στο προαναφερθέντος Eclipse Modeling Framework (EMF). Αποτελεί ένα περιβάλλον μοντελοποίησης, πιο πλούσιο από αυτό του EMF. Σε αυτό έχουν υλοποιηθεί αρκετά γνωστά πρότυπα προς χρήση όπως: UML 2.5.0, SysML 1.1 & 1.4, OCL 2.3.1, fUML 1.1, ALF 1.0.1, MARTE 1.1 (incubation), EAST-ADL (incubation), RobotML (incubation), UML-RT (incubation) and ISO/IEC 42010.

Μπορεί να συγκεκριμενοποιηθεί για κάποιο πεδίο γνώσης. Κάθε μέρος του Papyrus μπορεί σε αρκετές περιπτώσεις να προσαρμοστεί, όπως για παράδειγμα τα εξής: UML profile, model explorer, diagram notation and style, properties views, παλέτες και τα μενού δημιουργίας, αλλά και αρκετά άλλα.

Το εργαλείο αυτό προσφέρει επίσης την δυνατότητα εφαρμογής μοντελοκεντρικών τεχνικών όπως: η μοντελοκεντρική προσομοίωση, μοντελοκεντρικούς τυπικούς ελέγχους, εξερεύνηση αρχιτεκτονικών και πολλές άλλες.

6.4 Unified Modelling Language (UML)

[8] Η ενοποιημένη γλώσσα μοντελοποίησης (Unified Modeling Language – UML) είναι μια προδιαγραφή που έχει καθοριστεί με στόχο να προσφέρει στους αρχιτέκτονες συστημάτων, μηχανικών λογισμικού και προγραμματιστών εφαρμογών διάφορα εργαλεία για την ανάλυση, το σχεδιασμό και την υλοποίηση λογισμικών συστημάτων και εφαρμογών όπως γίνεται σε άλλες παρόμοιου τύπου διαδικασίες.

Από τα αρχικά κιόλας στάδια, η UML προέρχεται από αντικειμενοστραφείς μεθόδους και ενσωματώνει ένα αριθμό από τις καλύτερες πρακτικές όσο αφορά τη σχεδίαση γλωσσών μοντελοποίησης, τον αντικειμενοστραφή προγραμματισμό και τις γλώσσες αρχιτεκτονικών περιγραφών.

Κατά καιρούς η UML εμπλουτίστηκε με σημαντικούς και ακριβής ορισμούς σχετικά με την αφαιρετική σύνταξη των κανόνων της και της σημασιολογία τους, με μια πιο δομική γλώσσα και με μια σπουδαία και βελτιωμένη ικανότητα για μοντελοποίηση συστημάτων μεγάλης εμβέλειας (large-scale systems).

Η UML συνήθως αναπαρίσταται με διάφορα καλά ορισμένα δομικά στοιχεία που συνθέτουν μια σωρεία σχεδιαγραμμάτων τα οποία έχουν συγκεκριμένη σημασιολογία και σκοπό, όπως και το κάθε δομικό στοιχείο ξεχωριστά που τα αποτελεί.

6.5 Acceleo (Eclipse Plug-in)

[2] Το Acceleo είναι μια δογματική υλοποίηση της προδιαγραφής του Object Management Group (OMG) σχετικά με την μετατροπή μοντέλων σε γλώσσα κειμένου (Model to Text – M2T) [9], αναφερόμενο στο πρότυπο της Model Transformation Language (MTL).

Το εργαλείο αυτό προσφέρει μερικά υπό-εργαλεία ή άλλες υλοποιήσεις που βοηθούν στην δήλωση κανόνων σε MTL και στην προσπέλαση, την μετάφραση και την επεξεργασία των μοντέλων μέσω αυτής.

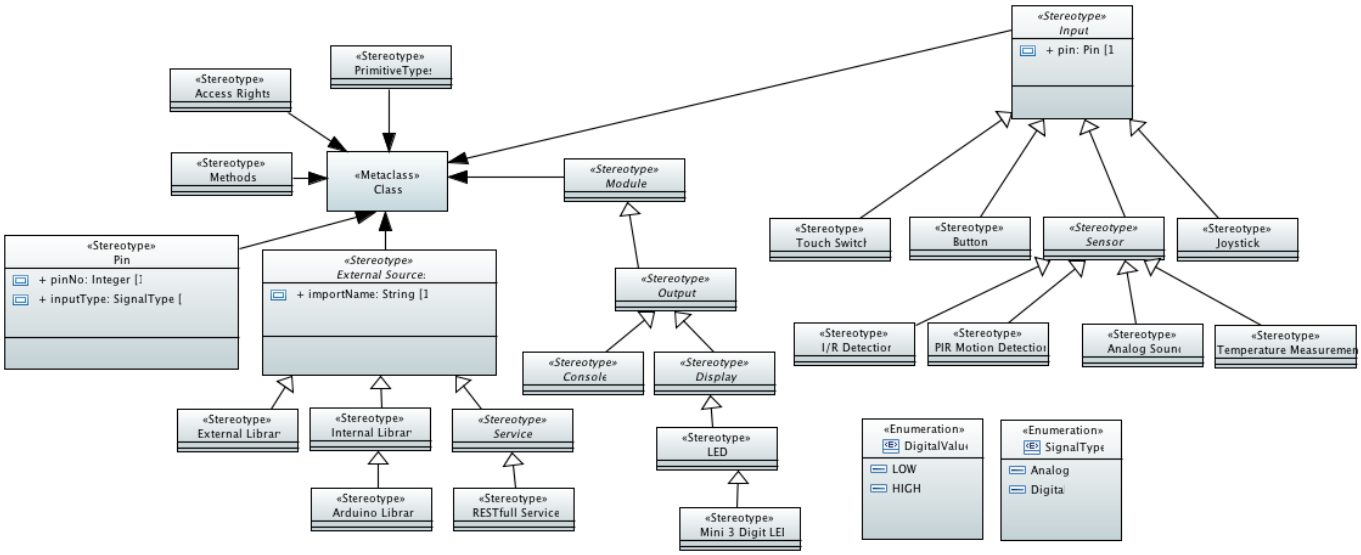
Είναι ένα Eclipse Plug-in το οποίο συνήθως χρησιμοποιείται σε συνεργασία με το προαναφερόμενο περιβάλλοντος μοντελοποίησης Papyrus ως συμπληρωματικό.

