Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή στα Πληροφοριακά Συστήματα

Μεταγραφή Εφαρμογής Κατασκευής Δέντρων Απόφασης με Χρήση Γενετικών Αλγορίθμων

Νικόλαος Δρόσου Πουγούνιας

Επιβλέπων Καθηγητής
Δρ. Δημήτριος Καλλές

Αύγουστος 2012

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταγραφή Εφαρμογής Κατασκευής Δέντρων Απόφασης με Χρήση Γενετικών Αλγορίθμων

Νικόλαος Δρόσου Πουγούνιας

Επιβλέπων Καθηγητής
Δρ. Δημήτριος Καλλές

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε
προς μερική εκπλήρωση των απαιτήσεων για απόκτηση

μεταπτυχιακού τίτλου σπουδών
στα Πληροφοριακά Συστήματα

από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου

Αύγουστος 2012

Περίληψη

Μελετάται η μεταγραφή εφαρμογής που κατασκευάζει δένδρα απόφασης με τον αλγόριθμο GATree. Πρόκειται για έναν ευρετικό γενετικό αλγόριθμο βελτιστοποίησης, ο οποίος αναπαριστά το χώρο αναζήτησης χρησιμοποιώντας δυαδικά δένδρα. Στόχος είναι η δυνατότητα χρήσης της εφαρμογής μέσω διαδικτυακού περιβάλλοντος και η προετοιμασία της για αξιοποίηση υποδομών πλέγματος.

Αρχικά, έγινε ανάλυση της προϋπάρχουσας εφαρμογής σε επίπεδο λειτουργικών και τεχνικών χαρακτηριστικών. Αφού αξιολογήθηκαν ορισμένες έτοιμες βιβλιοθήκες γενετικών αλγορίθμων, αποφασίστηκε να δημιουργηθεί από την αρχή η απαραίτητη αντικειμενοστραφής υποδομή. Με τον τρόπο αυτό εξασφαλίζεται η ευελιξία για τη μελλοντική επέκταση της εφαρμογής, αποφεύγοντας συγκεκριμένους δομικούς περιορισμούς που παρουσιάζουν οι βιβλιοθήκες που εξετάστηκαν. Επικουρικά, επαναχρησιμοποιήθηκαν βιβλιοθήκες τρίτων για χρήσιμες περιφερειακές λειτουργίες.

Ορίστηκε μια ανοιχτή αρχιτεκτονική, με τον πυρήνα της εφαρμογής να έχει εκλεπτυνθεί σε αυτοτελή υποσυστήματα τα οποία επικοινωνούν μέσω καθορισμένων διεπαφών. Σε επίπεδο υλοποίησης, επιλέχθηκε η τεχνολογία Java συνοδευόμενη από τη συλλογή εργαλείων ανάπτυξης εταιρικών εφαρμογών Spring, ενώ το Maven χρησιμοποιήθηκε για τη διαχείριση του κύκλου ζωής.

Το νέο σύστημα κατασκευάζει δένδρα με παρόμοια ακρίβεια, υλοποιεί το σύνολο των λειτουργιών και είναι συμβατό ως προς τις παραμέτρους, την είσοδο και την έξοδο του αρχικού συστήματος. Επιπλέον, διαμορφώνεται η κατάλληλη τεχνική και μεθοδολογική υποδομή για την περαιτέρω επιστημονική και εμπορική αξιοποίησή του.

Λέξεις-κλειδιά: Τεχνητή Νοημοσύνη, Μηχανική Μάθηση, Γενετικοί Αλγόριθμοι, Δένδρα Απόφασης, Βελτιστοποίηση, GATree, Java, Spring, Maven, ΕΛ/ΛΑΚ.

Summary

In the present thesis, we investigate the re-engineering of a software application which constructs decision trees using the GATree optimization algorithm. Particularly, GATree is a heuristic genetic algorithm which uses binary trees to represent points in the search space. We aim to make the application accessible over the internet and to prepare it for possible integration with grid systems.

At first, we analyzed the functionality and technical specifications of the application. Then, we evaluated a few available packages for genetic algorithms. Since we detected certain structural limitations, we decided to construct a new object oriented infrastructure from scratch. In this way, we ensure that the growth of the new system will be more flexible in the future. Additionally, we reused third-party libraries for some peripheral operations.

We defined an open architecture refined in self-contained modules, which communicate with each other via well-defined interfaces. At the level of implementation, we chose the Java technology accompanied by the Spring application development framework. Also, we used Maven for managing the life-cycle of the software.

The new system constructs trees with similar accuracy, implements all functionality, and is compatible with the arguments and the I/O of the previous one. Moreover, we created the appropriate technical and methodological foundation for further utilizing it in the scientific and commercial fields.

Key-words: Artificial Intelligence, Machine Learning, Genetic Algorithms, Decision Trees, Optimization, GATree, Java, Spring, Maven, Open-source software.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον επιβλέποντα Καθηγητή Δημήτρη Καλλέ για την άψογη συνεργασία που έχουμε, την ανάθεση αυτού του εξαιρετικά ενδιαφέροντος θέματος και τη διαρκώς εποικοδομητική καθοδήγηση.

Επίσης, θα ήθελα να αποτείνω τις ιδιαίτερες ευχαριστίες μου στον ερευνητή Δρ. Αθανάσιο Παπαγγελή για την ουσιαστικότατη συνεισφορά του στο σύνολο της εργασίας αυτής, και στους Δρ. Νικόλαο Κωνσταντίνο και Δρ. Δημήτριο Σκούτα για το σχολιασμό του κειμένου.

Στο σημείο αυτό θα ήθελα να ευχαριστήσω όλους τους διδάσκοντες των τμημάτων στα οποία συμμετείχα, καθένας από τους οποίους έχει ξεχωριστή συμβολή στην εκπαιδευτική μου πορεία στο Μεταπτυχιακό Πρόγραμμα Σπουδών «Πληροφοριακά Συστήματα»: Δρ. Δημήτρης Καλλές - ΠΛΗΣ50, Δρ. Κυριάκος Βλάχος - ΠΛΗΣ51, Δρ. Παναγιώτης Ζαχαριάς - ΠΛΗΣ60, και Δρ. Μιχάλης Ξένος - ΠΛΗΣ61.

Τέλος, ευχαριστώ από καρδιάς τη σύζυγό μου Ειρήνη για την πλήρη συμπαράσταση κατά τη διάρκεια εκπόνησης της Μεταπτυχιακής αυτής Διατριβής.

Περιεχόμενα

Ευρετήριο Εικόνων

Ευρετήριο Πινάκων

Κεφάλαιο 1
Εισαγωγή

Η παρούσα Μεταπτυχιακή Διατριβή έχει ως αντικείμενο τη μεταγραφή εφαρμογής που κατασκευάζει δένδρα απόφασης με τον αλγόριθμο GATree (Genetic Algorithm Tree). Ο αλγόριθμος GATree κατατάσσεται στην οικογένεια των ευρετικών γενετικών αλγορίθμων βελτιστοποίησης [34]. Παρουσιάστηκε το 2000 από τους Παπαγγελή και Καλλέ [19]. Έκτοτε δόθηκε προς χρήση στην επιστημονική κοινότητα μέσω της ομώνυμης εφαρμογής. Η βελτίωση των επιδόσεων του αλγορίθμου ως προς το χρόνο και το χώρο αποτελεί πρόσφορο έδαφος για περαιτέρω έρευνα [08].

Η μεταγραφή απαιτεί τη διαχείριση πολλαπλών τεχνικών προκλήσεων με ταυτόχρονη συμβατότητα ως προς το επιστημονικό αποτέλεσμα. Λαμβάνει χώρα από ένα μείγμα C++ και Delphi προς την ανοιχτού κώδικα τεχνολογία Java, η οποία έχει να επιδείξει μια μακρά πορεία αξιοσημείωτης ωρίμανσης σε εταιρικές εφαρμογές ευρείας κλίμακας. Επιπλέον, γίνεται μετάβαση από μια στατική, σε μια ανοιχτή αρχιτεκτονική αυτοτελών υποσυστημάτων η οποία επιτρέπει την ευκολότερη επέκτασή της.

