ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επισκόπηση μοντέλων χαρακτηριστικών και δημιουργία εργαλείου για την μετατροπή τους σε κώδικα λογισμικού

Ναϊκόπουλος Δημήτριος
ΑΜ: 725

Επιβλέπων Καθηγητής: Δρ. Μπίμπη Σταματία

ΚΟΖΑΝΗ, ΙΟΥΝΙΟΣ 2018

Τίτλος: Επισκόπηση μοντέλων χαρακτηριστικών και δημιουργία εργαλείου για την μετατροπή τους σε κώδικα λογισμικού

Περιγραφή: Διπλωματική εργασία στα πλαίσια των σπουδών για την απόκτηση του Διπλώματος που απονέμει το Πανεπιστήμιο Δυτικής Μακεδονίας με τίτλο «Μηχανικός Πληροφορικής και Τηλεπικοινωνιών»

Λέξεις κλειδιά: Feature models, software product lines, software maintainability, maintainability analysis, Feature-Oriented Software Development, Πρόγραμμα εξαγωγής κώδικα.

Δημιουργός: Ναϊκόπουλος Δημήτριος

Ημερομηνία δημιουργίας: 31-05-2018

Χρόνος έκδοσης: 2018

Χώρα έκδοσης: GR

Γλώσσα κειμένου: Gre

Η παρούσα Διπλωματική Εργασία
εκπονήθηκε στα πλαίσια των σπουδών
για την απόκτηση του Διπλώματος
που απονέμει το
Πανεπιστήμιο Δυτικής Μακεδονίας με τίτλο
«Μηχανικός Πληροφορικής και Τηλεπικοινωνιών»

Εγκρίθηκε από την εξεταστική επιτροπή την …./…./2018 αποτελούμενη από τους:

Ονοματεπώνυμο:	Βαθμίδα:	Υπογραφή:
1. …….	…….	…….
2. …….	…….	…….

Δήλωση Πνευματικών Δικαιωμάτων

Ευχαριστίες

Θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια του Τμήματος Μηχανικών Πληροφορικής και Τηλεπικοινωνιών του Πανεπιστημίου Δυτικής Μακεδονίας και συγκεκριμένα τη Λέκτορα κυρία Σταματία Μπίμπη για την εξαιρετική συνεργασία μας καθώς και για την πολύτιμη βοήθεια και καθοδήγηση της καθ' όλη τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω του κοντινούς μου φίλους που απέκτησα κατά την διάρκεια των σπουδών μου στο Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών, για τις όμορφες στιγμές, για τη στήριξη και την άψογη συνεργασία που είχαμε όλα αυτά τα χρόνια.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για την πνευματική και υλική υποστήριξη που μου προσέφεραν όλα αυτά τα χρόνια.

Δημήτριος Ναϊκόπουλος

Κοζάνη, Ιούνιος 2018

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως βασικό στόχο την μελέτη των μοντέλων χαρακτηριστικών καθώς και την ανάπτυξη του εργαλείου «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)» το οποίο υποστηρίζει την ανάλυση των μοντέλων χαρακτηριστικών όσον αφορά την συντηρησιμότητα της εφαρμογής. Επίσης, η εφαρμογή υποστηρίζει την εξαγωγή κώδικα λογισμικού στην αντικειμενοστραφή γλώσσα προγραμματισμού Java, όσο και την εξαγωγή του διαγράμματος κλάσεων του παραγόμενου κώδικα σε μορφή XML. Η εφαρμογή είναι σχεδιασμένη για να λειτουργεί σε ηλεκτρονικούς υπολογιστές όλων των λειτουργικών συστημάτων με την προϋπόθεση να υποστηρίζουν το Java Runtime Environment 8 και νεότερο. Ο χρήστης μπορεί να πλοηγηθεί στην εφαρμογή μέσω του ειδικά διαμορφωμένου και φιλικού προς το χρήστη περιβάλλον πλοήγησης.

Ολοένα και περισσότερες εταιρίες παραγωγής λογισμικού έχουν την ανάγκη να οργανώσουν της παραγωγή τους σε εκτενείς ουρές παραγωγής λογισμικού. Για την διαχείριση, την αποσφαλμάτοση και τον έλεγχο τέτοιον εκτενών συστημάτων θεωρήθηκε επιτακτική η δημιουργία ενός προτύπου σχέσεων λογισμικού όπου θα ήταν σε θέση να αποτυπώσει όχι μόνο την παραγωγή ενός συγκεκριμένου προϊόντος λογισμικού αλλά ολόκληρης της αλυσίδας, καθώς και να παρουσιάσει τα κοινά στοιχεία που μοιράζονται τα προϊόντα της ίδιας γραμμής παραγωγής. Σκοπός αυτής της αλλαγής ήταν η μαζική παραγωγή λογισμικού με την χρήση και την επαναχρησιμοποίηση κώδικα που έχει αναπτυχθεί για την παραγωγή παρόμοιων προϊόντων ίδιου τύπου. Για το λόγο αυτό χρησιμοποιούνται τα μοντέλα χαρακτηριστικών. Συνεπώς είναι απαραίτητο να δημιουργηθεί μία εφαρμογή για να καλύψει το κενό που υπάρχει ανάμεσα στην διαδικασία σχεδίασης και στην διαδικασία παραγωγής κώδικα λογισμικού καθώς και να παρέχει μετρήσεις για την ποιότητα της σχεδίασης της γραμμής προϊόντων λογισμικού.

Συνοψίζοντας τα βασικά στοιχεία αυτής της διπλωματικής εργασίας, επικεντρώνονται σε δύο θεματικούς άξονες. Ποιο συγκεκριμένα, στον πρώτο άξονα, γίνεται διερεύνηση και ανάλυση των μοντέλων χαρακτηριστικών και δίνεται έμφαση στον τρόπο αναπαράστασης της πληροφορίας σε αυτά. Επίσης, αναλύονται οι δύο βασικότερες έννοιες αυτής της διπλωματικής εργασίας όπου είναι η γραμμές προϊόντων λογισμικού και τα μοντέλα χαρακτηριστικών και παρουσιάζονται οι βασικές μετρικές συντηρησιμότητας που στην συνέχεια θα υλοποιηθούν. Τέλος, στο δεύτερο άξονα, γίνεται ανάλυση και παρουσίαση της εφαρμογής που αναπτύχθηκε με σκοπό την επέκταση της χρήσης των μοντέλων χαρακτηριστικών καθώς επίσης παρουσιάζεται και ο τρόπος λειτουργίας της εφαρμογής.

Λέξεις κλειδιά: Feature models, software product lines, software maintainability, maintainability analysis, Feature-Oriented Software Development, Πρόγραμμα εξαγωγής κώδικα.

General overview of feature models and creation of a tool for conversion into software code

Abstract

The present Diploma Thesis has as main goal, the study of the feature models as well as the development of the "Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A.)" tool which supports the analysis of the maintainability of the software product line. Also, the application supports the export of software code in the object-oriented programming language called Java, as well as the extraction of class diagram of the generated code in XML format. The application is designed to work on computers of all operating systems provided they support the Java Runtime Environment 8 and later. The user can navigate the application through the specially customized and user-friendly navigation environment.

More and more software companies have the need to organize their production into extensive software production lines. In order to manage, debug and control such extensive systems, it was considered imperative to create a software relationship model where it would be able to capture not only the production of a particular software product but the whole chain as well as present the common features shared by products on the same production line. The purpose of this change was the mass production of software by using and reusing code developed to produce similar products of the same type. For this reason, feature models are used. It is therefore necessary to create an application to fill the gap between the design process and the software code process and to provide measurements for the quality of the software product line design.

Summarizing the key elements of this Diploma Thesis, we focus on two main topics. Specifically, on the first topic, we investigate and analyze the feature models and emphasize in the way which the information is represented in them. It also analyzes the two main concepts of this Diploma Thesis, which is the software product lines and the feature models, and presents the basic maintainability metrics that will then be implemented in the software application. Finally, the second topic, analyzes and presents the application that has been developed to extend the use of the feature models as well as how the application works.

Keywords: Feature models, software product lines, software maintainability, maintainability analysis, Feature-Oriented Software Development, code export program.

Πίνακας περιεχομένων

1 Εισαγωγή 14
1.1 Σκοπός και στόχοι της εργασίας 15
1.2 Οργάνωση κειμένου 16
2 Θεωρητικό Υπόβαθρο 18
2.1 Μοντέλα χαρακτηριστικών (Feature models) 18
2.2 Γραμμές προϊόντων λογισμικού (Software product lines) 18
2.3 Μαζική παραγωγή και μαζική εξατομίκευση 19
2.3.1 Μαζική παραγωγή και εξατομίκευση σε προϊόντα λογισμικού 19
2.4 Ανάλυση και περιγραφή μοντέλων χαρακτηριστικών (Feature models) 20
2.4.1 Βασικά μοντέλα χαρακτηριστικών (Basic Feature models) 20
2.4.2 Μοντέλα χαρακτηριστικών βασισμένα στην πολλαπλότητα (Cardinality–based feature models) 21
2.4.3 Εκτεταμένα μοντέλα χαρακτηριστικών (Extended feature models) 22
2.4.4 Λειτουργίες ανάλυσης σε μοντέλα χαρακτηριστικών 22
2.4.4.1 Κενό μοντέλο χαρακτηριστικών (Void feature model) 22
2.4.4.2 Έγκυρο προϊόν (Valid product) 23
2.4.4.3 Μερικός έγκυρη διαμόρφωση (Valid partial configuration) 23
2.4.4.4 Εμφάνιση όλων των προϊόντων (All products) 23
2.4.4.5 Αριθμός προϊόντων (Number of products) 23
2.4.4.6 Φίλτρο Προϊόντων (Filter) 23
2.4.4.7 Ανίχνευσή ανωμαλιών (Anomalies detection) 24
2.4.4.8 “Νεκρά” χαρακτηριστικά (Dead features) 24
2.4.4.9 Υπό όρους “νεκρά” χαρακτηριστικά (Conditionally dead features) 24
2.4.4.10 Λάθος προαιρετικά χαρακτηριστικά (False optional features) 24
2.4.4.11 Wrong cardinalities 25
2.4.4.12 Redundancies 25
2.4.4.13 Επεξηγήσεις (Explanations) 25
2.4.4.14 Διορθωτικές εξηγήσεις (Corrective explanations) 25
2.4.4.15 Σχέσεις μοντέλων χαρακτηριστικών (Feature model relations) 26
2.4.4.16 Βελτιστοποίηση (Optimization) 27
2.4.4.17 Βασικά χαρακτηριστικά (Core features) 27
2.4.4.18 Χαρακτηριστικά παραλλαγής (Variant features) 27
2.4.4.19 Ατομικά σετ (Atomic sets) 27

2.4.4.20 Ανάλυση εξαρτήσεων (Dependency analysis) 27
2.4.4.21 Διαμόρφωση πολλαπλών βημάτων (Multi–step configuration) 27
2.4.4.22 Ομοιογένεια (Homogeneity) 27
2.4.4.23 Ομοιότητα (Commonality) 28
2.4.4.24 Παράγοντας μεταβλητότητας (Variability factor) 28
2.4.4.25 Βαθμός ορθογωνικότητας (Degree of orthogonality) 28
2.4.4.26 Extra Constraint Representativeness (ECR) 28
2.4.4.27 Lowest Common Ancestor (LCA) 28
2.4.4.28 Χαρακτηριστικά ρίζας (Root features) 29
2.4.5 Λογική βασισμένη σε προτασιακή λογική 29
2.4.6 Ανάλυση βασισμένη στον προγραμματισμό περιορισμών 30
2.5 Βασική ιδέα τις μετατροπής μοντέλων χαρακτηριστικών σε άλλα μοντέλα 30
2.6 Συνθήκες έμμεσης παρουσίας (Implicit Presence Conditions) 32
2.7 Διαδικασία προτύπου (Template Instantiation) 32
2.8 Αρχιτεκτονική βασιζόμενη σε μοντέλο (Model Driven Architecture) και μηχανική απαιτήσεων γραμμής προϊόντων. 33
2.8.1 Μετασχηματισμούς γραφήματος 34
2.9 Διατηρησιμότητα (Maintainability) των μοντέλων χαρακτηριστικών μίας γραμμής προϊόντων λογισμικού. 35
2.9.1 Σχεδιασμός μετρήσεων 36
2.9.2 Δομικές μετρήσεις μοντέλων χαρακτηριστικών 37
3 Ανάλυση της Εφαρμογής 39
3.1 Περιγραφή της εφαρμογής F.C.C.M.A 39
3.1.1 Δυνατότητα εξαγωγής κώδικα 39
3.1.2 Δυνατότητα εξαγωγής μετρήσεων συντηρησιμότητας 40
3.2 Πρότυπο εφαρμογής F.C.C.M.A 42
3.3 Προγράμματα που χρησιμοποιήθηκαν 44
3.3.1 Eclipse Oxygen, SceneBuilder και JavaFX 44
3.3.2 Πρόγραμμα σχεδίασης μοντέλων χαρακτηριστικών FeatureIDE 49
3.3.3 Αποθήκη μοντέλων χαρακτηριστικών S.P.L.O.T 51
4 Αρχιτεκτονική και Τεχνική ανάλυση του F.C.C.M.A 52
4.1 Περιγραφή Απαιτήσεων και Αρχιτεκτονική Ανάλυση του F.C.C.M.A 52
4.1.1 Περιπτώσεις χρήσης 52
4.1.2 Διάγραμμα κλάσεων 56
4.1.3 Αρχιτεκτονική αρχείων εισόδου 59
4.2 Τεχνική ανάλυση του F.C.C.M.A 61

4.2.1 Εισαγωγή και αναγνώριση κλάσεων αρχείου μοντέλου χαρακτηριστικών 61
4.2.2 Εισαγωγή, τροποποίηση και διαγραφή μεταβλητών στις κλάσεις 67
4.2.3 Αποθήκευση εργασιακού περιβάλλοντος 69
4.2.4 Εξαγωγή στοιχείων ανάλυσης συντηρησιμότητας 70
4.2.5 Εξαγωγή διαγράμματος κλάσεων σε μορφή XML 73
4.2.6 Εξαγωγή κώδικα Java 75
5 Παρουσίαση και εκτέλεση εφαρμογής F.C.C.M.A 77
5.1 Εκκίνηση εφαρμογής 77
5.2 Μοντέλο χαρακτηριστικών εισαγωγής 78
5.3 Εισαγωγή μοντέλου χαρακτηριστικών στο F.C.C.M.A 78
5.4 Διαχείριση Μεταβλητών 80
5.5 Εμφάνιση μετρικών συντηρησιμότητας 81
5.6 Αποθήκευση, εξαγωγή κώδικα και διαγράμματος κλάσης σε XML 81
6 Επίλογος 83
6.1 Συμπεράσματα 83
6.2 Μελλοντικές Επεκτάσεις 84
7 Βιβλιογραφικές αναφορές 86

Εικόνα 1: Παράδειγμα μοντέλου χαρακτηριστικών ενός απλού κινητού τηλεφώνου με χρήση όλων των ιδιοτήτων 21
Εικόνα 2: Παράδειγμα εκτεταμένου μοντέλου χαρακτηριστικών..... 22
Εικόνα 3: Παράδειγμα "νεκρού" χαρακτηριστικού (γκρι χρώμα) 24
Εικόνα 4: Παράδειγμα λάθους προαιρετικού χαρακτηριστικού 24
Εικόνα 5: Παράδειγμα ομάδας χαρακτηριστικών που δεν μπορεί να ικανοποιηθεί..... 25
Εικόνα 6: Παράδειγμα μη απαραίτητων περιορισμών..... 25
Εικόνα 7: Σχέσεις μεταξύ μοντέλων..... 26
Εικόνα 8: Μετατροπή σχέσεων μοντέλων χαρακτηριστικών σε προτασιακή λογική 29
Εικόνα 9: Μετατροπή σχέσεων μοντέλων χαρακτηριστικών σε ψευδοκώδικα..... 30
Εικόνα 10: Μετατροπή μοντέλων χαρακτηριστικών σε άλλα μοντέλα..... 32
Εικόνα 11: Παράδειγμα κανόνα μετατροπής του χαρακτηριστικού ρίζας σε κλάση. 34
Εικόνα 12: Παράδειγμα μετατροπής μοντέλου χαρακτηριστικών σε διάγραμμα κλάσεων.. 35
Εικόνα 13: Η σχέση μεταξύ δομικών ιδιοτήτων, γνωστικής πολυπλοκότητας και εξωτερικών χαρακτηριστικών ποιότητας. 37
Εικόνα 14: Αρχική οθόνη εφαρμογής 43
Εικόνα 15: Οθόνη εισαγωγής νέου μοντέλου χαρακτηριστικών (αριστερά). Οθόνη εισαγωγής έργου από αρχείο (δεξιά). 43
Εικόνα 16: Κύρια οθόνη του συστήματος. 44
Εικόνα 17: Λήψη και εγκατάσταση e(fx)clipse..... 45
Εικόνα 18: Στιγμιότυπο περιβάλλοντος SceneBuilder. 46
Εικόνα 19: Δημιουργία JavaFX project. 47
Εικόνα 20: Μέρος του κώδικα γραφικών της κεντρικής σελίδας της εφαρμογής. 48
Εικόνα 21: Λήψη και εγκατάσταση FeatureIDE. 49
Εικόνα 22: Περιβάλλον FeatureIDE. 50
Εικόνα 23: Εισαγωγή νέου χαρακτηριστικού στο δέντρο. 50
Εικόνα 24: Διαδικασία εξαγωγής μοντέλου χαρακτηριστικών σε XML..... 51
Εικόνα 25: Διάγραμμα περιπτώσεων χρήστη (use case diagram). 56
Εικόνα 26: Διάγραμμα κλάσεων 58
Εικόνα 27: Μετατροπή μοντέλου χαρακτηριστικών σε XML. 60
Εικόνα 28: Στιγμιότυπο αρχείου αποθήκευσης. 60
Εικόνα 29: Απόσπασμα κώδικα κλάσης χαρακτηριστικών..... 62
Εικόνα 30: Αρχείο μοντέλου χαρακτηριστικών με μη εκτυπώσιμους χαρακτήρες. 63
Εικόνα 31: Κώδικας μέτρησης εσοχών και διαγραφής αχρείαστων χαρακτηριστικών. 64
Εικόνα 32: Παράδειγμα κώδικα εισαγωγής χαρακτηριστικών. 66
Εικόνα 33:Κώδικας εισαγωγής περιορισμών και πλήθος παιδιών. 67
Εικόνα 34: Κώδικας εισαγωγής μεταβλητής. 68
Εικόνα 35: Κώδικας ανανέωσης μεταβλητής. 68
Εικόνα 36: Στιγμιότυπο αρχείου αποθήκευσής με περιορισμούς και κλάσεις. 69
Εικόνα 37: Κώδικας δημιουργίας αρχείου αποθήκευσης. 70
Εικόνα 38: Κώδικας εύρεσης συντελεστή συνδεσιμότητας-πυκνότητας. 71
Εικόνα 39: Κώδικας υπολογισμού μετρικής περιορισμών μεταξύ δέντρων. 71
Εικόνα 40: Κώδικας μετρήσεων συντηρησιμότητας. 73
Εικόνα 41: Μέρος του XML διαγράμματος κλάσεων. 74
Εικόνα 42: Παραγώμενος κώδικας Java..... 75
Εικόνα 43: Μέρος του κώδικα παραγωγής κώδικα Java 76
Εικόνα 44: Οθόνη εκκίνησης F.C.C.M.A. 77
Εικόνα 45: Μοντέλο χαρακτηριστικών αριθμομηχανής. 78

Εικόνα 46: Εισαγωγή μοντέλου χαρακτηριστικού αριθμομηχανής από XML 79
Εικόνα 47: Φόρτωση έργου αριθμομηχανής από αρχείο αποθήκευσης. 79
Εικόνα 48: Κεντρική οθόνη με της μεταβλητές του παραδείγματος. 80
Εικόνα 49: Οθόνη στοιχείων συντηρησιμότητας. 81
Εικόνα 50: Παραγόμενος κώδικας παραδείγματος. 82

1 Εισαγωγή

Καθώς ο κόσμος της τεχνολογίας εξελίσσεται, όλο και περισσότερες εταιρίες παραγωγής λογισμικού έχουν την ανάγκη να οργανώσουν της παραγωγή τους σε μεγάλες ουρές παραγωγής λογισμικού. Για την διαχείριση και τον έλεγχο τέτοιον εκτενών συστημάτων θεωρήθηκε επιτακτική η δημιουργία ενός προτύπου σχέσεων λογισμικού όπου θα ήταν σε θέση να αποτυπώσει όχι μόνο την παραγωγή ενός συγκεκριμένου προϊόντος λογισμικού αλλά ολόκληρης της αλυσίδας, καθώς και να παρουσιάσει τα κοινά στοιχεία που μοιράζονται τα προϊόντα της ίδιας γραμμής παραγωγής. Σκοπός αυτής της αλλαγής ήταν η μαζική παραγωγή λογισμικού με την χρήση και την επαναχρησιμοποίηση κώδικα που έχει αναπτυχθεί για την παραγωγή παρόμοιων προϊόντων ίδιου τύπου.

Η μαζική παραγωγή ορίζεται ως η παραγωγή μιας μεγάλης ποσότητας τυποποιημένων προϊόντων με τη χρήση τυποποιημένων διαδικασιών που παράγουν μεγάλο όγκο του ιδίου προϊόντος σε μειωμένο χρόνο στην αγορά [1]. Γενικά, οι απαιτήσεις των πελατών είναι οι ίδιες και δεν πραγματοποιείται προσαρμογή. Μετά τη βιομηχανική επανάσταση, μεγάλες εταιρείες άρχισαν να οργανώνουν την παραγωγή τους σε περιβάλλοντα μαζικής παραγωγής. Ωστόσο, σε αυτήν την άκρως ανταγωνιστική αγορά, η μαζική παραγωγή δεν είναι πια αρκετή και η μαζική προσαρμογή είναι αναγκαία για την επιτυχία στην αγορά. Σκοπός της μαζικής προσαρμογής είναι η ικανοποίηση όσο των δυνατών περισσότερων αναγκών του πελάτη [1]. Ένα απλό παράδειγμα πάνω σε αυτό είναι η προσαρμογή των δυνατοτήτων των κινητών τηλεφώνων πάνω στις ανάγκες του αγοραστικού κοινού. Είναι εύκολα κατανοητό ότι μεγάλες εταιρίες παραγωγής κινητών τηλεφώνων παράγουν διαφορετικά μοντέλα κινητών με σκοπό την απόκτηση μεγαλύτερου μέρους της αγοράς. Αυτό το επιτυγχάνουν παράγοντας συσκευές και λογισμικά που μοιράζονται κοινά χαρακτηριστικά όμως έχουν κάποιες βασικές διαφορές που γίνονται για την μείωση του κόστους. Με αυτόν τον τρόπο, εξασφαλίζεται η μαζική προσαρμογή διατηρώντας την μέγιστη δυνατή μαζική παραγωγή. Η αγορά συστημάτων πληροφορικής είναι ένας ιδιότυπος κλάδος της βιομηχανίας σε σύγκριση με τους πιο παραδοσιακούς κλάδους. Αν παραλληλίσουμε την εξέλιξη την εξέλιξη των εταιριών πληροφορικής με την ιστορία των παραδοσιακών βιομηχανιών, η εκβιομηχάνιση των πληροφοριακών συστημάτων ξεκίνησε με παραδοσιακές μεθόδους, εξελίχθηκε σε μαζική παραγωγή και τώρα επιδιώκει μαζική προσαρμογή για να πετύχει στην αγορά [1].

Η μαζική προσαρμογή των προϊόντων λογισμικού είναι γνωστή ως σειρές προϊόντων λογισμικού (Software Product Lines) ή οικογένειες προϊόντων λογισμικού (Software Product Families). Προκειμένου να καλυφθούν οι εκάστοτε ανάγκες των πελατών, η μηχανική της γραμμής παραγωγής λογισμικού προωθεί την παραγωγή μιας οικογένειας προϊόντων λογισμικού που αποτελούνται από κοινά χαρακτηριστικά, αντί να παράγονται αυτά τα χαρακτηριστικά ένα προς ένα από την αρχή. Οι Γραμμές Προϊόντος Λογισμικού (Software Product Lines) χρησιμοποιούνται ώστε να επιτυγχάνεται ταυτόχρονα υψηλό επίπεδο επαναχρησιμοποίησης και προσαρμογής. Λόγω του κόστους και των χρονικών περιορισμών, ειδικά σε εξελισσόμενους τομείς, η επέκταση των υφιστάμενων γραμμών προϊόντων λογισμικού και η δημιουργία νέων γραμμών προϊόντων λογισμικού από υπάρχοντα προϊόντα, δεν είναι οικονομικά

εφικτή. Για αυτόν τον λόγο κυρίαρχο ρόλο παίζει η διαδικασία προληπτικής προσέγγισής όπου σκοπό έχει να προβλέψει κατά πόσο είναι εύκολο να αναπτυχθεί και να συντηρηθεί η συγκεκριμένη γραμμή προϊόντων λογισμικού στο μέλλον [2]. Για την δημιουργία όμως των γραμμών παραγωγής λογισμικού έπρεπε να απαντηθούν κάποια ερωτήματα που προέκυψαν. Τα ερωτήματα αυτά ήταν πως μπορούμε να καθορίσουμε ένα συγκεκριμένο προϊόν λογισμού και πώς μπορούμε να καθορίσουμε μία συγκεκριμένη γραμμή παραγωγής λογισμικού. Τα προϊόντα σε μια σειρά προϊόντων λογισμικού διαφέρουν από τις δυνατότητές τους, όπου ένα χαρακτηριστικό (feature) είναι μια αύξηση της λειτουργικότητας του προγράμματος. Τα μεμονωμένα προϊόντα προσδιορίζονται με τη χρήση χαρακτηριστικών, ενώ οι σειρές προϊόντων λογισμικού καθορίζονται χρησιμοποιώντας μοντέλα χαρακτηριστικών (feature models) [1].

Οι γλώσσες μοντέλων χαρακτηριστικών είναι μια κοινή οικογένεια αναπαραστατικών γλωσσών που αντιπροσωπεύουν τις σειρές προϊόντων λογισμικού. Ένα μοντέλο χαρακτηριστικών αναπαριστά τις πληροφορίες μίας γραμμής προϊόντων λογισμικού σχετικά με κοινά και μη χαρακτηριστικά της γραμμής προϊόντων λογισμικού σε διαφορετικά επίπεδα. Ένα μοντέλο χαρακτηριστικών αντιπροσωπεύεται ως ένα ιεραρχικά διευθετημένο σύνολο χαρακτηριστικών με διαφορετικές σχέσεις μεταξύ αυτών των χαρακτηριστικών. Συνεπώς κρίνεται απαραίτητο να δημιουργηθεί ένα εργαλείο που θα παίρνει τις πληροφορίες μίας γραμμής προϊόντων λογισμικού που αναπαρίστανται σε ένα μοντέλο χαρακτηριστικών (feature model) και θα δίνει στατιστικές που αφορούν την δυνατότητα συντήρησης της γραμμής προϊόντων λογισμικού. Τέλος η εφαρμογή αυτήν θα δίνει την δυνατότητα στο χρήστη να παράγει κώδικα λογισμικού βάση του συγκεκριμένου μοντέλου χαρακτηριστικών.

1.1 Σκοπός και στόχοι της εργασίας

Η χρήση των μοντέλων χαρακτηριστικών (feature models) στην διαδικασία ανάπτυξης λογισμικού έχει απασχολήσει ιδιαίτερα τον κλάδο τις πληροφορικής τα τελευταία χρόνια. Η οργάνωση τις παραγωγικής διαδικασίας λογισμικού σε μορφή παρόμοια με αυτήν άλλων προϊόντων που παράγονται σε γραμμές παραγωγής θεωρείται επιτακτική. Για το λόφο αυτό μεγάλες εταιρίες τείνουν να χρησιμοποιούν όλο και περισσότερο τα μοντέλα χαρακτηριστικών (feature models) με σκοπό να πετύχουν αυτό το επίπεδο οργάνωσης της παραγωγής. Σκοπός της παρούσας εργασίας είναι να εξετάσει την χρήση αυτών των μοντέλων χαρακτηριστικών στην διαδικασία παραγωγής λογισμικού και να αναλύσει τις βασικές τους έννοιες και ιδιότητες. Επίσης στην συγκεκριμένη διπλωματική εργασία αναλύεται ο τρόπος και οι παραδοχές που γίνονται για την μετατροπή ενός μοντέλου χαρακτηριστικών (feature model) σε διάγραμμα κλάσεων. Με αυτόν τον τρόπο παρέχονται πληροφορίες που έχουν ως σκοπό να ενώσουν την διαδικασία σχεδίασης λογισμικού με την διαδικασία παραγωγής λογισμικού.

Μαζί με την παραπάνω ανάλυση των μοντέλων χαρακτηριστικών θα δημιουργηθεί και το κατάλληλο εργαλείο όπου θα δίνει την δυνατότητα στον χρήστη να εισάγει το πρότυπο ενός δέντρου χαρακτηριστικών (feature model tree) και το πρόγραμμα θα είσαι σε θέση να αναγνωρίσει τις κλάσεις που θα πρέπει να δημιουργηθούν καθώς και να δίνει την επιλογή στον χρήστη να εισάγει τις διάφορές μεταβλητές τις κάθε κλάσης. Επίσης η εφαρμογή θα είναι σε θέση να εξάγει κώδικα σε Java όπου θα αντιπροσωπεύει το δέντρο χαρακτηριστικών όπου έδωσε ο χρήστης καθώς και τις μεταβλητές της κάθε κλάσης που όρισε καθώς και το αντίστοιχο διάγραμμα κλάσεων σε μορφή XML. Τέλος

το πρόγραμμα θα έχει την δυνατότητα να εξάγει μετρήσεις οι οποίες θα έχουν ως σκοπό να βοηθήσουν τους μηχανικούς λογισμικού να προβλέψουν το κατά πόσο είναι δυνατό να συντηρηθεί μία τέτοια γραμμή προϊόντων λογισμικού στο μέλλον.

1.2 Οργάνωση κειμένου

Στην παρούσα διπλωματική εργασία αναλύεται η χρήση των μοντέλων χαρακτηριστικών στις γραμμές παραγωγής λογισμικού. Η οργάνωση του κειμένου γίνεται σε ενότητες όπου σε κάθε ενότητα αναλύετε μία διαφορετική θεματική ενότητα της διπλωματικής εργασίας. Ποιο συγκεκριμένα:

- **Ενότητα 1:** Γίνεται μία σύντομη αναφορά στην εξέλιξη της παραγωγής λογισμικού και πως αυτό συγκρίνεται με μεθόδους άλλων κλάδων παραγωγής προϊόντων. Έπειτα παρέχονται γενικές πληροφορίες σχετικά με την παρούσα διπλωματική εργασία, λόγους και στόχους ανάπτυξης του λογισμικού που υλοποιήθηκε. Τέλος, παρέχει ένα γενικό θεωρητικό υπόβαθρό για τους δύο κύριους όρους της εργασίας, τα μοντέλα χαρακτηριστικών και τις γραμμές παραγωγής ώστε να εξοπλίσει των αναγνώστη με τα απαραίτητα εφόδια για την κατανόηση της εργασίας.

- **Ενότητα 2:** Γίνεται εκτενής ανάλυση των μοντέλων χαρακτηριστικών (feature models) καθώς και του όρου της γραμμής παραγωγής λογισμικού. Έπειτα αναλύονται τα βασικά στοιχεία και ιδιότητες των μοντέλων χαρακτηριστικών καθώς παρουσιάζονται και οι τρόποι αναπαράστασης. Επίσης, εξετάζονται οι βασικές αρχές και τρόπους με τους οποίους μπορούμε να μετατρέψουμε ένα μοντέλο χαρακτηριστικών σε διάγραμμα κλάσεων και στην συνέχεια σε κώδικα. Τέλος γίνεται αναφορά στις μετρήσεις που θα χρησιμοποιηθούν για να συμπεράνουμε εάν είναι εύκολο να συντηρηθεί η συγκεκριμένη γραμμή προϊόντων λογισμικού

- **Ενότητα 3:** Παρουσιάζει το σκοπό και τα κύρια χαρακτηριστικά της εφαρμογής καθώς και τα κύρια εργαλεία που θα χρησιμοποιηθούν για την ανάπτυξη της όπως και τον λόγο που επιλέχθηκαν. Γίνεται αναφορά σε προγράμματα που χρησιμοποιήθηκαν για την κατασκευή του συγκεκριμένου εργαλείου καθώς και προγράμματα που χρησιμοποιήθηκαν για την δημιουργία των μοντέλων χαρακτηριστικών.

- **Ενότητα 4:** Εδώ περιγράφεται εκτενέστερα η αρχιτεκτονική της εφαρμογής καθώς δίνεται έμφαση στο λειτουργικότητα της. Αρχικά γίνεται μία σύντομη περιγραφή της εφαρμογής και στην συνέχεια αναφέρεται ο τρόπος σχεδίασης της. Στην συνέχεια αναλύονται και εξηγούνται οι σημαντικότερες από τις λειτουργίες της εφαρμογής καθώς και παρουσιάζονται τα εκάστοτε κομμάτια κώδικα.

- **Ενότητα 5:** Στην παρούσα ενότητα γίνεται έλεγχος της λειτουργικότητας της εφαρμογής και παρουσιάζονται παραδείγματα εκτέλεσης. Αρχικά εξηγείται το παράδειγμα όπου θα εκτελεστεί και τους λόγους που επιλέχθηκε. Στην συνέχεια

παρουσιάζονται και εξηγούνται όλες οι ενέργειες που χρειάστηκε να κάνει ο χρήστης καθώς δίνονται και τα εκάστοτε στιγμιότυπα χρήσης της εφαρμογής. Τέλος δίνονται εξηγήσεις σχετικά με τις μετρήσεις που πήραμε από την εφαρμογή καθώς παρουσιάζεται και ο εξαγόμενος κώδικας.

- **Ενότητα 6:** Σε αυτό το κεφάλαιο γίνεται επισκόπηση των συμπερασμάτων που εξάχθηκαν κατά της διάρκεια της διπλωματικής εργασίας. Τέλος παρέχονται πληροφορίες και προτάσεις για την επέκταση του συγκεκριμένου εργαλείου λογισμικού στο μέλλον.