Κεφάλαιο 7

Εργαλεία που υλοποιήθηκαν

7.1 Platform Independent Metamodel (PIM) for IoT

Σύμφωνα με τα προαναφερθέντα σχετικά με τις πρακτικές της μοντελοκεντρικής μηχανικής και την σημασία του καθορισμού των μεταμοντέλων, έχει υλοποιηθεί ένα μεταμοντέλο ανεξαρτήτου πλατφόρμας (Platform Independent Metamodel – PIM) για τους σκοπούς της παρούσας διπλωματικής εργασίας. Κύριος στόχος, η άντληση της αφαιρετικότητας των βασικών εννοιών σε μια πλατφόρμα μικροελεγκτών για υλοποίηση IoT εφαρμογών ώστε να μπορούν να παραχθούν σωστά και άρτια μοντέλα μέσω αυτού. Είναι υλοποιημένο ουσιαστικά ακολουθώντας το πρότυπο της UML 2.5 καθώς επίσης επεκτείνει έννοιες της για τους σκοπούς υιοθέτησης του IoT. Για την κατασκευή του χρησιμοποιήθηκε το περιβάλλον μοντελοποίησης Papyrus. Το μεταμοντέλο παρουσιάζεται παρακάτω ως γραφική αναπαράσταση:



Σχήμα 7.1

Στο μεταμοντέλο επεκτείνεται κατά κύριο λόγο η έννοια της μετακλάσης της υπάρχουσας κλάσης στο UML. Σε αυτήν επεκτείνονται οι έννοιες ύπαρξης επιπλέον μεθόδων για IoT, διάφορα δικαιώματα πρόσβασης για ασφάλεια, κάποιοι αρχέγονοι τύποι δεδομένων που χρησιμοποιούνται ανά πλατφόρμα μικροελεκτή, κάποια modules εξαρτημάτων εξόδου, κάποιους αισθητήρες και εξαρτήματα εισόδων και κάποιες επιπλέον εξωτερικές πηγές.

Στις εξωτερικές πηγές περιλαμβάνονται όλα όσα αφορούν εξωτερικές βιβλιοθήκες και APIs τρίτων κατασκευαστών, ενσωματωμένες βιβλιοθήκες των κατασκευαστών των ανάλογων πλατφόρμων μικροελεγκτών (πχ. Arduino Library), ενώ εντάσσονται σε αυτό κάποιες επιπλέον υπηρεσίες με τις οποίες μπορεί να επικοινωνήσει μια εφαρμογή IoT (πχ. RESTful Services).

Ως εισόδους έχουμε μια πληθώρα ενδεικτικών επιλεχθέντων αισθητήρων και εξαρτημάτων όπως αναφέραμε στο Κεφάλαιο 5. Η επιλογή έγινε με βάσει κάποιων «κοινών» χαρακτηριστικών ούτως ώστε να είναι εφικτή η μοντελοποίηση της αφαιρετικής τους έννοιας και της γενίκευσή τους για να εφαρμοστούν αργότερα σε άλλα παρόμοιου τύπου αισθητήρες και εξαρτήματα που απορρέουν από αυτά. Έχουμε λοιπών κάποιους διακόπτες αφής, κουμπιά, χειριστήρια, και αισθητήρες εκ των οποίων κάποιοι είναι υπέρυθρης τεχνολογίας, αισθητήρων κίνησης, αναλογικού ήχου και ελέγχου θερμότητας.

Για τον ίδιο λόγο με τα προαναφερθέν επιλέγηκαν και κάποια εξαρτήματα εξόδου όπως το κοινό console για παρουσίαση διαφόρων μηνυμάτων, μετρικών και μορφής της διεπαφής, ενώ ακόμη προστέθηκαν και κάποιες ψηφιακές οθόνες ή λαμπάκια φωτεινών ενδείξεων.

Καλό είναι να αναφερθεί πως το μοντέλο μπορεί να επεκταθεί για υποστήριξη περισσοτέρων εξαρτημάτων, ή και πλατφόρμων μικροελεγκτών, με ένα τρόπο πιο άρτιο αργότερα που ίσως καθορίζει και τα διάφορα επιμέρους θέματα, όπως για παράδειγμα τα θέματα ασφαλείας και δικαιωμάτων πρόσβασης.

7.2 The IoT UML Model Creator (Eclipse Plug-in / Framework)

Βάσει του προηγούμενου μεταμοντέλου που υλοποιήθηκε μέσω του Eclipse δημιουργήθηκε ένα αντίστοιχο εργαλείο, plug-in, που ορίζει ουσιαστικά ένα πλαίσιο κανόνων βάσει της εφαρμογής πλέον του μεταμοντέλου που έχει οριστεί για υποστήριξη του IoT. Ο αρχιτέκτονας ή μηχανικός λογισμικού έτσι μπορεί να σχεδιάσει δομικά στοιχεία μοντέλων που πλέον περιέχουν την λογική πίσω από το IoT και αναπαριστούν τα απαραίτητα επιθυμητά στοιχεία αναλόγως του στοιχείου στα οποία κάποιος πλέον θα αναφέρεται. Επίσης, το μεταμοντέλο μπορεί να εφαρμοστεί και με διάφορους άλλους τρόπους μέσω του εργαλείου Papyrus.

7.3 Παραγωγοί Κώδικα

Ακολουθώντας το πρότυπο του OMG για την Modeling Transformation Language (MTL) και με την χρήση του συμπληρωματικού εργαλείου Acceleo, υλοποιήθηκαν δύο παραγωγοί κώδικα στους οποίους έχουν δημιουργηθεί οι κατάλληλοι κανόνες παραγωγής για τις αντίστοιχες πλατφόρμες μικροελεγκτών, αισθητήρες, λοιπά εξαρτήματα και γλώσσες προγραμματισμού οι οποίες έχουν επιλεγεί.