Στόχος είναι η δυνατότητα χρήσης της εφαρμογής μέσω διαδικτυακού περιβάλλοντος και η προετοιμασία της για αξιοποίηση υποδομών πλέγματος ή άλλων τεχνικών παραλληλισμού.

1.1 Εισαγωγικές έννοιες

Εξόρυξη δεδομένων είναι η ανάλυση μεγάλων συλλογών δεδομένων και η εξαγωγή χρήσιμων πληροφοριών από αυτές [27]. Στον Πίνακας 1 παρουσιάζεται μέρος από μια συλλογή ιατρικών δεδομένων.

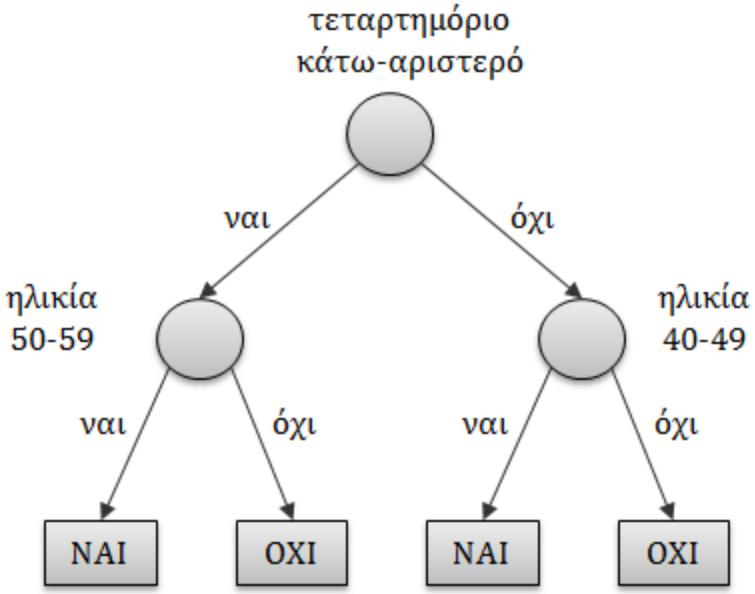
Ηλικία	Εμμηνόπαυση	Μέγεθος όγκου	Διακριτά οζίδια	Τεταρτημόριο	Επανεμφάνιση
40-49	προεμμηνόπαυση	15-19	0-2	άνω-αριστερό	ΝΑΙ
50-59	>= 40	15-19	0-2	κεντρικό	ΟΧΙ
50-59	>= 40	35-39	0-2	κάτω-αριστερό	ΝΑΙ
40-49	προεμμηνόπαυση	35-39	0-2	κάτω-αριστερό	ΟΧΙ
40-49	προεμμηνόπαυση	30-34	3-5	άνω-δεξί	ΝΑΙ
60-69	>= 40	15-19	0-2	άνω-αριστερό	ΟΧΙ
30-39	προεμμηνόπαυση	20-24	0-2	κεντρικό	ΟΧΙ

Πίνακας 1: Μέρος συλλογής ιατρικών δεδομένων για καρκινικούς όγκους στο στήθος γυναικών[1].

Στις στήλες αναγράφονται οι ιδιότητες (γνωρίσματα, attributes), η τελευταία εκ των οποίων είναι η κατηγορία (κλάση, class). Η τιμή της κλάσης χαρακτηρίζει όλη τη γραμμή. Η συλλογή αυτή καταγράφει εάν επανεμφανίστηκε καρκινικός όγκος σε μια γυναίκα που είχε νοσήσει στο παρελθόν. Το ζητούμενο είναι να μπορούν οι γιατροί να αποφανθούν, με βάση τα υπάρχοντα δεδομένα, εάν θα συμβεί επανεμφάνιση σε μια νέα ασθενή. Πρόκειται για ένα τυπικό πρόβλημα κατηγοριοποίησης (ταξινόμησης, κατάταξης, classification).

[1] Ljubljana University Medical Centre, Institute of Oncology, http://www.onko-i.si/eng

Κατηγοριοποίηση είναι η τοποθέτηση ενός αντικειμένου σε μία ή περισσότερες προκαθορισμένες κατηγορίες. Για το σκοπό αυτό ακολουθούμε τα εξής βήματα: Αρχικά, κατασκευάζουμε ένα μοντέλο με τον αλγόριθμο μάθησης (με χρήση των δεδομένων εκπαίδευσης). Στη συνέχεια, επικυρώνουμε το μοντέλο χρησιμοποιώντας πραγματικά δεδομένα (με χρήση των δεδομένων ελέγχου). Από εκεί και πέρα εφαρμόζουμε το μοντέλο σε καινούργια αντικείμενα [27].



Εικόνα 1: Δυαδικό δένδρο απόφασης

Το **δένδρο απόφασης** της **Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.** μοντελοποιεί τα δεδομένα του Πίνακας 1. Τα φύλλα αντιστοιχούν στο γνώρισμα της κλάσης, ενώ οι εσωτερικοί κόμβοι αντιστοιχούν σε ιδιότητες. Παρατηρούμε πως το δένδρο απόφασης, και μάλιστα το δυαδικό, είναι εύκολο στην κατανόηση. Επιπλέον, κατηγοριοποιεί γρήγορα νέα αντικείμενα (σε χρόνο γραμμικό ως προς το ύψος του δένδρου). Το γεγονός πως η κατασκευή βέλτιστων δένδρων απόφασης χαρακτηρίστηκε δυσεπίλυτο πρόβλημα, προκάλεσε έντονο ερευνητικό ενδιαφέρον και έδωσε ώθηση στην ανάπτυξη διαφόρων ευρετικών αλγορίθμων και τεχνικών.

Οι **γενετικοί αλγόριθμοι** είναι μια οικογένεια αλγορίθμων με επιρροές από συγκεκριμένα φαινόμενα της γενετικής, όπως η διασταύρωση, η μετάλλαξη και η φυσική επιλογή. Δημιουργούν έναν πληθυσμό από υποψήφιες λύσεις, τον οποίο αξιολογούν κι εξελίσσουν μέσα από πολλές επαναλήψεις [34]. Χαρακτηριστικό στοιχείο διαφοροποίησης σε σχέση με τους επαγωγικούς αλγόριθμους βελτιστοποίησης δένδρων (π.χ. ID3, C4.5), οι οποίοι υιοθετούν μιαν άπληστη

στρατηγική αναζήτησης παίρνοντας μια σειρά από τοπικά βέλτιστες αποφάσεις, είναι πως συνήθως αποφεύγουν τα τοπικά ακρότατα, στοχεύοντας στο γενικά βέλτιστο. Επιπλέον, είναι αχόρταγοι ως προς τους υπολογιστικούς πόρους (επεξεργαστική ισχύς, χρόνος, μνήμη) [19].

1.2 Μεθοδολογία

Αρχικά δόθηκε έμφαση στη μελέτη της βιβλιογραφίας με σκοπό την περαιτέρω εμβάθυνση στα δένδρα απόφασης, ως μέσο μηχανικής μάθησης. Επιπλέον, κρίθηκε απαραίτητη η στιβαρή κατανόηση των γενετικών αλγορίθμων, ιδιαίτερα όσον αφορά την εφαρμογή τους στη βελτιστοποίηση δένδρων απόφασης. Για το λόγο αυτό, η βιβλιογραφία διακρίνεται σε βασική και εκτεταμένη. Η πρώτη παρουσιάζει τις απαραίτητες έννοιες στις οποίες στηρίζεται η μεταπτυχιακή διατριβή (π.χ. δένδρα απόφασης, γενετικοί αλγόριθμοι, GATree). Στη δεύτερη έχουν επιλεγεί πηγές που ασχολούνται με ένα μεγαλύτερο εύρος θεμάτων μέσα από το οποίο μπορούν να δοθούν ερευνητικές κατευθύνσεις (π.χ. παράλληλοι μέθοδοι, τεχνικές βελτιστοποίησης). Η προεργασία αυτή συνετέλεσε στη δημιουργία της απαραίτητης επιστημονικής βάσης και στην ωρίμανση των θεμελιωδών εννοιών που αφορούν τη μεταπτυχιακή αυτή διατριβή.