- **Ενότητα 7:** Στο κεφάλαιο αυτό γίνεται αναφορά όλων των βιογραφικών πηγών που χρησιμοποιήθηκαν στην εκπόνηση της παρούσας διπλωματικής εργασίας. Τέλος γίνεται αναφορά και στους ηλεκτρονικούς ιστότοπους που βοήθησαν στην ανάπτυξη της εφαρμογής.

2 Θεωρητικό Υπόβαθρο

2.1 Μοντέλα χαρακτηριστικών (Feature models)

Τα μοντέλα χαρακτηριστικών ή αλλιώς Feature Models, χρησιμοποιούνται ευρέως για τη διαχείριση της μεταβλητότητας, της οργάνωσης καθώς και της κοινής χρήσης στις σειρές προϊόντων λογισμικού (Software Product Lines) [1]. Τα μοντέλα χαρακτηριστικών είναι μοντέλα πληροφοριών όπου ένα σύνολο προϊόντων αντιπροσωπεύεται ως σύνολο χαρακτηριστικών σε ένα ενιαίο μοντέλο. Στην ανάπτυξη λογισμικού, ένα μοντέλο χαρακτηριστικών (feature model) είναι μια αναπαράσταση όλων των προϊόντων της σειράς προϊόντων λογισμικού (Software Product Line) με την έννοια των "χαρακτηριστικών". Τα μοντέλα χαρακτηριστικών αντιπροσωπεύονται οπτικά μέσω ενός διαγράμματος χαρακτηριστικών. Τα μοντέλα χαρακτηριστικών χρησιμοποιούνται ευρέως κατά τη διάρκεια ολόκληρης της διαδικασίας ανάπτυξης της γραμμής προϊόντων λογισμικού τόσο στην διαδικασία σχεδίασης και συντήρησης όσο και στην διαδικασία παραγωγής. Συνήθως χρησιμοποιούνται ως πρότυπα για την παραγωγή άλλων παραγωγικών στοιχείων, όπως έγγραφα καθορισμού χαρακτηριστικών, ορισμός αρχιτεκτονικής ή κομμάτια κώδικα. Τα μοντέλα χαρακτηριστικών προτάθηκαν για πρώτη φορά στη μέθοδο ανάλυσης με βάσει τα χαρακτηριστικά γνωρίσματα η αλλιώς Feature-Oriented Domain Analysis (FODA) από τον Kang το 1990 [11]. Έκτοτε, η χρήση τους για την μοντελοποίηση και την περιγραφή των χαρακτηριστικών ενός προγράμματος έχει υιοθετηθεί ευρέως από την κοινότητα της γραμμής προϊόντων λογισμικού και έχουν προταθεί διάφορες προσθετικές επεκτάσεις ως προς την υλοποίησή τους. Τέλος να σημειωθεί ότι η χρήση των μοντέλων χαρακτηριστικών μπορεί να υιοθετηθεί όχι μόνο από για την παραγωγική διαδικασία προγραμμάτων λογισμικού αλλά και κάθε άλλου είδους γραμμής παραγωγής καθώς αποτελεί έναν εύκολο τρόπο αναπαράστασης και οργάνωσης των χαρακτηριστικών ενός προϊόντος [1][4].

2.2 Γραμμές προϊόντων λογισμικού (Software product lines)

Η μαζική παραγωγή ορίζεται ως η παραγωγή προϊόντων που έχουν κοινά χαρακτηριστικά σε μεγάλες ποσότητες χρησιμοποιώντας τυποποιημένες διαδικασίες που παράγουν ένα μεγάλο όγκο του ίδιου προϊόντος σε μειωμένο χρόνο στην αγορά. Μετά τη βιομηχανική επανάσταση, μεγάλες εταιρείες άρχισαν να οργανώνουν την παραγωγική τους διαδικασία σε περιβάλλον μαζικής παραγωγής. Όταν αναφερόμαστε σε γραμμές προϊόντων λογισμικού (Software Product Lines) εννοούμε τις μεθόδους, τα εργαλεία και τις τεχνικές παραγωγής και σχεδίασης λογισμικού για τη δημιουργία μιας συλλογής συστημάτων λογισμικού που εμφανίζουν παρόμοια χαρακτηριστικά από ένα κοινό σύνολο λογισμικού που χρησιμοποιεί ένα κοινό μέσο παραγωγής. Στο χώρο της τεχνολογίας λογισμικού, η προσαρμογή των προϊόντων σε μαζικό επίπεδο είναι γνωστή ως γραμμές προϊόντων λογισμικού ή αλλιώς ως οικογένειες προϊόντων λογισμικού (Software Product Families) [1]. Για την επιτάχυνση της δημιουργίας λογισμικού καθώς και την διευκόλυνση της κληρονόμησης χαρακτηριστικών από ένα λογισμικό σε άλλο, η μηχανική των γραμμών προϊόντων λογισμικού προάγει την

δημιουργία μίας οικογένειας προϊόντων που σχετίζονται με λογισμικό και κατέχουν κοινά χαρακτηριστικά, αντί να παράγουμε ένα προς ένα λογισμικά ή κομμάτια λογισμικών που έχουν ίδια χαρακτηριστικά. Γενικότερα, ο όρος γραμμές προϊόντων λογισμικού αναφέρεται στην υιοθετήσει ενός συστήματος ομαδοποίησης λογισμικών που εμφανίζουν κοινά χαρακτηριστικά. Σκοπός αυτής την ομαδοποίησης είναι η δημιουργία «οικογενειών» λογισμικού με σκοπό την επιτάχυνση της παραγωγικής διαδικασίας καθώς και την επέκταση της επαναχρησιμοποίησης κώδικα μεταξύ εφαρμογών της ίδιας οικογένειας. Ως παράδειγμα για την ευκολότερη κατανόηση του συγκεκριμένου όρου μπορεί να δοθεί η ανάγκη δημιουργίας λογισμικού για διαφορετικές συσκευές κινητής τηλεφωνίας από μία εταιρία κατασκευής κινητών τηλεφώνων. Η ίδια εταιρία έχει την δυνατότητα να παράγει διαφορετικά μοντέλα συσκευών με διαφορετικές συνθέσεις τόσο στο υλικό όσο και στο λογισμικό τους. Παρόλα αυτά κάποιες ιδιότητες μεταξύ των διαφορετικών μοντέλων παραμένουν ίδιες όπως είναι το βασικό μενού πλοήγησης. Όλες οι διαφορετικές εκδώσεις του λογισμικού για τα κινητά της ίδιας εταιρίας ανήκουν στην ίδια οικογένεια λογισμικού οπότε μπορούν να μοιραστούν τα ίδια χαρακτηριστικά όπου ένα από αυτά είναι και το μενού πλοήγησης. Οπότε δεν χρειάζεται η αναπαραγωγή του χαρακτηριστικών λογισμικού που μοιράζονται μεταξύ διαφορετικών συσκευών.

2.3 Μαζική παραγωγή και μαζική εξατομίκευση

Για την καλύτερη κατανόηση των μοντέλων χαρακτηριστικών (feature models) καθώς και της χρήσης τους στις γραμμές προϊόντων λογισμικού (Software Product Line), πρέπει να εξετάσουμε αρχικά πως λειτουργεί μία επιχείρηση με σκοπό να παράγει μεγάλο κέρδος με μικρό κόστος. Μετά την βιομηχανική επανάσταση οι μεγάλοι οργανισμοί άρχισαν να οργανώνουν την παραγωγή τους σε περιβάλλοντα μαζικής παραγωγής. Ως μαζική παραγωγή ορίζεται η παραγωγή προϊόντων με κοινά χαρακτηριστικά με την χρήση των ίδιων μεθόδων και εργαλείων με σκοπό την μείωση του χρόνου που το προϊόν χιάζεται για να βγει στην αγορά. Παρόλα αυτά, λόγου του μεγάλου ανταγωνισμού που υπάρχει στην αγορά η μαζική παραγωγή δεν είναι αρκετή και οι οργανισμοί στρέφονται στην μαζική εξατομίκευση προκειμένου να έχουν μεγαλύτερη επιτυχία στις αγορές. Η μαζική εξατομίκευση στοχεύει στην εξασφάλιση όσων των δυνατών περισσότερων αναγκών του αγοραστικού κοινού. Όμως, αυτό πρέπει να γίνει διατηρώντας όσο των δυνατών περισσότερο την δυνατότητα της μαζικής παραγωγής. Για να επιτευχθεί αυτό θα πρέπει να δημιουργηθούν διαφορετικά προϊόντα από υπάρχοντα αντικείμενα ώστε να αυξηθεί η κληρονομικότητα κώδικα όταν μιλάμε για προϊόντα λογισμικού και τα προϊόντα να μοιράζονται περισσότερες ομοιότητες παρά διαφορές.

2.3.1 Μαζική παραγωγή και εξατομίκευση σε προϊόντα λογισμικού

Στην τεχνολογία λογισμικού η μαζική εξατομίκευση λογισμικού είναι γνωστή ως γραμμές προϊόντων λογισμικού (Software Product Lines). Με σκοπό την εξασφάλιση της εξατομίκευσης σε μία γραμμή προϊόντων λογισμικού προωθείτε η χρήση οικογενειών λογισμικού. Μία οικογένεια λογισμικού

αντιπροσωπεύει προϊόντα λογισμικού που μοιράζονται περισσότερα κοινά στοιχεία παρά διαφορές [1]. Σκοπός αυτού είναι να μην απαιτείτε η δημιουργία κώδικα που έχει παραχθεί είδη στο παρελθών αλλά η κληρονομικότητα του μεταξύ των προϊόντων της ίδιας οικογένειας. Για αυτό τον λόγο οι γραμμές προϊόντων λογισμικού έχουν υιοθετηθεί από πολλούς οργανισμούς παραγωγής λογισμικού σε διάφορους τομείς όπως είναι τα ενσωματωμένα συστήματα για την παραγωγή κινητών τηλεφώνων, ενσωματωμένες εφαρμογές αυτοκινήτων και βιοιατρικές συσκευές. Παρόλα αυτά η υιοθετήσει αυτών των μεθόδων σε εφαρμογές υπολογιστών γραφείου και διαδικτύου αποτελεί ακόμα πρόκληση.

2.4 Ανάλυση και περιγραφή μοντέλων χαρακτηριστικών (Feature models)

Ένα μοντέλο χαρακτηριστικών είναι μία αναπαράσταση πληροφοριών σχετικά με ένα προϊών, στην δική μας περίπτωση ένα προϊόν λογισμικού, όπου αναπαριστά όλα τα δυνατά προϊόντα μίας γραμμής προϊόντων λογισμικού με όρους χαρακτηριστικών και συσχετίσεις μεταξύ τους. Τα μοντέλα χαρακτηριστικών αναπαρίστανται σε ιεραρχικά μοντέλα χαρακτηριστικών που απαρτίζονται από “σχέσεις” μεταξύ του γονέα και των παιδιών και διατμηματικές “σχέσεις” ανάμεσα σε διαφορετικά τμήματα ή αλλιώς κλαδιά του δέντρου. Οι δεύτερες “σχέσεις” συνήθως απαρτίζονται από εκφράσεις συμπερίληψης η αποκλεισμού όπως θα δούμε και στην συνέχεια. Τα μοντέλα χαρακτηριστικών χρησιμοποιούνται σε διαφορετικές φάσης της παραγωγής λογισμικού όπως από την ανάπτυξη του λογισμικού με την χρήση μοντέλου, των προγραμματισμό προσανατολισμένο στα χαρακτηριστικά καθώς και σε μεγάλες παραγωγικές διαδικασίες ή και στον γενικό προγραμματισμό.

2.4.1 Βασικά μοντέλα χαρακτηριστικών (Basic Feature models)

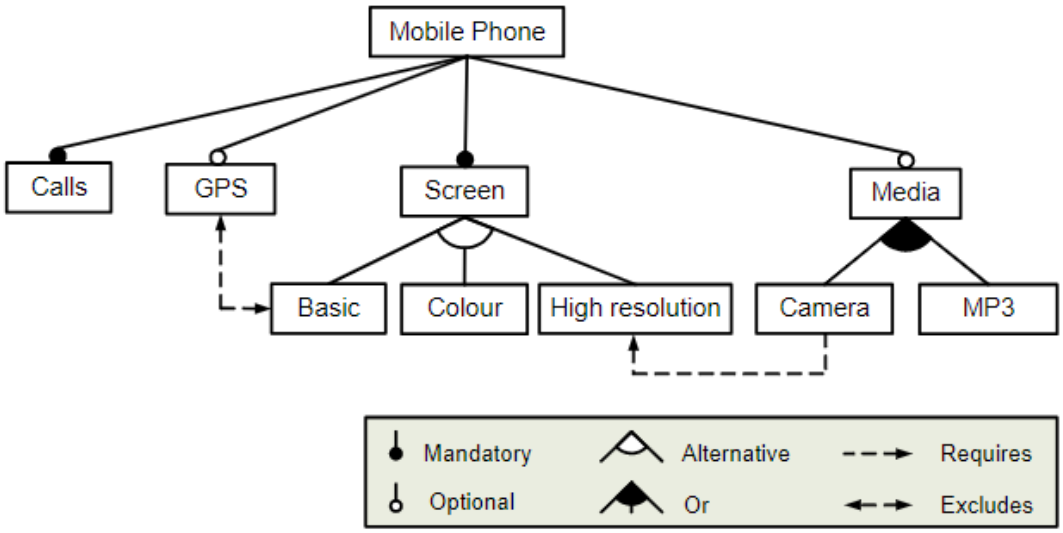
Καλούμε βασικά μοντέλα χαρακτηριστικών αυτά που επιτρέπουν τις παρακάτω σχέσεις μεταξύ των διαφόρων χαρακτηριστικών του δέντρου [1][4][6].

- **Υποχρεωτική (Mandatory):** Ένα χαρακτηριστικό παιδιού θεωρείται ότι έχει επιτακτική σχέση με το χαρακτηριστικό γονέα του όταν το συγκεκριμένο χαρακτηριστικό εμφανίζεται σε όλα τα προϊόντα όπου υπάρχει και το χαρακτηριστικό γονέα.
- **Προαιρετική (Optional):** Ένα χαρακτηριστικό παιδιού θεωρείται ότι έχει προαιρετική σχέση με το χαρακτηριστικό του γονέα όταν το χαρακτηριστικό μπορεί να εμφανίζεται προαιρετικά σε όλα τα προϊόντα που εμφανίζεται και ο γονέας του. Δηλαδή, μπορεί να υπάρξουν προϊόντα όπου υπάρχει το χαρακτηριστικό γονέα αλλά το συγκεκριμένο προαιρετικό χαρακτηριστικό παιδιού απουσιάζει.
- **Εναλλακτική (Alternative):** Μία ομάδα χαρακτηριστικών παιδιών θεωρείται ότι έχουν εναλλακτική σχέση με το χαρακτηριστικό του γονέα όταν μόνο ένα χαρακτηριστικό παιδιού μπορεί να επιλεχθεί όταν ο γονέας βρίσκεται σε κάποιο προϊόν.
- **Ή (Or):** Μία ομάδα χαρακτηριστικών παιδιών θεωρείται ότι έχουν σχέση or με το χαρακτηριστικό του γονέα όταν ένα ή και περισσότερα

παιδιά μπορούν να επιλεχθούν όταν ο γονέας βρίσκεται σε κάποιο προϊόν.

Επίσης είναι εύκολο να διεξαχθεί το συμπέρασμα ότι προκειμένου να έχουμε κάποιο χαρακτηριστικό παιδιού σε ένα από τα προϊόντα, θα πρέπει να συμπεριλαμβάνεται και ο κόμβος γονέα αυτού. Στην συνέχεια μπορούμε να διακρίνουμε δύο δια τμηματικές σχέσης μεταξύ διαφορετικών κλαδιών του δέντρου χαρακτηριστικών. Αυτές οι σχέσεις χωρίζονται σε:

- **Απαιτείται (Required):** Η σχέση αυτή απαιτεί την συμπερίληψη ενός άλλου χαρακτηριστικού Β σε περίπτωση που έχουμε στο προϊόν μας το χαρακτηριστικό Α.
- **Εξαιρείται (Excludes):** Η σχέση αποκλεισμού δηλώνει ότι σε περίπτωση που έχουμε το χαρακτηριστικό Α στο προϊόν μας δεν μπορούμε να έχουμε το χαρακτηριστικό Β.



Εικόνα 1: Παράδειγμα μοντέλου χαρακτηριστικών ενός απλού κινητού τηλεφώνου με χρήση όλων των ιδιοτήτων

2.4.2 Μοντέλα χαρακτηριστικών βασισμένα στην πολλαπλότητα (Cardinality–based feature models)

Για πρακτικούς και εννοιολογικής πληρότητας λόγους προτάθηκε η επέκταση των FODA μοντέλων χαρακτηριστικών με χρήση UML πολλαπλότητας [1]. Οι δύο νέες σχέσεις που προτάθηκαν είναι οι εξής:

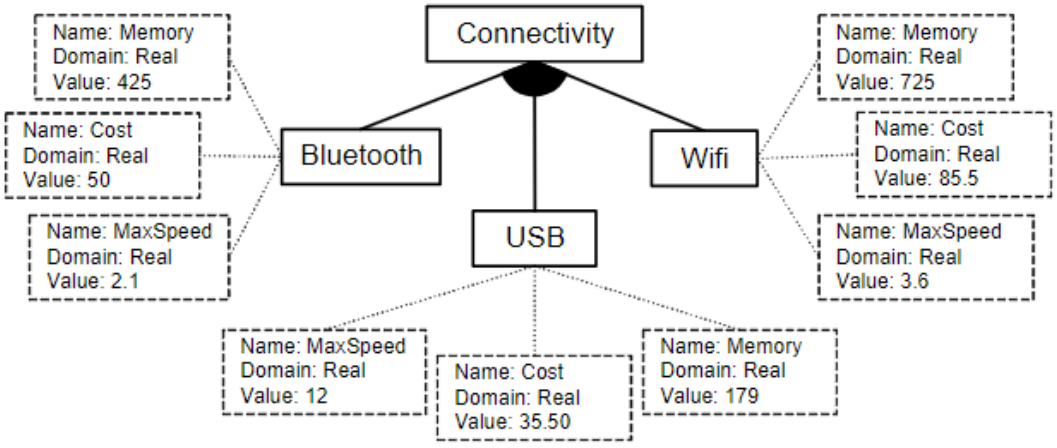
- **Feature cardinality:** Είναι μία ακολουθία διαστημάτων που συμβολίζονται με [n..m]. Ως n θεωρούμε το κάτω όριο και ως m το ανώτερο όριο. Αυτές οι μεταβλητές καθορίζουν τον αριθμό των περιπτώσεων του χαρακτηριστικού που μπορεί να είναι μέρος ενός συγκεκριμένου προϊόντος του δέντρου χαρακτηριστικών. Η σχέση αυτή έχει την δυνατότητα να χρησιμοποιηθεί ως γενίκευση για τις σχέσεις

υποχρεωτικού χαρακτηριστικού (mandatory) ([1,1]) και προαιρετικού χαρακτηριστικού (optional) ([0,1]).

- **Group cardinality:** Είναι μία ακολουθία διαστημάτων που συμβολίζεται με <n..m>. Ως n θεωρούμε το κάτω όριο και ως m το ανώτερο όριο. Αυτές οι μεταβλητές καθορίζουν τον αριθμό των χαρακτηριστικών παιδιού που μπορούν να βρίσκονται σε ένα προϊόν από κάποιον γονέα. Η σχέση αυτή είναι ισοδύναμη με την ενναλακτική σχέση (alternative) (<1..1>) και της or σχέσης (<1..N>).

2.4.3 Εκτεταμένα μοντέλα χαρακτηριστικών (Extended feature models)

Τα χαρακτηριστικά από μόνα τους δεν μας δίνουν πολλές πληροφορίες σχετικά με τις ιδιότητες του εκάστοτε χαρακτηριστικού. Για τον λόγο αυτό η επέκταση του θεωρείται αναγκαία με σκοπό να περιέχουν παραπάνω πληροφορίες. Αυτές οι πληροφορίες που περιγράφουν το εκάστοτε χαρακτηριστικό ονομάζονται γνωρίσματα. Το είδος των μοντέλων που περιέχουν γνωρίσματα ονομάζεται εκτεταμένο, προχωρημένο ή χαρακτηριστικό μοντέλο γνωρισμάτων. Στην παρακάτω εικόνα μπορούμε να δούμε ένα εκτεταμένο δέντρο χαρακτηριστικών όπου ως γνώρισμα για κάθε χαρακτηριστικό ορίζετε το όνομα, ο τομέας της μεταβλητής (ακέραια, δεκαδική, κτλπ) καθώς και η τιμή της.



Εικόνα 2: Παράδειγμα εκτεταμένου μοντέλου χαρακτηριστικών

2.4.4 Λειτουργίες ανάλυσης σε μοντέλα χαρακτηριστικών

Σε αυτό το κομμάτι αναλύονται οι διαφορετικές λειτουργίες ανάλυσης όπου προτάθηκαν. Για κάθε λειτουργία θα αναφέρεται και μία πιθανή περίπτωση πρακτικής εφαρμογής της εκάστοτε λειτουργίας.

2.4.4.1 Κενό μοντέλο χαρακτηριστικών (Void feature model)

Αυτήν η λειτουργία δέχεται ένα δέντρο χαρακτηριστικών ως είσοδο και επιστρέφει την πληροφορία αν το μοντέλο είναι κενό ή όχι. Ως κενό

μοντέλο ορίζεται το δέντρο χαρακτηριστικών που δεν παριστάνει κανένα προϊόν. Ο λόγος που μπορεί να συμβεί αυτό είναι η λανθασμένη χρήση των διατμηματικών περιορισμών του δέντρου. Η αυτοματοποίηση της συγκεκριμένης διαδικασίας μπορεί να βοηθήσει στην αποσφαλμάτοση μεγάλων δέντρων όταν για να γίνει αυτήν η διαδικασία χειροκίνητα απαιτείτε αρκετός χρόνος [1].

2.4.4.2 *Έγκυρο προϊόν (Valid product)*

Η συγκεκριμένη διαδικασία παίρνει ένα δέντρο χαρακτηριστικών και ένα προϊόν και επιστρέφει εάν το προϊόν ανήκει ή όχι στο συγκεκριμένο δέντρο. Αυτήν η λειτουργία μπορεί να βοηθήσει του SPL αναλυτές και διευθυντές να αποφασίσουν αν το δοθέν προϊόν μπορεί να υποστηριχτεί από την παρούσα γραμμή παραγωγής λογισμικού [1].

2.4.4.3 *Μερικός έγκυρη διαμόρφωση (Valid partial configuration)*

Η συγκεκριμένη λειτουργία παίρνει ένα δέντρο χαρακτηριστικών μαζί με μία μερική διαμόρφωση (partial configuration) ως είσοδο και επιστρέφει εάν η διαμόρφωση είναι έγκυρη η όχι. Αυτήν η λειτουργία μπορεί να φανεί ιδιαίτερα χρήσιμη στην διαδικασία παραγωγής καθώς μπορεί να δώσει στον χρήστη μία ιδέα για την εξέλιξη της διαμόρφωσης του προϊόντος [1].

2.4.4.4 *Εμφάνιση όλων των προϊόντων (All products)*

Σε αυτήν την διαδικασία ένα δέντρο χαρακτηριστικών δίνεται ως είσοδος. Ως έξοδος παράγονται όλα τα δυνατά προϊόντα που μπορούν να δημιουργηθούν από την συγκεκριμένη είσοδο. Αυτήν η διαδικασία είναι χρήσιμη για την αναγνώριση νέων συνδυασμών που μπορεί να μην δημιουργήθηκαν στο αρχικό σχέδιο της γραμμής παραγωγής [1].

2.4.4.5 *Αριθμός προϊόντων (Number of products)*

Αυτήν η διαδικασία επιστρέφει των αριθμό που προϊόντων που μπορούν να παραχθούν από το μοντέλο χαρακτηριστικών που δέχτηκε ως είσοδο. Αυτήν η μέθοδος εμφανίζει πόσο περίπλοκη αλλά και ελαστική μπορεί να είναι η SPL καθώς όσα περισσότερα προϊόντα τόσο μεγαλύτερη ελαστικότητα αλλά τόσο μεγαλύτερη πολυπλοκότητα [1].

2.4.4.6 *Φίλτρο Προϊόντων (Filter)*

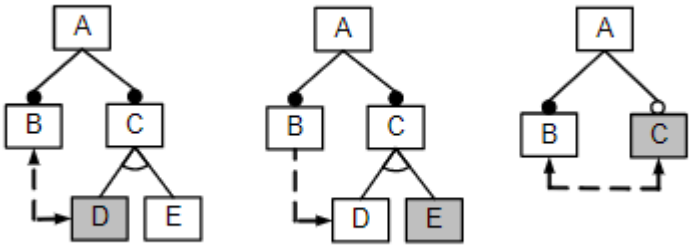
Η λειτουργία αυτή δέχεται ένα δέντρο χαρακτηριστικών καθώς και μία διαμόρφωση ή αλλιώς σύνθεση προϊόντος. Η λειτουργία επιστρέφει μία ομάδα προϊόντων συμπεριλαμβανομένου του δοθέντος που ικανοποιούν την παρούσα σύνθεση συνήθως μερικός. Η λειτουργία του φίλτρου προϊόντων μπορεί να βοηθήσει στην εξαγωγή των προϊόντων δοθέντων κάποιον συγκεκριμένων χαρακτηριστικών [1].

2.4.4.7 *Ανίχνευσή ανωμαλιών (Anomalies detection)*

Παρακάτω αναφέρονται οι ποιο γνωστές ανωμαλίες και λάθη που μπορεί να προκύψουν σε κάποιο μοντέλο χαρακτηριστικών [1].

2.4.4.8 *"Νεκρά" χαρακτηριστικά (Dead features)*

Ένα χαρακτηριστικό θεωρείτε νεκρό αν δεν μπορεί να εμφανιστεί σε κανένα προϊόν της SPL. Τα "νεκρά" χαρακτηριστικά προκαλούνται από την λάθος χρήση διατμηματικών περιορισμών μεταξύ διαφορετικών κλαδιών του δέντρου. Παρακάτω ακολουθούν τρία παραδείγματα. Τα χαρακτηριστικά με γκρι χρώμα είναι "νεκρά" [1].



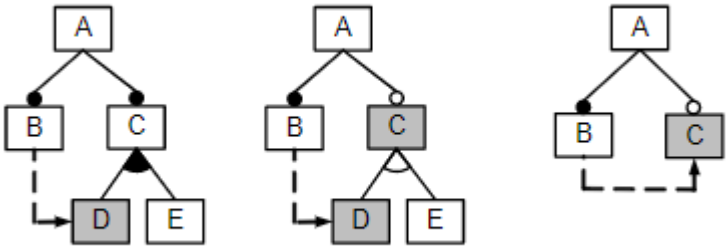
Εικόνα 3: Παράδειγμα "νεκρού" χαρακτηριστικού (γκρι χρώμα)

2.4.4.9 *Υπό όρους "νεκρά" χαρακτηριστικά (Conditionally dead features)*

Είναι χαρακτηριστικά όπου θεωρούνται "νεκρά" υπό συνθήκες. Για παράδειγμα εάν επιλεχτεί κάποιο άλλο χαρακτηριστικό του δέντρου που έχει κάποιον περιορισμό με το χαρακτηριστικό που θεωρούμε υπό όρους "νεκρό" [1] .

2.4.4.10 *Λάθος προαιρετικά χαρακτηριστικά (False optional features)*

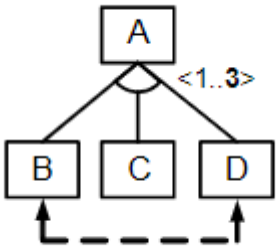
Ένα χαρακτηριστικό είναι λάθος προαιρετικό αν εμπεριέχεται σε όλα τα δυνατά προϊόντα του μοντέλου χαρακτηριστικών. Παρακάτω ακολουθούν τρία παραδείγματα όπου τα γκρι χαρακτηριστικά είναι κατηγοριοποιημένα λάθος ως προαιρετικά [1].



Εικόνα 4: Παράδειγμα λάθους προαιρετικού χαρακτηριστικού

2.4.4.11 Wrong cardinalities

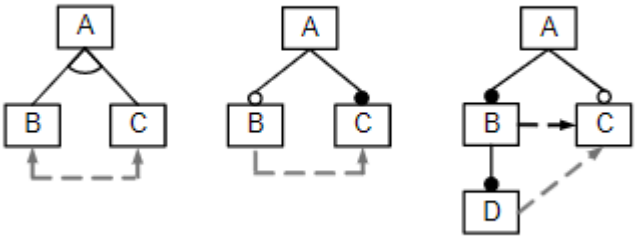
Θεωρείται μία ομάδα χαρακτηριστικών όταν δεν μπορεί να ικανοποιήσει το διάστημα <n..m>. Αυτό εμφανίζεται συνήθως σε μοντέλα σε cardinality–based feature models όπου κάποιος διατμηματικός περιορισμός εμπλέκεται όπως για παράδειγμα στην παρακάτω εικόνα [1].



Εικόνα 5: Παράδειγμα ομάδας χαρακτηριστικών που δεν μπορεί να ικανοποιηθεί

2.4.4.12 Redundancies

Ένα μοντέλο χαρακτηριστικών περιέχει ίδιες πληροφορίες σε πολλαπλούς τρόπους. Αυτό μπορεί να έχει αρνητική επίπτωση στην συντήρηση του μοντέλου. Στο παρακάτω παράδειγμα οι γκρι περιορισμοί είναι μη απαραίτητοι [1].



Εικόνα 6: Παράδειγμα μη απαραίτητων περιορισμών

2.4.4.13 Επεξηγήσεις (Explanations)

Αυτήν η λειτουργία δέχεται ως είσοδο ένα χαρακτηριστικό μοντέλο και μία ανάλυση λειτουργίας. Ως έξοδο επιστρέφει τους λόγους κάποιου προβλήματος που οφείλονται κυρίως στις σχέσης των χαρακτηριστικών και εξηγήσεις που συνήθως σχετίζονται με διάφορες ανωμαλίες. Οι επεξηγήσεις είναι μία δύσκολη λειτουργία και θεωρείται ως η αποσφαλμάτοση (debugging) των μοντέλων χαρακτηριστικών [1].

2.4.4.14 Διορθωτικές εξηγήσεις (Corrective explanations)

Η λειτουργία αυτή παίρνει ένα μοντέλο χαρακτηριστικών ως είσοδο και μία ανάλυση λειτουργίας. Ως έξοδο παράγει διάφορες βελτιωτικές

προτάσεις και διορθώσεις που πρέπει να γίνουν στο μοντέλο της αρχικής εισόδου [1].

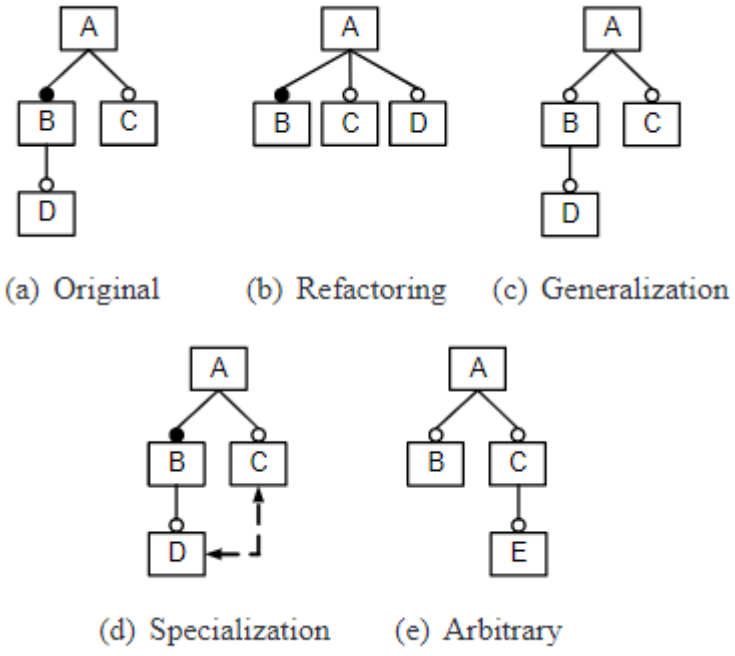
2.4.4.15 Σχέσεις μοντέλων χαρακτηριστικών (Feature model relations)

Αυτές οι λειτουργίες παίρνουν δύο διαφορετικά μοντέλα και εξετάζουν πόσο αυτά τα μοντέλα είναι κοινά μεταξύ τους. Αυτήν η διαδικασία είναι χρήσιμη για να πάρουμε πληροφορίες σχετικά με το πώς εξελίχθηκαν τα μοντέλα σε βάθος χρόνου [1].

Οι πιθανές σχέσεις μεταξύ δύο μοντέλων είναι οι ακόλουθες:

- **Επανεμφάνιση (Refactoring):** Ένα μοντέλο χαρακτηριστικών επανεμφανίζει ένα άλλο μοντέλο όταν και τα δύο μοντέλα αναπαριστούν τα ίδια προϊόντα αλλά έχουν διαφορετική δομή.
- **Γενίκευση (Generalization):** Ένα μοντέλο χαρακτηριστικών είναι γενίκευση ενός άλλου μοντέλου χαρακτηριστικών εάν το πρώτο περιέχει και επεκτείνει το σύνολο προϊόντων του δευτέρου.
- **Ειδίκευση (Specialization):** Ένα μοντέλο χαρακτηριστικών είναι ειδίκευση ενός άλλου μοντέλου χαρακτηριστικών εάν το πρώτο είναι υποσύνολο του δευτέρου.
- **Αυθεντική επεξεργασία (Arbitrary edit):** Δεν υπάρχει σαφής σχέση μεταξύ των δύο μοντέλων.

Παρακάτω φαίνονται σχηματικά οι παραπάνω σχέσεις.