Συγκεκριμένα ο ένας παραγωγός ή μετασχηματιστής κώδικα για Arduino, παράγει κώδικα στη γλώσσα προγραμματισμού C Wiring, ενώ ο δεύτερος παραγωγός κώδικα για Raspberry Pi, παράγει κώδικα στη γλώσσα προγραμματισμού Java με την χρήση της βιβλιοθήκης General Purpose Input Output (GPIO).

Οι παραγωγοί κώδικα που έχουν υλοποιηθεί δεν προσδίδουν μια πλήρη υλοποίηση εφαρμογής σε IoT, αλλά δημιουργούν τον κύριο σκελετό της εφαρμογής που εφαρμόζει σωστό τρόπο δήλωσης των αισθητήρων και των λοιπών εξαρτημάτων, καθώς και τις ιδιομορφίες της ανάλογης πλατφόρμας στο οποίο παράγεται. Μαζί με τον σκελετό, προσφέρει σε αρκετά σημεία σημάνσεις προς τους μηχανικούς λογισμικού, ώστε να προσθέσουν, αφαιρέσουν ή τροποποιήσουν τυχόν σημεία ενδιαφέροντος προβάλλοντάς τα σε μορφή κεφαλαίων σχολίων στον κώδικα.

Έχει υλοποιηθεί επίσης ένας τρίτος παραγωγός, ο οποίος ουσιαστικά εξάγει μέρος της απαραίτητης πληροφορίας από σχεδιαγράμματα καταστάσεων (state-machine diagrams) ορισμένα στη UML με στόχο της προσθήκης ελάχιστης λογικής στον παράγωγο κώδικα αργότερα.

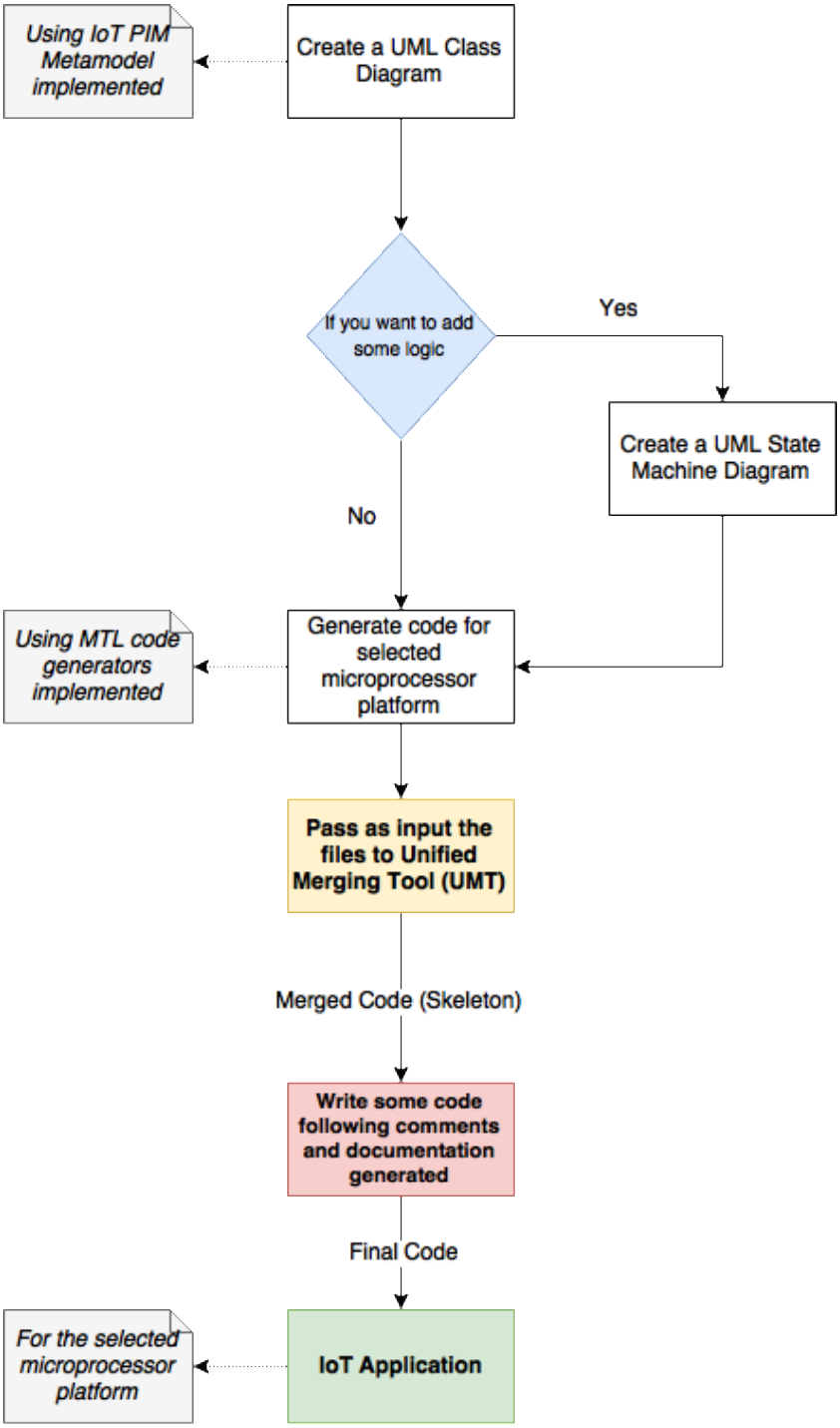
7.4 Unified Merging Tool (UMT)

Λόγω του ότι οι παραγωγοί κώδικα αντιμετώπιζαν διάφορα προβλήματα ή εμπόδιζαν την σωστή παραγωγή κώδικα λόγω απουσίας πλήρους υλοποίησης από τα εργαλεία που τα υποστηρίζουν, έχει υλοποιηθεί σε γλώσσα προγραμματισμού Java ένα επιπλέον εργαλείο που ονομάστηκε «Ενοποιημένο εργαλείο συνένωσης» για τους σκοπούς της παρούσας διπλωματικής εργασίας.

Το παρόν εργαλείο μπορεί να φιλτράρει μερικά βασικά λάθη στην παραγωγή κώδικα από τα προϊόντα των παραγωγών κώδικα, αντικαθιστώντας τα συγκεκριμένα σημεία με την ανάλογη επιθυμητή αναπαράσταση. Επίσης συνενώνει τα επί μέρους αρχεία που δημιουργούνται σε ένα ενιαίο λειτουργικό σκελετό κώδικα.