Έπειτα ξεκίνησε η απαραίτητη εξοικείωση με την υπάρχουσα εφαρμογή, μέσω της εκτεταμένης χρήσης της. Για το σκοπό αυτό πραγματοποιήθηκαν διάφορες εκτελέσεις πάνω στις 15 συλλογές δεδομένων (datasets) που συνοδεύουν την εφαρμογή. Πολύτιμες πληροφορίες και διευκρινήσεις αντλήθηκαν από τους επιστήμονες που την ανέπτυξαν [20] κατά τις συναντήσεις μαζί τους. Ακόμα, μελετήθηκε επιλεγμένο μέρος του πηγαίου κώδικα σε C++. Στα πλαίσια του σχηματισμού πρώιμων ιδεών, εντοπίστηκε η προγραμματιστική διεπαφή του Weka [06] η οποία διαβάζει datasets. Με στοχευμένη επισκόπηση του πηγαίου κώδικα του Weka, έγινε αξιολόγηση του τρόπου με τον οποίο αναπαρίσταται ένα dataset και αναγνωρίζονται τα επιμέρους κομμάτια του.

Στη συνέχεια δοκιμάστηκαν έτοιμες βιβλιοθήκες που επιτρέπουν την ανάπτυξη γενετικών αλγορίθμων. Πιο συγκεκριμένα, έγινε αξιολόγηση των ECJ (Java-based Evolutionary Computation Research System [11]) και JGAP (Java Genetic Algorithms Package [12]). Και τα δύο διαθέτουν εξαιρετικά ανεπτυγμένες δυνατότητες. Το μεν πρώτο παρέχει πολύ υψηλή προσαρμοστικότητα, μέσω μιας ιεραρχικής δομής απλών εξωτερικών αρχείων κειμένου [33]. Το δεύτερο ορίζει ένα σαφώς καθορισμένο προγραμματιστικό μοντέλο, κι επιπλέον παρέχει κάποιες ευκολίες

παραλληλισμού εργασιών. Ο βασικός τους περιορισμός, από την οπτική γωνία της παρούσας μεταπτυχιακής διατριβής, είναι πως δεν δίνουν στον προγραμματιστή τη δυνατότητα παρεμβάσεων σε βάθος. Για λόγους ευελιξίας κι επεκτασιμότητας, αποφασίστηκε να δημιουργηθεί από την αρχή η απαραίτητη αντικειμενοστραφής υποδομή για το γενετικό αλγόριθμο και την εκτέλεσή του. Αυτή είναι μια βασική σχεδιαστική διαφοροποίηση σε σχέση με την αρχική εφαρμογή, η οποία βασίζεται στο GAlib (Genetic Algorithms Library [28]).

Ο σχεδιασμός του νέου συστήματος επιστρατεύει επιλεγμένες πρακτικές της ICONIX [23][24]. Είναι εξάλλου εμφανής η επιρροή της συγκεκριμένης διαδικασίας στην ορολογία συγκεκριμένων κεφαλαίων. Πέραν τούτου, αξίζει να σημειωθεί η σημαντική επίδρασή της στον τρόπο εργασίας, ο οποίος δεν αποτυπώνεται εξ ολοκλήρου στο παρόν κείμενο.

Όσον αφορά την καθεαυτή υλοποίηση του νέου συστήματος, επιδιώχθηκε να γίνει χρήση της επαναληπτικής μεθοδολογίας SCRUM [22]. Το εγχείρημα προσαρμογής μιας ευέλικτης μεθοδολογίας στις ιδιαίτερες συνθήκες της παρούσας μεταπτυχιακής διατριβής, εν τέλει, δεν τελεσφόρησε. Παρ' όλα αυτά, αποτέλεσε ενδιαφέρουσα ιδέα η οποία χρήζει περαιτέρω διερεύνησης.

1.3 Δομή της Διατριβής

Στο Κεφάλαιο 2 παρουσιάζεται ο αλγόριθμος GATree. Περιγράφονται οι βασικές αρχές οι οποίες τον διέπουν, το γονίδιο που χρησιμοποιεί και οι τεχνικές εξέλιξης που επιστρατεύει. Επίσης, συγκρίνεται με κλασικούς άπληστους αλγορίθμους και διερευνώνται διάφορες ιδέες επέκτασής του.

Στο Κεφάλαιο 3 γίνεται επισκόπηση της υπάρχουσας εφαρμογής, η οποία είναι προς μεταγραφή. Καταγράφονται τα τεχνικά χαρακτηριστικά της, η λειτουργικότητα που επιτελεί, η μορφή της εισόδου και της εξόδου της, καθώς και το σύνολο των παραμέτρων που δέχεται.

Στο Κεφάλαιο 4 αποτυπώνεται το πεδίο του προβλήματος υπό μορφή μοντέλου. Πρόκειται για το «λεξικό δεδομένων» του έργου – μια βάση κατανόησης των αντικειμένων που αποτελούν τα συστατικά κομμάτια της μεταγραφής.

Στο Κεφάλαιο 5 τεκμηριώνεται η τεχνική αρχιτεκτονική. Δίνεται μια συνολική εικόνα του συστήματος, ορίζεται με σαφή τρόπο η δομή και λαμβάνονται οι απαραίτητες τεχνολογικές αποφάσεις.

Στο Κεφάλαιο 6 καταγράφονται μία-προς-μία οι βελτιώσεις που έχει να επιδείξει η νέα εφαρμογή σε επίπεδο υποδομής, λειτουργικότητας και παραμετροποίησης, σε σχέση με την παλαιότερη.

Στο Κεφάλαιο 7 προτείνονται τεχνικές ταυτοχρονισμού και αξιοποίησης υποδομών παράλληλου υπολογισμού.

Στο Κεφάλαιο 8 γίνεται ανασκόπηση της δουλειάς και συνοψίζονται τα τελικά συμπεράσματα.

Στο Παράρτημα Α εξηγείται η διαχείριση έργου με το Maven και πώς αυτή εντάσσεται σε έναν ολοκληρωμένο κύκλο ζωής του λογισμικού.

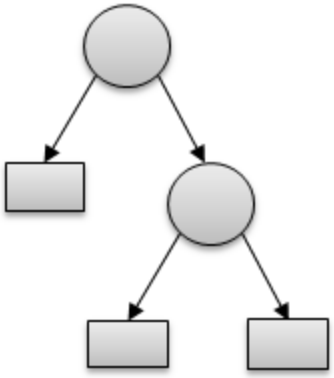
Στο Παράρτημα Β παρουσιάζεται η ευέλικτη μεθοδολογία SCRUM και μελετάται η προσπάθεια εφαρμογής της στα πλαίσια της παρούσας μεταπτυχιακής διατριβής.

Κεφάλαιο 2
Ο αλγόριθμος GATree

Ο γενετικός αλγόριθμος GATree εφαρμόζεται στη βελτιστοποίηση δένδρων απόφασης. Αρχικά, δημιουργείται ένας πληθυσμός αποτελούμενος από ελάχιστα δυαδικά δένδρα, σε κάθε κόμβο των οποίων προσδίδεται μία τυχαία ιδιότητα και μια τυχαία τιμή. Έπειτα, βαθμολογείται η ακρίβεια κάθε δένδρου με τη χρήση της συνάρτησης αξιολόγησης. Στη συνέχεια, ο πληθυσμός εξελίσσεται μέσω των τεχνικών της μετάλλαξης, της διασταύρωσης και της επιλογής, μέχρι να συμπληρωθεί ο ζητούμενος αριθμός γενεών. Το δένδρο με τη μεγαλύτερη βαθμολογία αποτελεί το αποτέλεσμα της διαδικασίας [19].

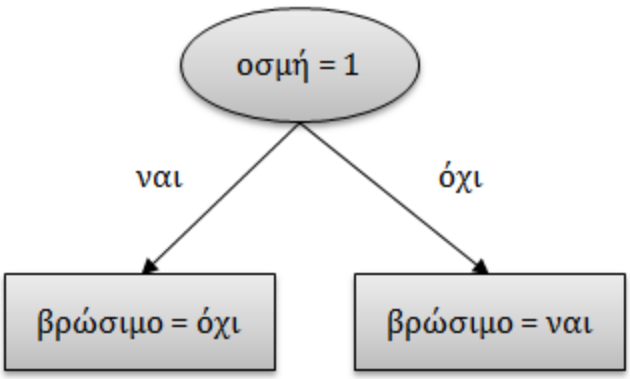
2.1 Χρωμόσωμα

Βασικό συστατικό στοιχείο αποτελεί το δυαδικό δένδρο (Εικόνα 2), με τη βοήθεια του οποίου αναπαρίσταται ο χώρος αναζήτησης. Επιπλέον, είναι προϋπόθεση να χρησιμοποιούνται πραγματικά δυαδικά δένδρα κατά την υλοποίηση του αλγορίθμου. Το γεγονός αυτό ευνοεί τη χρήση αντικειμενοστρεφούς τεχνολογίας.