Εικόνα 7: Σχέσεις μεταξύ μοντέλων

2.4.4.16 Βελτιστοποίηση (Optimization)

Αυτήν η λειτουργία δέχεται ένα μοντέλο χαρακτηριστικών ως είσοδο και μία λειτουργία. Ως έξοδο επιστρέφει το προϊόν που τηρεί τα κριτήρια που θέτει η λειτουργία. Αυτήν η λειτουργία είναι ιδιαίτερα χρήσιμη όταν έχουμε να κάνουμε με εκτεταμένα μοντέλα χαρακτηριστικών που περιέχουν γνωρίσματα [1].

2.4.4.17 Βασικά χαρακτηριστικά (Core features)

Αυτήν η διαδικασία παίρνει ως είσοδο ένα μοντέλο χαρακτηριστικών. Επιστρέφει ένα σύνολο από χαρακτηριστικά που είναι μέλη σε όλα τα προϊόντα της γραμμής παραγωγής λογισμικού. Τα βασικά χαρακτηριστικά είναι τα ποιο συνηθισμένα χαρακτηριστικά μίας SPL καθώς εμφανίζονται συνέχεια [1].

2.4.4.18 Χαρακτηριστικά παραλλαγής (Variant features)

Αυτήν η διαδικασία δέχεται ένα μοντέλο και επιστρέφει ένα σύνολο χαρακτηριστικών που δεν εμφανίζονται σε όλα τα προϊόντα της SPL [1].

2.4.4.19 Ατομικά σετ (Atomic sets)

Η λειτουργία αυτή παίρνει ως είσοδο ένα μοντέλο χαρακτηριστικών. Επιστρέφει ένα σύνολο από ομάδες χαρακτηριστικών που μπορούν να χρησιμοποιηθούν σαν μονάδα σε συγκεκριμένες αναλύσεις. Όταν αυτήν η λειτουργία τα σύνολα που εξάχθηκαν μπορούν να χρησιμοποιηθούν για την δημιουργία ποιο απλών μοντέλων [1].

2.4.4.20 Ανάλυση εξαρτήσεων (Dependency analysis)

Αυτήν η λειτουργία δέχεται ως είσοδο ένα μοντέλο χαρακτηριστικών και μία μερική διαμόρφωση. Επιστρέφει μία νέα διαμόρφωση με τα χαρακτηριστικά που θα έπρεπε να επιλεχτούν και να διαγραφούν από το μοντέλο [1].

2.4.4.21 Διαμόρφωση πολλαπλών βημάτων (Multi–step configuration)

Ένα πρόβλημα διαμόρφωσης πολλαπλών βημάτων ορίζεται ως η διαδικασία παραγωγής μίας σειράς ενδιάμεσων διαμορφώσεων. Η διαδικασία δέχεται ως είσοδο ένα μοντέλο χαρακτηριστικών, μία αρχική διάταξη, μια επιθυμητή κατάσταση, έναν αριθμό βημάτων, έναν καθολικό περιορισμό που δεν μπορεί να παραβιαστεί και μία συνάρτηση κόστους. Ως αποτέλεσμα παράγει την λίστα με τα βήματα που πρέπει να ακολουθηθούν για να πάμε από τον αρχικό στον τελικό στόχο [1].

2.4.4.22 Ομοιογένεια (Homogeneity)

Δέχεται ένα μοντέλο χαρακτηριστικών και παράγει μία ένδειξη κατά πόσο το μοντέλο είναι ομοιογενές [1]. Η ομοιογένεια υπολογίζεται με τον παρακάτω τύπο

$$Homogeneity = 1 - \frac{\#uf}{\#products}$$

Το #uf είναι ο αριθμός των μοναδικών χαρακτηριστικών σε ένα προϊόν και ως #products ορίζεται ο συνολικός αριθμός των προϊόντων που παράγονται από το μοντέλο χαρακτηριστικών [1].

2.4.4.23 *Ομοιότητα (Commonality)*

Αυτήν η λειτουργία δέχεται ένα μοντέλο χαρακτηριστικών και μία σύνθεση ως είσοδο και παράγει το ποσοστό των προϊόντων που παράγονται βάση της συγκεκριμένης σύνθεσης [1]. Ο τύπος είναι: Ομοιότητα = σύνθεση / σύνολο παραγόμενων προϊόντων.

2.4.4.24 *Παράγοντας μεταβλητότητας (Variability factor)*

Αυτήν η λειτουργία δέχεται ένα μοντέλο χαρακτηριστικών και παράγει την αναλογία ανάμεσα στα προϊόντα και στο 2^n όπου n ο αριθμός των χαρακτηριστικών που περιέχει το μοντέλο. Υπολογίζεται από τον τύπο N.Products / 2^n [1].

2.4.4.25 *Βαθμός ορθογωνικότητας (Degree of orthogonality)*

Αυτήν η διαδικασία δέχεται ένα μοντέλο χαρακτηριστικών και ένα υπό δέντρο. Ως έξοδο παράγει το βαθμό ορθογωνικότητας. Ως βαθμός ορθογωνικότητας ορίζεται την αναλογία μεταξύ τον συνολικό αριθμό προϊόντων και τον αριθμό προϊόντων του υποδέντρου. Υπολογίζεται από τον τύπο Orthogonality = N.Products / N.ofTreeProducts. Ένας μεγάλος βαθμός ορθογωνικότητας υποδεικνύει ότι η αποφάσεις μπορούν να ληφθούν τοπικά χωρίς να μας ενδιαφέρει για τις επιρροές άλλων κλαδιών [1].

2.4.4.26 *Extra Constraint Representativeness (ECR)*

Αυτήν η λειτουργία παίρνει ως είσοδο ένα μοντέλο χαρακτηριστικών και επιστρέφει τον βαθμό αντιπροσωπευτικότητας των διατμηματικών περιορισμών του δέντρου. Ως ECR ορίζεται η αναλογία του αριθμού των χαρακτηριστικών που εμπλέκονται σε μία διατμηματική σχέση προς τον αριθμό των χαρακτηριστικών στο δέντρο[1]. Υπολογίζεται με τον τύπο ECR = N.ofFeaturesOfCrossTreeCon / N.ofFeaturesInTree [1].

2.4.4.27 *Lowest Common Ancestor (LCA)*

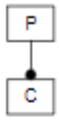
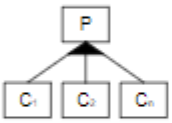
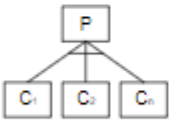
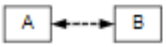
Αυτήν η διαδικασία δέχεται ένα μοντέλο χαρακτηριστικών και ένα σύνολο από χαρακτηριστικά σαν είσοδο. Ως έξοδο παράγει ένα χαρακτηριστικό που είναι ο λιγότερο πιθανός πρόγονος της εισόδου. Ως LCA ορίζεται ο κοινός πρόγονος όπου βρίσκεται ποιο μακριά από την ρίζα του δέντρου [1].

2.4.4.28 Χαρακτηριστικά ρίζας (Root features)

Αυτήν η λειτουργία έχει ως είσοδο ένα μοντέλο χαρακτηριστικών και ένα σύνολο από χαρακτηριστικά. Ως έξοδο παράγει ένα σύνολο χαρακτηριστικών που είναι οι ρίζες στο μοντέλο [1].

2.4.5 Λογική βασισμένη σε προτασιακή λογική

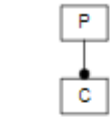
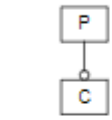
Μία προτασιακή φόρμουλα αποτελείται από ένα σύνολο συμβόλων ή μεταβλητών και από λογικές συνδέσεις που περιορίζουν τις τιμές των μεταβλητών. Τα παρακάτω αποτελούν λογικά σύμβολα της προτασιακής λογικής: ¬,Λ,V,⇒,⇔ όπου το ¬ είναι η άρνηση, Λ είναι η σύζευξη, V είναι η διάζευξη, ⇒ είναι η συνεπαγωγή και ⇔ είναι η ισοδυναμία. Στον παρακάτω πίνακα παρουσιάζονται οι μετατροπές των σχέσεων στα μοντέλα χαρακτηριστικών σε εκφράσεις προτασιακής λογικής [1].

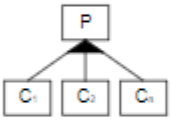
Relationship		PL Mapping
MANDATORY		$P \leftrightarrow C$
OPTIONAL		$C \rightarrow P$
OR		$P \leftrightarrow (C_1 \vee C_2 \vee \ldots \vee C_n)$
ALTERNATIVE		$(C_1 \leftrightarrow (\neg C_2 \wedge \ldots \wedge \neg C_n \wedge P)) \wedge$ $(C_2 \leftrightarrow (\neg C_1 \wedge \ldots \wedge \neg C_n \wedge P)) \wedge$ $(C_n \leftrightarrow (\neg C_1 \wedge \neg C_2 \wedge \ldots \wedge \neg C_{n-1} \wedge P))$
IMPLIES		$A \rightarrow B$
EXCLUDES		$\neg(A \wedge B)$

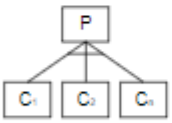
Εικόνα 8: Μετατροπή σχέσεων μοντέλων χαρακτηριστικών σε προτασιακή λογική

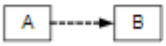
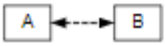
2.4.6 Ανάλυση βασισμένη στον προγραμματισμό περιορισμών

Ένα πρόβλημα περιορισμού ικανοτήτων (Constraint Satisfaction Problem CSP) [1] συνιστάται από ένα σύνολο μεταβλητών , ένα σύνολο πεπερασμένων περιοχών για αυτές τις μεταβλητές και μία ομάδα περιορισμών των μεταβλητών. Ο προγραμματισμός περιορισμών μπορεί να οριστεί να οριστεί ως σύνολο τεχνικών όπως αλγόριθμοι ή heuristics. Ένα CSP επιλύεται βρίσκοντας τις τιμές των μεταβλητών όπου όλοι οι περιορισμοί ικανοποιούνται. Η χαρτογράφηση ενός μοντέλου χαρακτηριστικών σε CSP μπορεί να ποικίλει ανάλογα με τον συγκεκριμένο πρόγραμμα που χρησιμοποιείτε στην συνέχεια για ανάλυση. Ο παρακάτω πίνακας αποτελεί χαρτογράφηση των σχέσεων από το μοντέλο χαρακτηριστικών σε CSP.

Relationship		CSP Mapping
MANDATORY		P = C
OPTIONAL		if (P = 0) C = 0

OR		if (P > 0)
		Sum(C1,C2,...Cn) in {1..n}
		else
		C1= 0, C2=0,…., Cn=0

ALTERNATIVE		if (P > 0)
		Sum(C1,C2,...Cn) in {1..1}
		else
		C1= 0, C2=0,…., Cn=0

REQUIRES		if (A > 0) B>0
EXCLUDES		if (A > 0) B=0

Εικόνα 9: Μετατροπή σχέσεων μοντέλων χαρακτηριστικών σε ψευδοκώδικα

2.5 Βασική ιδέα τις μετατροπής μοντέλων χαρακτηριστικών σε άλλα μοντέλα

Μια οικογένεια μοντέλων αντιπροσωπεύεται από ένα μοντέλο χαρακτηριστικών και ένα πρότυπο μοντέλο. Το μοντέλο χαρακτηριστικών καθορίζει μία ιεραρχία μεταξύ των χαρακτηριστικών. Επίσης περιέχουν περιορισμούς ανάμεσα στα χαρακτηριστικά που καθορίζουν τα διάφορα πιθανά προϊόντα που μπορεί να παραχθούν από το

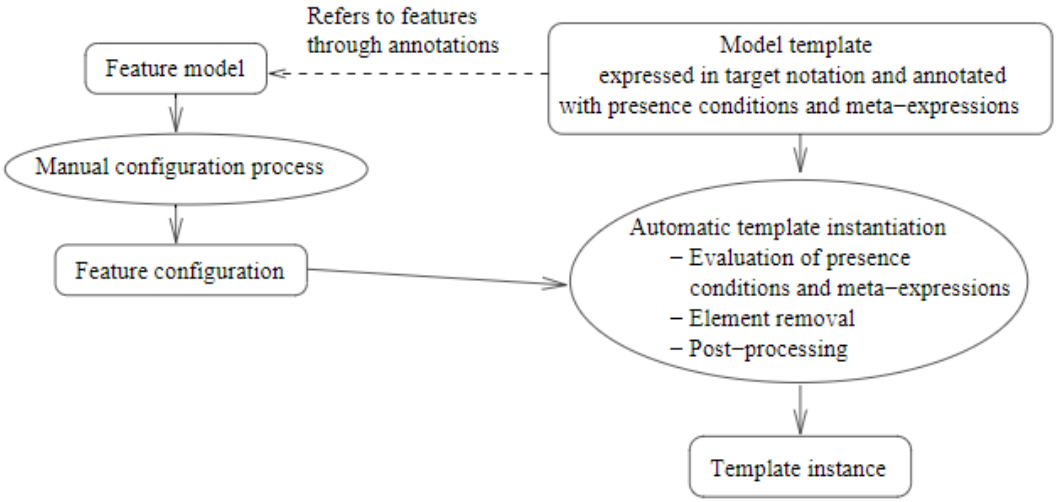
μοντέλο. Το πρότυπο μοντέλου περιέχει την ένωση μεταξύ των στοιχείων του μοντέλου σε όλες τις έγκυρες περιπτώσεις. Το σύνολο των έγκυρων προτύπων αντιστοιχεί στο μέγεθος της οικογένειας μοντέλων. Τα χαρακτηριστικά ενός πρότυπου μοντέλου μπορούν να μετασχηματιστούν χρησιμοποιώντας συνθήκες παρουσίας (presence conditions) και εκφράσεις (meta-expressions). Αυτές οι σημειώσεις ορίζονται με όρους χαρακτηριστικών και ιδιοτήτων χαρακτηριστικών από το μοντέλο χαρακτηριστικών και μπορούν να εκτιμηθούν σε σχέση με τη διαμόρφωση χαρακτηριστικών.

Μία συνθήκη παρουσίας επάνω σε ένα στοιχείο του μοντέλου υποδεικνύει αν το στοιχείο αυτό θα πρέπει να αφαιρεθεί ή όχι από ένα πρότυπο. Οι εκφράσεις χρησιμοποιούνται για τον υπολογισμό των χαρακτηριστικών του μοντέλου όπως το όνομα το του χαρακτηριστικού ή τον τύπο του. Για να δημιουργήσουμε μία οικογένεια μοντέλων μπορούμε να δημιουργήσουμε μία διαμόρφωση χαρακτηριστικών βάση του μοντέλου χαρακτηριστικών. Οι εκφράσεις μπορούν να χρησιμοποιηθούν για τον υπολογισμό των χαρακτηριστικών των βασικών τύπων, καθώς και των αναφορών σε στοιχεία του μοντέλου. Σε αυτή την δημοσίευση, εξετάζονται μόνο οι αναφορές υπολογισμών σε ήδη υπάρχοντα στοιχεία. Μία κλάση μπορεί να αναπαρασταθεί στο πρότυπο ένα το χαρακτηριστικό που την υποδεικνύει έχει επιλεχθεί.

Η πραγματοποίηση της παρούσας προσέγγισης δοθέντος ενός στόχου ακολουθεί τα παρακάτω βήματα:

- Αρχικά πρέπει να ληφθεί η απόφαση σχετικά με την μορφή των συνθηκών παρουσίασης (presence conditions) και των εκφράσεων (meta-expressions).
- Λαμβάνονται αποφάσεις σχετικά με τις συνθήκες έμμεσης παρουσίας που θα πρέπει να έχουν τα χαρακτηριστικά εφόσον ο χρήστης δεν έχει ορίσει κάποια συνθήκη για αυτά.
- Λαμβάνονται αποφάσεις σχετικά με τον μηχανισμό που τα χειριστεί τις επισημάνσεις καθώς και πως θα προβληθούν σχηματικά.
- Λήψη αποφάσεων ένα είναι αναγκαία κάποια επιπλέων επεξεργασία.

Το παρακάτω διάγραμμα δείχνει μία γενική εικόνα της προσέγγισης που ακολουθεί η συγκεκριμένη δημοσίευση.



Εικόνα 10: Μετατροπή μοντέλων χαρακτηριστικών σε άλλα μοντέλα

2.6 Συνθήκες έμμεσης παρουσίας (Implicit Presence Conditions)

Όταν ένα χαρακτηριστικό του μοντέλου δεν έχει εκχωρηθεί ρητά τον χρήστη, είναι μία συνθήκη έμμεσης παρουσίας (IPC). Γενικά, υποθέτοντας ότι μία συνθήκη παρουσίας είναι αληθής είναι μια απλή επιλογή που είναι κατά βάση επαρκής στην πράξη. Ωστόσο, μερικές φορές μία πιο χρήσιμη IPC μπορεί να παρέχεται για ένα στοιχείο δεδομένου με βάση τις συνθήκες παρουσίας άλλων στοιχείων και τη σύνταξη και τη σημασιολογία του σημειώματος στόχου [3]. Για παράδειγμα, σύμφωνα με τη σύνταξη σε UML, μια δυαδική συσχέτιση απαιτεί έναν ταξινομητή σε κάθε άκρο της. Έτσι, μια λογική επιλογή IPC για μια δυαδική συσχέτιση θα ήταν η σύνδεση των συνθηκών παρουσίασης και των δύο ταξινομητών. Με αυτόν τον τρόπο, η κατάργηση οποιουδήποτε από τους ταξινομητές θα οδηγούσε επίσης στην κατάργηση της σύνδεσης. Τα IPC μειώνουν την προσπάθεια σχολιασμού των χαρακτηριστικών του μοντέλου με συνθήκες παρουσίας από τον χρήστη.

2.7 Διαδικασία προτύπου (Template Instantiation)

Μια απλή και γενική διαδικασία δημιουργίας προτύπου προϋποθέτει τον υπολογισμό των εκφράσεων και την αφαίρεση στοιχείων των οποίων οι συνθήκες παρουσίασης είναι ψευδείς. Ωστόσο, η γενική διαδικασία μπορεί να είναι εξειδικευμένη για μια ποιο συγκεκριμένη σημείωση με μερικά πρόσθετα βήματα επεξεργασίας. Εντοπίζονται δύο κατηγορίες επιπρόσθετων βημάτων όπως η εφαρμογή και απλούστευση μεθόδων μετασχηματισμού που διορθώνει αυτόματα τα προβλήματα που μπορεί να δημιουργηθούν από την αφαίρεση χαρακτηριστικών. Καθορίζεται για καταστάσεις στις οποίες υπάρχει μία μοναδική λύση σε ένα πρόβλημα που δημιουργείται από την αφαίρεση στοιχείων. Η απλοποίηση περιλαμβάνει την αφαίρεση στοιχείων που έχουν απολυθεί μετά την αφαίρεση άλλων στοιχείων [3].

Ο πλήρης αλγόριθμός δημιουργίας προτύπου που αναπτύχθηκε σε αυτήν την δημοσίευση μπορεί να συνοψιστεί ως εξής:

- Αξιολόγηση των συνθηκών έμμεσης παρουσίασης και των εκφράσεων. Η αξιολόγηση γίνεται κατά τη διέλευση της ιεραρχίας περιορισμού των στοιχείων στο πρότυπο πρώτα σε βάθος. Τα παιδιά των στοιχείων των οποίων οι συνθήκες παρουσίασης αξιολογούνται ως ψευδείς δεν επισκέπτονται από τον αλγόριθμο επειδή θα αφαιρεθούν.
- Η ανάλυση απομάκρυνσης περιλαμβάνει τον υπολογισμό των IPC και τις πληροφορίες που απαιτούνται για την εφαρμογή της εφαρμογής αυτόματης διόρθωσης
- Αφαίρεση στοιχείων και εφαρμογή αυτόματης διόρθωσης. Στο βήμα αυτό, τα στοιχεία των οποίων οι συνθήκες παρουσίας είναι ψευδείς αφαιρούνται και εφαρμόζονται τυχόν διορθώσεις.
- Απλούστευση. Η απλοποίηση εκτελείται στο τέλος.

2.8 Αρχιτεκτονική βασιζόμενη σε μοντέλο (Model Driven Architecture) και μηχανική απαιτήσεων γραμμής προϊόντων.

Η αρχιτεκτονική βασιζόμενη σε μοντέλα (Model Driven Architecture) εισήχθη από την ομάδα διαχείρισης αντικειμένων (Object Management Group) και βασίζεται στην ιδέα της ανεξάρτητης πλατφόρμας (Platform Independent Model - PIM). Το PIM είναι μια προδιαγραφή ενός συστήματος όσον αφορά τις έννοιες τομέα και την ανεξαρτησία των πλατφόρμων. Το σύστημα μπορεί στη συνέχεια να μετασχηματίσει το PIM σε ένα συγκεκριμένο μοντέλο πλατφόρμας. Καθώς η κύρια δύναμη της αρχιτεκτονικής βασιζόμενη σε μοντέλα (Model Driven Architecture) είναι ο χειρισμός και ο μετασχηματισμός διαφορετικών μοντέλων και τα μοντέλα στόχων και χαρακτηριστικών που εισάγονται στη διαδικασία μας, θα διερευνηθούν οι σχέσεις αυτών των μοντέλων με UML [5]. Οι απαιτήσεις των γραμμών προϊόντων λογισμικού περιλαμβάνουν διάφορες δραστηριότητες. Η κύρια δραστηριότητα περιλαμβάνει την προδιαγραφή του μοντέλου τομέα, το οποίο αποτελείται από τα χαρακτηριστικά τομέα. Ο σχεδιασμός μιας λύσης για αυτές τις απαιτήσεις αποτελεί τη βάση των αρχιτεκτονικών στοιχείων του προϊόντος. Το κύριο πλεονέκτημα της αρχιτεκτονικής βασιζόμενη σε μοντέλα (Model Driven Architecture) σε σχέση με την παραδοσιακή ανάπτυξη είναι ότι η διαχείριση του σημείου μεταβολής της πλατφόρμας γίνεται αυτόματα από το βήμα μετασχηματισμού και δεν αποτελεί ανησυχία του μηχανικού του προϊόντος. Σύμφωνα με το [5] υπάρχουν δύο είδη μεταμορφώσεων:

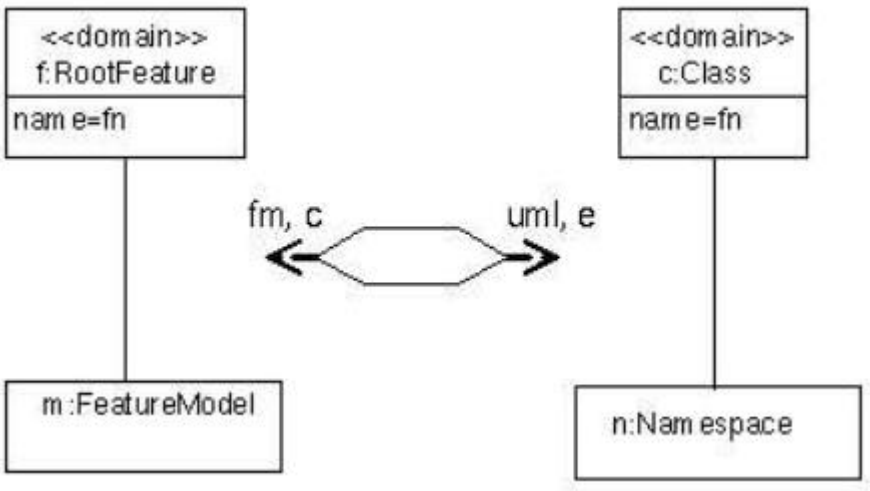
- **Οριζόντια:** επιλογή στόχων ή συνδυασμό στόχων, διαμόρφωση μοντέλου χαρακτηριστικών και παράσταση πλαισίου.
- **Κάθετη:** μοντέλο στόχου γραμμής προϊόντων λογισμικού σε μοντέλο χαρακτηριστικών, χαρακτηριστικό μοντέλο σε αρχιτεκτονική γραμμής προϊόντων λογισμικού και τα παράλληλα εφαρμοσμένα ισοδύναμα.

Το συμβατικό βήμα διαμόρφωσης ενός μοντέλου χαρακτηριστικών αποτελείται από την επιβολή ενός συνόλου περιορισμών που ξεκινά την επιλογή ενός υπο-διαγράμματος χαρακτηριστικών, ενδεχομένως με ορισμένες εναλλακτικές παραλλαγές που αναβάλλονται στο χρόνο εκτέλεσης.

2.8.1 Μετασχηματισμούς γραφήματος

Ο ορισμός του μετασχηματισμού μπορεί να δει στην πιο ώριμη κατάσταση του ως μεταγλωττιστής για μια συγκεκριμένη γλώσσα. Ο συνδυασμός μοντέλων / στόχων θα καταρτίζεται σε μια εφαρμογή εργασίας που χρησιμοποιεί τον ορισμό του μετασχηματισμού, τα στοιχεία και τις απαιτήσεις του πελάτη.

Οι κανόνες μετασχηματισμού γραφήματος αποτελούνται από τρία μέρη: μια αριστερή πλευρά ή μια θετική κατάσταση εφαρμογής, μία δεξιά πλευρά και ένα σύνολο αρνητικών συνθηκών εφαρμογής. Η αριστερή πλευρά του κανόνα αναφέρει ένα μορφισμό που πρέπει να βρεθεί στο γράφημα για να εφαρμοστεί αυτός ο κανόνας. Η δεξιά πλευρά του κανόνα ορίζει ποια στοιχεία της αριστερής πλευράς διαγράφονται, ποια διατηρούνται και ποια νέα στοιχεία πρέπει να δημιουργηθούν στο προκύπτον γράφημα. Οι αρνητικές συνθήκες εφαρμογής καθορίζουν υπογράμματα που δεν πρέπει να υπάρχουν στο γράφημα για να μπορέσουν να εφαρμόσουν τον κανόνα [5].

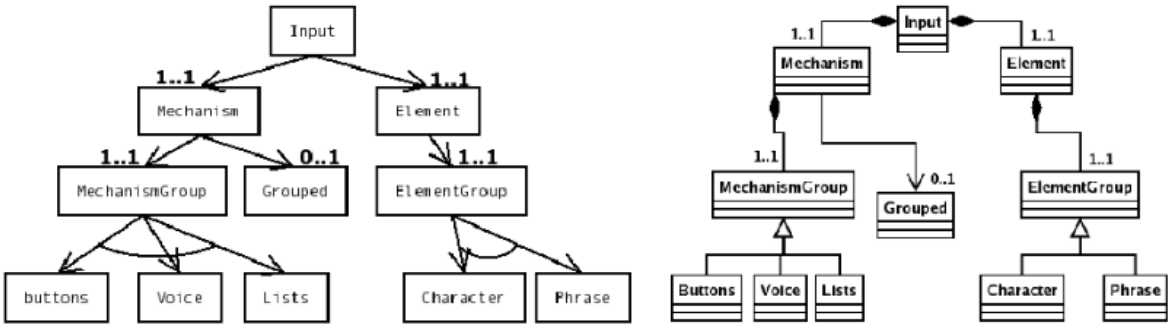


Εικόνα 11: Παράδειγμα κανόνα μετατροπής του χαρακτηριστικού ρίζας σε κλάση.

Αυτό το σύνολο μπορεί να θεωρηθεί ως διαδικασία μετασχηματισμού τριών σταδίων.

- **Δημιουργία κλάσεων:** Απαιτείται ένας κανόνας για τη δημιουργία των τάξεων αρχιτεκτονικής της γραμμής προϊόντων λογισμικού από το αντίστοιχο χαρακτηριστικό του μοντέλου χαρακτηριστικών. Οι αρνητικές συνθήκες εφαρμογής πρέπει να είναι ξεκάθαρες για να διασφαλιστεί η εκτέλεση ακριβώς ενός μετασχηματισμού ανά χαρακτηριστικό.
- **Μετακίνηση παιδιών:** Απαιτείται ένας ή περισσότεροι κανόνες για τη μετακίνηση όλων των συνδέσμων μεταξύ του κόμβου και των παιδιών κόμβων του. Οι σχέσεις μετακινούνται από τον κόμβο λειτουργιών στον κόμβο τάξης. Ένας κανόνας ανά τύπο κόμβου παιδιών ορίζεται.

- **Διαγραφή του κόμβου λειτουργίας:** Τέλος, ο κόμβος λειτουργίας διαγράφεται από το γράφημα. Κανένας κόμβος με συνδέσμους για παιδιά δεν μπορεί να διαγραφεί εξαιτίας του περιορισμού των άκρων.



Εικόνα 12: Παράδειγμα μετατροπής μοντέλου χαρακτηριστικών σε διάγραμμα κλάσεων.

2.9 Διατηρησιμότητα (Maintainability) των μοντέλων χαρακτηριστικών μίας γραμμής προϊόντων λογισμικού.

Δεδομένου του γεγονότος ότι πολλά διαφορετικά συστήματα λογισμικού μπορούν να δημιουργηθούν από μια ενιαία γραμμή προϊόντων λογισμικού, μπορεί να αναμένεται ότι ένας σχεδιασμός χαμηλής ποιότητας μπορεί να έχει αρνητική επίπτωση σε πολλά συστήματα λογισμικού που δημιουργούνται. Επομένως, μπορούμε να αναγνωρίσουμε την ανάγκη δημιουργίας έγκαιρων δεικτών εξωτερικών χαρακτηριστικών ποιότητας προκειμένου να αποφευχθούν οι δημιουργίες ελαττωματικού και χαμηλής ποιότητας σχεδιασμού κατά τα τελευταία στάδια της παραγωγής [4].

Στον τομέα της ανάπτυξης αντικειμενοστραφών λογισμικών έχει αναγνωριστεί ευρέως ότι για να είναι σε θέση να σχεδιάσουν συστήματα λογισμικού υψηλής ποιότητας, οι προγραμματιστές θα πρέπει να επικεντρωθούν στην ανάπτυξη υψηλής ποιότητας εννοιολογικών μοντέλων των προβλεπόμενων συστημάτων στα αρχικά στάδια της ανάπτυξης. Ως εκ τούτου, θα πρέπει να δοθεί έμφαση στη διατήρηση της ποιότητας των σχεδιασμένων εννοιολογικών μοντέλων του λογισμικού παρά στην προσπάθεια επιβολής της ποιότητας στα τελικά στάδια της ανάπτυξης, όπως για παράδειγμα στα στάδια κωδικοποίησης. Σε ένα τέτοιο πρότυπο ανάπτυξης, η ποιότητα των μοντέλων είναι εξαιρετικά σημαντική επειδή επηρεάζει άμεσα την ποιότητα του τελικού προϊόντος λογισμικού. Ως αποτέλεσμα, η επιβολή της ποιότητας στα εννοιολογικά μοντέλα των γραμμών προϊόντων λογισμικού είναι απαραίτητη. Αυτό οφείλεται στο γεγονός ότι οι σειρές προϊόντων λογισμικού χρησιμεύουν ως βάση για την ανάπτυξη οικογενειών συστημάτων λογισμικού που έχουν κοινή λειτουργικότητα και συμπεριφορά. Ως εκ τούτου, ενώ ένα χαμηλής ποιότητας μοντέλο ενός ενιαίου συστήματος λογισμικού θα επηρεάσει αυτό το λογισμικό και ίσως και τις μεταγενέστερες εκδόσεις του, σε σειρές προϊόντων λογισμικού, αυτή η ανεπάρκεια θα μεταφραστεί σε πολλά διαφορετικά συστήματα λογισμικού χαμηλής ποιότητας ή ακόμη και λανθασμένα που δημιουργήθηκαν από τη γραμμή προϊόντων.

Όσον αφορά τα χαρακτηριστικά στοιχεία του λογισμικού, τα χαρακτηριστικά ποιότητας μπορούν να κατηγοριοποιηθούν σε δύο κατηγορίες: εσωτερικά χαρακτηριστικά ποιότητας και εξωτερικά χαρακτηριστικά ποιότητας [4].

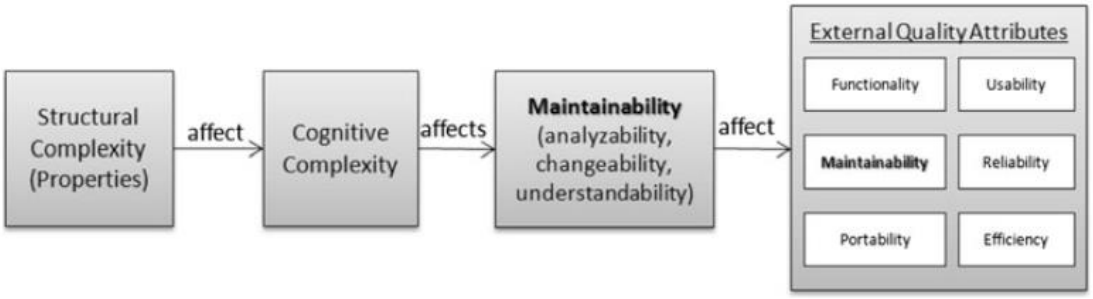
- **Εσωτερικά χαρακτηριστικά:** Τα εσωτερικά χαρακτηριστικά ποιότητας είναι αυτά που μπορούν να μετρηθούν με βάση χαρακτηριστικά προϊόντων όπως μέγεθος, μήκος ή πολυπλοκότητα.
- **Εξωτερικά χαρακτηριστικά:** τα χαρακτηριστικά εξωτερικής ποιότητας, όπως η απόδοση, η αξιοπιστία και η συντηρησιμότητα, είναι εκείνα τα χαρακτηριστικά ποιότητας που μπορούν να μετρηθούν μόνο σε σχέση με το πώς ένα σύστημα λογισμικού σχετίζεται με το περιβάλλον του και, ως εκ τούτου, μπορούν να μετρηθούν μόνο όταν το σύστημα λογισμικού έχει πλήρως αναπτυχθεί. Η διατήρηση (maintainability) ως ένα από τα χαρακτηριστικά εξωτερικής ποιότητας αφορά την αξιολόγηση της κατανόησης, αλλαγής και ανάλυσης των αναπτυγμένων μοντέλων χαρακτηριστικών.