Μια επιπλέον λειτουργία που προστέθηκε είναι η δυνατότητα προσθήκης μερικής λογικής στον παράγωγο τελικό κώδικα όπως έχει οριστεί από σχεδιαγράμματα καταστάσεων (state-machine diagrams) και παραχθεί από τον αντίστοιχο παραγωγό κώδικα.

Στο σχήμα 7.2 μπορείτε να δείτε γραφικά την διαδικασία που ακολουθείται πριν τη χρήση του UMT, κατά την διάρκεια και με το πέρας της χρήσης του για παραγωγή του τελικού προϊόντος κώδικα.


Using IoT PIM Metamodel implemented
Create a UML Class Diagram
If you want to add some logic
Yes
Create a UML State Machine Diagram
No
Using MTL code generators implemented
Generate code for selected microprocessor platform
Pass as input the files to Unified Merging Tool (UMT)
Merged Code (Skeleton)
Write some code following comments and documentation generated
Final Code
For the selected microprocessor platform
IoT Application

Σχήμα 7.2

Κεφάλαιο 8

Παραδείγματα Επιλεγμένων IoT Εφαρμογών

8.1 Εισαγωγή

Έχοντας ως σκοπό την μελέτη χαρακτηριστικών εφαρμογών στις επιλεχθέντες πλατφόρμες, στις γλώσσες προγραμματισμού τους και στους διάφορους αισθητήρες και εξαρτήματα που μπορούν να χρησιμοποιηθούν σε μια IoT εφαρμογή, επιλέχτηκαν συγκεκριμένα χαρακτηριστικά έργα για την κάθε μια.

Η ύπαρξη του συγκεκριμένου συνόλου εφαρμογών βοήθησε όχι μόνο στην υλοποίηση των αντίστοιχων κανόνων, αλλά επίσης και στην επαλήθευση των παραγόντων προϊόντων από αυτούς.

Σε αυτό το κεφάλαιο θα αναφερθούν τα έργα IoT που έχουν επιλεγεί, περιγράφοντας το σκοπό τους και για ποιες πλατφόρμες μικροελεγκτών και γλώσσες προγραμματισμού αποσκοπούν ή και βιβλιοθήκες. Τα έργα αυτά έχουν περισυλλεγεί από αποθετήρια όπως το hackster.io και το GitHub.

Τέλος θα σχολιαστούν περιεκτικά τα αποτελέσματα που παράγονται μετά την ολοκλήρωση όλης της διαδικασίας. Δηλαδή πρώτα της σχεδίασης των αντίστοιχων μοντέλων για την συγκεκριμένη εφαρμογή, την εκτέλεση των παραγώγων κώδικα για

την πλατφόρμα και γλώσσα επιλογής τους, την «προαιρετική» προσθήκη λογικής μέσω σχεδιαγραμμάτων καταστάσεων και την μετάφρασή τους σε μορφή κειμένου, και τέλος την συνένωση και φιλτράρισμα των απαραίτητων πτυχών του παραγόμενου κώδικα ούτως ώστε να δημιουργηθεί μια άρτια αρχική λύση στην οποία ο μηχανικός λογισμικού θα μπορεί να βασιστεί.

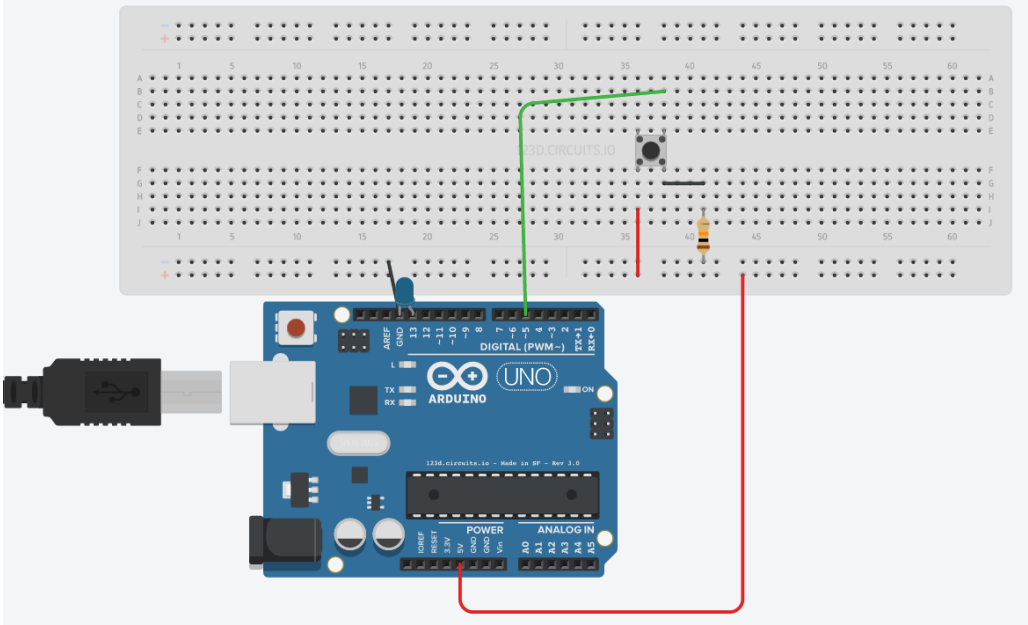
8.2 Έργα IoT που επιλέχτηκαν

8.2.1 The Twitter Button

Ένα απλό ενδεικτικό έργο που συνδυάζει κώδικα και για τις δύο πλατφόρμες ελεγκτών που έχουν επιλεγεί. Στο συγκεκριμένο έργο όμως, θα επικεντρωθούμε όμως μόνο στον κώδικά του για Arduino. Κάνει χρήση ενός και μόνου εξαρτήματος, κατακρίβειαν ένα ψηφιακό κουμπί.