Εικόνα 2: Ένα δυαδικό δένδρο.

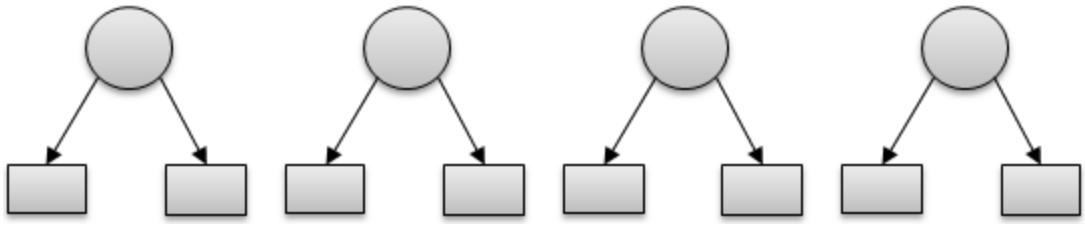
Η ρίζα και οι εσωτερικοί κόμβοι του δένδρου σχετίζονται με κάποια ιδιότητα, ενώ τα φύλλα με την κατάταξη του στιγμιότυπου. Επίσης, κατά σύμβαση η αριστερή ακμή αποτυπώνει το «αληθές γεγονός» (ναι, true), ενώ η δεξιά ακμή το «ψευδές γεγονός» (όχι, false). Στην Εικόνα 3 παρουσιάζεται ένα απλοποιημένο παράδειγμα, που αποφαίνεται εάν ένα μανιτάρι είναι βρώσιμο [09].



Εικόνα 3: Ένα δυαδικό δένδρο απόφασης.

2.2 Αρχικοποίηση

Ο αρχικός πληθυσμός αποτελείται από έναν αριθμό ελάχιστων δυαδικών δένδρων απόφασης, όπως φαίνεται στην Εικόνα 4. Ελάχιστο είναι το δυαδικό δένδρο που αποτελείται από τη ρίζα και δύο ακριβώς φύλλα· συνολικά από τρείς κόμβους.



Εικόνα 4: Ένας πληθυσμός από ελάχιστα δυαδικά δένδρα απόφασης.

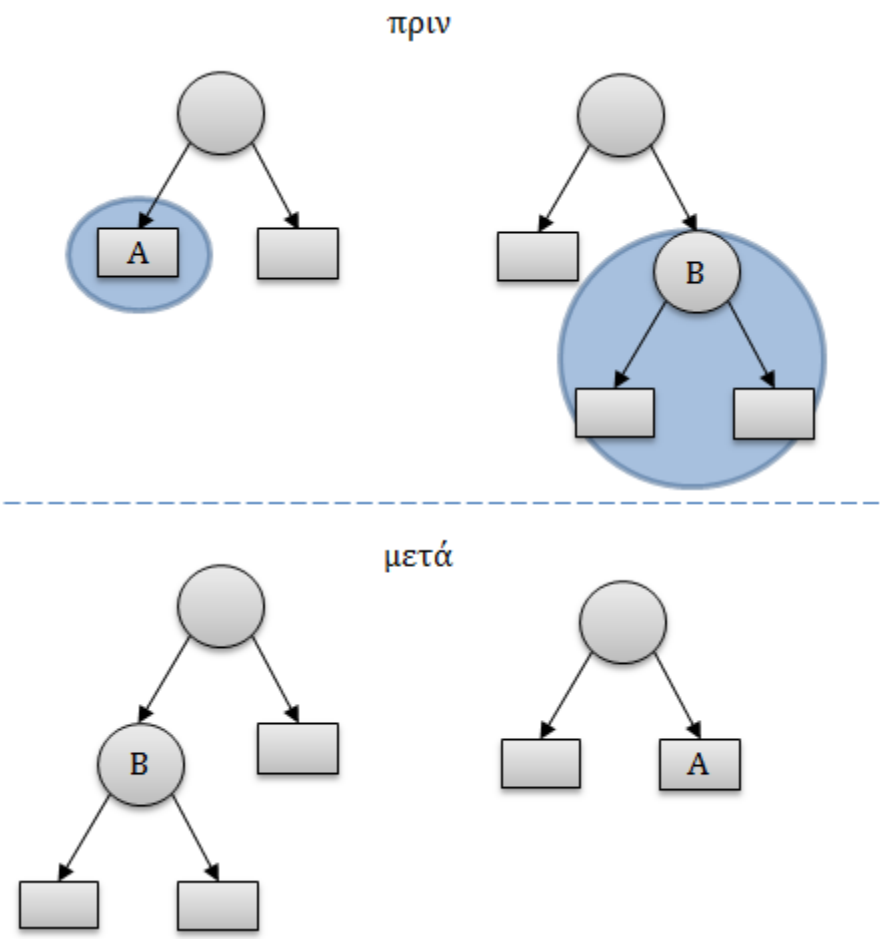
Χαρακτηριστικό κάθε δένδρου του αρχικού πληθυσμού είναι ο τυχαίος τρόπος με τον οποίο κατασκευάζεται. Για κάθε εσωτερικό κόμβο επιλέγεται τυχαία μια ιδιότητα και εν συνεχεία μια τυχαία τιμή της συγκεκριμένης ιδιότητας. Με τον ίδιο τρόπο, για κάθε φύλλο επιλέγεται μια τυχαία κλάση και μια τυχαία τιμή της κλάσης.

2.3 Διαφοροποίηση

2.3.1 Διασταύρωση

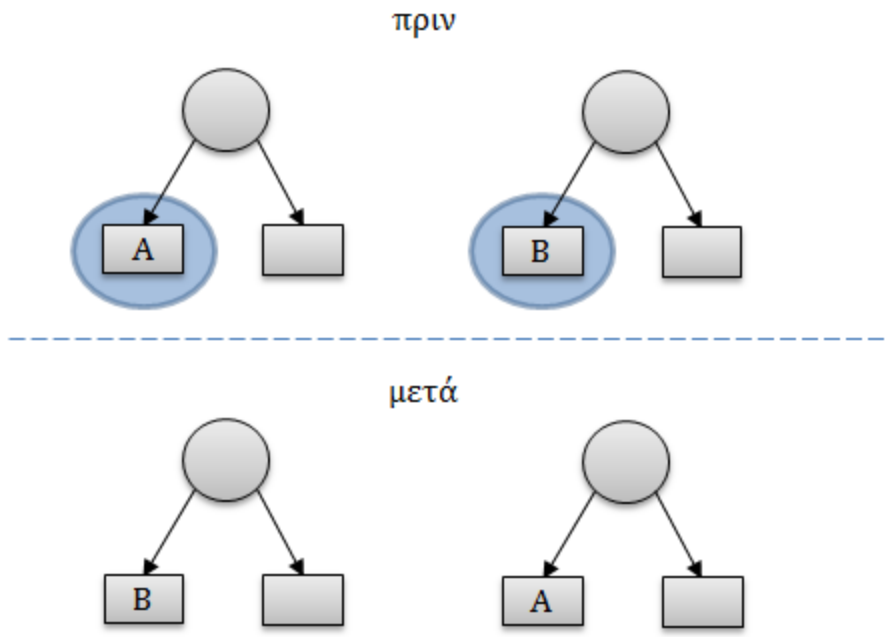
Κατά τη διασταύρωση δύο δένδρων, επιλέγεται ένας τυχαίος κόμβος από το καθένα και ανταλλάσσονται τα υποδένδρα τους. Πρόκειται για τη λεγόμενη «διασταύρωση ενός σημείου» [32].