Θεωρούμε ότι η συντηρησιμότητα είναι μια σύνθεση τριών χαρακτηριστικών, δηλαδή της ανάλυσης, της μεταβλητότητας και της κατανόησης. Όπως προτείνεται στο πρότυπο ISO 9126 για την ποιότητα του λογισμικού (ISO 2001), οι τρεις αυτές έννοιες ερμηνεύονται ως εξής:

- **Δυνατότητα ανάλυσης**: είναι η ικανότητα του εννοιολογικού μοντέλου ενός συστήματος λογισμικού να διαγνωστεί για ανεπάρκεια.
- **Μεταβλητότητα:** είναι η δυνατότητα και η ευκολία αλλαγής σε ένα μοντέλο όταν απαιτούνται τροποποιήσεις.
- **Κατανόηση:** είναι η προοπτική και η πιθανότητα το μοντέλο του συστήματος λογισμικού να κατανοηθεί από τους χρήστες του ή άλλους σχεδιαστές μοντέλων.

2.9.1 Σχεδιασμός μετρήσεων

Η υποκείμενη λογική για την ανάπτυξη ποσοτικών μοντέλων για τη συσχέτιση ιδιοτήτων δομικών μοντέλων και εξωτερικών χαρακτηριστικών ποιότητας έχει αναγνωριστεί ότι σχετίζεται με την έννοια της γνωστικής πολυπλοκότητας. Σύμφωνα με έρευνες έχει υποστηριχθεί ότι οι άνθρωποι ενεργούν ορθολογικά στο βαθμό της γνωστικής τους ικανότητας. Ως εκ τούτου, μπορούν να χάσουν το δρόμο για το τι κάνουν και να κάνουν παράλογες αποφάσεις σε πολύπλοκα και μεγάλα περιβάλλοντα. Αυτό ισχύει και για τους μηχανικούς λογισμικού που μπορεί να κατακλύζονται από την πολυπλοκότητα του τομέα στον οποίο ασχολούνται και ως εκ τούτου μπορούν να εισάγουν σφάλματα στα μοντέλα που σχεδιάζονται. Με άλλα λόγια, η γνωστική πολυπλοκότητα είναι η ψυχική επιβάρυνση των ανθρώπων που εκτελούν σχετικές λειτουργίες στο λογισμικό. Ως εκ τούτου, το μεγαλύτερο και πιο περίπλοκο σύστημα λογισμικού είναι, τόσο υψηλότερο θα είναι ο βαθμός γνωστικής πολυπλοκότητάς του. Η υψηλή γνωστική πολυπλοκότητα ενός λογισμικού μπορεί να επηρεάσει τα εξωτερικά του χαρακτηριστικά ποιότητας.



Εικόνα 13: Η σχέση μεταξύ δομικών ιδιοτήτων, γνωστικής πολυπλοκότητας και εξωτερικών χαρακτηριστικών ποιότητας.

2.9.2 Δομικές μετρήσεις μοντέλων χαρακτηριστικών

Στο βαθμό που γνωρίζουμε, έχουν διεξαχθεί πολύ λίγες προκαταρκτικές μελέτες για τον καθορισμό κατάλληλων δομικών μετρήσεων μοντέλων χαρακτηριστικών. Ακόμη και μέσα σε αυτά τα υπάρχοντα έργα, δεν έχει γίνει πολύ θεωρητική ή εμπειρική αξιολόγηση των προτεινόμενων μετρήσεων. Δεδομένου έχει αποδειχτεί επισήμως ότι μια μετρική δεν μπορεί να συλλάβει όλες τις πτυχές της πολυπλοκότητας, προτείνονται μερικές απλές αλλά διαισθητικές δομικές μετρήσεις ως αναφορές μετρήσεων για τα μοντέλα χαρακτηριστικών των γραμμών προϊόντων λογισμικού. Η κύρια ιδέα πίσω από το σχεδιασμό αυτών των μετρήσεων είναι η πληρότητα και η απλότητα. Για αυτόν τον λόγο αυτές οι μερικές προσπαθούν να καλύψουν όσο το δυνατόν περισσότερα δομικά χαρακτηριστικά ενός μοντέλου χαρακτηριστικών. Για να επιτευχθεί αυτό, ελήφθησαν υπόψη οι διαρθρωτικές μετρήσεις που προτείνονται στους τομείς του αντικειμενοστραφούς σχεδιασμού, των σχέσεων οντοτήτων-σχέσεων και ακόμη και των μοντέλων επιχειρηματικών διαδικασιών. Με αυτά στο νου οι μετρήσεις που θα αναλυθούν σε αυτήν την διπλωματική εργασία όσον αφορά την συντηρισιμότητα έχουν να κάνουν με τρείς κύριους τύπους, το μέγεθος, το μήκος του μοντέλου καθώς και την πολυπλοκότητά του. Για το μέγεθος του μοντέλου έχουμε τις μετρήσεις για τον αριθμό των χαρακτηριστικών στο μοντέλο, τον αριθμό των κορυφαίων χαρακτηριστικών και τον αριθμό των χαρακτηριστικών φίλων. Για την μέτρηση της δομικής πολυπλοκότητας ενός μοντέλου χαρακτηριστικών έχουμε την κυκλική πολυπλοκότητα, τους περιορισμούς μεταξύ δέντρων, την αναλογία μεταβλητότητας, τον συντελεστή συνδεσιμότητας-πυκνότητας και την ευελιξία διαμόρφωσης. Τέλος, για την μέτρηση του μήκους του μοντέλου χαρακτηριστικών έχουμε το βάθος του δέντρου [4]. Οι παραπάνω μετρήσεις θα αναλυθούν περισσότερο τόσο σημασιολογικά όσο και ως προς τον τρόπο που τις εξάγουμε σε επόμενο κεφάλαιο της εργασίας.

Οι μετρήσεις που παρουσιάστηκαν είναι ανεξάρτητες μεταβλητές. Θεωρούνται ανεξάρτητες, επειδή μέσα στη σχέση αιτίας-αποτελέσματος που μας ενδιαφέρει, αντιπροσωπεύουν την αιτία, δηλαδή, μας ενδιαφέρει να δούμε αν οι δομικές μετρήσεις συσχετίζονται με τη συντηρησιμότητα και τα υποσυστήματα της ή όχι. Σε μια τέτοια ρύθμιση, οι δομικές μετρήσεις θεωρούνται οι ανεξάρτητες μεταβλητές. Κάθε μία από αυτές τις μετρήσεις είναι

μια ανεξάρτητη μεταβλητή που αντιπροσωπεύει μια δομική ιδιότητα ενός μοντέλου χαρακτηριστικών. Όλες οι μεταβλητές καθώς και ο τρόπος που συλλέγουμε πληροφορίες για αυτές έχουν γίνει με τέτοιο τρόπο ώστε να σέβονται τις διαφορετικές σχέσεις και ιδιότητες που έχουν τα μοντέλα χαρακτηριστικών. Σύμφωνα με την έρευνα [4] μπορούμε να ομαδοποιήσουμε τις παραπάνω μεταβλητές έτσι ώστε να εξάγουμε στοιχεία για τρείς κύριες ομάδες, την αναλυσιμότητα, την κατανοητότητα και την μεταβλητότητα των μοντέλων χαρακτηριστικών. Για την αναλυσιμότητα η έρευνα καταλήγει ότι οι μετρήσεις των αριθμών των φίλων χαρακτηριστικών και του συνόλου των διαφορετικών δυνατών προϊόντων που μπορούν να παραχθούν είναι οι ποιο σημαντικές για να καθορίσουν αυτήν την μετρική. Για την κατανοητότητα χρησιμοποιούνται οι μεταβλητές του αριθμού των φίλων χαρακτηριστικών καθώς και της ευελιξίας του μοντέλου χαρακτηριστικών. Τέλος, για την μεταβλητότητα η [4] προτείνει την χρήση των μεταβλητών ευελιξίας του μοντέλου χαρακτηριστικών, του αριθμού των φίλων χαρακτηριστικών και της κυκλικής πολυπλοκότητας. Με βάση αυτές τις επιλεγμένες μετρήσεις, μπορούμε να συμπεράνουμε ότι το σύνολο του αριθμού των φίλων χαρακτηριστικών, του συνόλου των διαφορετικών δυνατών προϊόντων που μπορούν να παραχθούν, της κυκλικής πολυπλοκότητας και της ευελιξίας του μοντέλου χαρακτηριστικών είναι το επαρκές υποσύνολο των προτεινόμενων μετρήσεων για την αξιολόγηση της διατηρησιμότητας.

3 Ανάλυση της Εφαρμογής

Σε αυτό το κεφάλαιο θα γίνει ανάλυση και αναλυτική περιγραφή του εργαλείου μετασχηματισμού ενός μοντέλου χαρακτηριστικού σε κώδικα λογισμικού καθώς και οι τεχνολογίες που χρησιμοποιήθηκαν. Ποιο συγκεκριμένα, θα ακολουθήσει εκτενής περιγραφή της εφαρμογής «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)» καθώς και περιγραφή των εργαλείων που χρησιμοποιήθηκαν για την δημιουργία της εφαρμογής. Τέλος θα δοθούν λεπτομέρειες σχετικά με την εφαρμογή «FeatureIDE» και πως με την χρήση της μπορούμε να σχεδιάσομε μοντέλα χαρακτηριστικών για να τα χρησιμοποιήσουμε ως είσοδο στην εφαρμογή που αναπτύχθηκε στην συγκεκριμένη διπλωματική εργασία.

3.1 Περιγραφή της εφαρμογής F.C.C.M.A

Η εφαρμογή μετασχηματισμού μοντέλων χαρακτηριστικών σε κώδικα λογισμικού υλοποιήθηκε ώστε να προσφέρει ένα νέο εργαλείο στους μηχανικούς λογισμικού με σκοπό να φέρει ποιο κοντά την διαδικασία σχεδίαση λογισμικού με την διαδικασία παραγωγής κώδικα. Η εφαρμογή έχει την δυνατότητα να δέχεται ως είσοδο ένα αρχείο τύπου xml όπου θα περιγράφει το μοντέλο χαρακτηριστικών και αναγνωρίζει τις βασικές κλάσεις που πρέπει να δημιουργηθούν για την υλοποίηση του συγκεκριμένου μοντέλου χαρακτηριστικών. Το μοντέλο χαρακτηριστικών δημιουργείτε με την χρήση του «FeatureIDE» προσθέτου του «Eclipse Oxygen» και έπειτα εξάγεται σε μορφή xml ώστε να μπορέσει να επεξεργαστεί από το «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)». Η εφαρμογή δίνει στον χρήστη την δυνατότητα να αποθηκεύσει την δουλειά του και να επιστρέψει αργότερα ώστε να την ολοκληρώσει. Με αυτόν τον τρόπο δίνεται η δυνατότητα για την ανάλυση και σχεδίαση μεγάλων γραμμών προϊόντων λογισμικού που είναι αδύνατον να ολοκληρωθούν σε μία μέρα. Τέλος, το πρόγραμμα «F.C.C.M.A» δίνει δύο κύριες επιλογές στον χρήστη, την επιλογή εξαγωγής κώδικα και την επιλογή προβολής μετρήσεων που σκοπό έχουν να βοηθήσουν τον μηχανικό λογισμικού να κατανοήσει κατά πόσο θα είναι δυνατή η συντήρηση της συγκεκριμένης γραμμής προϊόντων λογισμικού μέσα από το μοντέλο χαρακτηριστικών.

3.1.1 Δυνατότητα εξαγωγής κώδικα

Αυτήν η δυνατότητα έχει ως σκοπό να παράγει κώδικα σε JAVA σύμφωνα με τα στοιχεία και τις μεταβλητές που έχει δώσει ο χρήστης για κάθε μία κλάση που δημιουργήθηκε από το μοντέλο χαρακτηριστικών (feature model). Για να γίνει αυτό ο χρήστης εισάγει τις παραμέτρους κάθε μεταβλητής της κάθε κλάσεις στην εφαρμογή και αυτήν με την εκτέλεση της διαδικασίας εξαγωγής κώδικά παράγει όλες τις κλάσεις μαζί με τις μεταβλητές τους καθώς και τις απαραίτητες συναρτήσεις. Με αυτόν τον τρόπο επιταχύνει την δημιουργία κώδικα καθώς πλέον ο απαραίτητος κώδικας για την κάθε κλάση δημιουργείτε ένα βήμα μετά την διαδικασία σχεδίασης του λογισμικού. Ο χρήστης προκειμένου να εισάγει στην εφαρμογή κάποια μεταβλητή πρέπει να εισάγει τις παρακάτω πληροφορίες σχετικά με αυτήν:

- **Όνομα μεταβλητής (Value Name):** Το όνομα όπου θα χαρακτηρίζει αυτήν την μεταβλητή. Το όνομα μπορεί να είναι οποιαδήποτε τιμή τύπου String.
- **Πρόσβαση της μεταβλητής (Value Access):** Αυτήν η τιμή καθορίζει το επίπεδο πρόσβασης της μεταβλητής. Ο χρήστης μπορεί να επιλέξει από μία λίστα μεταξύ των ιδιωτικών (private), δημόσιων (public) και καθολικών (global).
- **Τύπος της μεταβλητής (Value Type):** Ο χρήστης έχει την δυνατότητα να επιλέξει από μία λίστα τον τύπο της μεταβλητής. Για παράδειγμα ως τύπο μεταβλητής αν θέλει να επιλέξει η μεταβλητή να είναι ακέραιος αριθμός επιλέξει «int» από την λίστα.

Επίσης, ο χρήστης μπορεί να κάνει τροποποιήσεις σε μεταβλητές που έχει είδη κάνει εισαγωγή σε κάποια κλάση απλά επιλέγοντάς τες και έπειτα τροποποιώντας τα στοιχεία που θέλει να αλλάξει. Τέλος, δίνεται η δυνατότητα διαγραφής μεταβλητών από την βάση σε περίπτωση που δεν χρειάζονται.

3.1.2 Δυνατότητα εξαγωγής μετρήσεων συντηρησιμότητας

Αυτήν η δυνατότητα δίνει τα απαραίτητα στοιχεία στον μηχανικό λογισμικού ώστε να είναι σε θέση να αποφασίσει κατά πόσο είναι δυνατών η συντήρηση της συγκεκριμένης γραμμής προϊόντων λογισμικού. Για να είναι σε θέση η εφαρμογή να παράγει τις συγκεκριμένες μετρικές ο χρήστης πρέπει να έχει εισάγει ένα μοντέλο χαρακτηριστικών πρώτα στο σύστημα με την μορφή αρχείου xml. Διαφορετικά, θα πρέπει να έχει εισάγει το κατάλληλο αρχείο απλού κειμένου όπου αποθηκεύει η εφαρμογή με σκοπό να βοηθήσει τον χρήστη να συνεχίσει την εργασία του κάποια άλλη στιγμή.

Έχοντας αυτά ως δεδομένα η εφαρμογή έπειτα παίρνει το μοντέλο χαρακτηριστικών και το επεξεργάζεται με σκοπό να βγάλει τις παρακάτω μετρικές. Οι μετρικές αυτές είναι οι εξής:

1. Ο αριθμός των χαρακτηριστικών στο μοντέλο
2. Ο αριθμός των κορυφαίων χαρακτηριστικών
3. Ο αριθμός των χαρακτηριστικών φίλων
4. Η κυκλική πολυπλοκότητα
5. Οι περιορισμοί μεταξύ δέντρων
6. Η αναλογία μεταβλητότητας
7. Ο συντελεστής συνδεσιμότητας-πυκνότητας
8. Η ευελιξία διαμόρφωσης
9. Το βάθος του δέντρου

Στον παρακάτω πίνακα εξηγούνται αναλυτικά τα στοιχεία που εξετάζει η κάθε μεταβλητή καθώς χωρίζονται οι μεταβλητές σε τρείς βασικές κατηγορίες [4].

Τύπος Μέτρησης	**Όνομα Μέτρησης**	**Επεξήγηση**
Μέγεθος	Αριθμός χαρακτηριστικών	Ο συνολικός αριθμός των δυνατοτήτων που υπάρχουν σε ένα μοντέλο χαρακτηριστικών. Αυτό περιλαμβάνει τόσο τα φύλλα όσο και τα γονικά χαρακτηριστικά. Ο Αριθμός χαρακτηριστικών μετρά όλους τους κόμβους στο δέντρο μοντέλου χαρακτηριστικών.
	Αριθμός κορυφαίων χαρακτηριστικών	Ο αριθμός των χαρακτηριστικών που είναι οι πρώτοι άμεσοι απόγονοι της ρίζας του μοντέλου χαρακτηριστικών. Με άλλα λόγια, ο αριθμός των κόμβων σε βάθος ένα στο δέντρο.
	Αριθμός χαρακτηριστικών φίλων	Ο αριθμός των χαρακτηριστικών χωρίς παιδιά ή περαιτέρω εξειδίκευση. Αυτά αντιστοιχούν στα φύλλα του δέντρου του μοντέλου χαρακτηριστικών.
Δομική πολυπλοκότητα	Κυκλική πολυπλοκότητα	Ο αριθμός των ξεχωριστών κύκλων που μπορεί να βρεθεί στο μοντέλο χαρακτηριστικών. Δεδομένου ότι τα μοντέλα χαρακτηριστικών έχουν τη μορφή δέντρων, δεν μπορούν να υπάρχουν κύκλοι σε ένα μοντέλο χαρακτηριστικών. Ωστόσο, οι περιορισμοί ακεραιότητας μεταξύ των διαθέσιμων χαρακτηριστικών μπορούν να προκαλέσουν κύκλους. Είναι απλό να δείξουμε ότι ο αριθμός των διακεκριμένων κύκλων και, συνεπώς, η κυκλική πολυπλοκότητα ενός μοντέλου χαρακτηριστικών είναι ισοδύναμη με τον αριθμό των περιορισμών ακεραιότητας ενός μοντέλου χαρακτηριστικών. Αυτό οφείλεται στην δομή τύπου δέντρου (χωρίς κύκλους) των μοντέλων χαρακτηριστικών
	Περιορισμοί μεταξύ δέντρων	Ο λόγος του αριθμού των μοναδικών χαρακτηριστικών που εμπλέκονται στον περιορισμό ακεραιότητας μοντέλου χαρακτηριστικών προς όλο τον αριθμό των χαρακτηριστικών στο μοντέλο χαρακτηριστικών. Το μέτρο αυτό αντιπροσωπεύει το βαθμό εμπλοκής των χαρακτηριστικών στον ορισμό των περιορισμών ακεραιότητας

	Αναλογία μεταβλητότητας	Ο μέσος συντελεστής διακλάδωσης των γονικών χαρακτηριστικών στο μοντέλο χαρακτηριστικών. Με άλλα λόγια, ο μέσος αριθμός παιδιών των κόμβων στο δέντρο μοντέλου χαρακτηριστικών
	Συντελεστής συνδεσιμότητας-πυκνότητας	Η αναλογία του αριθμού των άκρων σε σχέση με τον αριθμό των χαρακτηριστικών ενός μοντέλου χαρακτηριστικών. Στη θεωρία γραφημάτων, ο συντελεστής συνδεσιμότητας αντιπροσωπεύει πόσο καλά συνδέονται τα στοιχεία γραφικών παραστάσεων
	Ευελιξία Διαμόρφωσης	Αυτή είναι η αναλογία του αριθμού των προαιρετικών λειτουργιών σε όλες τις διαθέσιμες λειτουργίες του μοντέλου χαρακτηριστικών. Το σκεπτικό πίσω από αυτό είναι ότι όσο περισσότερα προαιρετικά χαρακτηριστικά υπάρχουν στο μοντέλο χαρακτηριστικών, τόσο περισσότερες επιλογές είναι διαθέσιμες για τους σχεδιαστές από τις οποίες μπορεί να επιλέξουν κατά τη διαμόρφωση του μοντέλου χαρακτηριστικών
Μήκος	Βάθος Δέντρου	Το μήκος της μεγαλύτερης διαδρομής από το χαρακτηριστικό μοντέλο ρίζας σε κάποιο φύλλο του μοντέλου χαρακτηριστικών

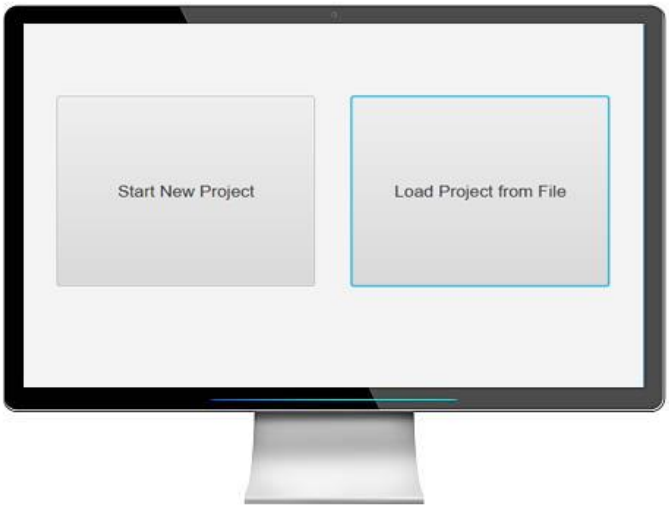
Έπειτα αφού ο χρήστης επιλέξει την εξαγωγή των μετρήσεων η εφαρμογή πηγαίνει τον χρήστη σε κατάλληλη σελίδα όπου παρουσιάζονται οι μετρήσεις αυτές. Ο χρήστης δεν είναι απαραίτητο να έχει εισάγει μεταβλητές πριν εξάγει τις μετρικές συντηρησιμότητας καθώς αυτές εξάγονται απευθείας από το μοντέλο χαρακτηριστικών. Αυτό δίνει την δυνατότητα στον μηχανικό λογισμικού να προβεί σε οποιαδήποτε αρχιτεκτονική αλλαγή κρίνει αυτός απαραίτητη πριν να αρχίσει να καθορίζει τις μεταβλητές της κάθε κλάσης.

3.2 Πρότυπο εφαρμογής F.C.C.M.A

Για την υλοποίηση της εφαρμογής «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)», δημιουργήθηκε ένα πρότυπο των βασικών οθονών και λειτουργιών του συστήματος. Το πρότυπο αυτό έπειτα θα αποτελέσει το κύριο γραφικό περιβάλλον της εφαρμογής. Μέσω του παρακάτω προσχεδίου γίνεται απεικόνιση τόσο των στατικών όσο και των δυναμικών στοιχείων της εφαρμογής. Για την δημιουργία

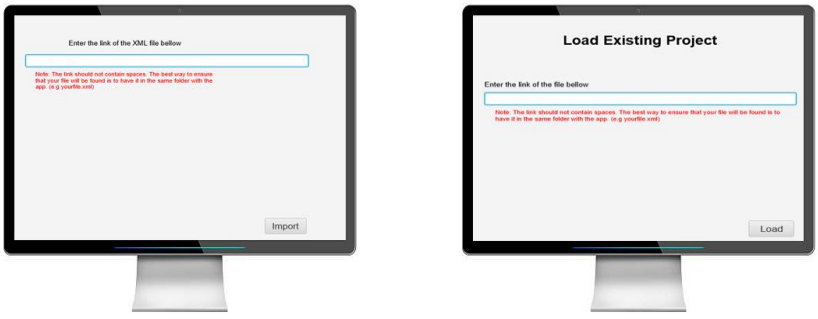
των οθονών χρησιμοποιήθηκε το πρόγραμμα SceneBuilder όπου θα γίνει εκτενέστερη αναφορά για αυτό σε επόμενο κεφάλαιο.

Στην Εικόνα 14 απεικονίζετε η αρχική οθόνη της εφαρμογής όπου θα δίνει στον χρήστη δύο επιλογές. Η πρώτη επιλογή θα είναι η δημιουργία ενός νέου έργου (Start new Project) όπου ο χρήστης θα έχει την δυνατότητα να εισάγει ένα μοντέλο χαρακτηριστικών στο πρόγραμμα από xml αρχείο. Η δεύτερη επιλογή είναι η φόρτωση ενός παλιού έργου στο πρόγραμμα (Load Project from File) όπου δίνει την δυνατότητα στον χρήστη να συνεχίσει την εργασία του που έχει από πριν αποθηκεύσει.



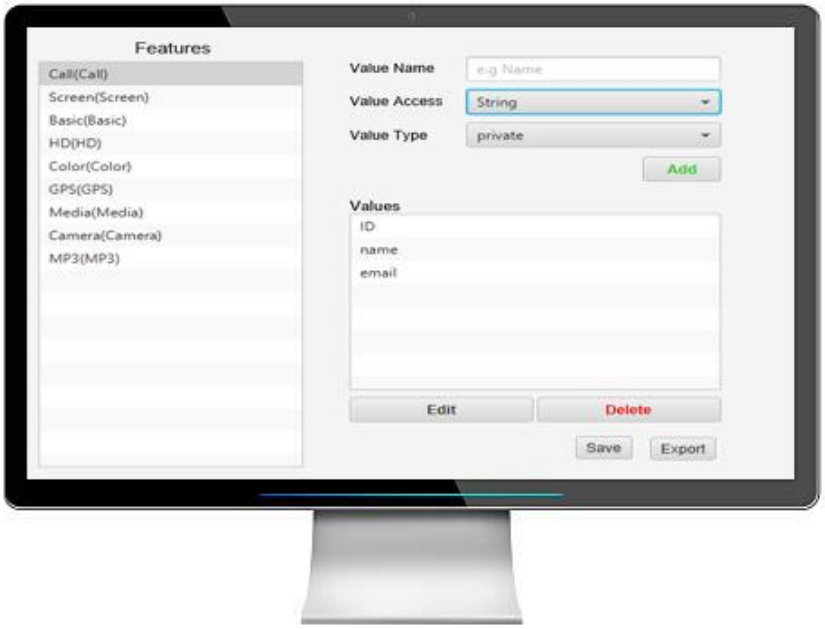
Εικόνα 14: Αρχική οθόνη εφαρμογής

Στην Εικόνα 15 μπορούμε να δούμε στα αριστερά την οθόνη όπου μεταβαίνει ο χρήστης μετά την επιλογή να εισάγει ένα νέο μοντέλο χαρακτηριστικών στο πρόγραμμα. Σε αυτήν την οθόνη δίνεται η δυνατότητα πληκτρολόγησης του προορισμού που βρίσκεται το αρχείο xml ώστε το πρόγραμμα να μπορέσει να το εισάγει. Η δεξιά οθόνη είναι αυτήν που μεταβαίνει ο χρήστης εάν επιλέξει να φορτώσει ένα υπάρχον έργο στο πρόγραμμα αφού το έχει αποθηκεύσει κάποια προηγούμενη χρονική στιγμή. Για να γίνει αυτό, ο χρήστης εισάγει την διαδρομή του αρχείου όπου είναι αποθηκευμένο το παλιό έργο στο κατάλληλο πεδίο.



Εικόνα 15: Οθόνη εισαγωγής νέου μοντέλου χαρακτηριστικών (αριστερά). Οθόνη εισαγωγής έργου από αρχείο (δεξιά).

Στην Εικόνα 16 απεικονίζεται η κεντρική οθόνη της εφαρμογής όπου ο μηχανικός λογισμικού μπορεί να εισάγει τις μεταβλητές στις κλάσεις που έχουν δημιουργηθεί από το μοντέλο χαρακτηριστικών αυτόματα από το πρόγραμμα. Επίσης δίνονται και οι επιλογές να αποθηκεύσει την εργασία του και να συνεχίσει κάποια άλλη στιγμή καθώς και να εξάγει τον κώδικα. Τέλος σε αυτήν την οθόνη βλέπου όλες τις δυνατές επιλογές που θα έχει ο χρήστης.



Εικόνα 16: Κύρια οθόνη του συστήματος.

3.3 Προγράμματα που χρησιμοποιήθηκαν

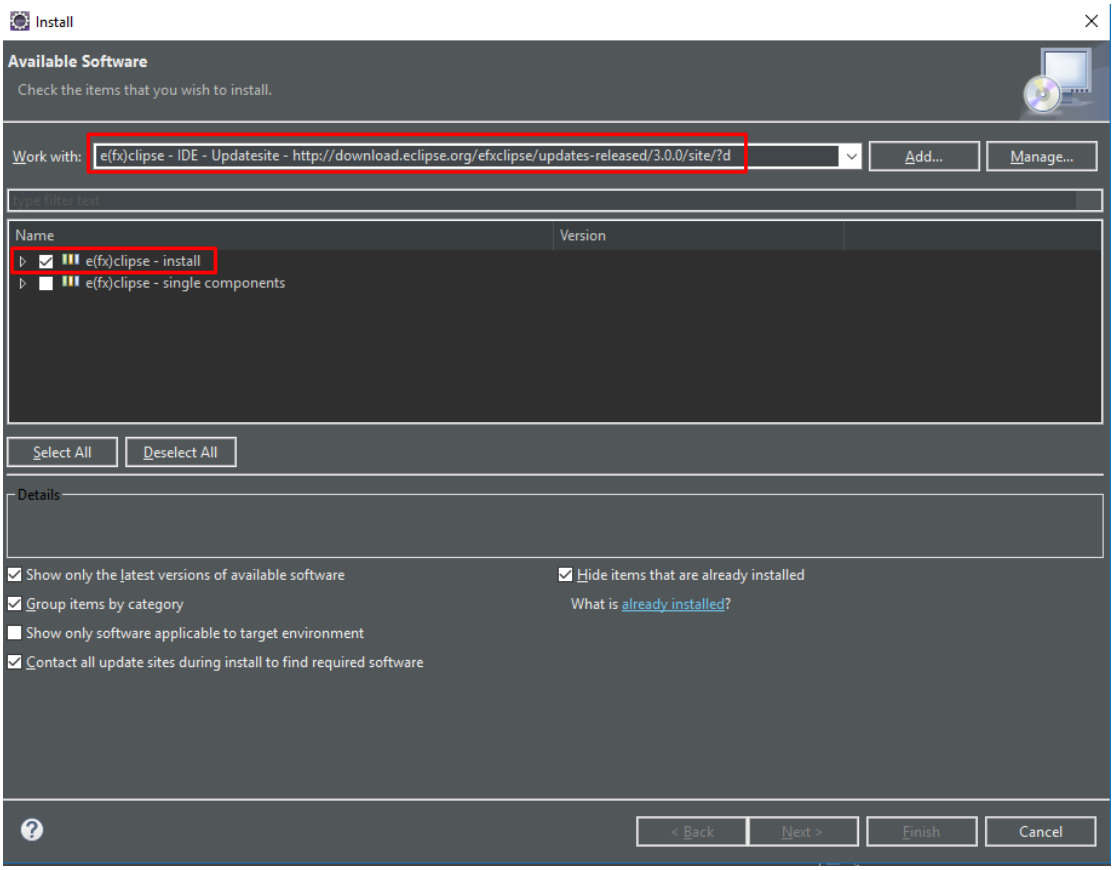
Για την ανάπτυξη του εργαλείου «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)», χρησιμοποιήθηκε μια πληθώρα εργαλείων τόσο για την ανάπτυξη του κώδικα όσο και για την ανάπτυξη του γραφικού περιβάλλοντος της εφαρμογής. Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε το ολοκληρωμένο περιβάλλον ανάπτυξης Eclipse Oxygen [10] στο οποίο προστέθηκε η επέκταση FeatureIDE [7][8] για τον σχεδιασμό των μοντέλων χαρακτηριστικών καθώς και η επέκταση SceneBuilder [12] για την κατασκευή του γραφικού περιβάλλοντος της εφαρμογής. Για την υλοποίηση του συστήματος χρησιμοποιήθηκε επίσης η γλώσσα προγραμματισμού Java [10] καθώς συνδυάστηκε και με XML για την εξαγωγή και εισαγωγή αρχείων στο σύστημα όπως επίσης και με JavaFX [9] όπου είναι η κύρια δομή των γραφικών.

3.3.1 Eclipse Oxygen, SceneBuilder και JavaFX

Το μεγαλύτερο μέρος της εφαρμογής υλοποιήθηκε στο ολοκληρωμένο περιβάλλον ανάπτυξης λογισμικού του Eclipse Oxygen. Το Eclipse είναι ένα εργαλείο το οποίο μας επιτρέπει μέσα στο περιβάλλον χώρο του να δημιουργήσουμε και να επεξεργαστούμε αρχεία Java αλλά και άλλων γλωσσών

προγραμματισμού. Το Eclipse επίσης δίνει την δυνατότητα στον χρήστη να κατεβάσει πρόσθετα τα οποία επεκτείνουν τις δυνατότητες του προγράμματος δίνοντας στον χρήστη τα απαραίτητα εργαλεία να αναπτύξει κάθε είδους εφαρμογή [10].

Για την δημιουργία των γραφικών πρέπει πρώτα να εγκαταστήσουμε στο Eclipse Oxygen την επέκταση e(fx)clipse [15] για να μπορούμε να δημιουργήσουμε ένα JavaFX project. Για να το κάνουμε αυτό πρέπει από το μενού βοήθεια να επιλέξουμε την λήψη και εγκατάσταση προσθέτων και στην συνέχεια να επιλέξουμε την κατάλληλη αποθήκη [9] ώστε να πραγματοποιήσομε λήψη της επέκτασης. Στην παρακάτω εικόνα φαίνεται αναλυτικά το παράθυρο λήψης καθώς και οι επιλογές που δώσαμε.

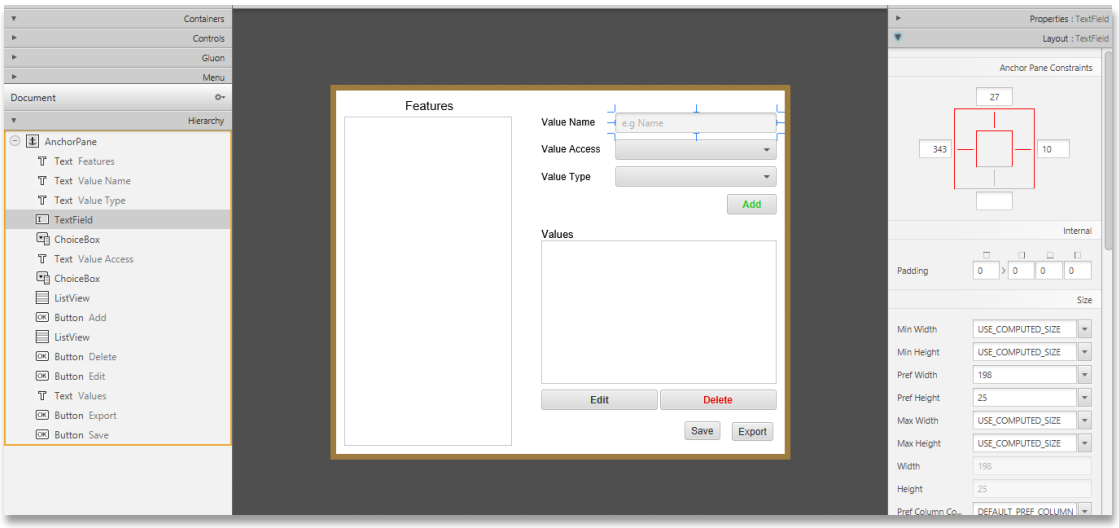


Εικόνα 17: Λήψη και εγκατάσταση e(fx)clipse.