Σκοπός του συγκεκριμένου έργου είναι να αποφασίσει ο χρήστης το τι θα φορέσει την μέρα εκείνη. Με το πάτημα ενός κουμπιού, ο χρήστης θα ενημερώνεται μέσω του Twitter Feed του σχετικά με το τι θα πρέπει να επιλέξει από την γκαρνταρόμπα του. Το μόνο που έχει να κάνει είναι να έχει ένα φάκελο με όλες τις φωτογραφίες της γκαρνταρόμπας του.

Με την χρήση ενός Arduino Uno πρέπει να γίνει η πιο κάτω συνδεσμολογία:



Σχήμα 8.1

Το παρόν έργο πάρθηκε από το Hackster.io και βρίσκεται στον ακόλουθο σύνδεσμο: https://www.hackster.io/gatoninja236/what-to-wear-twitter-button-7fa34b?ref=channel&ref_id=424_trending___&offset=29

8.2.2 Security Motion Sensor

Ένα επίσης αρκετά απλό έργο που συνδυάζει κώδικα και για τις δύο πλατφόρμες ελεγκτών που έχουν επιλεγεί. Θα επικεντρωθούμε επίσης στον κώδικα για Arduino. Κάνει χρήση ενός και μόνου αισθητήρα κίνησης, συγκεκριμένα ένα PIR Sensor.

Σκοπός του συγκεκριμένου έργου είναι να ανιχνεύσει και να ειδοποιήσει το χρήστη εάν κάποιος έχει παραβιάσει φυσικά την ιδιωτικότητά του. Για παράδειγμα, εάν κάποιος κλέφτης έχει εισέλθει στο σπίτι του ή κάποιο άτομο έχει εισέλθει σε κάποιο απαγορευμένο μέρος για το οποίο δεν είναι εξουσιοδοτημένος. Το Arduino ανιχνεύει την κίνηση στο χώρο με την χρήση του αισθητήρα και αποστέλλει την κατάστασή του στο Raspberry Pi για να γίνει η ανάλογη ενημέρωση του χρήστη. Ο κώδικας σε Arduino μπορεί να λειτουργήσει και ως αυτούσιος καθώς υπάρχει και η ανάλογη έξοδος μηνυμάτων από αυτό.

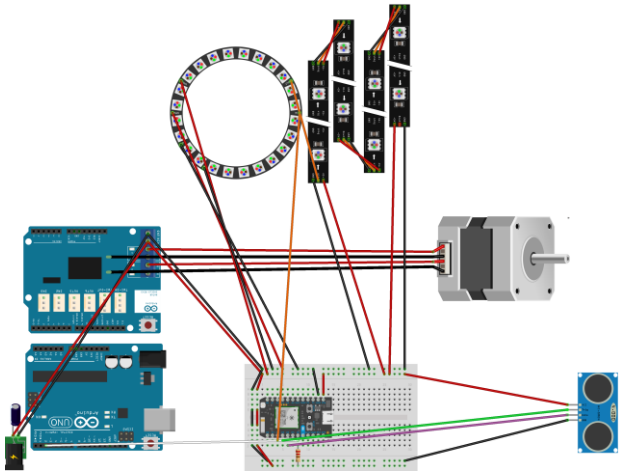
Μπορείτε να βρείτε τον κώδικα του παρόντος έργου στο Παράρτημα Α.

8.2.3 WiFi Death Lamp

Ένα έργο αποκλειστικά για τις πλατφόρμες μικροελεγκτών Arduino. Κάνει χρήση αρκετών αισθητήρων και πρόσθετων εξαρτημάτων.

Σκοπός του συγκεκριμένου έργου είναι βάσει εντολών από τον χρήστη να διαχειρίζεται τη συγκεκριμένη λάμπα, στην ενεργοποίηση, απενεργοποίηση, αλλαγή χρώματος LED ενδείξεων και άνοιγμα – κλείσιμο των μερών της.

Το έργο πρέπει να ακολουθεί την παρακάτω συνδεσμολογία:



Σχήμα 8.2

Το παρόν έργο πάρθηκε από το Hackster.io και βρίσκεται στον ακόλουθο σύνδεσμο: https://www.hackster.io/jhnskog/death-lamp-196c55?ref=channel&ref_id=424_trending___&offset=45

8.2.4 GPIO Ri4J Example Project

Ένα έργο υλοποιημένο μόνο για Raspberry Pi σε γλώσσα προγραμματισμού Java. Κύριο συστατικό του αποτελεί η βιβλιοθήκη General Purpose Input Output (GPIO). Λόγω της εξαιρετικά μεγάλης ανομοιότητας και κατακερματισμού μεταξύ των τεχνολογιών και των βιβλιοθηκών που υπάρχουν, επιλέγηκε τελικά η συγκεκριμένη η οποία είναι γενικού σκοπού, ώστε να μπορεί να καθοριστεί μια πρώτη δουλειά πάνω σε πλατφόρμες μικροελεγκτών Raspberry Pi τουλάχιστον σε μια πιο δομημένη γλώσσα όπως η Java.

Το παρόν έργο πάρθηκε από το pi4j.com και βρίσκεται στον ακόλουθο σύνδεσμο: http://pi4j.com/example/control.html

8.3 Αποτελέσματα

Κατά την εκτέλεση της απαιτούμενης διαδικασίας που περιγράφεται στην εισαγωγή του παρόντος κεφαλαίου, τα αποτελέσματα αν και αρχικά, είναι αρκετά ενθαρρυντικά καθώς παράγεται ο ανάλογος απαιτούμενος σκελετός κώδικα με κάποιες ενδείξεις σε μορφή σχολίων προς τον μηχανικό λογισμικού. Ο σκελετός κώδικα που παράγεται φαίνεται να συνάδει ορθά στις αντίστοιχες γραμμές κώδικα του αναφερθέντος έργου σε κάθε περίπτωση. Ένα ενδεικτικό ποσοστό παραγωγής κώδικα που συνήθως παράγεται, βάση του υπολογισμού <γραμμές κώδικα που παράγονται> / <συνολικές γραμμές κώδικα που απαιτούνται για την πλήρη υλοποίηση>, είναι γύρω στο 30,7%.