Στο παράδειγμα της Εικόνα 5, έχουν επιλεγεί ο κόμβος Α από το πρώτο δένδρο και ο κόμβος Β από το δεύτερο δένδρο. Ο κόμβος Α είναι φύλλο, οπότε αποτελεί από μόνος του ένα υποδένδρο. Αντίθετα, ο κόμβος Β αποτελεί τη ρίζα ενός υποδένδρου μεγέθους 3. Κατά τη διασταύρωση, λαμβάνει χώρα ανταλλαγή των συγκεκριμένων υποδένδρων.



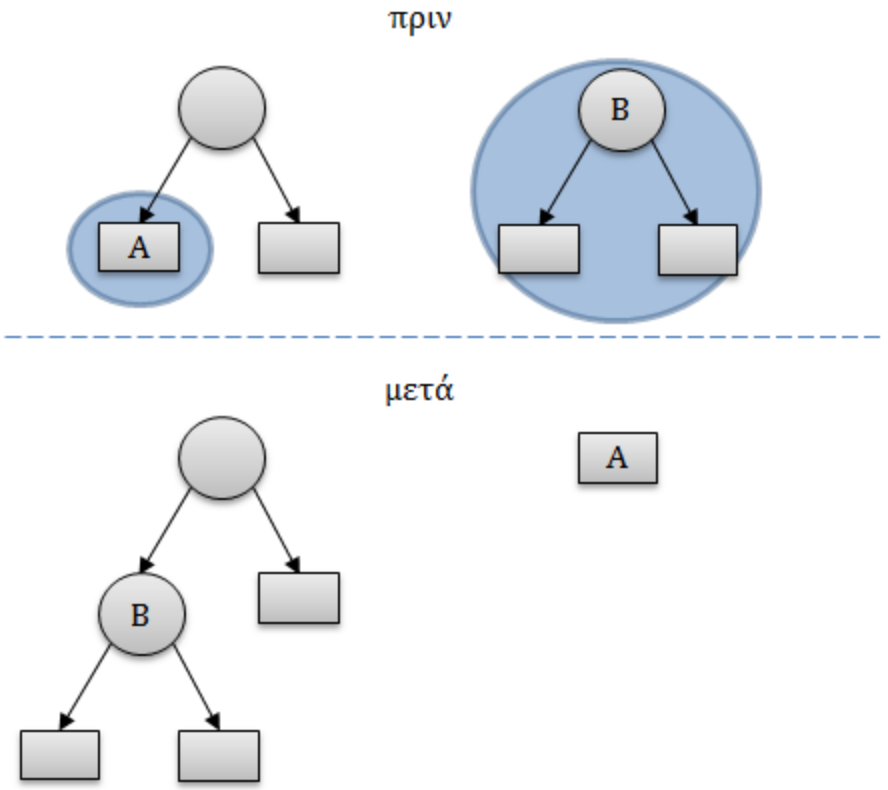
Εικόνα 5: Παράδειγμα διασταύρωσης.

Μια απλή περίπτωση παρουσιάζεται όταν και οι δύο επιλεγμένοι κόμβοι είναι φύλλα (Εικόνα 6).



Εικόνα 6: Διασταύρωση όπου οι επιλεγμένοι κόμβοι είναι φύλλα.

Όταν η διασταύρωση λαμβάνει χώρα μεταξύ ρίζας και φύλλου, τότε ένα από τα παραγόμενα δένδρα έχει βάθος 0 (Εικόνα 7).

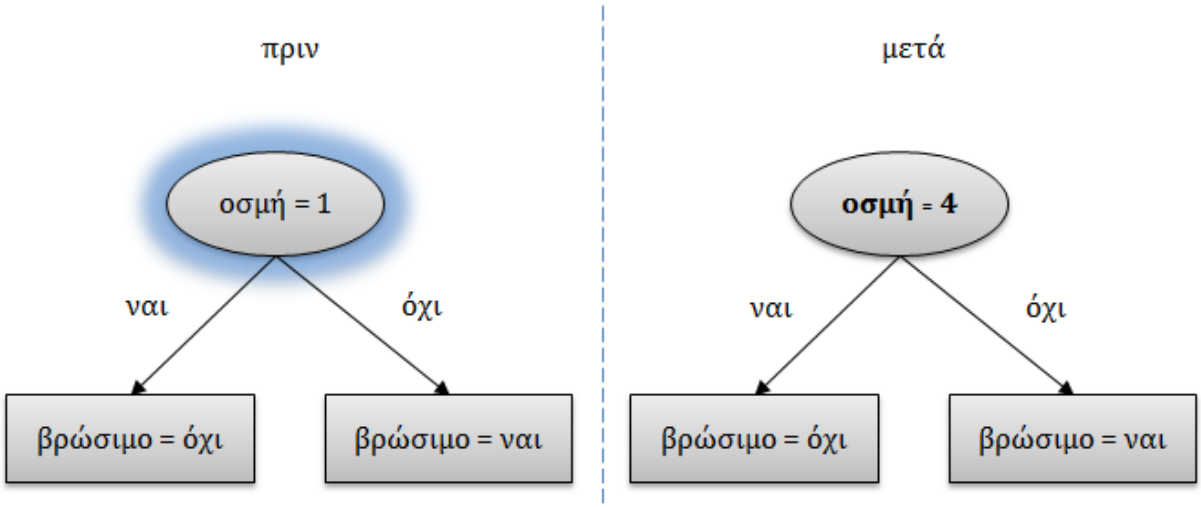


Εικόνα 7: Διασταύρωση μεταξύ ενός φύλλου και μιας ρίζας.

Τέλος, τα δένδρα παραμένουν ανέπαφα όταν επιλεγούν οι ρίζες τους.

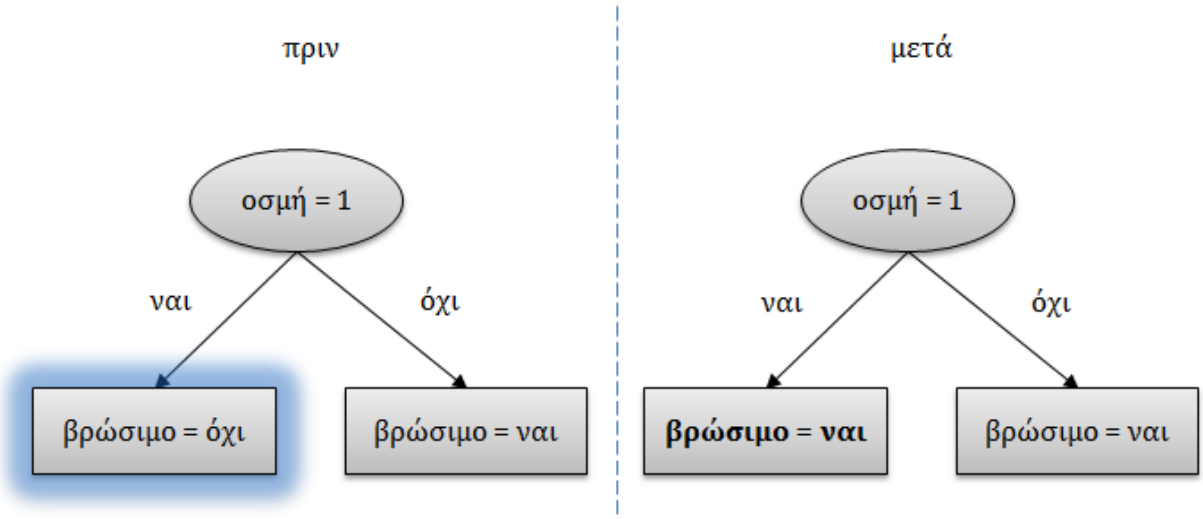
2.3.2 Μετάλλαξη

Κατά τη μετάλλαξη ενός κόμβου επιλέγεται τυχαία μια διαφορετική τιμή για την ιδιότητα που φέρει, όπως παρουσιάζεται στην Εικόνα 8.



Εικόνα 8: Μετάλλαξη ενός εσωτερικού κόμβου.

Παρόμοια, όταν ένα φύλλο μεταλλάσσεται απλά αλλάζει η τιμή της κλάσης του (Εικόνα 9).



Εικόνα 9: Μετάλλαξη ενός φύλλου.