Το JavaFX είναι μια πλατφόρμα λογισμικού για τη δημιουργία και την παροχή εφαρμογών γραφείου, καθώς και πλούσιες εφαρμογές διαδικτύου που μπορούν να λειτουργήσουν σε μεγάλη ποικιλία συσκευών. Το JavaFX προορίζεται να αντικαταστήσει την Swing ως την τυπική βιβλιοθήκη GUI για το Java SE. Το JavaFX υποστηρίζει χρήση σε επιτραπέζιους υπολογιστές και προγράμματα περιήγησης ιστού σε λειτουργικά συστήματα Microsoft Windows, Linux και macOS.

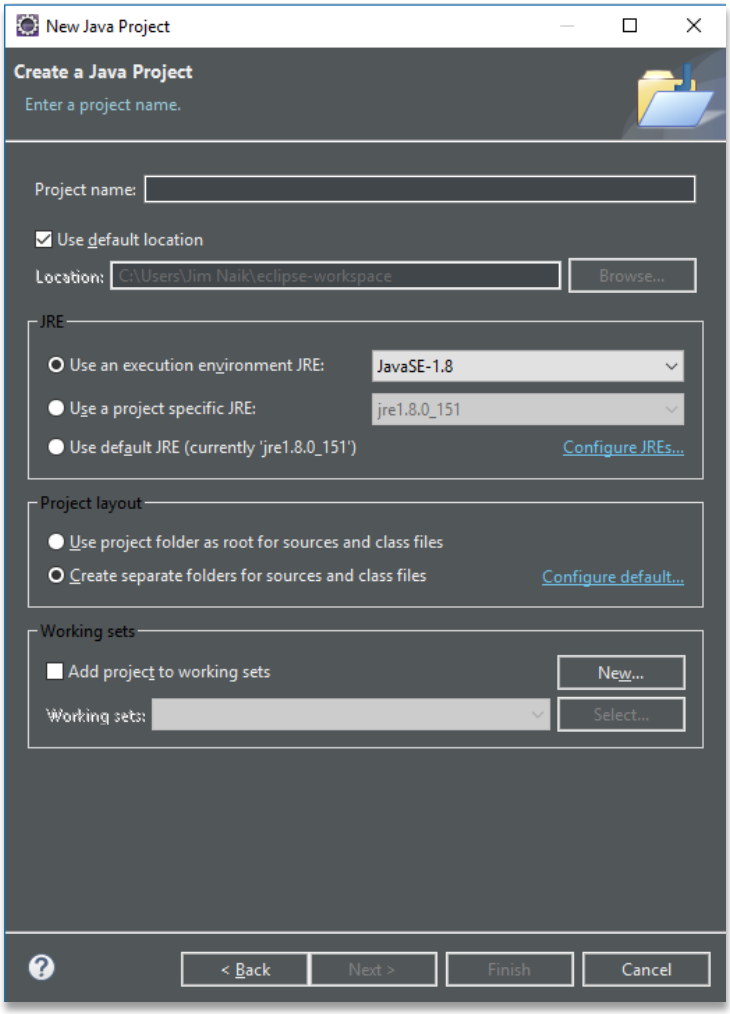
Στην συνέχεια κάνουμε λήψη και εγκατάσταση του SceneBuilder ώστε να μπορέσουμε να σχεδιάσουμε το γραφικό περιβάλλον της εφαρμογής μέσα από το συγκεκριμένο πρόγραμμα. Το SceneBuilder είναι μία εφαρμογή όπου

δουλεύει μαζί με το JavaFX περιβάλλον του Eclipse Oxygen ώστε να παράγει πλούσια σε περιεχόμενο γραφικά περιβάλλοντα που μπορούν να προβληθούν σε διαφορετικού τύπου συσκευές και λειτουργικά χωρίς κανένα πρόβλημα. Το παρακάτω στιγμιότυπο παρουσιάζει το γραφικό περιβάλλον του SceneBuilder καθώς και μία από τις οθόνες που δημιουργήθηκαν για την εφαρμογή.



Εικόνα 18: Στιγμιότυπο περιβάλλοντος SceneBuilder.

Για την δημιουργία της εφαρμογής δημιουργήθηκε ένα project σε JavaFX ώστε να υποστηρίζονται τα κατάλληλα γραφικά καθώς και ο τυπικός προγραμματισμός σε γλώσσα Java. Για την δημιουργία του JavaFX έργου επιλέγουμε την αντίστοιχη επιλογή από την δημιουργία νέου έργου και στην συνέχεια δίνουμε το επιθυμητό όνομα, επιλέγουμε την τοποθεσία που θέλουμε να αποθηκευτή και τέλος επιλέγουμε το JavaSE-1.8 ως επιθυμητό περιβάλλον εκτέλεσης της εφαρμογής. Η παρακάτω εικόνα δείχνει το στιγμιότυπο δημιουργίας ενός παρόμοιου project με αυτό που χρησιμοποιήθηκε για την δημιουργία της εφαρμογής.



Εικόνα 19: Δημιουργία JavaFX project.

Με την χρήση του SceneBuilder μπορούμε να δημιουργούμε γραφικά περιβάλλοντα τύπου fxml που χρησιμοποιούνται από την JavaFX με την χρήση του γραφικού περιβάλλοντος που είδαμε στην Εικόνα 18. Το γραφικό περιβάλλον αυτό δίνει την δυνατότητα στον χρήστη με μεθόδους drag and drop να δημιουργήσει γραφικά περιβάλλοντα που στην συνέχεια θα το SceneBuilder θα τα μεταφράσει σε κώδικα fxml ώστε να μπορέσουν να χρησιμοποιηθούν από το Eclipse Oxygen. Στην Εικόνα 20 φαίνεται ένα κομμάτι του κώδικα που δημιουργήθηκε από το γραφικό παράθυρο που σχεδιαζόταν στην Εικόνα 18.

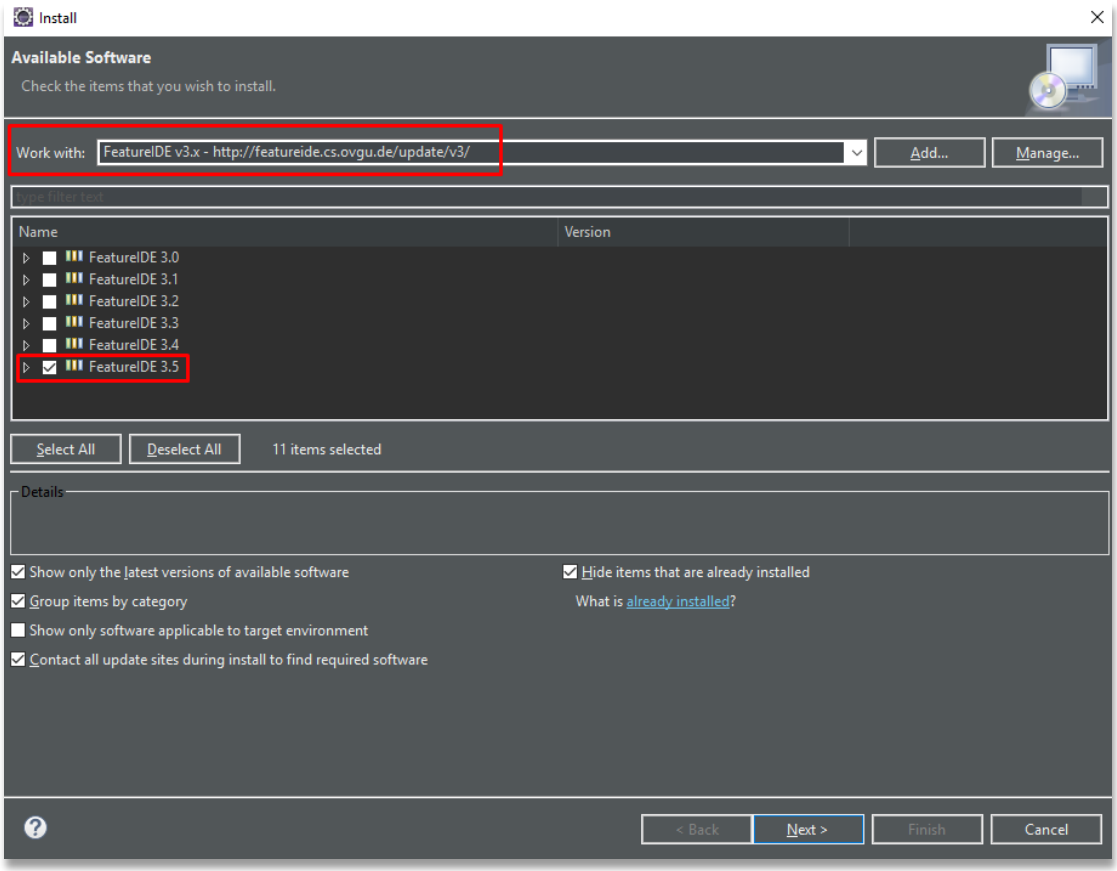
```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ChoiceBox?>
<?import javafx.scene.control.ListView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>

<AnchorPane minHeight="400.0" minWidth="550.0"
xmlns="http://javafx.com/javafx/9.0.1" xmlns:fx="http://javafx.com/fxml/1"
fx:controller="application.homePage">
   <children>
      <Text layoutX="85.0" layoutY="19.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Features" textAlignment="CENTER"
AnchorPane.bottomAnchor="419.033203125" AnchorPane.leftAnchor="85.0"
AnchorPane.topAnchor="10.0">
         <font>
            <Font name="Arial" size="15.0" />
         </font>
      </Text>
      <Text layoutX="252.0" layoutY="43.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Value Name" textAlignment="CENTER"
AnchorPane.leftAnchor="252.0" AnchorPane.topAnchor="32.13671875">
         <font>
            <Font name="Arial" size="12.0" />
         </font>
      </Text>
      <Text layoutX="255.0" layoutY="76.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Value Type" AnchorPane.leftAnchor="252.0"
AnchorPane.topAnchor="99.13671875">
         <font>
            <Font name="Arial" size="12.0" />
         </font>
      </Text>
      <TextField fx:id="valueName" layoutX="343.0" layoutY="27.0"
prefHeight="25.0" prefWidth="198.0" promptText="e.g Name"
AnchorPane.leftAnchor="343.0" AnchorPane.rightAnchor="10.0"
AnchorPane.topAnchor="27.0" />
      <ChoiceBox fx:id="dropListType" layoutX="343.0" layoutY="60.0"
prefWidth="150.0" AnchorPane.leftAnchor="343.0" AnchorPane.rightAnchor="10.0"
AnchorPane.topAnchor="60.0" />
      <Text layoutX="249.0" layoutY="76.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Value Access" AnchorPane.leftAnchor="252.0">
         <font>
            <Font name="Arial" size="12.0" />
         </font>
      </Text>
      <ChoiceBox fx:id="dropListAccess" layoutX="343.0" layoutY="93.0"
prefHeight="25.0" prefWidth="198.0" AnchorPane.leftAnchor="343.0"
AnchorPane.rightAnchor="10.0" AnchorPane.topAnchor="93.0" />
      <ListView fx:id="listFeatures" layoutX="14.0" layoutY="32.0"
onMouseClicked="#onFeatureSelect" prefHeight="399.0" prefWidth="206.0"
```

Εικόνα 20: Μέρος του κώδικα γραφικών της κεντρικής σελίδας της εφαρμογής.

3.3.2 Πρόγραμμα σχεδίασης μοντέλων χαρακτηριστικών FeatureIDE

Το FeatureIDE είναι ένα πλαίσιο ανοικτού κώδικα ενός IDE για τους μηχανικούς της γραμμής προϊόντων λογισμικού που βασίζεται στην ανάπτυξη λογισμικού με γνώμονα τα χαρακτηριστικά (Feature-Oriented Software Development). Το FeatureIDE υποστηρίζει ολόκληρο τον κύκλο ζωής μιας γραμμής προϊόντων λογισμικού σε μια συνεκτική υποδομή εργαλείων, ξεκινώντας με την ανάλυση χαρακτηριστικών και τη μοντελοποίηση χαρακτηριστικών [7][8], αλλά καλύπτει επίσης το σχεδιασμό, την υλοποίηση και τη συντήρηση. Σε αντίθεση με παλαιότερες εκδόσεις, το FeatureIDE δεν καλύπτει μόνο μία μόνο γλώσσα, αλλά αρκετές γλώσσες βασισμένες στην έννοια του FOSD.

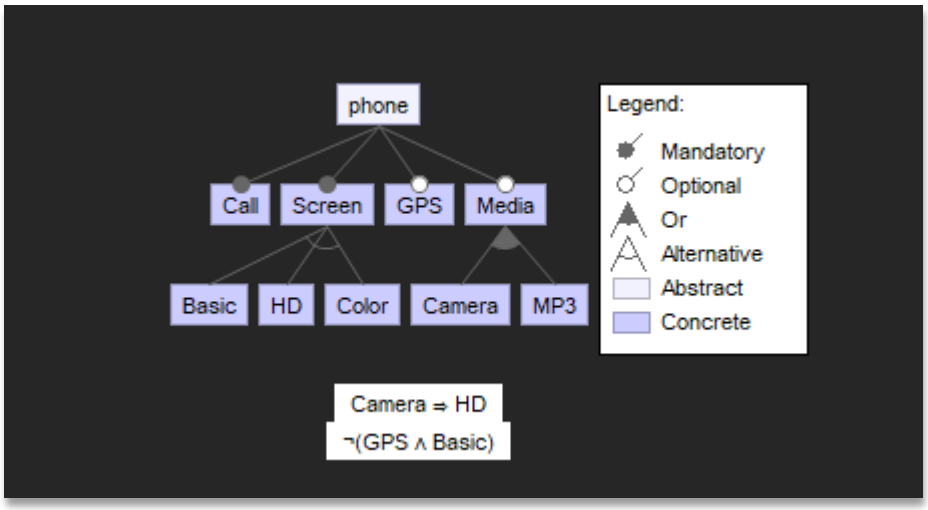
Για την εγκατάσταση του FeatureIDE θα ακολουθηθεί η διαδικασία που ακολουθήθηκε προηγουμένως πηγαίνοντας στην βοήθεια του Eclipse Oxygen και έπειτα λήψη και εγκατάσταση προσθέτων [14]. Στην συνέχεια επιλέγουμε την κατάλληλη αποθήκη ώστε να πραγματοποιήσομε λήψη της επέκτασης. Στην παρακάτω εικόνα φαίνεται αναλυτικά το παράθυρο λήψης καθώς και οι επιλογές που δώσαμε.



Εικόνα 21: Λήψη και εγκατάσταση FeatureIDE.

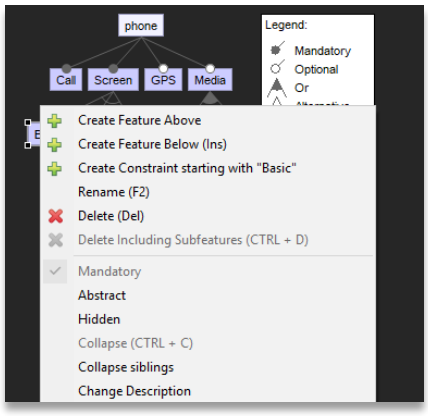
Μετά την εγκατάσταση του FeatureIDE μπορούμε να δημιουργήσουμε ένα μοντέλο χαρακτηριστικών από το μενού δημιουργίας νέου έργου. Κάνοντας το

παραπάνω μας ανοίγει ένα νέο περιβάλλον μέσα στο Eclipse Oxygen όπου επιτρέπει την κατασκευή και επεξεργασία μοντέλων χαρακτηριστικών. Το αρχείο του μοντέλου χαρακτηριστικών αποθηκεύεται σε μορφή XML. Στην παρακάτω εικόνα μπορούμε να δούμε το περιβάλλον λειτουργίας του FeatureIDE καθώς και ένα απλό μοντέλο χαρακτηριστικών που σχεδιάσαμε. Επίσης, μπορούμε να διακρίνουμε τις διαφορετικές σχέσεις που υποστηρίζονται από τα μοντέλα χαρακτηριστικών καθώς εμφανίζονται στο υπόμνημα που προσφέρει το γραφικό περιβάλλον. Κάτω από το μοντέλο χαρακτηριστικών μπορούμε να δούμε τους περιορισμούς που υπάρχουν στο συγκεκριμένο δέντρο χαρακτηριστικών καθώς και ότι εμφανίζονται με την μορφή προτασιακής λογικής όπως αναφέρθηκε και στο Κεφάλαιο 2 της παρούσας διπλωματικής εργασίας.



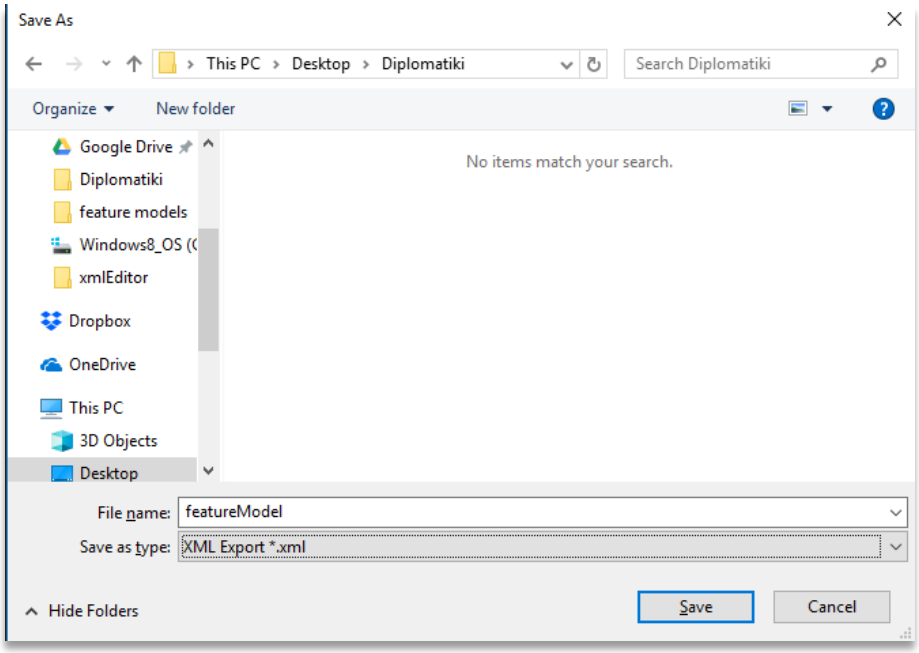
Εικόνα 22: Περιβάλλον FeatureIDE.

Για την εισαγωγή νέων χαρακτηριστικών στο δέντρο κάνουμε δεξί κλικ σε στο χαρακτηριστικό που θέλουμε να εισάγουμε ένα καινούριο παιδί ή γονέα και στην συνέχεια επιλέγουμε να εισάγουμε αντικείμενο από κάτω εάν θέλουμε να εισάγουμε ένα παιδί ή από επάνω εάν επιθυμούμε να εισάγουμε γονέα. Επίσης μας δίνεται η επιλογή να καθορίσουμε αν το συγκεκριμένο χαρακτηριστικό θα είναι υποχρεωτικό ή όχι.



Εικόνα 23: Εισαγωγή νέου χαρακτηριστικού στο δέντρο.

Τέλος, μπορούμε να εξάγουμε το συγκεκριμένο μοντέλο τόσο σε μορφή εικόνας ώστε να το εισάγουμε σε κάποια τεχνική αναφορά όσο και σε μορφή XML όπου θα χρησιμοποιηθεί έπειτα για την εισαγωγή των στοιχείων του μοντέλου χαρακτηριστικών στην εφαρμογή «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)». Για να μπορέσουμε να κάνουμε αποθήκευση σε έναν από τους δύο τύπους που προαναφέρθηκαν κάνουμε δεξί κλικ στην επιφάνεια εργασίας του FeatureIDE και έπειτα επιλέγουμε «εξαγωγή ως» (Export As). Στο αναδυόμενο παράθυρο όπου εμφανίζεται επιλέγουμε το μέρος όπου θέλουμε να το αποθηκεύσουμε το όνομα καθώς και τον τύπου του αρχείου και επιλέγουμε αποθήκευση. Στην Εικόνα 24 φαίνεται αναλυτικά η διαδικασία εξαγωγής ενός μοντέλου χαρακτηριστικών.



Εικόνα 24: Διαδικασία εξαγωγής μοντέλου χαρακτηριστικών σε XML

3.3.3 Αποθήκη μοντέλων χαρακτηριστικών S.P.L.O.T

Το S.P.L.O.T. (Software Product Lines Online Tools) είναι ένα σύστημα βασισμένο σε ιστοσελίδα που αναπτύχθηκε σε Java το οποίο χρησιμοποιεί μια μηχανή τύπου HTML για να δημιουργήσει διεπαφές με τους χρήστες που θέλουν να το χρησιμοποιήσουν. Καθώς το σύστημα είναι διαδικτυακό, διευκολύνει έντονα την ανταλλαγή γνώσεων και δεν απαιτεί λήψη ενημερώσεων λογισμικού. Το S.P.L.O.T. υποστηρίζεται από εξελιγμένες μηχανές ρύθμισης παραμέτρων και αποτελεσματικά αυτοματοποιημένα συστήματα συλλογιστικής [10]. Επίσης, το S.P.L.O.T προσφέρει ένα αρχείο με μοντέλα χαρακτηριστικών άλλων χριστών όπου διατίθενται ελεύθερα στο διαδίκτυο. Η εφαρμογή αυτήν χρησιμοποιήθηκε στην μελέτη και την σύγκριση διαφορετικών μοντέλων χαρακτηριστικών κατά την διάρκεια της διπλωματικής εργασίας με σκοπό την εξέταση διαφορετικών περιπτώσεων γραμμών προϊόντων λογισμικού.

4 Αρχιτεκτονική και Τεχνική ανάλυση του F.C.C.M.A

Στο συγκεκριμένο κεφάλαιο θα γίνει περιγραφή του τρόπου όπου σχεδιάστηκε το εργαλείο «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)» καθώς και ο τρόπος με τον οποίο υλοποιήθηκε. Ποιο αναλυτικά, θα γίνει περιγραφή των απαιτήσεων της εφαρμογής, θα αναλυθούν οι βασικές λειτουργίες του καθώς και οι περιπτώσεις χρήσης. Επιπροσθέτως, θα γίνει λεπτομερής τεχνική ανάλυση της εφαρμογής παρουσιάζοντας και εξηγώντας τα σημαντικότερα σημεία στον κώδικα καθώς και στα εξαγόμενα αρχεία.

4.1 Περιγραφή Απαιτήσεων και Αρχιτεκτονική Ανάλυση του F.C.C.M.A

4.1.1 Περιπτώσεις χρήσης

Η ανάλυση των περιπτώσεων χρήσης (use case analysis) ή ανάλυση ευρωστίας (robustness analysis) είναι μία μέθοδος για τον προσδιορισμό της συμπεριφοράς του συστήματος για την κάλυψη των απαιτήσεων που είναι καταγεγραμμένες στις περιπτώσεις χρήσης. Η ανάλυση των περίπτωσης χρήσης αποτελεί την κύρια μορφή για τη συλλογή των απαιτήσεων χρήσης για ένα νέο πρόγραμμα λογισμικού ή μια εργασία που πρέπει να ολοκληρωθεί. Οι κύριοι στόχοι μιας ανάλυσης περιπτώσεων χρήσης είναι: ο σχεδιασμός ενός συστήματος από την οπτική γωνία του χρήστη, η επικοινωνία της συμπεριφοράς του συστήματος με τους όρους του χρήστη και ο καθορισμός όλων των εξωτερικών ορατών συμπεριφορών. Ένα άλλο σύνολο στόχων για μια ανάλυση περιπτώσεων χρήσης είναι η σαφής επικοινωνία όπως για παράδειγμα οι απαιτήσεις του συστήματος, ο τρόπος χρήσης του συστήματος, οι ρόλοι που παίζει ο χρήστης στο σύστημα, το τι κάνει το σύστημα ως απάντηση στο ερέθισμα του χρήστη, τι λαμβάνει ο χρήστης από το σύστημα και ποια αξία θα λάβει ο πελάτης ή ο χρήστης από το σύστημα [13]. Στην παρούσα διπλωματική εργασία θα μελετηθούν οι περιπτώσεις χρήσης για την πλοήγηση του χρήστη στην εφαρμογή ως μηχανικός λογισμικού.

Οι εφαρμογή περιλαμβάνει τις παρακάτω οντότητες, όπως περιγράφονται και στο διάγραμμα κλάσεων:

- **Μηχανικός Λογισμικού:** Είναι ο κύριος χρήστης της εφαρμογής.
- **Προϊόν:** Είναι τα παραγόμενα στοιχεία της εφαρμογής. Αυτά τα στοιχεία είναι ο κώδικα σε Java και οι μετρικές συντηρησιμότητας όπου εξάγει το εργαλείο.

Η οντότητα που θα αναλυθεί στην συνέχεια είναι ο «Μηχανικός Λογισμικού» καθώς αποτελεί τον κύριο και μοναδικό χρήστη της εφαρμογής.

Οι λειτουργίες που εκτελούνται από τον «Μηχανικό Λογισμικού» είναι οι εξής:

- **Φόρτωση αρχείου μοντέλου χαρακτηριστικών**: Ο χρήστης δίνει στο εργαλείο τον προορισμό που βρίσκεται το αρχείο XML που είναι αποθηκευμένο το μοντέλο χαρακτηριστικών. Έπειτα με την εκκίνηση της διαδικασίας εισαγωγής του αρχείου το πρόγραμμα βρίσκει το αρχείο και το διαβάζει. Τέλος το πρόγραμμα μεταφέρει τον χρήστη

στην κεντρική σελίδα της εφαρμογής έχοντας εισάγει αυτόματα όλα τα χαρακτηριστικά που θα μετατραπούν σε κλάσεις σε αυτήν την σελίδα.

- **Φόρτωση αρχείου υπάρχοντος έργου:** Ο χρήστης δίνει στο εργαλείο τον προορισμό που βρίσκεται το αρχείο TXT που είναι αποθηκευμένο το μοντέλο χαρακτηριστικών μαζί με ότι αλλαγές έχει κάνει ο χρήστης. Έπειτα με την εκκίνηση της διαδικασίας εισαγωγής του αρχείου το πρόγραμμα βρίσκει το αρχείο και το διαβάζει. Τέλος το πρόγραμμα μεταφέρει τον χρήστη στην κεντρική σελίδα της εφαρμογής έχοντας εισάγει αυτόματα όλες τις κλάσεις αρχείων που είχαν δημιουργηθεί την πρώτη φοράς του μοντέλου χαρακτηριστικών στο σύστημα. Επιπροσθέτως, εισάγονται όλες οι μεταβλητές που τυχών είχε εισάγει ο χρήστης στο παρελθόν.

- **Εισαγωγή μεταβλητών:** Για την εισαγωγή μεταβλητών στις κλάσεις που παρήχθησαν από το μοντέλο χαρακτηριστικών ο χρήστης πρώτα πρέπει να έχει κάνει την εισαγωγή του απαραίτητου αρχείου στο σύστημα. Έπειτα, και αφού η διαδικασία αναγνώρισης των κλάσεων του συστήματος ολοκληρωθεί ο χρήστης μπορεί να επιλέξει την κλάση που θέλει από την λίστα των χαρακτηριστικών. Στην συνέχεια, ο χρήστης πρέπει να δώσει το όνομα της μεταβλητής (value name) να επιλέξει από την λίστα την προσβασιμότητα της μεταβλητής (value access)(private, public, global) καθώς και τον τύπο της μεταβλητής (value type) (String, char, int, float, double, long, boolean, List, ArrayList, byte, short). Τέλος, ο χρήστης επιλέγοντας την επιλογή προσθήκης νέας μεταβλητής το σύστημα εισάγει την μεταβλητή στην κλάση και η μεταβλητή εμφανίζεται στην λίστα μεταβλητών της κλάσης.

- **Τροποποίηση μεταβλητής:** Για την τροποποίηση μεταβλητών στις κλάσεις που παρήχθησαν από το μοντέλο χαρακτηριστικών ο χρήστης πρώτα πρέπει να έχει κάνει την εισαγωγή του απαραίτητου αρχείου στο σύστημα. Έπειτα, και αφού η διαδικασία αναγνώρισης των κλάσεων του συστήματος ολοκληρωθεί ο χρήστης μπορεί να επιλέξει την κλάση που θέλει από την λίστα των χαρακτηριστικών. Στην συνέχεια, αφού ο χρήστης επιλέξει την κλάση που θέλει να επεξεργαστεί, στη λίστα μεταβλητών φαίνονται οι μεταβλητές που υπάρχουν στην συγκεκριμένη κλάση. Ο χρήστης μπορεί να επιλέξει μία από αυτές τις μεταβλητές και στην συνέχεια να αλλάξει οποιαδήποτε από τις ιδιότητές όπου είναι το όνομα της μεταβλητής (value name) να επιλέξει από την λίστα την προσβασιμότητα της μεταβλητής (value access)(private, public, global) καθώς και τον τύπο της μεταβλητής (value type) (String, char, int, float, double, long, boolean, List, ArrayList, byte, short).

- **Διαγραφή μεταβλητής:** Για την διαγραφή μεταβλητών στις κλάσεις που παρήχθησαν από το μοντέλο χαρακτηριστικών ο χρήστης πρώτα πρέπει να έχει κάνει την εισαγωγή του απαραίτητου αρχείου στο σύστημα. Έπειτα, και αφού η διαδικασία αναγνώρισης των κλάσεων του συστήματος ολοκληρωθεί ο χρήστης μπορεί να επιλέξει την κλάση που θέλει από την λίστα των χαρακτηριστικών. Στην συνέχεια, αφού ο χρήστης επιλέξει την κλάση που θέλει να επεξεργαστεί, στη λίστα μεταβλητών φαίνονται οι μεταβλητές που υπάρχουν στην συγκεκριμένη κλάση. Ο χρήστης επιλέγει την μεταβλητή που θέλει να διαγράψει και στην συνέχεια επιλέγει την επιλογή της διαγραφής μεταβλητής (Delete) ώστε η μεταβλητή να αφαιρεθεί από την συγκεκριμένη κλάση.

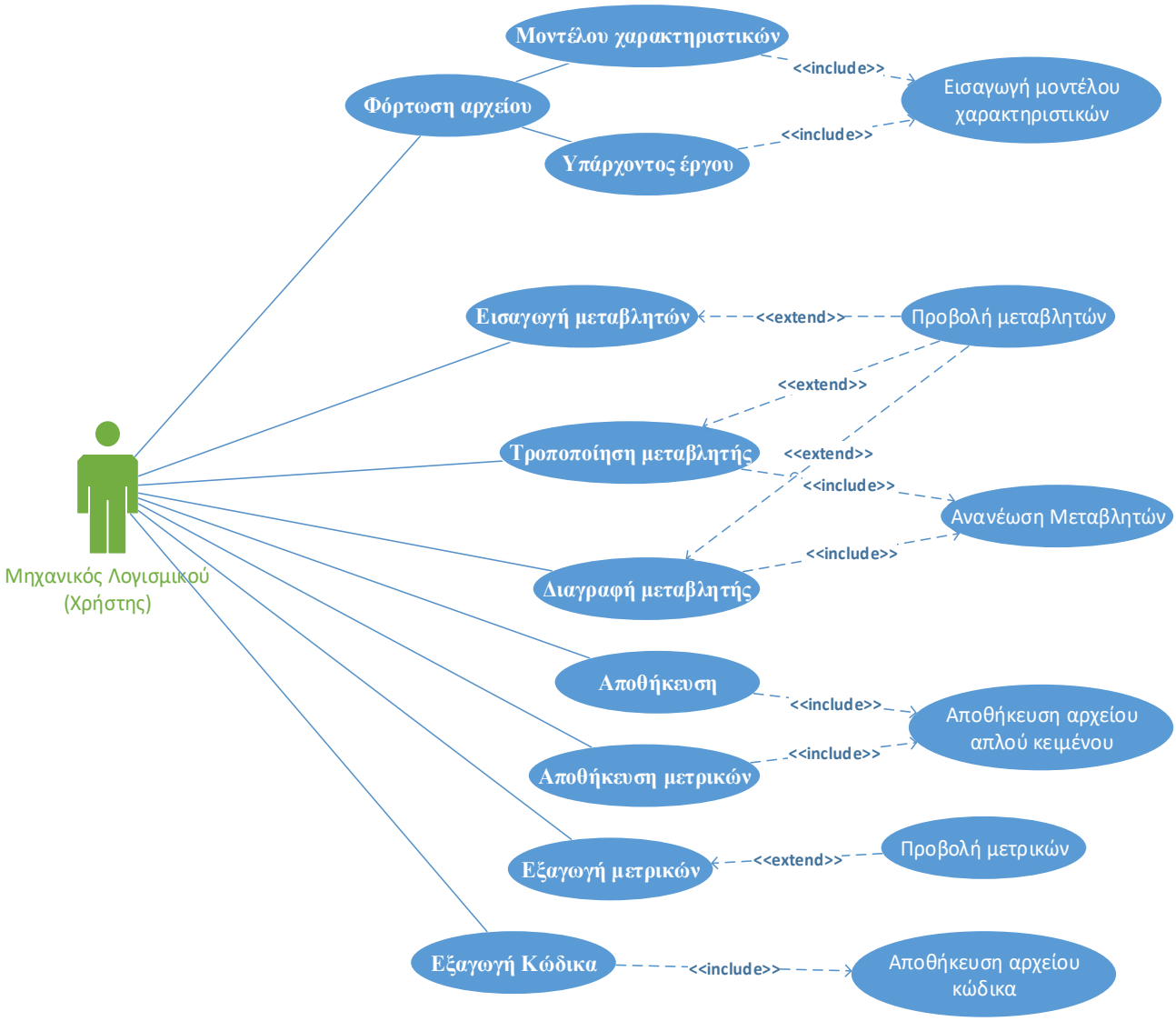
- **Αποθήκευση:** Για την αποθήκευση του χώρου εργασίας και όλων τον μεταβλητών που έχει εισάγει ο χρήστης πρώτα πρέπει να έχει κάνει την εισαγωγή του απαραίτητου αρχείου στο σύστημα. Έπειτα, και αφού η διαδικασία αναγνώρισης των κλάσεων του συστήματος ολοκληρωθεί η εφαρμογή ανακατευθύνει στον χρήστη στην κεντρική οθόνη της εφαρμογής. Από εκεί ο χρήστης μπορεί να επιλέξει την εντολή αποθήκευσης (Save) και το σύστημα να ξεκινήσει την διαδικασία αποθήκευσης. Με την ολοκλήρωση της διαδικασίας εμφανίζεται κατάλληλο μήνυμα στον χρήστη.