Κεφάλαιο 9

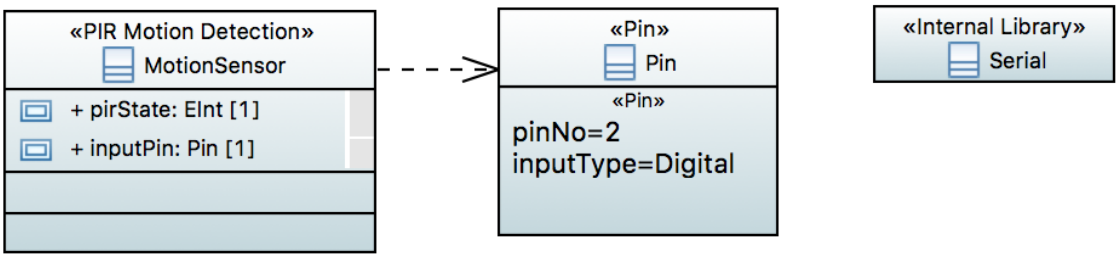
Δοκιμές

9.1 Εισαγωγή

Για επαλήθευση των αποτελεσμάτων διεξήχθησαν κάποιες δοκιμές βάση κάποιων παραδειγμάτων που επιλέχθηκαν αλλά και κάποιων άλλων πιο απλών σεναρίων χρήσης. Στο παρόν κεφάλαιο παρατίθενται κάποια από αυτά αντιστοιχούμενα από τα UML διαγράμματα κλάσεων ή state machine διαγραμμάτων που έχουν σχεδιαστεί, μαζί με τον παράγωγο κώδικα στο οποίο καταλήγουμε μετά την εκτέλεση ολόκληρης της διαδικασίας.

9.2 Σενάριο χρήσης βάση επιλεγμένου έργου

Το επιλεγμένο έργο που θα δειχθεί σε αυτό το έγγραφο για παρουσίαση της δοκιμής του είναι το Motion Detection Project όπως αναφέρθηκε σε προηγούμενο κεφάλαιο. Ακολουθεί το UML διάγραμμα κλάσεων του, ενώ τον ολοκληρωμένο κώδικα του έργου και τον παράγωγό του μπορείτε να τα βρείτε στο Παράρτημα Α.



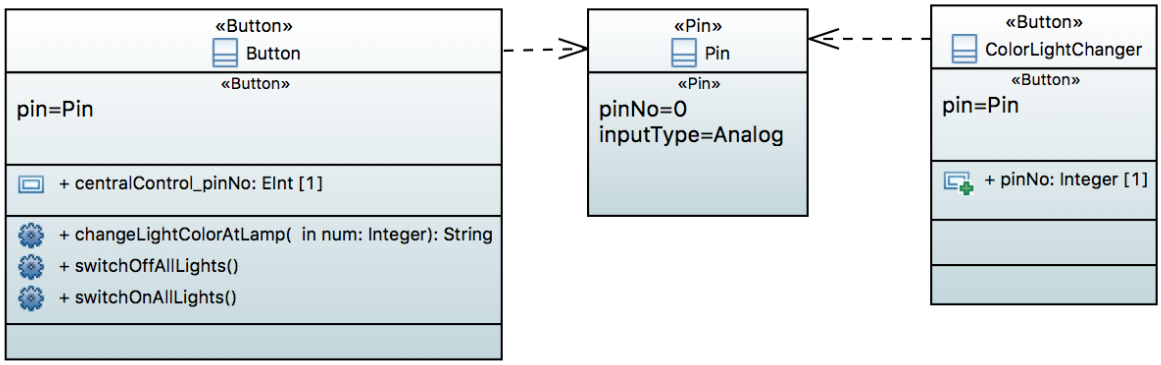
Σχήμα 9.1: UML Class Diagram for Motion Detection Project

9.3 Σενάριο χρήσης βάση παραδείγματος

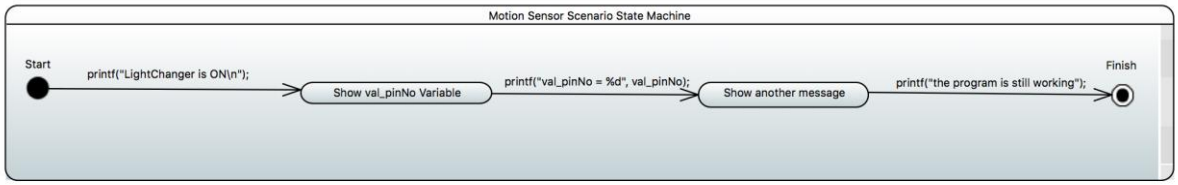
Για τους σκοπούς της παρουσίασης της παρούσας διπλωματικής, κατασκευάστηκε ένα παράδειγμα έργου για επαλήθευση της όλης διαδικασίας, πέραν των τυπικών έτοιμων έργων που έχουν επιλεγεί.

Το συγκεκριμένο παράδειγμα λοιπόν αφορά μια εφαρμογή σε πλατφόρμα μικροελεγκτών Arduino. Με την χρήση ενός κουμπιού και ενός χρωματικού ρυθμιζόμενου λαμπτήρα, ο χρήστης μπορεί με κάθε πάτημα να αλλάζει το χρώμα του λαμπτήρα.

Ακολουθεί η αναπαράσταση σε UML διαγράμματα κλάσεων, UML διαγράμματα καταστάσεων για προσθήκη λογικής και τον τελικό παράγωγο κώδικα μετά την χρήση του εργαλείου UMT.



Σχήμα 9.2: UML Class Diagram



Σχήμα 9.3: UML State Machine Diagram

Τον παράγωγο κώδικα μπορείτε να τον βρείτε στο Παράρτημα Α.

Κεφάλαιο 10

Συμπεράσματα

10.1 Γενικά σχόλια

Ένα από τα δυσκολότερα μέρη κατά την διάρκεια της συγκεκριμένης ατομικής διπλωματικής εργασίας ήταν η εκμάθηση καινούριων τεχνολογιών, μεθοδολογιών και πρακτικών αλλά και υιοθέτησή τους.