2.4 Αξιολόγηση

Η αντικειμενική συνάρτηση δέχεται ως είσοδο ένα δένδρο και επιστρέφει έναν αριθμό που υποδηλώνει το πόσο «κατάλληλο» είναι. [33]. Πιο συγκεκριμένα, η συνάρτηση του αλγορίθμου GATree ισορροπεί μεταξύ ακρίβειας και μεγέθους [20].

$$payoff(tree_i) = CorrectClassified_i^2 * \frac{x}{size_i^2 + x}$$

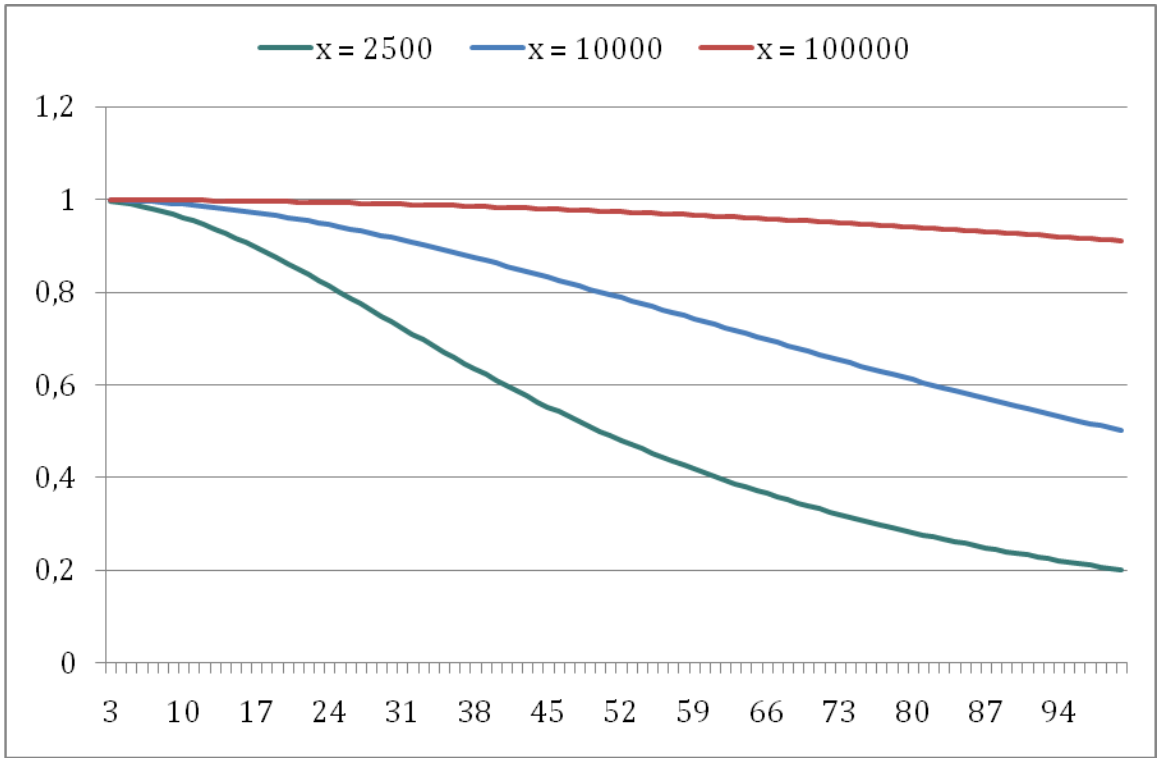
Όπου:

- $size$ είναι το μέγεθος του δένδρου.
- $CorrectClassified$ είναι η ακρίβεια του δένδρου [27].
- x είναι μια παράμετρος η οποία εξηγείται παρακάτω.

Με την πρώτη επαφή, θεωρείται μια καλή συνάρτηση καθώς είναι συνεχής και μονότονη. Με τη συνέχεια αποφεύγονται τα νεκρά σημεία, όπου ενδεχομένως χρειάζονται προσεγγίσεις. Επιπλέον, αποφεύγονται οι απότομες μεταβολές στις τιμές της. Η μονοτονία έρχεται να συμπληρώσει τη συνέχεια παρέχοντας τη δυνατότητα ποιοτικής εκτίμησης. Για παράδειγμα, αναμένουμε όσο μικρότερο να είναι το όρισμα, τόσο μικρότερη να είναι η τιμή της. Τέλος η συνάρτηση είναι απλή, καθώς δεν φαίνεται να απαιτεί πολύπλοκους υπολογισμούς [33].

Ο πρώτος παράγοντας του γινομένου δηλώνει πως όσο μεγαλύτερη είναι η ακρίβεια, τόσο μεγαλύτερη είναι η βαθμολογία του δένδρου. Από το δεύτερο παράγοντα του γινομένου προκύπτει πως όσο μεγαλύτερο είναι το μέγεθος του δένδρου, τόσο μικρότερη είναι η βαθμολογία του.

Σημείο κλειδί αποτελεί η παράμετρος x. Στο διάγραμμα της Εικόνα 10 αποτυπώνεται ο δεύτερος παράγοντας της αντικειμενικής συνάρτησης (κατακόρυφος άξονας) σε σχέση με το μέγεθος του δένδρου (οριζόντιος άξονας) για διαφορετικές τιμές της παραμέτρου x.



Εικόνα 10: Ο παράγοντας μεγέθους της αντικειμενικής συνάρτησης σε σχέση με το μέγεθος του δένδρου για δεδομένες τιμές της παραμέτρου x.

- Είναι προφανές πως ευνοούνται τα μικρότερα δένδρα, καθώς –ανεξάρτητα από την τιμή της παραμέτρου- οι καμπύλες έχουν αρνητική κλίση.

- Η εύνοια προς τα μικρότερα δένδρα μεγαλώνει όσο μικραίνει η παράμετρος. Πράγματι, ο παράγοντας των μεγαλύτερων δένδρων μειώνεται γρηγορότερα όσο μικρότερη είναι η παράμετρος. Στην περίπτωση αυτή, τα μεγαλύτερα δένδρα έχουν σαφώς λιγότερες πιθανότητες να επιζήσουν κατά το πέρασμα στην επόμενη γενιά.

- Η αναζήτηση αναμένεται να εκτείνεται σε μεγαλύτερο χώρο, όσο μεγαλύτερη είναι η παράμετρος. Αυτό συμβαίνει επειδή εμπλέκονται δένδρα ολοένα και μεγαλύτερου μεγέθους.

Από τα παραπάνω, συμπεραίνουμε πως η επιλογή της παραμέτρου x είναι στενά συνδεδεμένη με το μέγεθος του δένδρου. Δίνεται έτσι η δυνατότητα στον ερευνητή να κατευθύνει το πείραμα προς συγκεκριμένο αποτέλεσμα - μικρό x σημαίνει μικρότερα δένδρα και το αντίστροφο.

Κεφάλαιο 3
Επισκόπηση υπάρχουσας εφαρμογής

Στο παρόν κεφάλαιο γίνεται επισκόπηση της εφαρμογής που είναι προς μεταγραφή. Με τον τρόπο αυτό αναδεικνύονται οι λειτουργικές και μη απαιτήσεις για το νέο σύστημα.

3.1 Τεχνικά χαρακτηριστικά

Το υπάρχον σύστημα είναι μια εφαρμογή desktop η οποία είναι σχεδιασμένη σε 2 επίπεδα. Ο πυρήνας είναι γραμμένος σε C++, ενώ η γραφική διεπαφή και οι περιφερειακές λειτουργίες σε Delphi[2]. Ο πυρήνας είναι ανεξάρτητος από τη Delphi, κι έτσι μπορεί να χρησιμοποιείται αυτόνομα στη γραμμή εργασιών.

Σε επίπεδο εξαρτήσεων, αξίζει να σημειωθεί πως το σύστημα βασίζεται στη βιβλιοθήκη γενετικών αλγορίθμων GAlib [28], η οποία σχεδιάστηκε ώστε να είναι εύκολη η δοκιμή διάφορων αντικειμενικών συναρτήσεων, αναπαραστάσεων, γενετικών τελεστών και αλγορίθμων.