- **Εξαγωγή μετρικών:** Για την εξαγωγή μετρικών συντηρησιμότητας του μοντέλου χαρακτηριστικών του έργο που έχει αναπτύξει ο χρήστης πρώτα πρέπει να έχει κάνει την εισαγωγή του απαραίτητου αρχείου στο σύστημα. Έπειτα, και αφού η διαδικασία αναγνώρισης των κλάσεων του συστήματος ολοκληρωθεί η εφαρμογή ανακατευθύνει στον χρήστη στην κεντρική οθόνη της εφαρμογής. Από εκεί ο χρήστης επιλέγει την επιλογή ανάλυσης του μοντέλου χαρακτηριστικών. Η διαδικασία ανάλυσης του μοντέλου εκκινείτε και η εφαρμογή αφού ολοκληρωθεί η διεργασία εξαγωγής των αποτελεσμάτων μεταφέρει τον χρήστη στην οθόνη προβολής των μετρικών συντηρησιμότητας.

- **Αποθήκευση μετρικών:** Για την αποθήκευση μετρικών συντηρησιμότητας του μοντέλου χαρακτηριστικών του έργο που έχει αναπτύξει ο χρήστης πρώτα πρέπει να έχει κάνει την εισαγωγή του απαραίτητου αρχείου στο σύστημα. Έπειτα, και αφού η διαδικασία αναγνώρισης των κλάσεων του συστήματος ολοκληρωθεί η εφαρμογή ανακατευθύνει στον χρήστη στην κεντρική οθόνη της εφαρμογής. Από εκεί ο χρήστης επιλέγει την επιλογή ανάλυσης του μοντέλου χαρακτηριστικών. Η εφαρμογή ανακατευθύνει τον χρήστη στην κατάλληλη οθόνη όπου από εκεί επιλέγει αποθήκευση. Αφού ο χρήστης επιλέξει αποθήκευση η εφαρμογή αποθηκεύει τις μετρικές σε ένα αρχείο απλού κειμένου και ενημερώνει τον χρήστη.

- **Εξαγωγή Κώδικα:** Για την εξαγωγή κώδικα των κλάσεων που του μοντέλου χαρακτηριστικών και όλων τον μεταβλητών που έχει εισάγει ο χρήστης πρώτα πρέπει να έχει κάνει την εισαγωγή του απαραίτητου αρχείου στο σύστημα. Έπειτα, και αφού η διαδικασία αναγνώρισης των κλάσεων του συστήματος ολοκληρωθεί η εφαρμογή ανακατευθύνει στον χρήστη στην κεντρική οθόνη της εφαρμογής. Από εκεί ο χρήστης μπορεί να επιλέξει την εντολή εξαγωγής (Export) και το σύστημα να ξεκινήσει την διαδικασία παραγωγής του κώδικα και αποθήκευσης του σε κατάλληλα αρχεία Java. Με την ολοκλήρωση της διαδικασίας εμφανίζεται κατάλληλο μήνυμα στον χρήστη.

Στην Εικόνα 25 έχουμε το διάγραμμα περιπτώσεων χρήσης της εφαρμογής «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)». Στα αριστερά του διαγράμματος μπορούμε να δούμε τον χρήστη του συστήματος που είναι ο «Μηχανικός Λογισμικού». Στην αριστερή μεριά του διαγράμματος μπορούμε να δούμε με λεπτομέρεια τις λειτουργίες που μπορεί να εκτελέσει η κάθε οντότητα.



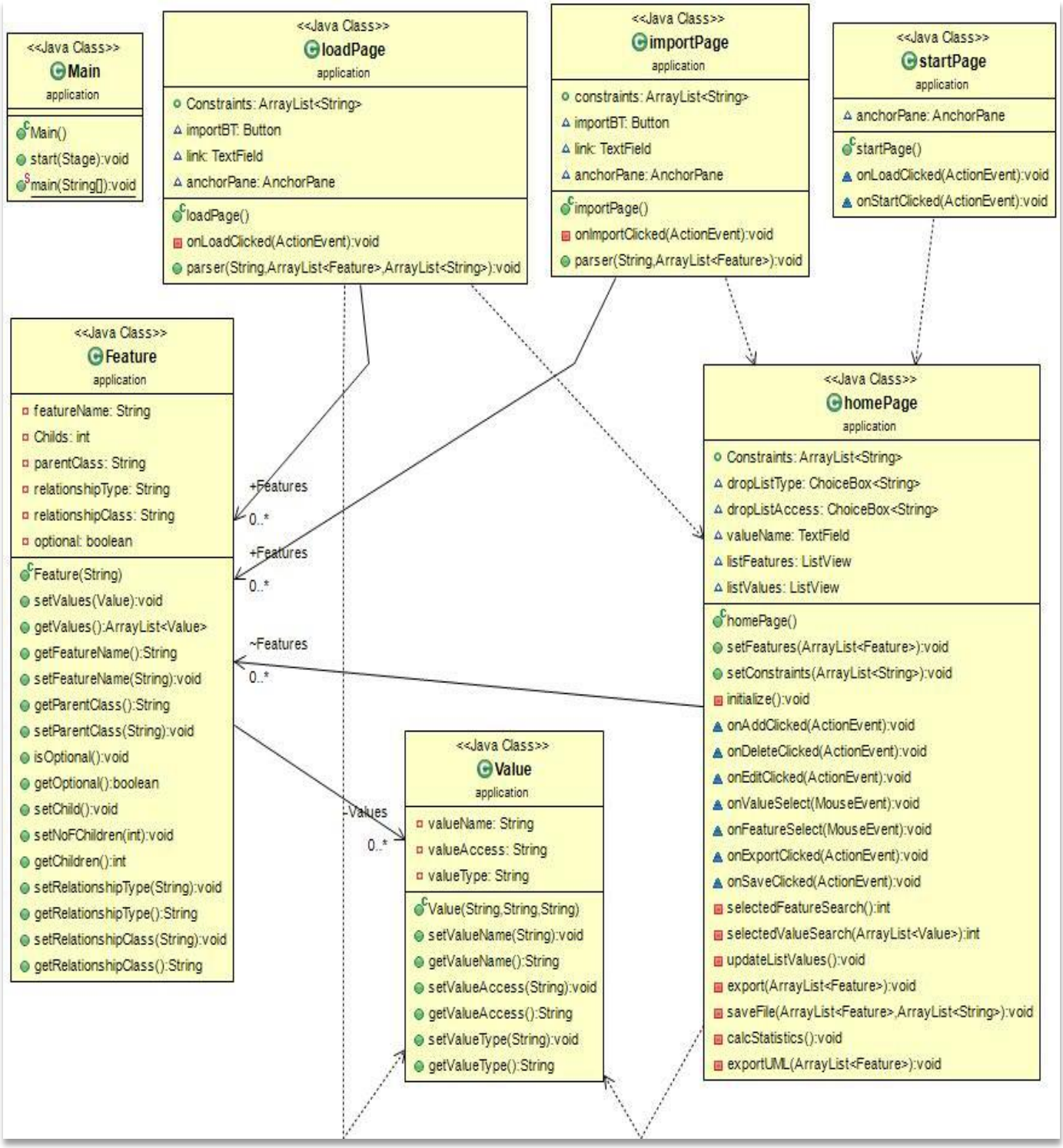
Εικόνα 25: Διάγραμμα περιπτώσεων χρήστη (use case diagram).

4.1.2 Διάγραμμα κλάσεων

Τα διαγράμματα κλάσεων χρησιμοποιούνται στην ανάλυση για την ανάδειξη των σημαντικότερων εννοιών του προβλήματος, όπως επίσης και στην σχεδίαση για να παρέχουν μία λεπτομερέστερη περιγραφή των βασικότερων στοιχείων του αντικειμενοστραφούς προγραμματισμού που είναι οι κλάσεις του εκάστοτε λογισμικού. Στην τεχνολογία λογισμικού, ένα διάγραμμα κλάσης στην Unified Modeling Language (UML) είναι ένας τύπος διαγράμματος στατικών δομών που περιγράφει τη δομή ενός συστήματος, δείχνοντας τις κλάσεις του συστήματος, τα χαρακτηριστικά του, τις λειτουργίες (ή τις μεθόδους) και τις σχέσεις μεταξύ των αντικειμένων. Το διάγραμμα είναι το κύριο δομικό στοιχείο της αντικειμενοστραφούς μοντελοποίησης. Χρησιμοποιείται για γενική εννοιολογική μοντελοποίηση του συστήματος της εφαρμογής και για λεπτομερή μοντελοποίηση μεταφράζοντας τα μοντέλα σε κώδικα προγραμματισμού. Τα διαγράμματα κλάσεων μπορούν επίσης να χρησιμοποιηθούν για τη μοντελοποίηση δεδομένων. Οι κλάσεις σε ένα

διάγραμμα κλάσεων αντιπροσωπεύουν τόσο τα κύρια στοιχεία, τις αλληλεπιδράσεις στην εφαρμογή, όσο και τις κατηγορίες που πρόκειται να προγραμματιστούν [13].

Στην Εικόνα 26 βλέπουμε το διάγραμμα κλάσεων που σχεδιάστηκε στα πλαίσια της δημιουργίας του εργαλείου «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)». Η υλοποίηση της εφαρμογής επιλέχθηκε να πραγματοποιηθεί με την βοήθεια της γλώσσας αντικειμενοστραφούς προγραμματισμού Java. Για τον λόγο ότι η Java έχει αντικειμενοστραφής φύση, κρίθηκε απαραίτητο να δημιουργηθεί ένα διάγραμμα με τις κλάσεις του συστήματος που θα χρειαστούν για την υλοποίησή του. Στο διάγραμμα αυτό αποτυπώνονται όλα τα στοιχεία και χαρακτηριστικά που απαρτίζουν την συγκεκριμένη εφαρμογή. Είναι εμφανές ότι στο διάγραμμα αυτό αποτυπώνονται όλα τα δημόσια (public) και ιδιωτικά (private) δεδομένα στοιχεία των κλάσεων καθώς και τις μεθόδους που υλοποιεί η κάθε κλάση. Επίσης, μπορούμε εύκολα να διακρίνουμε τις σχέσεις μεταξύ των διαφορετικών κλάσεων του συστήματος. Το «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)» αποτελείται από εφτά κλάσεις εκ των οποίων η μία είναι η κλάση main. Στο πάνω μέρος της κάθε κλάσεις απεικονίζονται οι μεταβλητές. Με κόκκινο τετράγωνο με σταυρό στην μέση διακρίνονται οι ιδιωτικές (private) μεταβλητές των κλάσεων ενώ με μπλε τρίγωνο και σταυρό στην μέση διακρίνονται οι δημόσιες (public). Στο κάτω μέρος της κάθε κλάσης απεικονίζονται οι μέθοδοι και οι συναρτήσεις. Οι δημόσιες (public) μέθοδοι της κάθε κλάσεις χαρακτηρίζονται με έναν πράσινο κύκλο ενώ με κόκκινο τετράγωνο χαρακτηρίζονται οι ιδιωτικές (private). Η κλάση «Feature» αποτελεί την κύρια κλάση αποθήκευσης της πληροφορίας κάθε μοντέλου χαρακτηριστικών αφού κάθε χαρακτηριστικό αποτελεί αντικείμενο αυτής της κλάσης. Τέλος, με μπλε τρίγωνο στο τμήμα τον συναρτήσεων χαρακτηρίζονται οι συναρτήσεις των γραφικών. Αυτές οι συναρτήσεις είναι που δίνουν την λειτουργικότητα στο γραφικό περιβάλλον και βάση αυτών καλούνται οι ιδιωτικές συναρτήσεις ώστε να γίνει η ανάλογη ενέργεια από το σύστημα.



Εικόνα 26: Διάγραμμα κλάσεων

4.1.3 Αρχιτεκτονική αρχείων εισόδου

Τα αρχεία εισόδου του προγράμματος αποτελούν ένα σημαντικό μέρος της διπλωματικής εργασίας καθώς πάνω στην δομή τους βασίζεται το πώς δημιουργούνται οι κλάσεις από τα μοντέλα χαρακτηριστικών. Το εργαλείο «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)» δέχεται ως είσοδο δύο διαφορετικούς τύπους αρχείων. Ο πρώτος τύπος είναι αρχεία Extensible Markup Language (XML) και παράγονται από το εργαλείο FeatureIDE ώστε να αποθηκεύουν την πληροφορία του μοντέλου χαρακτηριστικών. Ο δεύτερος τύπος αρχείων είναι αρχεία απλού κειμένου (txt) και εξάγονται από την ίδια την εφαρμογή κατά την διάρκεια της αποθήκευσης με σκοπό να βοηθήσουν τον χρήστη του συστήματος ώστε να συνεχίσει την εργασία του κάποια άλλη χρονική στιγμή. Επίσης με αυτόν τον τρόπο μπορεί να επιτευχθεί και η εύκολή μεταβίβαση του χώρου εργασίας από το ένα μέρος σε κάποιο άλλο είτε ακόμα και η αποστολή του συγκεκριμένου αρχείου σε τρίτους.

Στην Εικόνα 27 βλέπουμε στη δεξιά μεριά το ένα μοντέλο χαρακτηριστικών ενός απλού τηλεφώνου και στην αριστερή το αντίστοιχο παραγόμενο κώδικα XML. Στη σχηματική αναπαράσταση του μοντέλου χαρακτηριστικών μπορούμε να παρατηρήσουμε ότι υπάρχουν όλοι οι τύποι διαφορετικών συσχετίσεων μεταξύ των χαρακτηριστικών (mandatory, optional, or, alternative, required, excludes) [1][4][6]. Το μοντέλο χαρακτηριστικών αποτελείται από 10 χαρακτηριστικά εκ τον οποίον το ένα είναι η ρίζα του δέντρου χαρακτηριστικών. Κάτω από το μοντέλο μπορούμε να δούμε τους δύο διαφορετικούς περιορισμούς που υπάρχουν στο δέντρο εκφρασμένους σε σχεσιακή άλγεβρα. Μπορούμε εύκολα να διακρίνομε ότι το δέντρο έχει βάθος δύο καθώς εκτίνετε σε δύο επίπεδα μετά από την ρίζα. Στην αριστερή μεριά της εικόνας βλέπουμε τον παραγόμενο XML κώδικα και που αντιστοιχεί στο δέντρο αρχίζοντας με την δήλωση της έκδοσης της XML καθώς και της κωδικοποίησης στην αρχή του αρχείου. Έπειτα έχουμε την δήλωση αρχής του μοντέλου χαρακτηριστικών καθώς και το όνομα του. Βλέπουμε ότι η χρήσιμη πληροφορία που είναι το δέντρο χαρακτηριστικών δεν βρίσκετε σε μορφή XML και για αυτόν τον λόγο χρειάστηκε να αναπτυχθεί ειδική συνάρτηση ανάγνωσης του συγκεκριμένου τμήματος κώδικα ώστε να μπορέσουμε να αποκομίσουμε την χρήσιμη πληροφορία και να σχηματίσουμε τις απαραίτητες κλάσεις καθώς και να ορίσουμε και τις σχέσεις μεταξύ των κλάσεων. Η διαδικασία και ο τρόπος υλοποίησης της συγκεκριμένης συνάρτησης αναλύεται στην συνέχεια της διπλωματικής εργασίας με περισσότερες λεπτομέρειες. Παρόλο που τα χαρακτηριστικά του δέντρου δεν είναι σε μορφή XML βλέπουμε ότι τηρούν την διάταξη που είχαν και στο δέντρο χαρακτηριστικών. Επίσης στο τέλος του αρχείου βλέπουμε το τμήμα των περιορισμών που ακολουθούν την ίδια σειρά όπου είχαν και στο δέντρο χαρακτηριστικών. Το μοντέλο χαρακτηριστικό τελειώνει με την παρουσίαση όλων των χαρακτηριστικών καθώς και των περιορισμών του δέντρου.

Εικόνα 27: Μετατροπή μοντέλου χαρακτηριστικών σε XML.

Το δεύτερο αρχείο που δέχεται ως είσοδο στο σύστημα παρουσιάζεται στην Εικόνα 28 και είναι το αρχείο απλού κειμένου (txt) όπου περιέχει τις πληροφορίες του μοντέλου χαρακτηριστικών καθώς και τις πληροφορίες των μεταβλητών που έχει εισάγει ο χρήστης. Για να δημιουργηθεί και να εισαχθεί το προαναφερθείσα αρχείο στην εφαρμογή δημιουργήθηκαν ειδικές συναρτήσεις όπου η ανάπτυξη τους καθώς και η λειτουργικότητά τους αναλύετε εκτενέστερα στην συνέχεια της διπλωματικής εργασίας.

Παρατηρώντας το αρχείο στην Εικόνα 28 μπορούμε εύκολα να διακρίνουμε ότι κάθε χαρακτηριστικό ομαδοποιείται ώστε να έχουμε όλες τις πληροφορίες μαζί καθώς το σύστημα τις διαβάζει για να τις εισάγει στο κατάλληλο αντικείμενο κατά την εισαγωγή. Επίσης στο τέλος του αρχείου μπορούμε να δούμε ομαδοποιημένους όλους τους περιορισμούς του συγκεκριμένου μοντέλου χαρακτηριστικών ώστε να εισαχθούν στην κατάλληλη λίστα όταν εισάγουμε το αρχείο στο σύστημα.

```
Feature: Basic(Basic) Parent: Screen(Screen)
RelationshipType: Generalization RelationshipClass: Screen(Screen)
Optional: true
Children: 0
FeatureEnd
Feature: HD(HD) Parent: Screen(Screen)
RelationshipType: Generalization RelationshipClass: Screen(Screen)
Optional: true
Children: 0
FeatureEnd
Feature: Color(Color) Parent: Screen(Screen)
RelationshipType: Generalization RelationshipClass: Screen(Screen)
Optional: true
Children: 0
FeatureEnd
```

Εικόνα 28: Στιγμιότυπο αρχείου αποθήκευσης.

4.2 Τεχνική ανάλυση του F.C.C.M.A

Η εφαρμογή «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)» αναπτύχθηκε στο Eclipse Oxygen με την χρήση της αντικειμενοστραφούς γλώσσας προγραμματισμού Java. Σε αυτό το κομμάτι της διπλωματικής εργασίας θα αναλύσουμε τις σημαντικότερες λειτουργίες της εφαρμογής καθώς και θα δούμε σημαντικά κομμάτια κώδικα. Αρχικά αναλύουμε το πώς η εφαρμογή αναγνωρίζει τις κλάσεις και όλα τα στοιχεία από το μοντέλο χαρακτηριστικών. Στην συνέχεια, θα αναλυθεί η εσωτερικές λειτουργίες τις εφαρμογής που είναι η παραγωγή κώδικα και μετρήσεων σε σχέση με την συντηρησιμότητα του μοντέλου.

4.2.1 Εισαγωγή και αναγνώριση κλάσεων αρχείου μοντέλου χαρακτηριστικών

Για αναγνώριση των κλάσεων ο χρήστης εισάγει το αρχείο XML όπου περιέχει το μοντέλο χαρακτηριστικών στο σύστημα. Το αρχείο αυτό παράγεται από το FeatureIDE που είναι ένα πρόσθετο το Eclipse του οποίου την χρήση την μελετήσαμε σε προηγούμενα κεφάλαια. Για την ανάγνωση του συγκεκριμένου αρχείου κατασκευάστηκε μία μέθοδος όπου εκτελείται στην οθόνη εισαγωγής του της διαδρομής του αρχείου XML. Η συνάρτηση αυτήν ανοίγει το αρχείο και αφού αναγνωρίσει τα βασικά χαρακτηριστικά του αρχείου στην συνέχεια μετακινητέ στο κομμάτι της πληροφορίας. Όπως αναφέραμε και στο Κεφάλαιο 4.1.3 η χρήσιμη πληροφορία του μοντέλου χαρακτηριστικών δεν βρίσκεται σε μορφή XML οπότε χρειάστηκε να δημιουργηθούν ειδικές μέθοδοι για την εξαγωγή της χρήσιμης πληροφορίας. Για την αποθήκευση των χαρακτηριστικών του μοντέλου χρησιμοποιείται ένα ArrayList «Features» τύπου Feature όπου Feature είναι η κλάση όπου θα δημιουργείτε κάθε αντικείμενο χαρακτηριστικών.

Η κλάση «Feature» χρησιμοποιείτε ως η κύρια δομή αποθήκευσης πληροφορίας της εφαρμογής. Η κλάση αυτή εμπεριέχει εφτά μεταβλητές για την αποθήκευση των απαραίτητων πληροφοριών καθώς και όλες τις απαραίτητες συναρτήσεις. Η μεταβλητή «featureName» αποθηκεύει το όνομα του χαρακτηριστικού που θα είναι και το όνομα της κλάσης. Το ArrayList τύπου Value «Values» αποθηκεύει τις τιμές του συγκεκριμένου χαρακτηριστικού και χρησιμοποιείτε μόνο για την αποθήκευση των μεταβλητών που έχει εισάγει ο χρήστης στην κεντρική σελίδα της εφαρμογής για την συγκεκριμένη κλάση. Η μεταβλητή «Child» είναι μία μεταβλητή μέτρηση των παιδιών που έχει η συγκεκριμένη κλάση/χαρακτηριστικό και θα χρησιμοποιηθεί στην ανάλυση συντηρησιμότητας. Η μεταβλητή «parentClass» αποθηκεύει το όνομα του γονέα της συγκεκριμένης κλάσης εάν έχει γονέα η κλάση. Οι μεταβλητές «relationshipType» και «relationshipClass» αποθηκεύουν την σχέση της συγκεκριμένης κλάσης με κάποια άλλη και θα αναλυθεί στην συνέχεια πως αποφασίζεται αυτό. Τέλος, η μεταβλητή «optional» χρησιμοποιείται στην ανάλυση συντηρησιμότητας της εφαρμογής και μας δείχνει εάν το συγκεκριμένο χαρακτηριστικό είναι προαιρετικό ή όχι. Ο απόσπασμα του κώδικα της κλάσης «Feature» παρουσιάζεται στην Εικόνα 29.

```java
import java.util.ArrayList;

public class Feature {
	private String featureName;
	private ArrayList<Value> Values = new ArrayList<Value>();
	private int Child = 0;
	private String parentClass = null;
	private String relationshipType = null;
	private String relationshipClass = null;
	private boolean optional=false;

	public Feature(String name) {
		this.setFeatureName(name);
	}

	public void setValues(Value value) {
		Values.add(value);
	}

	public ArrayList<Value> getValues() {
		return Values;
	}

	public String getFeatureName() {
		return featureName;
	}

	public void setFeatureName(String featureName) {
		this.featureName = featureName;
	}

	public String getParentClass() {
		return parentClass;
	}

	public void setParentClass(String parentClass) {
		this.parentClass = parentClass;
	}

	public void isOptional() {
		this.optional=true;
	}

	public boolean getOptional() {
		return this.optional;
	}

	public void setChild () {
		Child++;
	}

	public void setNoFChildren (int i) {
		Child = i;
	}
```

Εικόνα 29: Απόσπασμα κώδικα κλάσης χαρακτηριστικών

Η συνάρτηση «parser» οπού εκτελείται για την εισαγωγή του μοντέλου χαρακτηριστικού δημιουργεί ένα αντικείμενο τύπου Scanner με όνομα «scanning» με σκοπό την ανάγνωση του αρχείου. Αρχικά θα χωρίζουμε το κείμενο του αρχείου ανά κενό και ανά αλλαγή γραμμής. Όπως φαίνεται στην Εικόνα 30 η χρήσιμη πληροφορία του αρχείου οργανώνεται μέσα στις ετικέτες «feature_tree» και «constraints». Το είδος του κάθε χαρακτηριστικού καθορίζεται από τέσσερα διαφορετικά σύμβολα. Το πρώτο σύμβολο το «r» καθορίζει την ρίζα του δέντρου, το δεύτερο το «m» καθορίζει εάν ένα χαρακτηριστικό είναι υποχρεωτικό, το τρίτο σύμβολο το «o» καθορίζει εάν το χαρακτηριστικό είναι προαιρετικό και τέλος το τέταρτο σύμβολο το «g» καθορίζει όταν έχουμε ένα σύνολο χαρακτηριστικών είτε τύπου «or» εάν έχουμε μπορούμε να επιλέξουμε πάνω από ένα [1, *], είτε τύπου «Alternative» εάν μπορούμε να επιλέξουμε μόνο μία εναλλακτική [1, 1].

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
    <feature_model name="FeatureIDE model">
        <feature_tree>
            :r phone(phone)
                :m Call(Call)
                :m Screen(Screen)
                    :g [1,1]
                        : Basic(Basic)
                        : HD(HD)
                        : Color(Color)
                :o GPS(GPS)
                :o Media(Media)
                    :g [1,*]
                        : Camera(Camera)
                        : MP3(MP3)
        </feature_tree>
        <constraints>
            C1:~Camera or HD
            C2:~GPS or ~Basic
        </constraints>
    </feature_model>
```

Εικόνα 30: Αρχείο μοντέλου χαρακτηριστικών με μη εκτυπώσιμους χαρακτήρες.

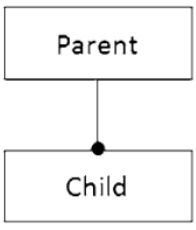
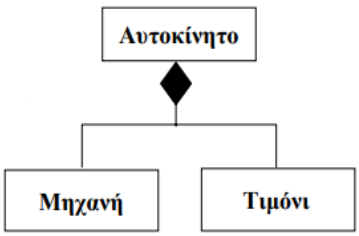
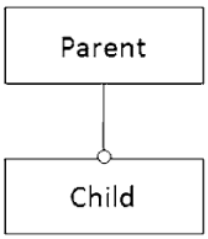
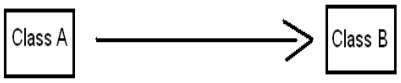
Βλέπουμε ότι ο μη εμφανίσιμος λευκός χαρακτήρας της εσοχής χρησιμοποιείται για την διάταξη του δέντρο και παριστάνεται με κίτρινο βέλος. Όπου μία εσοχή ποιο μέσα σημαίνει ότι το συγκεκριμένο χαρακτηριστικό έχει για γονέα το αμέσως προηγούμενο χαρακτηριστικό που είχε μία λιγότερη εσοχή από αυτό. Για την εξασφάλιση της κληρονομικότητας έπρεπε πρώτα να αποθηκεύουμε σε μία δομή δεδομένων τύπου ArrayList όλα τα χαρακτηριστικά που διαβάζει η εφαρμογή και να αφαιρεί από αυτήν την λίστα το τελευταίο χαρακτηριστικό όταν το επόμενο χαρακτηριστικό για εισαγωγή έχει τόσες εσοχές από μπροστά όσα και το προηγούμενο ή λιγότερες. Αυτό γίνεται γιατί σε περίπτωση που διαβαστεί χαρακτηριστικό με τον ίδιο ή μικρότερο αριθμό εσοχών από το προηγούμενο βάση διάταξης σημαίνει ότι αυτό το χαρακτηριστικό δεν μπορεί να έχει ως γονέα το προηγούμενο και επίσης βάση

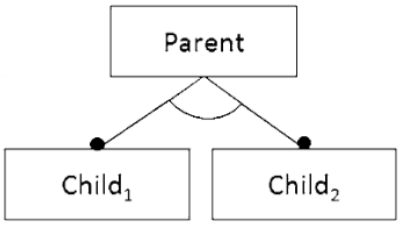
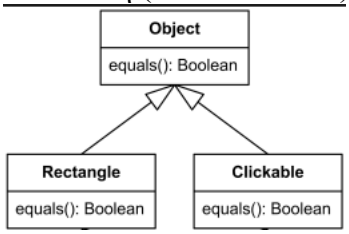
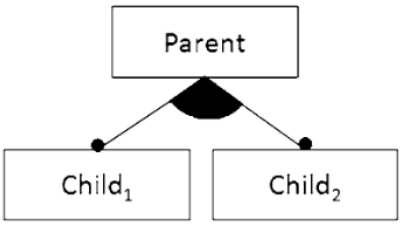
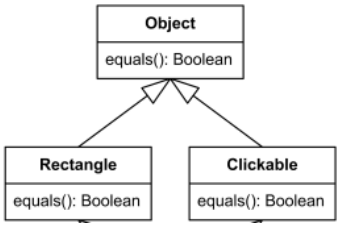
διάταξης γνωρίζουμε ότι και τα επόμενα χαρακτηριστικά που θα εισαχθούν στο πρόγραμμα σε περίπτωση που έχουν περισσότερες εσοχές θα έχουν ως γονέα το χαρακτηριστικό που το πρόγραμμα διάβασε τώρα και όχι το προηγούμενο. Στην Εικόνα 31 υπάρχει απόσπασμα του κώδικα που μετράει τα κενά καθώς και του κώδικα που αφαιρεί τα χαρακτηριστικά που δεν χρειάζονται πλέον από την δομή δεδομένων.

```
for (int i=0; i<temp.length(); i++) {
      if (temp.toCharArray()[i] == '   ') {
            tabCount++;
      }
}
//remove unnecessary parents from the list
if (parentsSave.size() != 0 && oldTabCount >= tabCount && flag) {
      delFeatures=0;
      for(int i=parentsSave.size()-1; i>=tabCount-3;i--) {
                              parentsSave.remove(i);
      }

}
```

Εικόνα 31: Κώδικας μέτρησης εσοχών και διαγραφής αχρείαστων χαρακτηριστικών.

Το είδος της σχέσης που θα ενώνει τον γονέα με τα παιδιά καθορίζεται από τον τύπο της σχέσης που είχε από πριν αυτήν η σχέση στο μοντέλο χαρακτηριστικών [5]. Επομένως η μεταφορά των σχέσεων αυτών από σχέσεις μοντέλων χαρακτηριστικών σε σχέσεις διαγράμματος κλάσεων γίνεται σύμφωνα με τον παρακάτω πίνακα. Στην αριστερή μεριά παρουσιάζονται οι σχέσεις μορφή μοντέλου χαρακτηριστικών και στην δεξιά μεριά οι σχέσεις σε μορφή διαγράμματος κλάσεων.

Μοντέλο Χαρακτηριστικών	**Διάγραμμα Κλάσεων**
Υποχρεωτική (Mandatory) 	Σύνθεση (Composition) 
Προαιρετική (Optional) 	Σύνδεση (Association) 

Εναλλακτική (Alternative) 	Γενίκευση (Generalization) 
Διάζευξη (ή) (Or) 	Γενίκευση (Generalization) 

Οπότε βάση του παραπάνω πίνακα είναι εύκολο να διακρίνουμε την μετατροπή των τεσσάρων σχέσεων από τα μοντέλα χαρακτηριστικών στις τρείς σχέσεις στα διαγράμματα κλάσεων. Επομένως στον κώδικα της εφαρμογής θα ορίζουμε την κατάλληλη σχέση μεταξύ γονέα και παιδιού ανάλογα με τον τύπο του χαρακτηριστικού που εισάγουμε. Επίσης εισάγουμε στην κάθε νέα κλάση που δημιουργούμε και τον γονέας της και επιλέγουμε εάν είναι προαιρετική η όχι ανάλογα με το είδος του χαρακτηριστικού που εισάγουμε. Στην Εικόνα 32 φαίνεται κομμάτι κώδικα από την μετατροπή των διαφορετικών χαρακτηριστικών σε κλάσεις. Όλα τα δεδομένα αποθηκεύονται σε ένα αντικείμενο «feature» τύπου «Feature» μέσο τον κατάλληλων συναρτήσεων που διαθέτη η συγκεκριμένη κλάση. Αυτό γίνεται ώστε να έχουμε εύκολη πρόσβαση σε όλα τα δεδομένα. Έπειτα μετά την εισαγωγή όλων το στοιχείων του μοντέλου χαρακτηριστικού στο αντικείμενο, αυτό αποθηκεύεται σε ένα ArrayList ώστε να γίνει ποιο εύκολη η μεταφορά των χαρακτηριστικών από την οθόνη εισαγωγής στην κύρια οθόνη του προγράμματος αλλά και να μπορούμε να τα επεξεργαστούμε ποιο εύκολα. Το παρακάτω στιγμιότυπο δείχνει την εισαγωγή των υποχρεωτικών, των προαιρετικών και του πρώτου στοιχείου μίας ομάδας (εναλλακτική σχέση ή διάζευξη (ή)).