Υπάρχουν διάφορα εργαλεία και τεχνολογίες οι οποίες δεν είναι πλήρως ή καλά καθορισμένες ακόμη, με αποτέλεσμα να αποτελεί τροχοπέδη της όλης διπλωματικής. Ένα άλλο μεγάλο εμπόδιο ήταν η ανομοιότητα και η εν μέρους ή καθόλου συμβατότητα μεταξύ των διαφόρων εργαλείων, πλατφόρμων μικροελεγκτών ή αισθητήρων και εξαρτημάτων που χρησιμοποιούνται. Υπάρχει μια πληθώρα αισθητήρων και εξαρτημάτων, που πολλές φορές το καθένα φέρει μαζί του ιδιαίτερα χαρακτηριστικά για υποστήριξη της ανάλογης λειτουργικότητάς του. Αυτό δυσκολεύει την γενίκευσή του στην αφαιρετικότητα ώστε να μοντελοποιηθεί σωστά για μια πιο ευρέα γκάμα χρήσεως.

Άλλα εμπόδια μπορούν να χαρακτηριστούν και η ανομοιότητα υλοποίησης εφαρμογών στις πλατφόρμες μικροελεγκτών καθώς υπάρχουν σχεδόν άπειροι τρόπου, όπου ο καθένας χρησιμοποιεί συνήθως και μια διαφορετική γλώσσα κάνοντας χρήση διαφορετικών βιβλιοθηκών, όχι και τόσο γνωστών, που κατακερματίζουν την γενική φιλοσοφία του IoT και αυτής της μοντελοκεντρικής μηχανικής. Πράγμα που καθιστά και δύσκολη τη μοντελοποίησης τους.

10.2 Εισηγήσεις για μελλοντική εργασία

Θα ήταν καλό να γίνουν επί μέρους μελέτες όσο αφορά την δημοτικότητα των εργαλείων, των αισθητήρων, των εξαρτημάτων, των γλωσσών προγραμματισμού και βιβλιοθηκών που χρησιμοποιούνται ανά πλατφόρμα μικροελεγκτών, με στόχο πάντοτε την ομαλή αφαιρετική γενίκευσή τους σε ένα ακόμη πιο πλούσιο μεταμοντέλο για να μπορούν να μοντελοποιηθούν ορθά.

Μια άλλη πρόταση είναι ο επανακαθορισμός ή και βελτίωση των εργαλείων που υλοποιήθηκαν βάση των καινούριων πλέον αποφάσεων που θα απορρέουν από τις νέες μελέτες στα επί μέρους θέματα.

Συστήνεται να υλοποιηθούν εξαρχής εργαλεία για παραγωγή κώδικα. Εάν δηλαδή υπάρχει η δυνατότητα ορισμού μιας καινούριας γλώσσας σήμανσης για μετατροπή των μοντέλων, πλήρως λειτουργικής και καλά ορισμένης, θα ήταν άκρως επιθυμητό.

Ακόμη χρειάζεται συνένωση όλων των εργαλείων σε ένα κοινό πλαίσιο για να μειωθεί ο κόπος που θα ανατεθεί σε κάποιον μηχανικό λογισμικού για να εφαρμόσει την όλη διαδικασία.

Τονίζεται και πάλι η σημαντικότητα της αναθεώρησης, της ανάλυσης και διαλεύκανσης των επί μέρους θεμάτων για να μπορεί το θέμα με το οποίο καταπιάνεται η συγκεκριμένη ατομική διπλωματική εργασία, να βελτιωθεί στο μέγιστο.

Βιβλιογραφία

[1] Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey." Computer networks 54.15 (2010): 2787-2805.

[2] Acceleo, https://www.eclipse.org/acceleo/

[3] Arduino, https://www.arduino.cc

[4] DZONE. “DZONE’s guide to Internet of Things”, Volume III (2016)

[5] Eclipse Papyrus, https://eclipse.org/papyrus/

[6] Gubbi, Jayavardhana, et al. "Internet of Things (IoT): A vision, architectural elements, and future directions." Future Generation Computer Systems 29.7 (2013): 1645-1660.

[7] Kent Stuart. “Model driven Engineering”. University of Kent, Canterbury, UK

[8] OMG, “Unified Modeling Language”, Version 2.5 (2015)

[9] OMG, “MOF Model to Text Transformation Language”, Version 1.0 (2008)

[10] Raspberry Pi, https://www.raspberrypi.org

[11] Schmidt, Douglas C. "Model-driven engineering." COMPUTER-IEEE COMPUTER SOCIETY- 39.2 (2006): 25.

[12] Sendall Shane and Kozaczynski Wojtek, “Model Transformation – the Heart and Soul of Model-Driven Software Development”, (2003)

[13] 10 Best Microcontroller Boards For Hobbyists And Engineers, http://wonderfulengineering.com/10-best-microcontroller-boards-for-hobbyists-and-engineers/

[14] MDT-UML2 Tools How to Use UML Profiles, https://wiki.eclipse.org/MDT-UML2Tools_How_To_Use_UML_Profiles

Παράρτημα Α

Παράγωγος Κώδικας παραδείγματος Υποενότητας 9.3

```
//Button.ino Generated via ArduinoM2T

//Sensor Pin Declarations
const int centralControl_pinNo = 1;    //DEFINE THIS PIN

//Attribute Declarations
int val_centralControl_pinNo = 0;

//ColorLightChanger.ino Generated via ArduinoM2T

//Sensor Pin Declarations
const int pinNo = 1;    //DEFINE THIS PIN

//Attribute Declarations
int val_pinNo = 0;

//DECLARE YOUR OWN VARIABLES HERE

void setup() {
  pinMode(centralControl_pinNo, INPUT);     // declare input pin centralControl_pinNo
  pinMode(pinNo, INPUT);     // declare input pin pinNo
  Serial.begin(9600); //PLEASE VERIFY THIS LINE
}
void loop() {
  val_centralControl_pinNo = digitalRead(centralControl_pinNo);  // read input value of pin centralControl_pinNo
  val_pinNo = digitalRead(pinNo);  // read input value of pin pinNo


  // State Machine functions

   printf("LightChanger is ON\n");
  // Show val_pinNo Variable
   printf("the program is still working");
  // Show another message
   printf("val_pinNo = %d", val_pinNo);

    //DECLARE THE PROGRAM LOGIC BELOW

  }


  //Auto-generated Operations



  /*
   * AutoGenerated Operation for changeLightColorAtLamp()
   * Please write a brief description about the logic of this method.
   */

  public String changeLightColorAtLamp(){

    return resString;
  }

  /*
   * AutoGenerated Operation for switchOffAllLights()
   * Please write a brief description about the logic of this method.
   */

  public void switchOffAllLights(){

    return ;
  }
```

```
/*
 * AutoGenerated Operation for switchOnAllLights()
 * Please write a brief description about the logic of this method.
 */
public void switchOnAllLights(){

  return ;
}
```

```
//DECLARE YOUR OWN OPERATIONS BELOW
```