[2] http://en.wikipedia.org/wiki/Object_Pascal

Επίσης, χρησιμοποιείται το εργαλείο VCG [25] για την οπτική αναπαράσταση των αποτελεσμάτων. Το γεγονός αυτό, μεταξύ άλλων, καθορίζει τη μορφή των αρχείων εξόδου. Το VCG απεικονίζει γράφους [01], διαβάζοντας τις προδιαγραφές από ένα αρχείο κειμένου με συγκεκριμένη σύνταξη. Περιλαμβάνει διάφορες ευρετικές μεθόδους, ιδιαίτερα για την καλύτερη τοποθέτηση των κόμβων στο χώρο.

3.2 Είσοδος

Ένα αρχείο τύπου ARFF (Attribute – Relation File Format) αποτυπώνει πλήρως μια συλλογή δεδομένων (dataset). Περιλαμβάνει τόσο τον ορισμό των ιδιοτήτων της, όσο και τα ίδια τα στιγμιότυπα [09]. Πρόκειται για μια μορφή κειμένου ASCII που αναπτύχθηκε από την ομάδα του Weka [06]. Να σημειώσουμε πως το υπάρχον σύστημα δεν υλοποιεί όλο το ARFF.

3.2.1 Επικεφαλίδα

Καταρχήν δηλώνεται το όνομα της συλλογής με το πεδίο @Relation. Ακολουθεί ο ορισμός των ιδιοτήτων με το πεδίο @Attribute (Εικόνα 11).

```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute windy {true, false}
@attribute temperature numeric
@attribute humidity numeric
@attribute play {yes, no}
```

Εικόνα 11: Παράδειγμα επικεφαλίδας ενός αρχείου ARFF. [31]

Οι πλέον χρησιμοποιούμενοι τύποι ιδιοτήτων είναι ο αριθμητικός και ο ονομαστικός. Μια ιδιότητα δηλώνεται αριθμητική με τη λέξη numeric. Εάν η ιδιότητα είναι ονομαστική, τότε ο ορισμός της παραθέτει τις επιτρεπόμενες τιμές.

3.2.2 Δεδομένα

Το πεδίο @Data ορίζει την έναρξη παράθεσης των στιγμιότυπων (Εικόνα 12).

```
@data
sunny, 80, 90, true, no
rainy, 65, 70, true, no
sunny, 85, 85, false, no
rainy, 70, 96, false, yes
rainy, 68, 80, false, yes
overcast, 83, 86, false, yes
```

Εικόνα 12: Παράδειγμα δεδομένων ενός αρχείου ARFF [31].

Οι τιμές κάθε στιγμιότυπου χωρίζονται με κόμμα (,) ενώ το λατινικό ερωτηματικό (?) δηλώνει μια μη γνωστή τιμή.

3.2.3 Ιδιαιτερότητες

Ο τύπος αρχείου ARFF είναι ακόμα στενά συνδεδεμένος με το Weka. Κατά συνέπεια δεν έχει ξεκινήσει ακόμα η διαδικασία προτυποποίησής του. Το γεγονός αυτό οδηγεί στην εμφάνιση μικρών συντακτικών διαφορών ανάμεσα σε διαφορετικές συλλογές δεδομένων.

1. Η χρήση κεφαλαίων – πεζών: Τα πεδία ορίζονται με κεφαλαία (@ATTRIBUTE), μικρά (@attribute) ή μεικτά (@Attribute) γράμματα.

2. Η χρήση κενού χαρακτήρα για το διαχωρισμό διακριτών τιμών: Κάποια datasets χρησιμοποιούν κενό μετά το κόμμα (1, 2, 3), ενώ κάποια άλλα όχι (1,2,3). Αυτό εμφανίζεται κατά τον ορισμό ιδιοτήτων ή κατά την παράθεση των τιμών σε ένα στιγμιότυπο.

3. Η χρήση κενών γραμμών. Πολλές συλλογές χρησιμοποιούν κενές γραμμές για εικαστικούς λόγους, π.χ. για να διαχωρίσουν την επικεφαλίδα από τα δεδομένα. Αντίθετα, άλλες συλλογές δεν έχουν καμία κενή γραμμή.

4. Η χρήση εισαγωγικών. Παρατηρείται η χρήση απλών εισαγωγικών για τη δήλωση του ονόματος μιας ιδιότητας, των τιμών μιας ιδιότητας, ακόμα και της άγνωστης τιμής.

5. Η χρήση χαρακτήρα διαφυγής (escape character) στις τιμές μιας ιδιότητας.

6. Η χρήση παλαιότερης σύνταξης. Παρατηρείται συχνά η χρήση συντακτικών μορφωμάτων τα οποία δεν υποστηρίζονται πλέον από το Weka, όπως οι λέξεις integer και real.

3.3 Έξοδος

Η εφαρμογή υποστηρίζει τρεις τύπους αρχείων εξόδου, ανάλογα με τη χρήση τους.

3.3.1 Αρχείο αποτελέσματος

Πρόκειται για ένα απλό αρχείο κειμένου το οποίο παρουσιάζει το τελικό αποτέλεσμα, όπως φαίνεται στην Εικόνα 13.

```
100 0.733007 0.861386 0.711218 35 0.861386 100000 0
 if toothed=true then
 |-if venomous=true then
 | |-if hair=true then
 | | |-bird
 | | +-fish
 | +-if eggs=false then
 |   |-mammal
 |   +-if toothed=false then
 |     |-bird
 |     +-fish
```

Εικόνα 13: Το αρχείο εξόδου με το αποτέλεσμα της βελτιστοποίησης.

Στην πρώτη γραμμή αναγράφονται τα εξής στοιχεία, με τη σειρά που εμφανίζονται:

- Ο αριθμός της γενιάς στην οποία κατέληξε ο αλγόριθμος.
- Η αξιολόγηση του καλύτερου δένδρου, με βάση την αντικειμενική συνάρτηση.
- Η ακρίβεια του καλύτερου δένδρου ως προς τα δεδομένα εκπαίδευσης (train data).
- Ο μέσος όρος αξιολόγησης.

- Το μέγεθος του καλύτερου δένδρου.
- Η ακρίβεια του καλύτερου δένδρου ως προς τα δεδομένα ελέγχου (test data).
- Η τιμή της παραμέτρου x.

Στη συνέχεια αποτυπώνεται το καλύτερο δένδρο με έναν απλό δομημένο τρόπο. Η σύνταξη αποτελείται από ορισμένες λέξεις-κλειδιά γλώσσας προγραμματισμού (if, then, false, true), σε συνδυασμό με σήμανση εντολών τύπου tree των λειτουργικών συστημάτων.

3.3.2 Αρχείο οπτικής απεικόνισης

Πρόκειται για το αρχείο που δίνεται στο εργαλείο VCG για την οπτική απεικόνιση του αποτελέσματος (Εικόνα 14). Ακολουθείται η γλώσσα GDL (Graph Description Language), η οποία περιγράφει γραφήματα και τη διάταξη των στοιχείων που το αποτελούν [25].

```
graph:{
textmode:center
layoutalgorithm:tree
xspace:5
yspace:15
scaling:0.8
smanhattan_edges:yes
node:{title:"0047DC10" label:"if Output6<=0 then " textcolor:yellow color:red}
edge:{thickness:3 sourcename:"0047DC10" targetname:"0047A398"}
node:{title:"0047A398" label:"if Chooser2<=0 then " textcolor:yellow color:red}
edge:{thickness:3 sourcename:"0047A398" targetname:"00493C50"}
node:{title:"00493C50" label:"1" textcolor:yellow color:red}
edge:{thickness:3 sourcename:"0047A398" targetname:"00481440"}
node:{title:"00481440" label:"0" textcolor:yellow color:red}
```

Εικόνα 14: Μέρος του αρχείου εξόδου για την οπτική απεικόνιση του δένδρου.

Στην Εικόνα 15 παρουσιάζεται ένα μικρό υποσύνολο του συντακτικού της γλώσσας GDL. Το γράφημα αποτελείται από κόμβους και ακμές. Κάθε κόμβος φέρει έναν μοναδικό τίτλο. Επίσης, κάθε ακμή ορίζεται μεταξύ ακριβώς δύο κόμβων.

```
graph: {
    <ιδιότητες γράφου>
    <κόμβοι>
    <ακμές>
}

node: {
    title: <τίτλος κόμβου>
    <ιδιότητες κόμβου>
}

edge: {
    sourcename: <τίτλος κόμβου αφετηρίας>
    targetname: <τίτλος κόμβου προορισμού>
    <ιδιότητες ακμής>
}
```

Εικόνα 15: Υποσύνολο της σύνταξης GDL για την οπτική απεικόνιση γραφημάτων.