```
else if(temp.equals(":m")) {
                        //save the parent
      parent = scanning.next();
      featureCount++;
      Feature feature = new Feature(parent);
      feature.setRelationshipType("Composition");
      feature.setRelationshipClass(parentsSave.get(tabCount-4));
      Features.add(feature);
      parentsSave.add(parent);
}
else if(temp.equals(":o")) {
      //save the parent
      parent = scanning.next();
      featureCount++;
      Feature feature = new Feature(parent);
      feature.setRelationshipType("Association");
      feature.setRelationshipClass(parentsSave.get(tabCount-4));
      feature.isOptional();
      Features.add(feature);
      parentsSave.add(parent);
}
else if(temp.equals(":g")) {
      //take the scanner in the last position of the :g line so the next
scan will be :
      //this procedure will skip all the useless for out class
construction information
      temp =scanning.next().replaceAll("\\s+","");
      while (!(temp.equals(":"))) {
            temp =scanning.next().replaceAll("\\s+","");
      }
      parent = scanning.next();
      featureCount++;
      Feature feature = new Feature(parent);
      feature.setParentClass(parentsSave.get(tabCount-4));
      feature.setRelationshipType("Generalization");
      feature.setRelationshipClass(parentsSave.get(tabCount-4));
      feature.isOptional();
      Features.add(feature);
      parentsSave.add(parent);
}
```

Εικόνα 32: Παράδειγμα κώδικα εισαγωγής χαρακτηριστικών.

Για την εισαγωγή των περιορισμών στο σύστημα γίνεται ειδικός έλεγχος όπου όταν διαβαστεί η ετικέτα «<constraints>» από το σύστημα τότε με την χρήση μίας μεταβλητής που λειτουργεί σαν διακόπτης μεταφέρουμε την ροή του προγράμματος στο τμήμα αναγνώρισης περιορισμών. Εκεί γίνεται έλεγχος εάν έχουμε φτάσει στο τέλος των περιορισμών σε περίπτωση που το πρόγραμμα διαβάσει την κατάλληλη ετικέτα, αλλιώς εισάγει τους περιορισμούς στο σύστημα αποθηκεύοντάς τους σε ένα ArrayList τύπου String. Με τον τερματισμό της διαδικασίας αναγνωρίσεις περιορισμών το σύστημα αναθέτει τον αριθμό τον παιδιών που έχει η κάθε κλάση ώστε να χρησιμοποιηθεί στην συνέχεια στην ανάλυση μετρικών διατηρησιμότητας και έπειτα τερματίζει η συνάρτηση εισαγωγής. Οι ο κώδικας των δύο αυτό διεργασιών προβάλετε στην

Εικόνα 33 του εγγράφου. Μετά την ολοκλήρωση της διαδικασίας το πρόγραμμα αποστέλλει τα δεδομένα που συλλέχθηκαν στην κύρια οθόνη της εφαρμογής και μεταφέρει και τον χρήστη στην συγκεκριμένη οθόνη.

```
while(flag2) {
	temp = scanning.next();
	if (temp.replaceAll("\\s+","").equals("</constraints>")) {
		flag2=false;
	}
	else {
		String constraint = temp.substring(temp.indexOf(":")+1);
		constraint.trim();
		constraints.add(constraint);
	}
}
//set children in parents
for(int i=0; i<Features.size()-1;i++) {
	for(int j=i; j<Features.size();j++) {
		if(Features.get(i).getFeatureName().equals(Features.get(j).get
		ParentClass())) {
		Features.get(i).setChild();
		}
	  }
}
```

Εικόνα 33:Κώδικας εισαγωγής περιορισμών και πλήθος παιδιών.

4.2.2 Εισαγωγή, τροποποίηση και διαγραφή μεταβλητών στις κλάσεις

Για την εισαγωγή μεταβλητών στις κλάσεις που δημιουργήθηκαν από το μοντέλο χαρακτηριστικών ο χρήστης πρέπει να βρίσκεται στην κεντρική οθόνη της εφαρμογής. Έπειτα μέσω ειδικής συνάρτησης γίνεται σύνδεση του γραφικού περιβάλλοντος με των κώδικα ώστε αφού ο χρήστης δώσει τις απαραίτητες πληροφορίες το σύστημα να μπορεί να τις αποθηκεύσει.

Για την αποθήκευση και διαχείριση αυτών των πληροφοριών χρησιμοποιείται το ArrayList Values που είναι τύπου Value. Αυτή η κλάση περιέχει τα τρία χαρακτηριστικά της κάθε μεταβλητής που είναι το όνομά της, ο τύπος της και ο βαθμός προσβασιμότητας της καθώς και τις κατάλληλες συναρτήσεις για την διαχείριση αυτών των μεταβλητών. Για να γίνει η εισαγωγή μεταβλητής ψάχνουμε και ανακτούμε τον δείκτη του επιλεγμένου χαρακτηριστικού στο ArrayList με σκοπό να εισάγουμε στην συνέχεια σε αυτό το αντικείμενο της μεταβλητής που θα δημιουργήσουμε. Έπειτα, δημιουργούμε το αντικείμενο σύμφωνα με τις πληροφορίες που έδωσε ο χρήστης και το αποθηκεύουμε στο ArrayList Values του χαρακτηριστικού που έχουμε επιλέξει. Τέλος καλούμε την συνάρτηση ανανέωσης του γραφικού περιβάλλοντος όπου εμφανίζονται οι μεταβλητές ώστε να εμφανιστεί η συγκεκριμένη μεταβλητή στον χρήστη. Στην Εικόνα 34 μπορούμε να δούμε τον κώδικα όπου υλοποιεί την συγκεκριμένη διεργασία.

```
@FXML
    void onAddClicked(ActionEvent event) {
            System.out.println("AddClicked");
            //We take the value of the index of the feature in arrayList
            int index = selectedFeatureSearch();
            //We take the information given by the user in GUI
            String type = dropListType.getValue();
            String access = dropListAccess.getValue();
            String name = valueName.getText();
            //Make an Object Value and add it to the Values arrayList in-
            side the feature Object
            Value value = new Value(name, access, type);
            this.Features.get(index).setValues(value);
            updateListValues();
    }
```

Εικόνα 34: Κώδικας εισαγωγής μεταβλητής.

Για την τροποποίηση των μεταβλητών καλείται η συνάρτηση τροποποίησης όπου ανανεώνει παίρνει τις ανανεωμένες τιμές που έχει δώσει ο χρήστης στο γραφικό περιβάλλον της εφαρμογής. Έπειτα, ανανεώνει τις υπάρχουσες τιμές στο αντικείμενο Values. Για να γίνει αυτό ψάχνουμε και ανακτούμε τον δείκτη του επιλεγμένου χαρακτηριστικού στο ArrayList με σκοπό να τροποποιήσουμε στην συνέχεια από αυτό το αντικείμενο τις μεταβλητές που θα έχει τροποποιήσει ο χρήστης. Τέλος καλούμε την συνάρτηση ανανέωσης του γραφικού περιβάλλοντος όπου εμφανίζονται οι μεταβλητές ώστε να εμφανιστεί η συγκεκριμένη μεταβλητή στον χρήστη. Στην Εικόνα 35 μπορούμε να δούμε τον κώδικα όπου υλοποιεί την συγκεκριμένη διεργασία.

```
@FXML
void onEditClicked(ActionEvent event) {
      System.out.println("EditClicked");
      //We take the values of the indexes of the feature and value in
      arrayLists
      int index = selectedFeatureSearch();
      int index2 =
      selectedValueSearch(this.Features.get(index).getValues());
      We take the information given by the user in GUI and we set in the
      value object
      this.Features.get(index).getValues().get(index2).setValueType(dropLi
      stType.getValue());
      this.Features.get(index).getValues().get(index2).setValueAccess(drop
      ListAccess.getValue());
      this.Features.get(index).getValues().get(index2).setValueName(valueN
      ame.getText());
      updateListValues();
}
```

Εικόνα 35: Κώδικας ανανέωσης μεταβλητής.

Για την διαγραφή μίας μεταβλητής πρέπει να επιλέξουμε την κλάση όπου θέλουμε για να ανακτήσουμε τον δείκτη του επιλεγμένου χαρακτηριστικού στο ArrayList με σκοπό να διαγράψομε το αντικείμενο της μεταβλητής που θέλουμε στην συνέχεια. Έπειτα, καλείτε η συνάρτηση διαγραφή όπου βρίσκει και διαγράφη την συγκεκριμένη μεταβλητή που επιλέξαμε.

4.2.3 Αποθήκευση εργασιακού περιβάλλοντος

Για την αποθήκευση του περιβάλλοντος εργασίας του χρήστη ώστε να μπορέσει να συνεχίσει κάποια άλλη χρονική στιγμή δημιουργήθηκε κατάλληλη συνάρτηση όπου αποθηκεύει τόσο το μοντέλο χαρακτηριστικών όσο και τις μεταβλητές σε ένα αρχείο απλού κειμένου με συγκεκριμένη μορφή. Για την κλίση αυτής της συνάρτησης ο χρήστης πρέπει να πατήσει το κουμπί «Save» από το γραφικό περιβάλλον. Έπειτα, η συνάρτηση αποθήκευσης καλείται όπου παίρνει ως ορίσματα το ArrayList Features όπου είναι αποθηκευμένες οι κλάσεις με τις μεταβλητές και το ArrayList Constraints όπου είναι αποθηκευμένοι οι περιορισμοί. Η συνάρτηση δημιουργεί ένα αρχείο απλού κειμένου με όνομα «saveFile.txt» και κωδικοποίησης UTF-8. Έπειτα με μία δομή επανάληψης εισάγουμε τα στοιχεία για όλα τα χαρακτηριστικά/κλάσεις στο αρχείο και στην συνέχεια με μία δεύτερη δομή επανάληψης εισάγουμε όλους τους περιορισμούς στο τέλος του αρχείου. Στιγμιότυπο του αρχείου αποθήκευσης καθώς και της δομής του μπορούμε να δούμε στην Εικόνα 36. Η δομή του αρχείου έγινε έτσι ώστε να εξυπηρετεί την εύκολη αποθήκευση και ανάκτηση των δεδομένων. Επίσης η αποθήκευση του μοντέλου χαρακτηριστικών με τις μεταβλητές σε εξωτερικό αρχείο δίνει την δυνατότητα για την εύκολη μετακίνηση και αποστολή του αρχείου. Ο κώδικας της συγκεκριμένης διαδικασίας φαίνεται στην Εικόνα 37.

```
Feature: Media(Media) Parent: null
RelationshipType: Association RelationshipClass: noParent
Optional: true
Children: 2
FeatureEnd
Feature: Camera(Camera) Parent: Media(Media)
RelationshipType: Generalization RelationshipClass: Media(Media)
Optional: true
Children: 0
FeatureEnd
Feature: MP3(MP3) Parent: Media(Media)
RelationshipType: Generalization RelationshipClass: Media(Media)
Optional: true
Children: 0
FeatureEnd
Constraint: ~Camera  or  HD
ConstraintEnd
Constraint: ~GPS  or  ~Basic
ConstraintEnd
```

Εικόνα 36: Στιγμιότυπο αρχείου αποθήκευσής με περιορισμούς και κλάσεις.

```
//save Function
      private void saveFile(ArrayList<Feature> features, ArrayList<String>
      constraints) throws IOException, UnsupportedEncodingException {
            PrintWriter writer = new PrintWriter("saveFile.txt","UTF-8");
          for (Feature fm : features) {
             writer.println("Feature: "+fm.getFeatureName()+" Parent:
             "+fm.getParentClass());
             writer.println("RelationshipType: "+fm.getRelationshipType()+"
             RelationshipClass: "+fm.getRelationshipClass());
             writer.println("Optional: "+fm.getOptional());
             writer.println("Children: "+fm.getChildren());
             for (Value fm2 : fm.getValues()) {
                   writer.println("Variable: ");
                   writer.println("Access: "+fm2.getValueAccess()+" Type:
                   "+fm2.getValueType()+" Name: "+fm2.getValueName());
             }
             writer.println("FeatureEnd ");
          }
          for (String con : constraints) {
             writer.println("Constraint: "+ con);
             writer.println("ConstraintEnd ");
          }
          writer.close();
      }
```

Εικόνα 37: Κώδικας δημιουργίας αρχείου αποθήκευσης.

4.2.4 Εξαγωγή στοιχείων ανάλυσης συντηρησιμότητας

Για την εξαγωγή στοιχείων ανάλυσης συντηρησιμότητας ο χρήστης πρέπει να βρίσκεται στην κεντρική οθόνη της εφαρμογής και να έχει εισάγει ένα μοντέλο χαρακτηριστικών στην εφαρμογή. Για την εξαγωγή των στοιχείων συντηρησιμότητας της εφαρμογής δεν απαιτείται από τον χρήστη να έχει εισάγει μεταβλητές στις κλάσης που δημιουργήθηκαν. Με την κατάλληλη επιλογή από το γραφικό περιβάλλον του συστήματος ενεργοποιείται η συνάρτηση εξαγωγή στατιστικών. Έπειτα η συνάρτηση παίρνει ως ορίσματα το ArrayList των χαρακτηριστικών και το ArrayList των περιορισμών ώστε να εξάγει τις διάφορες μετρικές όπου αναλύθηκαν στο Κεφάλαιο 3.1.2 της παρούσας διπλωματικής εργασίας.

Για την εξαγωγή της μέτρησης του πλήθους των χαρακτηριστικών του μοντέλου χαρακτηριστικών εισάγουμε στην κατάλληλη μεταβλητή το πλήθος των στοιχείων του ArrayList Features με την χρήση της συνάρτησης size() που υποστηρίζεται για ArrayList. Έτσι μπορούμε να βρούμε το πλήθος χωρίς να χρειάζεται να κάνουμε κάποια δομή επανάληψης.

Για την εξαγωγή της κυκλικής πολυπλοκότητας εισάγουμε στην κατάλληλη μεταβλητή το πλήθος των στοιχείων του ArrayList Constraints με την χρήση της συνάρτησης size() που υποστηρίζεται για ArrayList. Έτσι μπορούμε να βρούμε το πλήθος χωρίς να χρειάζεται να κάνουμε κάποια δομή επανάληψης.

Για την εύρεση του συντελεστή συνδεσιμότητας-πυκνότητας προσθέτουμε το μέγεθος του ArrayList των χαρακτηριστικών με το ArrayList των περιορισμών

και το διαιρούμε με τον αριθμό των χαρακτηριστικών στο μοντέλο. Με αυτό τον τρόπο θέλουμε να διαιρέσουμε τον αριθμό όλων των συνδέσεων που υπάρχουν στο μοντέλο χαρακτηριστικών με το πλήθος των χαρακτηριστικών. Ξέροντας ότι κάθε χαρακτηριστικό και κάθε περιορισμός αποτελούν μία σύνδεση με την πρόσθεση των δύο μεγεθών παίρνουμε το πλήθος των συνδέσεων. Ο κώδικας για αυτήν την διαδικασία φαίνεται στην Εικόνα 38.

```
CoC = (double) (Features.size()+Constraints.size()) / Features.size();
```

Εικόνα 38: Κώδικας εύρεσης συντελεστή συνδεσιμότητας-πυκνότητας.

Για την εύρεση της μεταβλητής των περιορισμών μεταξύ δέντρων θέλουμε να υπολογίσουμε τον αριθμό των μοναδικών χαρακτηριστικών που συμμετέχουν σε περιορισμούς. Για να γίνει αυτό πριν αποθηκεύσουμε ένα χαρακτηριστικό που συμμετέχει σε περιορισμό ελέγχουμε πρώτα αν αυτό υπάρχει είδη στην λίστα των χαρακτηριστικών που έχουμε μετρήσει. Με αυτόν τον τρόπο εξασφαλίζουμε ότι δεν θα υπάρχουν διπλότυπα στην λίστα. Τέλος αφού βρούμε τον αριθμό των χαρακτηριστικών που συμμετέχουν σε περιορισμούς το διαιρούμε με το πλήθος των χαρακτηριστικών του δέντρου για να πάρουμε την μετρική των περιορισμών μεταξύ των δέντρων. Στην Εικόνα 39 μπορούμε να δούμε των κώδικα που εκτελείται με σκοπό να εξαχθεί αυτήν η μετρική.

```
for (int i=0; i<Constraints.size(); i++) {
	temp = Constraints.get(i);
	temp = temp.replaceAll("~","");
	String[] constraint = temp.split("\\s*or\\s*|\\s*and\\s*");
	for (int h=0; h<constraint.length; h++) {
		System.out.println(constraint[h]);
		if (LoC.isEmpty()) {
			LoC.add(constraint[h]);
		}
		if(!LoC.contains(constraint[h])) {
			LoC.add(constraint[h]);
		}
	}
}
if (!LoC.isEmpty()) {
	CTC = (double) LoC.size() / Features.size();
}
```

Εικόνα 39: Κώδικας υπολογισμού μετρικής περιορισμών μεταξύ δέντρων.

Για να βρούμε τον αριθμό των κορυφαίων χαρακτηριστικών αρκεί να βρούμε τα χαρακτηριστικά που δεν έχουν γονέα. Από την φύση του το δέντρο χαρακτηριστικό ταξινομεί αυτά τα χαρακτηριστικά στην κορυφή του δέντρου ενώνοντάς τα με την ρίζα. Στην δική μας περίπτωση η ρίζα επειδή δεν υπάρχει ως κλάση του συστήματος τα χαρακτηριστικά αυτά δεν έχουν γονέα οπότε η μεταβλητή parent θα έχει την τιμή «noParent». Επομένως με μία επανάληψη και έναν έλεγχο της τιμής αυτής μπορούμε να βρούμε τον αριθμό των κορυφαίων χαρακτηριστικών.

Για να βρούμε την ευελιξία της διαμόρφωσης πρέπει να βρούμε τον αριθμό των προαιρετικών χαρακτηριστικών και να τον διαιρέσουμε με το πλήθος των χαρακτηριστικών του μοντέλου. Για να βρούμε το πλήθος των προαιρετικών χαρακτηριστικών αρκεί να ελέγξουμε την μεταβλητή «optional» που έχουν τα αντικείμενα των χαρακτηριστικών.

Για να βρούμε τον αριθμό των χαρακτηριστικών φίλων αρκεί να ελέγξουμε την μεταβλητή που περιέχει τον αριθμό των παιδιών του συγκεκριμένου χαρακτηριστικού. Εάν η μεταβλητή αυτήν είναι μηδέν τότε το χαρακτηριστικό δεν έχει παιδιά οπότε είναι φίλο του δέντρου χαρακτηριστικών.

Για να βρούμε την αναλογία μεταβλητότητας του δέντρου χαρακτηριστικών αρκεί να βρούμε τον μέσο όρο τον διακλαδώσεών του. Για να το κάνουμε αυτό αθροίσαμε το σύνολο των παιδιών που έχουν όλα τα χαρακτηριστικά ελέγχοντας σε μία δομή επανάληψης το πλήθος παιδιών κάθε χαρακτηριστικού. Έπειτα, διαιρέσαμε αυτόν τον αριθμό με τον πλήθος των χαρακτηριστικών του δέντρου.

Για να υπολογίσουμε το βάθος του δέντρου σε μία δομή επανάληψης μετράμε το μέγιστο βάθος που πηγαίνει κάθε φορά. Στην συνέχεια συγκρίνουμε αυτό το βάθος ώστε να κρατήσουμε το βάθος μόνο από το μεγαλύτερο κλαδί και αυτό θα είναι το βάθος του δέντρου.

Με την ολοκλήρωση όλων των μετρικών ανάλυσης η εφαρμογή ανακατευθύνει τον χρήστη στην κατάλληλη οθόνη όπου παρουσιάζονται οι μετρικές συντηρησιμότητας του μοντέλου χαρακτηριστικών. Ο κώδικας για τις παραπάνω μετρικές φαίνεται στην Εικόνα 39.

```
for (int i=0; i<Features.size(); i++) {
	if (Features.get(i).getRelationshipClass().equals("noParent")) {
		NoTF++;
	}
	if(Features.get(i).getOptional()) {
		FoC++;
	}
}
for (int i=0; i<Features.size()-1;i++) {
	if (Features.get(i).getChildren() == 0) {
		NoLF++;
		tempDoT=1;
	}
	else {
		//if the feature is not a leaf then we go deeper into the tree
		so the depth increases. We will increase only the temporary
		value of DoT
		//and check when is higher than our previous max value in
		order to find the max depth and not the depth of the last
		branch of the tree.
		tempDoT++;
		if (tempDoT > DoT) {
			DoT = tempDoT;
		}
		//because this feature is a parent we will add the number of
		children it has and in the end we will find the average number
		of children that a feature has.
		RoV = Features.get(i).getChildren()+RoV;
	}
}
//we divide the sum of children with the number of parents. The number of
parents comes from the subtraction of leafs from overall features.
RoV = RoV / LoC.size();
//we divide the sum of optional with the number of all features to find the
average.
FoC= FoC/NoF;
```

Εικόνα 40: Κώδικας μετρήσεων συντηρησιμότητας.

4.2.5 Εξαγωγή διαγράμματος κλάσεων σε μορφή XML

Κατά την διάρκεια εξαγωγής κώδικα της εφαρμογής δημιουργείται και ένα αρχείο τύπου XML όπου αποθηκεύεται το διάγραμμα κλάσεων του παραγόμενου κώδικα. Για την δημιουργία αυτού του αρχείου δημιουργήθηκε μία συνάρτηση η οποία δημιουργεί το αρχείο «umlClassDiagram.xml» με κωδικοποίηση UTF-8. Για την εγγραφή των πληροφοριών σε αρχείο δημιουργείται ένα αντικείμενο τύπου «PrintWriter». Στην κορυφή του αρχείου δηλώνεται η έκδοση της XML που θα χρησιμοποιηθεί καθώς και η κωδικοποίησή. Έπειτα με την χρήση μίας δομής επανάληψης ανατρέχουμε στο ArrayList όπου είναι αποθηκευμένα τα χαρακτηριστικά και εισάγουμε τις πληροφορίες τους με συγκεκριμένη δομή.

Για την δημιουργία ενός διαγράμματος κλάσεων σε μορφή XML έπρεπε να κατασκευαστεί μία δομή όπου θα περιέχει όλα τα απαραίτητα χαρακτηριστικά μίας κλάσης στον αντικειμενοστραφή προγραμματισμό καθώς και την βασική

ιεραρχική δομή παρουσίασης πληροφοριών που χαρακτηρίζει ένα XML αρχείο. Η δομή που αναπτύχθηκε περιέχει έξι διαφορετικές ετικέτες XML (Class, Relationship, Values, Value, Functions, Function) που καθορίζουν και την δομή του αρχείου. Επίσης, για την αποθήκευση των πληροφοριών της κάθε κλάσης δημιουργήθηκαν και τα εξής χαρακτηριστικά των ετικετών: name, type, class και access. Τέλος, η δομή του αρχείου ακολουθεί ιεραρχική στοίχιση με εσοχές ώστε τα στοιχεία που περιέχονται σε κάποια ετικέτα να είναι στο εσωτερικό της.

Η χρήση XML για την αναπαράσταση του διαγράμματος κλάσεων προσφέρει ένα εύχρηστο και ευανάγνωστο τρόπο παρουσιάσεις ενός διαγράμματος κλάσεων. Επίσης η αποθήκευση της πληροφορίας με αυτήν την μορφή μπορεί να βοηθήσει στην μεταφορά των πληροφοριών του συγκεκριμένου προγράμματος σε κάποια άλλη εφαρμογή αφού όλες οι κλάσεις με όλα τα στοιχεία τους μπορούν πολύ εύκολα να ανακτηθούν.

Στην Εικόνα 41 παρουσιάζεται ένα απόσπασμα ενός διαγράμματος κλάσεων με την χρήση XML.

```
<?xml version="1.0" encoding="UTF-8" ?>
    <Class name="Call(Call)">
        <Relationship type="Composition" class="root">
        <Values>
        </Values>
        <Functions>
            <Function access="public" name="Call(Call)">
        </Functions>
    </Class>
    <Class name="Screen(Screen)">
        <Relationship type="Composition" class="root">
        <Values>
        </Values>
        <Functions>
            <Function access="public" name="Screen(Screen)">
        </Functions>
    </Class>
    <Class name="Basic(Basic)">
        <Relationship type="Generalization" class="Screen(Screen)">
        <Values>
        </Values>
        <Functions>
            <Function access="public" name="Basic(Basic)">
        </Functions>
    </Class>
    <Class name="HD(HD)">
        <Relationship type="Generalization" class="Screen(Screen)">
        <Values>
        </Values>
        <Functions>
            <Function access="public" name="HD(HD)">
        </Functions>
    </Class>
```

Εικόνα 41: Μέρος του XML διαγράμματος κλάσεων.

4.2.6 Εξαγωγή κώδικα Java

Για την εξαγωγή κώδικα ο χρήστης θα πρέπει να είναι στην κεντρική οθόνη της εφαρμογής και να έχει εισάγει ένα μοντέλο χαρακτηριστικών στο πρόγραμμα. Έπειτα, με την επιλογή της κατάλληλης διαδικασίας από το γραφικό περιβάλλον της εφαρμογής εκκινείτε η συνάρτηση παραγωγής κώδικα. Για την διαδικασία αυτήν χρειάστηκε να κατασκευαστεί ειδική μέθοδος όπου δέχεται ως είσοδο το ArrayList που είναι αποθηκευμένες όλες οι κλάσεις του προγράμματος που επεξεργάζεται ο μηχανικός λογισμικού.

Για την εγγραφή των αρχείων κώδικα Java δημιουργείτε ένα αντικείμενο τύπου «PrintWriter». Για να δημιουργηθούν τα αρχεία ορίζεται μία δομή επανάληψης ώστε να προσπελάσουμε όλες τις κλάσεις που βρίσκονται αποθηκευμένες στην λίστα. Το όνομα κάθε κλάση χρησιμοποιείται για το όνομα του αρχείου που θα δημιουργηθεί με την επέκταση .java και κωδικοποίηση UTF-8. Στην συνέχεια παράγεται ο κώδικάς σύμφωνα με την ορθή σύνταξη κώδικα της Java.

Η συνάρτηση αυτήν παίρνει το όνομα μίας κλάσης και δημιουργεί το αρχείο της κλάσης. Στην συνέχεια ελέγχει εάν υπάρχει κάποια κληρονομικότητα σε αυτήν την κλάση και αν ναι την προσθέτει στον κώδικα. Έπειτα διαβάζει όλες τις μεταβλητές του συστήματος και εισάγει τις μεταβλητές στην εισαγωγή του εγγράφου. Έπειτα δημιουργεί τον Constructor βάση του ονόματος της κλάσης και των μεταβλητών της. Τέλος, η συνάρτηση δημιουργεί τις απαραίτητες συναρτήσεις ώστε να είναι δυνατό κάποιος να θέση και να πάρει τις τιμές των μεταβλητών από το πρόγραμμα.

Παράδειγμα του εξαγόμενου κώδικα φαίνεται στην Εικόνα 42 και μέρος του κώδικα της συνάρτησης φαίνεται στην Εικόνα 43.

```
public class Call(Call){

	private int ID;
	private String name;

	 public Call(Call) (int ID, String name){
		this.ID = ID;
		this.name = name;

}

	public void setID (int ID){
		this.ID = ID;
	}

	public int getID (){
		return ID;
	}

	public void setname (String name){
		this.name = name;
	}

	public String getname (){
		return name;
	}
}
```

Εικόνα 42: Παραγώμενος κώδικας Java

```
private void export(ArrayList<Feature> features ) throws
FileNotFoundException, UnsupportedEncodingException {
		//Save the path of every value instead of calling the same
		path again and again
		Value values;
		for(int i=0; i<features.size();i++) {
			System.out.println(features.get(i).getFeatureName());
			PrintWriter writer = new
			PrintWriter(features.get(i).getFeatureName()+".java",
			"UTF-8");
			//create class
			if (features.get(i).getParentClass()=="noParent") {
				writer.println("public class
				"+features.get(i).getFeatureName()+" {");
				writer.println();
			}
			else {
				writer.println("public class
				"+features.get(i).getFeatureName()+" extends
				"+features.get(i).getParentClass()+" {");
				writer.println();
			}
			//create values
			for(int j=0; j<features.get(i).getValues().size();j++)
{
				values=features.get(i).getValues().get(j);