Αυθεντικός κώδικας έργου Security Sensor

```
// Uses a PIR sensor to detect movement


int inputPin = 2;               // choose the input pin (for PIR sensor)
int pirState = LOW;             // we start, assuming no motion detected
int val = 0;                    // variable for reading the pin status

void setup() {
  pinMode(inputPin, INPUT);     // declare sensor as input
  Serial.begin(9600);
}

void loop(){
  val = digitalRead(inputPin);  // read input value
  if (val == HIGH) {            // check if the input is HIGH
    delay(150);


    if (pirState == LOW) {
      // we have just turned on
      Serial.println("Motion detected!");
      // We only want to print on the output change, not state
      pirState = HIGH;
    }
  }
  else {
    delay(300);
    if (pirState == HIGH){
      // we have just turned off
      Serial.println("Motion ended!");
      // We only want to print on the output change, not state
      pirState = LOW;
    }
  }
}
```

Παράγωγος κώδικας έργου Security Sensor

```
//MotionSensor.ino Generated via ArduinoM2T

//Sensor Pin Declarations
int pirState = 1;    //DEFINE THIS PIN
const int inputPin = 2;    //DEFINE THIS PIN

//Attribute Declarations
int val_pirState = 0;
int val_inputPin = 0;

//Pin.ino Generated via ArduinoM2T

//Sensor Pin Declarations

//Attribute Declarations

//DECLARE YOUR OWN VARIABLES HERE

void setup() {
  pinMode(inputPin, INPUT);     // declare input pin inputPin
  Serial.begin(9600); //PLEASE VERIFY THIS LINE
}
void loop() {
  val_inputPin = digitalRead(inputPin);  // read input value of pin inputPin
  //DECLARE THE PROGRAM LOGIC BELOW

}


//Auto-generated Operations

//DECLARE YOUR OWN OPERATIONS BELOW
```

Οδηγίες Χρήσης Εργαλείων

Γενικά

Για να δουλέψουν τα εργαλεία χρειάζονται κάποιες ρυθμίσεις ή ακολουθούμενες διαδικασίες για να γίνει η σωστή προσαρμογή τους και να εκτελούνται ορθά. Όλα τα απαραίτητα εργαλεία δόθηκαν μαζί με το ψηφιακό δίσκο της παρούσας διπλωματικής εργασίας κατά την παράδοση. Ακολουθούν βέβαια κάποιες οδηγίες για προσαρμογή και τρόπο χρήσης.

Παραγωγοί κώδικα

Πρέπει να δημιουργηθεί από ένα Acceleo έργο για κάθε ένα αρχείο .mtl παραγωγού κώδικα. Πρέπει να ρυθμιστούν από το ίδιο το εργαλείο ο προορισμός του παράγωγου κώδικα, το αρχείο generate.mtl που θα χρησιμοποιεί καθώς και ο στόχος-διάγραμμα που θα ακολουθήσει για να παράξει τον αντίστοιχο κώδικα.

Σημείωση πως μαζί με τα αρχεία αυτά, για κάθε πλατφόρμα μικροελεκτή και γλώσσα, παρατίθεται και ένα αρχείο Generate.java το οποίο πρέπει να αντικαταστήσει ο χρήστης προτού χρησιμοποιήσει τον παραγωγό. Αυτό γιατί υπάρχουν κάποια προβλήματα όσο αφορά διάφορα dependencies τα οποία επιλύνονται σε αυτό με κώδικα που τρέχει προτού γίνει η εκτέλεση του αρχείου .mtl.

Άλλα αναλυτικά dependencies που πρέπει να ακολουθεί το έργο στο εργαλείο Acceleo είναι τα εξής:

Τα συγκεκριμένα μπορούν να ρυθμιστούν από τον οδηγό δημιουργίας του Acceleo κατά την έναρξη ενός νέου έργου παραγωγής, στην αρχή.

Εργαλείο UMT

Εφόσον είναι δημιουργημένο με γλώσσα προγραμματισμού Java μπορεί να εκτελεστεί ανάλογα σε αντίστοιχα περιβάλλοντα που υποστηρίζουν την γλώσσα. Κατά την εκτέλεση του εργαλείου ο χρήστης καλείται να δηλώσει τη λειτουργία που θέλει να κάνει με το εργαλείο, ενώ μετέπειτα ακολουθούν επιπλέον κατευθυντήριες οδηγίες για εισόδους ονομάτων αρχείων προς σύμπτυξη και διόρθωση.

```
Welcome to Unified Merging Tool for IoT

        Menu
        1. Merge Arduino .ino Files
        2. Merge Raspberry Pi .java Files

Please select an option > 1
Please give a name for the output file > MotionProject.ino
Is there any program logic you want to add? [y/n] > n
Please give the files in a straight line separated with spaces and a # at the end: MotionSensor.c Pin2.c #
..[Done] Filenames imported: [MotionSensor.c, Pin2.c]
Reading...Done
Filtering and Merging......Done.....Done
Exporting.................................................Done
```

Από το μενού ο χρήστης έχει την δυνατότητα να επιλέξει τη λειτουργία σύμπτυξης είτε για αρχεία .ino για παραγωγή τελικού Arduino έργου σε γλώσσα προγραμματισμού C (Wiring), είτε για αρχεία .java για παραγωγή τελικού Raspberry Pi έργου σε γλώσσα προγραμματισμού Java.

Νοείται πως τα αρχεία εισόδου, είναι τα αρχεία που παράγονται από τους αντίστοιχους παραγώγους κώδικα για κάθε πλατφόρμα μικροελεγκτών και γλώσσα της επιλογής του μηχανικού λογισμικού.