				writer.println("\t"+values.getValueAccess()+"
				"+values.getValueType()+"
				"+values.getValueName()+";");
			}
			//create constructor
			writer.println();
			writer.print("\t public
			"+features.get(i).getFeatureName()+" (");
			for(int j=0; j<features.get(i).getValues().size()-
			1;j++) {
				values=features.get(i).getValues().get(j);
				writer.print(values.getValueType()+"
				"+values.getValueName()+", ");
			}
			//add the last value to constructor and if the class
			has not any values do nothing
			if (features.get(i).getValues().size() != 0){
				values=features.get(i).getValues().get(features.g
				et(i).getValues().size()-1);
				writer.println(values.getValueType()+"
				"+values.getValueName()+"){");
			}
			else {
				writer.println("){}");
			}
			for(int j=0; j<features.get(i).getValues().size();j++)
{
				values=features.get(i).getValues().get(j);
				//setter
				writer.println("\t\t"+"this."+values.getValueName
				()+" = "+values.getValueName()+";");
			}
```

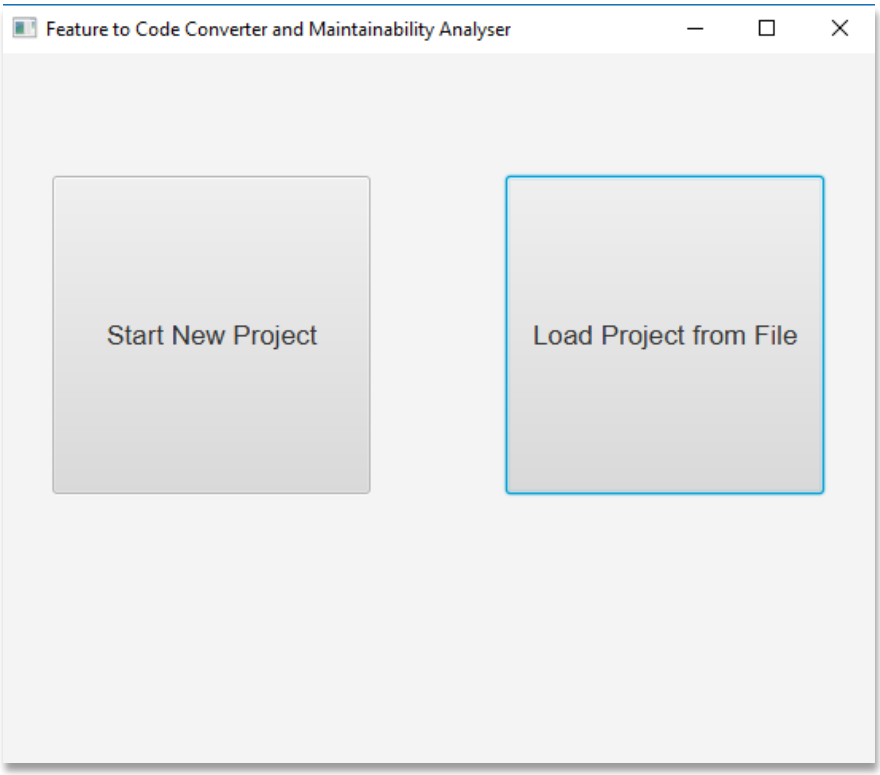
Εικόνα 43: Μέρος του κώδικα παραγωγής κώδικα Java

5 Παρουσίαση και εκτέλεση εφαρμογής F.C.C.M.A

Στο συγκεκριμένο κεφάλαιο θα γίνει παρουσίαση του εργαλείου «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)» με την χρήση ενός παραδείγματος εκτέλεσης της εφαρμογής. Για την απεικόνισή των διαφορετικών διαδικασιών θα δοθούν στιγμιότυπα των διαφορετικών οθονών του συστήματος. Η λήψη των στιγμιότυπων θα γίνει σε φορητό υπολογιστή λειτουργικού συστήματος Windows και ανάλυση οθόνης 1366 x 768. Η παρουσίαση και εκτέλεση του συστήματος θα καλύψει όλες τις δυνατότητες της εφαρμογής καθώς και όλες τις περιπτώσεις χρήσης που αναλύθηκαν στο Κεφάλαιο 4.1.1 της συγκεκριμένης διπλωματικής εργασίας. Για την αποθήκευση των στιγμιότυπων οθόνης χρησιμοποιήθηκε η ελεύθερη προς χρήση εφαρμογή LightShot ώστε να γίνεται απεικόνιση μόνο του παραθύρου εκτέλεσης της εφαρμογής.

5.1 Εκκίνηση εφαρμογής

Εκκινώντας την εφαρμογή «Feature to Code Converter and Maintainability Analyser (F.C.C.M.A)» εμφανίζεται η αρχική οθόνη πλοήγησης. Σε αυτήν την οθόνη μπορούμε να δούμε αριστερά την επιλογή εκκίνησης νέου έργου όπου απαιτεί την εισαγωγή ενός μοντέλου χαρακτηριστικών σε μορφή XML και δεξιά την επιλογή φόρτωσης ενός παλιού έργου στο πρόγραμμα ώστε ο χρήστης να συνεχίσει τις εργασίες του. Στην δικιά μας περίπτωση θα επιλέξουμε την επιλογή δημιουργίας ενός νέου έργου και θα εισάγουμε το μοντέλο χαρακτηριστικών που θα αναλύσουμε. Στην Εικόνα 44 προβάλετε η οθόνη εκκίνησης του συστήματος.



Εικόνα 44: Οθόνη εκκίνησης F.C.C.M.A.

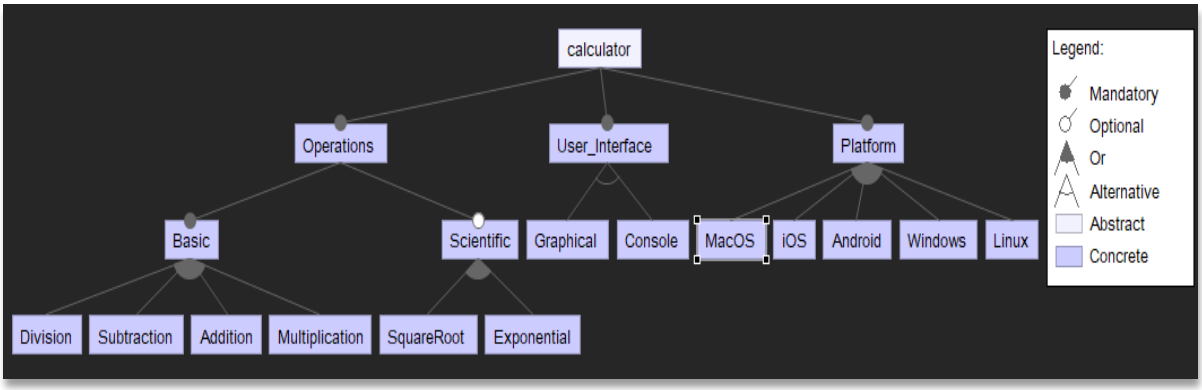
5.2 Μοντέλο χαρακτηριστικών εισαγωγής

Για την επίδειξη της εφαρμογής κατασκευάστηκε ένα μοντέλο χαρακτηριστικών στο FeatureIDE όπου απεικονίζει μερικός την γραμμή παραγωγής λογισμικού που θα είχε η κατασκευή διαφορετικών ειδών αριθμομηχανών.

Μπορούμε να διακρίνουμε ότι ως ρίζα είναι η γενική οντότητα της αριθμομηχανής (calculator). Στην συνέχεια υπάρχουν τρία χαρακτηριστικά παιδιά της ρίζας όπου είναι οι πράξεις που θα μπορεί να τελεί η εφαρμογή (Operations), το είδος του γραφικού περιβάλλοντος (User Interface) της εφαρμογής καθώς και το είδος των υποστηριζόμενων λειτουργικών (Platform) της εφαρμογής. Για το χαρακτηριστικό των υποστηριζόμενων πράξεων της εφαρμογής βλέπουμε ότι χωρίζονται σε δύο υπό χαρακτηριστικά. Οι βασικές (Basic) πράξεις (πρόσθεση, αφαίρεση, πολλαπλασιασμός, διαίρεση) είναι υποχρεωτικό χαρακτηριστικό του δέντρου που σημαίνει ότι όλες οι διαφορετικές εφαρμογές θα περιέχουν έστω μία από αυτές καθώς ο σύνδεσμος Or επιτρέπει την επιλογή από μία μέχρι όλες. Οι επιστημονικές (Scientific) πράξεις (τετραγωνική ρίζα, εκθετικά) δεν είναι υποχρεωτικό να υπάρχουν σε όλες τις διαφορετικές εκδόσεις της εφαρμογής αφού είναι προαιρετικό χαρακτηριστικό.

Για το υποχρεωτικό χαρακτηριστικό του γραφικού περιβάλλοντος έχουμε δύο επιλογές. Η μία είναι να έχει γραφικό περιβάλλον και η άλλη να έχει περιβάλλον κονσόλας. Σε αυτήν την περίπτωση μπορούμε να επιλέξουμε μόνο μία από της δύο σχέσεις διότι η σχέση alternative επιτρέπει την επιλογή μόνο ενός χαρακτηριστικού. Τέλος για το υποχρεωτικό χαρακτηριστικό του λειτουργικού συστήματος η δίνονται πέντε επιλογές (MacOS, iOS, Android, Windows, Linux). Μπορούν να επιλεχθούν ταυτόχρονα πολλά διαφορετικά χαρακτηριστικά λειτουργικού καθώς η επιλογή or δίνει αυτήν την δυνατότητα.

Στην Εικόνα 45 παρουσιάζεται το διάγραμμα μοντέλου χαρακτηριστικών για το παραπάνω παράδειγμα.



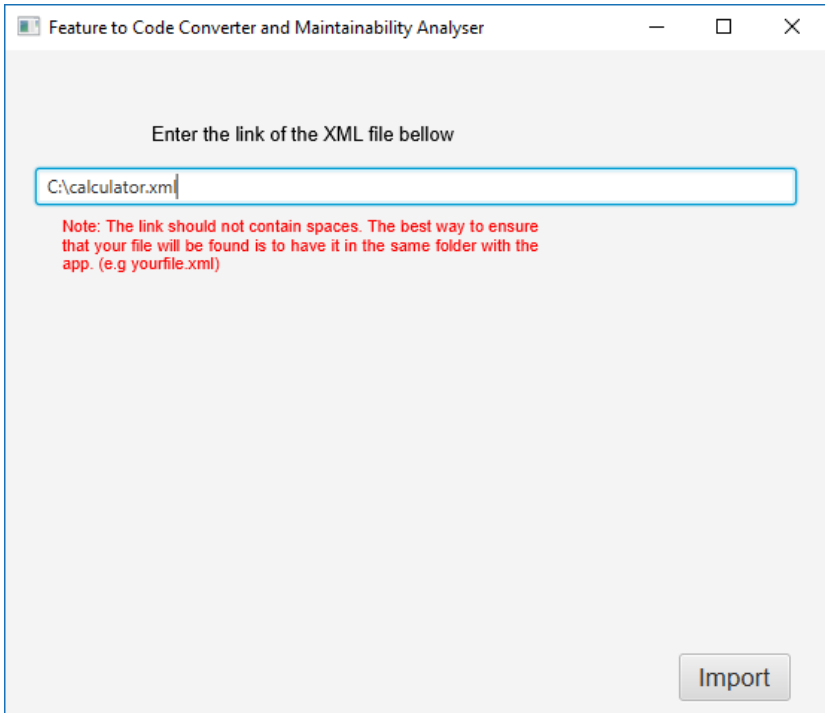
Εικόνα 45: Μοντέλο χαρακτηριστικών αριθμομηχανής.

5.3 Εισαγωγή μοντέλου χαρακτηριστικών στο F.C.C.M.A

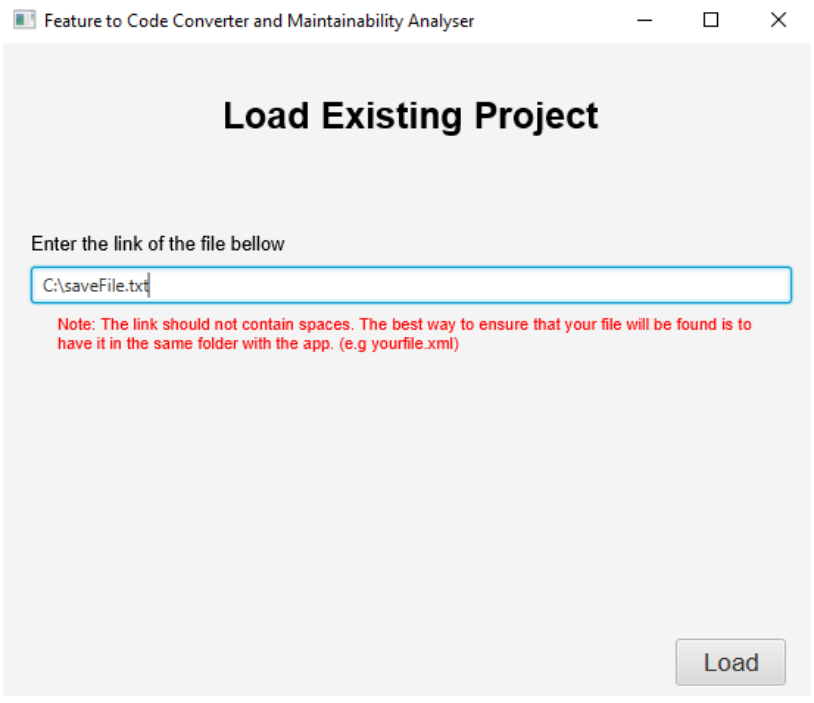
Για την εισαγωγή του μοντέλου χαρακτηριστικών χρησιμοποιήθηκε το αρχείο XML του διαγράμματος χαρακτηριστικών που είδαμε προηγουμένως στο Κεφάλαιο 5.2. Ο χρήστης αφού επιλέξει την δημιουργία νέου έργου μεταφέρεται στην οθόνη εισαγωγής νέου μοντέλου χαρακτηριστικών. Σε περίπτωση επιλογής να φορτώσει ένα παλιό έργο

που έχει αποθηκεύσει στο παρελθόν η εφαρμογή ανακατευθύνει το χρήστη στην κατάλληλη σελίδα όπου ο χρήστης πρέπει να δώσει την διαδρομή του αρχείου που θέλει να εισάγει.

Στην Εικόνα 46 βλέπουμε την οθόνη όπου ο χρήστης εισάγει το μοντέλο χαρακτηριστικών για πρώτη φορά στο σύστημα με την χρήση του XML αρχείου. Στην Εικόνα 47 βλέπουμε την οθόνη φόρτωσης ενός υπάρχων έργου στο σύστημα.



Εικόνα 46: Εισαγωγή μοντέλου χαρακτηριστικού αριθμομηχανής από XML

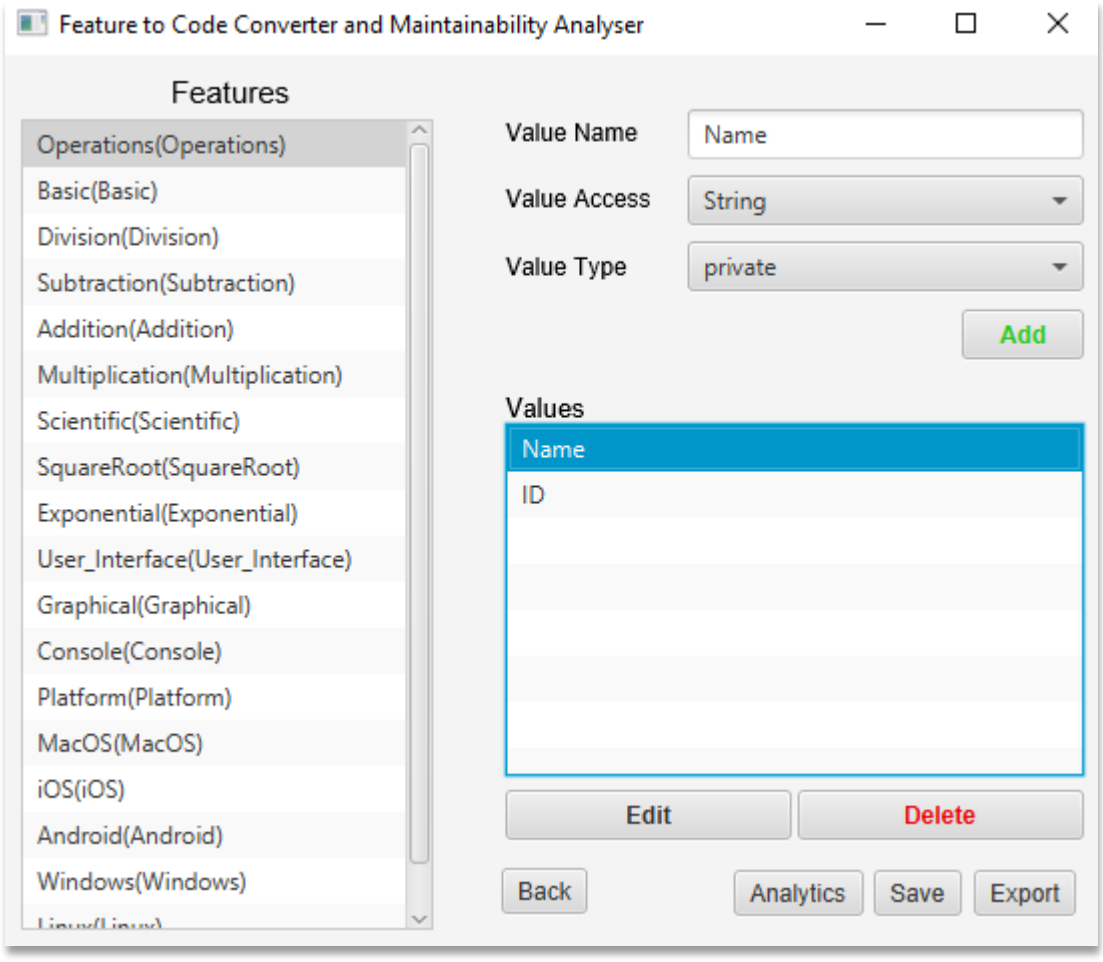


Εικόνα 47: Φόρτωση έργου αριθμομηχανής από αρχείο αποθήκευσης.

5.4 Διαχείριση Μεταβλητών

Σε αυτήν την οθόνη ο χρήστης εισάγει τις μεταβλητές που θέλει στο σύστημα. Οι κλάσεις του συστήματος που δημιουργήθηκαν φαίνονται στο αριστερό μέρος του παραθύρου. Για τους λόγους τις συγκεκριμένης παρουσίασης της εφαρμογής θα δημιουργήσουμε δύο μεταβλητές στην κλάση των πράξεων (operations). Η πρώτη μεταβλητή είναι το όνομα (name) του αντικειμένου που θα δημιουργείτε βάση όχι της συγκεκριμένης κλάσης αλλά από τους απόγονους της κλάσης αυτής αφού έχουμε κληρονομικότητα και αυτήν είναι η κλάση γονέας. Αυτήν η μεταβλητή είναι τύπου String και είναι ιδιωτική (private). Η δεύτερη μεταβλητή που θα δημιουργήσουμε είναι η μεταβλητή ID όπου θα χαρακτηρίζει το μοναδικό κωδικό που θα έχει το κάθε αντικείμενο. Αυτή η μεταβλητή είναι τύπου ακέραιος αριθμός (int) και ιδιωτική (private).

Στην Εικόνα 48 παρουσιάζεται η κεντρική οθόνη του συστήματος με τις μεταβλητές που περιγράψαμε ότι εισήγαμε ποιο πάνω.



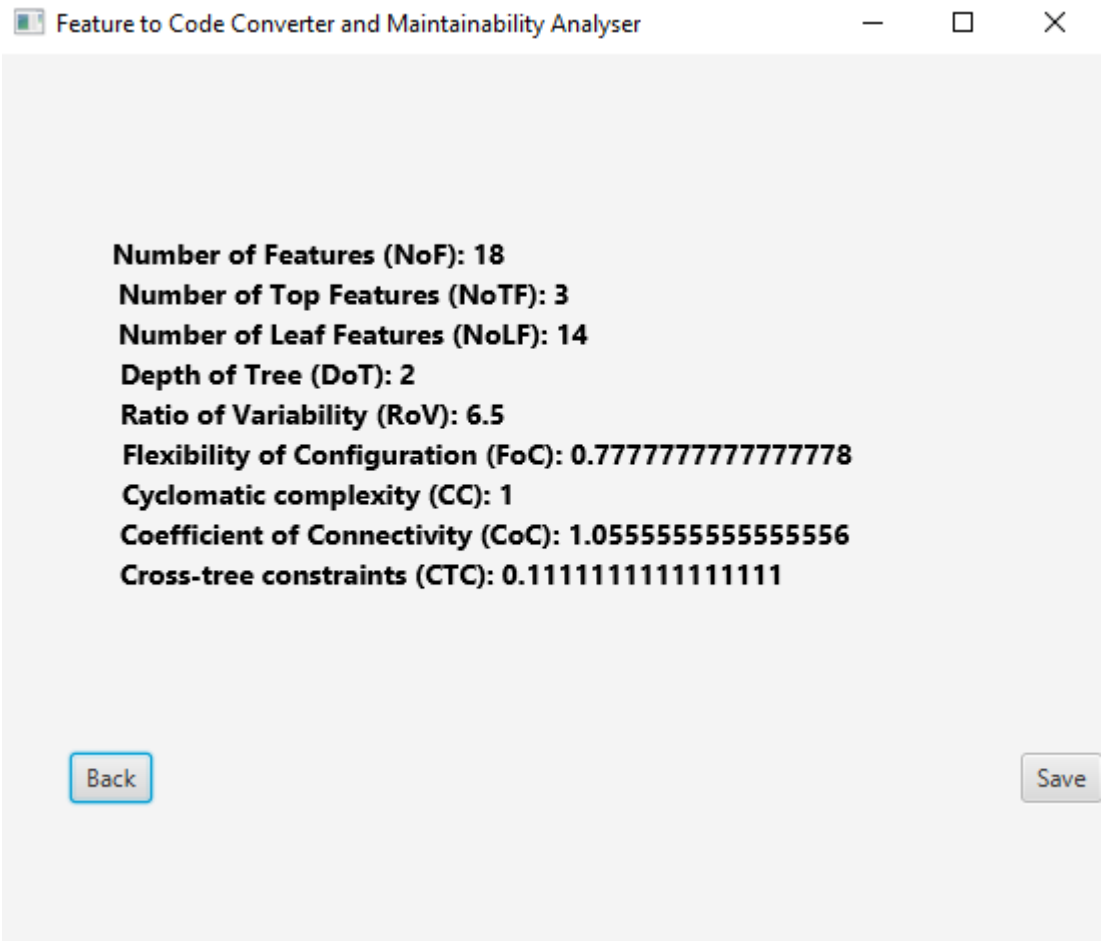
Εικόνα 48: Κεντρική οθόνη με της μεταβλητές του παραδείγματος.

Για την αλλαγή του ονόματος της μεταβλητής «Name» επιλέγουμε την μεταβλητή της δίνουμε άλλο όνομα και πατάμε το κουμπί Edit. Για την διαγραφή της μεταβλητής «ID» ο χρήστης επιλέγει την μεταβλητή και πατάει το κουμπί Delete.

5.5 Εμφάνιση μετρικών συντηρησιμότητας

Για την εμφάνιση των μετρήσεων συντηρησιμότητας του μοντέλου χαρακτηριστικών ο χρήστης πρέπει να επιλέξει το κατάλληλο κουμπί στην κεντρική οθόνη του συστήματος. Έπειτα, το σύστημα επεξεργάζεται το μοντέλο χαρακτηριστικών και εξάγει τις μετρικές. Η εφαρμογή ανακατευθύνει το χρήστη σε μία καινούρια σελίδα όπου μπορεί να δει τις μετρικές καθώς και να τις αποθηκεύσει σε αρχείο απλού κειμένου. Τέλος με την επιλογή της επιστροφής ο χρήστης μπορεί να επιστρέψει στην κεντρική σελίδα της εφαρμογής.

Οι μετρήσεις όπου αναφέρθηκαν και αναλύθηκαν σε προηγούμενα κεφάλαια παρουσιάζονται στην Εικόνα 49 για την περίπτωση χρήσης αυτού του παραδείγματος.



Εικόνα 49: Οθόνη στοιχείων συντηρησιμότητας.

5.6 Αποθήκευση, εξαγωγή κώδικα και διαγράμματος κλάσης σε XML

Για την αποθήκευση του αρχείου ο χρήστης επιλέγει το κουμπί Save και η εφαρμογή δημιουργεί και αποθηκεύει το παρών έργο σε ένα αρχείο απλού κειμένου. Έπειτα ο χρήστης μπορεί να εισάγει αυτό το αρχείο στο πρόγραμμα ώστε να συνεχίσει την εργασία του από το σημείο που είχε μείνει.

Για την εξαγωγή κώδικα Java καθώς και την παραγωγή του διαγράμματος κλάσεων σε XML μορφή ο χρήστης επιλέγει την επιλογή Export. Έπειτα η εφαρμογή εκκινεί τις κατάλληλες συναρτήσεις που εξηγήσαμε σε προηγούμενα κεφάλαια ώστε να εξάγει τα

απαραίτητα αρχεία. Παράδειγμα του αρχείου κώδικα της κλάσης «Operations» μπορούμε να δούμε στην Εικόνα 50.

```
public class Operations extends null {

	private String Name;
	private int ID;

	 public Operations (String Name, int ID){
		this.Name = Name;
		this.ID = ID;

}

	public void setName (String Name){
		this.Name = Name;
	}

	public String getName (){
		return Name;
	}

	public void setID (int ID){
		this.ID = ID;
	}

	public int getID (){
		return ID;
	}
}
```

Εικόνα 50: Παραγόμενος κώδικας παραδείγματος.

6 Επίλογος

Στο κεφάλαιο αυτό θα γίνει καταγραφή όλων των συμπερασμάτων της μελέτης όπως και της έρευνας που πραγματοποιήθηκε στο πλαίσιο της διπλωματικής εργασίας. Επιπλέον, θα γίνει παρουσίαση κάποιον παρατηρήσεων που έγιναν αντιληπτές κατά τη διάρκεια εκπόνησης της. Τέλος, θα προταθούν μελλοντικές επεκτάσεις της εφαρμογής που δημιουργήθηκε οι οποίες σκοπό έχουν να βοηθήσουν στην ανάπτυξη του λογισμικού «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)».

6.1 Συμπεράσματα

Στις μέρες μας τα μοντέλα χαρακτηριστικών χρησιμοποιούνται όλο και ποιο πολύ για την σχεδίαση, την διαχείριση και τον έλεγχο εκτενών συστημάτων. Τα μοντέλα χαρακτηριστικών παρέχουν όλα τα απαραίτητα χαρακτηριστικά για ανάλυση μίας εφαρμογής σε πρώιμο στάδιο ώστε να εντοπιστούν πιθανά σχεδιαστικά λάθη πριν την διαδικασία της παραγωγή κώδικα. Επίσης, τα μοντέλα χαρακτηριστικών μπορούν να θεωρηθούν ως ένα πρώτο στάδιο σχεδιαστικής δομής καθώς μπορούν να παραχθούν ποιο γρήγορα από άλλα μοντέλα όπως για παράδειγμα τα διαγράμματα κλάσεων καθώς δεν απαιτούν τόσες πληροφορίες.

Στόχος της παρούσας διπλωματικής εργασίας ήταν να μελετηθεί η χρήση των μοντέλων χαρακτηριστικών στις γραμμές προϊόντων λογισμικού αλλά και ενός εργαλείου για την επεξεργασίας τους. Ο σκοπός αυτός επιτεύχθηκε μέσω τις εκτενείς μελέτης που έγινε πάνω στα μοντέλα χαρακτηριστικών καθώς και πως αυτά μπορούν να μεταφραστούν σε άλλου είδους μοντέλα καθώς και σε κώδικα. Επίσης, μέσο της μελέτης και της έρευνας που έγινε διαπιστώθηκε για το πώς μπορεί να γίνει η μετάφραση των σχέσεων μεταξύ των χαρακτηριστικών του μοντέλου χαρακτηριστικών σε σχέσεις του διαγράμματος κλάσεων.

Ο στόχος της εξαγωγής κώδικα καθώς, μετρικών συντηρησιμότητας καθώς και του διαγράμματος κλάσεων σε μορφή XML έγινα με την υλοποίηση της εφαρμογής «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)». Με την εφαρμογή αυτήν είμαστε σε θέση παράγουμε το βασικό κώδικα όπου χρειάζεται για να δημιουργηθούν οι κλάσεις επιταχύνοντας έτσι την διαδικασία παραγωγής κώδικα. Επίσης, οι μετρήσεις συντηρησιμότητας μπορούν να δώσουν πληροφορίες στο μηχανικό λογισμικού εάν η συγκεκριμένη γραμμής παραγωγής είναι εύκολο να συντηρηθεί και να επεκταθεί στο μέλλον. Με αυτόν τον τρόπο εξασφαλίζεται η καλή ποιότητα της γραμμής παραγωγής λογισμικού και προλαμβάνονται τυχόν λάθη σχεδίασης.

Τα εργαλεία και οι τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση του συγκεκριμένου εγχειρήματος, επιλέχθηκαν με μεγάλη προσοχή και ύστερα από έρευνα. Τα κριτήρια που έπαιξαν καταλυτικό ρόλο στην επιλογή των εργαλείων αυτών ήταν αρχικά η προσβασιμότητα σε αυτά, δηλαδή εάν έχουν ελεύθερη πρόσβαση και χρήση ή απαιτείται κάποια σχετική άδεια χρήσης. Δεύτερο κριτήριο αποτέλεσε η βιβλιογραφικές όσο και δικτυακές πληροφορίες που θα μπορούσαν να βρεθούν για αυτές της εφαρμογές ώστε να είναι δυνατή η ανάπτυξη του λογισμικού. Τόσο το Eclipse όσο και το FeatureIDE γνωρίζουν μία ευρεία χρήση στο χώρο του αντικειμενοστραφή προγραμματισμού και σχεδιασμού μοντέλων χαρακτηριστικών. Η

χρήση της XML για εξαγωγή του διαγράμματος κλάσεων έγινε με σκοπό την απλούστευση της σχεδίασης του διαγράμματος και για την εύκολη ανάγνωση των στοιχείων κάθε κλάσης από κάποιο πρόγραμμα.

Κατά την διάρκεια της παρούσας διπλωματικής παρατηρήθηκαν και μερικές παρατηρήσεις. Η μεγαλύτερη από αυτές ήταν ότι παρόλο που το FeatureIDE παράγει το μοντέλο χαρακτηριστικών σε αρχείου της μορφής XML η χρήσιμη πληροφορία βρίσκεται σε μορφή απλού κειμένου και για το λόγο αυτό έπρεπε να δημιουργηθεί ειδική συνάρτηση ώστε να μπορεί να αναγνωρίζει τα χαρακτηριστικά. Στην συνέχεια μία δεύτερη παρατήρηση είναι η ελλιπής οργάνωσης ενός repository με μοντέλα χαρακτηριστικών καθώς και κατάλληλων σχολίων εάν στην πράξη αυτά τα μοντέλα ήταν αποδοτικά ή όχι.

6.2 Μελλοντικές Επεκτάσεις

Το εργαλείο «Feature to Code Converter and Maintainability Analyzer (F.C.C.M.A)» αναπτύχθηκε με σκοπό να βοηθήσει τους μηχανικούς λογισμικού στην εξέλιξη των μοντέλων χαρακτηριστικών και να επεκτείνει την χρήση τους. Παρά το γεγονός ότι η εφαρμογή αυτή πληροί τις προδιαγραφές για να γίνει ένα βασικό εργαλείο στην ανάλυση των μοντέλων χαρακτηριστικών, θα γίνει παρουσίαση κάποιων προτεινόμενων επεκτάσεων οι οποίες θα μπορέσουν να βοηθήσουν στην περαιτέρω ανάπτυξη των δυνατοτήτων της εφαρμογής.

Η πρώτη επέκταση που θα μπορούσε να γίνει στην εφαρμογή με σκοπό να αυξήσει τις δυνατότητές της είναι η κατασκευή ενός περιβάλλοντος σχεδίασης μοντέλων χαρακτηριστικών. Στην τωρινή μορφή της, η εφαρμογή διαβάζει τα μοντέλα χαρακτηριστικών από ένα αρχείο XML που παράγεται από το πρόσθετο FeatureIDE του Eclipse. Με την κατασκευή ενός γραφικού περιβάλλοντος σχεδίασης μοντέλων χαρακτηριστικών η εφαρμογή θα μπορούσε να γίνει αυτόνομη και να μην εξαρτάτε πλέον από το FeatureIDE και το Eclipse. Με αυτόν τον τρόπο η χρήση της εφαρμογής θα γινόταν ποιο εύκολη για το μηχανικό λογισμικού καθώς δεν θα χρειαζόταν να μεταβαίνει από την μία εφαρμογή στην άλλη. Επίσης με την δημιουργία ενός γραφικού περιβάλλοντος εντός της εφαρμογής η ανάλυση συντηρησιμότητας θα γινόταν κατά την διαδικασία κατασκευής του μοντέλου χαρακτηριστικών. Έτσι θα ήταν ποιο εύκολο και γρήγορο να εντοπισθούν σχεδιαστικά λάθη στο μοντέλο παρά του να πρέπει πρώτα να σχεδιαστεί ολόκληρο το μοντέλο χαρακτηριστικών και έπειτα να γίνει εισαγωγή του στην εφαρμογή. Τέλος, με την δημιουργία ενός γραφικού περιβάλλοντος σχεδίασης μοντέλων χαρακτηριστικών θα ήταν εύκολο αυτό να επεκταθεί και σε περιβάλλον σχεδίασης και προβολής διαγραμμάτων κλάσεως. Έτσι θα μπορούσε να γίνει και άλλη μία επέκταση στην εφαρμογή, αυτήν της εξαγωγής στατιστικών και για τα διαγράμματα κλάσεων κλίνοντας έτσι τον κύκλο της γραμμής προϊόντων λογισμικού.

Μία δεύτερη μελλοντική επέκταση δεν αφορά κατά βάση την ανάπτυξη της εφαρμογής «Feature to Code Converter and Maintainability Analyser (F.C.C.M.A)» αλλά την γενικότερη επέκταση των δυνατοτήτων των μοντέλων χαρακτηριστικών. Αυτή η επέκταση αφορά την κατασκευή ενός ηλεκτρονικού αποθετηρίου όπου θα αποθηκεύονται μοντέλα χαρακτηριστικών καθώς και οι μετρικές τους. Έπειτα, οι κατασκευαστές αυτών των μοντέλων χαρακτηριστικών θα μπορούν να επιστρέψουν στην σελίδα της εφαρμογής και βάση ενός ερωτηματολογίου να μπορούν να δώσουν

πληροφορίες σχετικά με το πόσο εύκολη ήταν η συντήρηση και η ανάπτυξη της συγκεκριμένης γραμμής προϊόντων λογισμικού. Σκοπός αυτής της επέκτασης θα είναι να δημιουργηθεί ένα ηλεκτρονικό αποθετήριο όπου θα περιέχει πληροφορίες όχι μόνο για τα μοντέλα χαρακτηριστικών αλλά και πως αυτά ανταποκρίθηκαν στην πράξη. Έπειτα μέσα από τις απαντήσεις και τα σχόλια των χρηστών θα μπορούν να εξαχθούν πληροφορίες σχετικά με το ποια μεταβλητή από την ανάλυση συντηρησιμότητας φαίνεται να επηρέασε στην πράξη ποιο πολύ την εξέλιξη της συγκεκριμένης σειράς προϊόντων λογισμικού. Με αυτόν τον τρόπο μπορούμε να επεκτείνουμε την λειτουργία της εφαρμογής μας με σκοπό όχι μόνο να εμφανίζει τις μετρήσεις συντηρησιμότητας αλλά να μπορεί να προτείνει στο χρήστη πιθανών λάθη στην σχεδίαση αλλά και λύσεις για την καλύτερη κατασκευή του μοντέλου. Έτσι θα διασφαλιστεί η ποιότητα κατασκευής των γραμμών προϊόντων λογισμικού και από αυτό θα επωφεληθεί όλη η κοινότητα όπου θα χρησιμοποιεί την συγκεκριμένη εφαρμογή αλλά και θα ανεβάσει τις απόψεις και τα στοιχεία που σύλλεξε από την πραγματική δοκιμή αυτών στην παραγωγική διαδικασία.

Οι παραπάνω μελλοντικές επεκτάσεις μπορούν να υλοποιηθούν ώστε όχι μόνο να αναπτυχθεί η λειτουργικότητα της συγκεκριμένης εφαρμογής, αλλά και για να δημιουργηθεί μία κοινότητα όπου θα επωφελείται τα δεδομένα που συλλέξανε όλοι οι χρήστες της. Με την διαρκή ανάπτυξη των τεχνολογιών λογισμικού και ειδικότερα των μοντέλων χαρακτηριστικών συγκεκριμένες προτάσεις θα μπορούσαν να αποτελέσουν τον πυλώνα της δημιουργίας και της ανάπτυξης μίας κοινότητας λογισμικού. Τέλος, οι επεκτάσεις αυτές θα μπορούσαν να φέρουν τις γραμμές προϊόντων λογισμικού και τα μοντέλα χαρακτηριστικών ποιο κοντά στις μικρότερες επιχειρήσεις που δεν έχουν τους απαραίτητους πόρους για να αναπτύξουν τον τομέα ανάλυσης λογισμικού.

7 Βιβλιογραφικές αναφορές

[1] David Benavides, Sergio Segura, Antonio Ruiz-Corte's. "Automated Analysis of Feature Models 20 Years Later: A Literature Review". Information Systems Data: Creation, Management and Utilization, Vol 35, Issue 6, p. 615-636, September 2010.

[2] Benjamin Klatt, Klaus Krogmann. "Model-Driven Product Consolidation into Software Product Lines". Proceedings of the 1st Workshop on Model-Based and Model-Driven Software Modernization, March 2012.

[3] Krzysztof Czarnecki, Michal Antkiewicz. "Mapping Features to Models: A Template Approach Based on Superimposed Variants". In: Glück R., Lowry M. (eds) Generative Programming and Component Engineering. GPCE 2005. Lecture Notes in Computer Science, vol 3676. Springer, Berlin, Heidelberg, 2005.

[4] Ebrahim Bagheri, Dragan Gasevic. "Assessing the maintainability of software product line feature models using structural metrics". Software Quality Journal, Volume 19, Issue 3, p. 579–612, September 2011.

[5] Javier Perez, Miguel A. Laguna, Yania Crespo, Bruno González-Baixauli. "Requirements Variability Support Through MDA™ and Graph Transformation". Electronic Notes in Theoretical Computer Science 152(1), p. 161-173, March 2006.

[6] K. Czarnecki, S. Helsen. "Feature-based survey of model transformation approaches". IBM SYSTEMS JOURNAL, VOL 45, NO 3, p.621-645.

[7] Christian Kästner, Thomas Thüm, Gunter Saake, Janet Feigenspan, Thomas Leich, Fabian Wielgorz, Sven Apel. "FeatureIDE: A Tool Framework for Feature-Oriented Software Development". Proceedings of the 31th International Conference on Software Engineering (ICSE), May 2009.

[8] Thomas Leich, Sven Apel, Laura Marnitz, Gunter Saake. "FeatureIDE: An Eclipse-Based Approach". Proceedings of the 2005 OOPSLA workshop on Eclipse Technology eXchange, ETX 2005, San Diego, California, USA, October 16-17, 2005.

[9] G.Kruk, O.Alves, L.Molinari, E.Roux. "PRACTICES FOR EFFICIENT DEVELOPMENT OF JAVAFX APPLICATIONS". 16th Int. Conf. on Accelerator and Large Experimental Control Systems ICALEPCS2017, Barcelona, Spain, 2017.

[10] Eclipse Juno IDE. https://www.eclipse.org/oxygen/. Ιούνιος 2018.

[11] K. Kang, S. Cohen, J. Hess, W. Novak, S. Peter-son. “Feature–Oriented Domain Analysis (FODA) Feasibility Study”. Technical Report CMU/SEI-90-TR-21,Software Engineering Institute, Carnegie Mellon University, November 1990.

[12] SceneBuilder. http://gluonhq.com/products/scene-builder/. Ιούνιος 2018.

[13] Μανόλης Γιακουμάκης, Νίκος Διαμαντίδης. “Τεχνολογία Λογισμικού”. Εκδόσεις Σταμούλη, p. 207, 255, Μάρτιος 2009.

[14] FeatureIDE, http://featureide.cs.ovgu.de/update/v3/. Ιούνιος 2018.

[15] E(fx)clipse, http://download.eclipse.org/efxclipse/updates-released/3.0.0/site/?d. Ιούνιος 2018.