Αξίζει να σημειωθεί πως η ιεραρχική αυτή δομή αναπτύχθηκε σε μια εποχή που δεν υπήρχε ακόμα η XML[3] για την αναπαράσταση δεδομένων. Στις μέρες μας, η ίδια σύνταξη χρησιμοποιείται ακόμα ευρέως σε αντικειμενοστρεφείς γλώσσες παραγωγής πλούσιων οπτικά εφαρμογών όπως η JavaFX[4].

3.3.3 Αρχείο επικύρωσης με διασταύρωση

Παρουσιάζει το αποτέλεσμα της επικύρωσης με διασταύρωση τμημάτων (k-fold cross validation), όπως φαίνεται στην Εικόνα 16.

[3] http://en.wikipedia.org/wiki/XML
[4] http://javafx.com/

```
Training size:337 Testing Size:112

Results for Validation:1  Accuracy:0.794643 Correct classified:89 out of 112
Results for Validation:2  Accuracy:0.8125 Correct classified:91 out of 112
Results for Validation:3  Accuracy:0.9375 Correct classified:105 out of 112
Results for Validation:4  Accuracy:0.839286 Correct classified:94 out of 112

------------------------------------------------------------
Average Accuracy:0.845982 Average Fitness:0.714403 AverageSize:11
```

Εικόνα 16: Αρχείο εξόδου επικύρωσης με διασταύρωση 4 τμημάτων.

Αρχικά αναγράφεται ο αριθμός των στιγμιότυπων εκπαίδευσης και ελέγχου. Έπειτα, παρουσιάζονται τα αποτελέσματα της επικύρωσης ανά τμήμα. Πιο συγκεκριμένα, καταγράφεται ο αριθμός του τμήματος, η ακρίβεια και ο αριθμός των στιγμιότυπων για τα οποία έγινε σωστή κατάταξη.

3.4 Παράμετροι

Στον Πίνακας 2 ακολουθεί καταγραφή των παραμέτρων της εφαρμογής (command line arguments), οι πιθανές τιμές τους και η σημασία τους.

-gen	100	Αριθμός γενεών.
-pop	50	Αριθμός πληθυσμού.
-c	0.95	Πιθανότητα διασταύρωσης.
-m	0.01	Πιθανότητα μετάλλαξης.
-r	0.25	Ποσοστό αναπλήρωσης πληθυσμού.
-ch	0	Τύπος διασταύρωσης.
-mh	1	Τύπος μετάλλαξης.

-cv	10	Ο αριθμός των τμημάτων για την επικύρωση με διασταύρωση.
-split	2	Σε πόσα datasets να χωριστεί το δοσμένο αρχείο συλλογής δεδομένων.
-xmin	40000	Η ελάχιστη τιμή του παράγοντα x.
-xmax	4000000	Η μέγιστη τιμή του παράγοντα x.
-e	0.95	Αποδεκτός βαθμός σφάλματος για επιτάχυνση της εξέλιξης.
-s	123456789	Το όρισμα της γεννήτριας τυχαίων αριθμών.
-t	C:\GATree\Data\anneal\anneal0.arff	Η θέση του αρχείου με τα δεδομένα εκπαίδευσης.
-output	C:\GATree\out\outputfile22204.txt	Η θέση του αρχείου αποτελέσματος.
-output3	C:\GATree\out\cv24093.txt	Η θέση του αρχείου επικύρωσης με διασταύρωση.
-output4	C:\GATree\out\vcg46180.vcg	Η θέση του αρχείου οπτικής απεικόνισης.

Πίνακας 2: Οι παράμετροι σε γραμμή εντολών.

3.5 Λειτουργίες

Ας δούμε μια σύνοψη των κυριότερων λειτουργιών που επιτελεί η παλαιά εφαρμογή.

3.3.1 Εκτέλεση γενετικού αλγορίθμου

Καταρχήν εκτελείται ο γενετικός αλγόριθμος GATree. Πρόκειται για την πιο σημαντική λειτουργία της εφαρμογής. Η εκτέλεση αποτελείται από τα εξής γενικά βήματα:

- Δημιουργία αρχικού πληθυσμού με N υποψήφιες λύσεις.
- Υπολογισμός της καταλληλότητας κάθε λύσης.
- Όσο ο πληθυσμός δεν συγκλίνει σε μια λύση:

  Επανάληψη για N/2 φορές των ακόλουθων βημάτων:

  a. Επιλογή δύο λύσεων από τον πληθυσμό.
  b. Συνδυασμός των δύο λύσεων για την αναπαραγωγή δύο απογόνων.
  c. Υπολογισμός της καταλληλότητας των δύο απογόνων.
  d. Εισαγωγή των δύο απογόνων στο νέο πληθυσμό [33].

3.3.1 Επικύρωση με διασταύρωση K τμημάτων

Κατά την επικύρωση με διασταύρωση, η συλλογή δεδομένων χωρίζεται σε K τμήματα. Αρχικά επιλέγεται το 1ο τμήμα για έλεγχο, ενώ τα υπόλοιπα K-1 τμήματα διατίθενται για εκπαίδευση. Εκτελείται ο γενετικός αλγόριθμος και καταγράφεται η ακρίβεια του βέλτιστου δέντρου (Εικόνα 17).

Εικόνα 17: Πρώτο βήμα επικύρωσης με διασταύρωση 4 τμημάτων.

Έπειτα επιλέγεται το $2^{ο}$ τμήμα για έλεγχο, τα υπόλοιπα Κ-1 τμήματα για εκπαίδευση (Εικόνα 18) κοκ.

Εικόνα 18: Δεύτερο βήμα επικύρωσης με διασταύρωση 4 τμημάτων.

Τέλος, υπολογίζεται ο μέσος όρος της ακρίβειας των δένδρων που προέκυψαν από κάθε βήμα. Η τελική αυτή ακρίβεια αποτελεί μέτρο σύγκρισης με άλλους αλγορίθμους για τη συγκεκριμένη συλλογή δεδομένων [27].

Κεφάλαιο 4
Μοντελοποίηση

Στο κεφάλαιο αυτό παρουσιάζεται, βήμα προς βήμα, η μοντελοποίηση του νέου συστήματος, με σκοπό να αποτελέσει μια γερή βάση κατανόησης του έργου, ένα «λεξικό δεδομένων» [23]. Το αρχικό αυτό μοντέλο εξελίσσεται καθ' όλη τη διάρκεια ζωής του έργου ώστε να αποτελεί πάντοτε την πιο πρόσφατη αποτύπωση του πεδίου του προβλήματος. Σε πρακτικό επίπεδο, η απεικόνιση γίνεται με απλοποιημένα διαγράμματα κλάσεων [04].

4.1 Λεξικό δεδομένων

Οι βασικές οντότητες, που αποτελούν το χώρο της λύσης και περιγράφουν με αντικείμενα του πραγματικού κόσμου το πεδίο του προβλήματος, παρουσιάζονται στον Πίνακας 3.

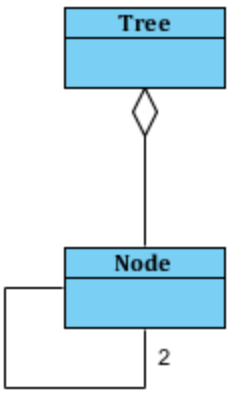
Tree	Δυαδικό δένδρο αναζήτησης.
Node	Κόμβος ενός δυαδικού δένδρου αναζήτησης.
Relation	Συλλογή δεδομένων (dataset). Προτιμήθηκε το όνομα Relation, σε σχέση με το DataSet, επειδή ο όρος αυτός χρησιμοποιείται στον

	ορισμό των αρχείων ARFF [31].
Attribute	Ιδιότητα μιας συλλογής δεδομένων.
Instance	Στιγμιότυπο μιας συλλογής δεδομένων.

Πίνακας 3: Το λεξικό με τις βασικές οντότητες.

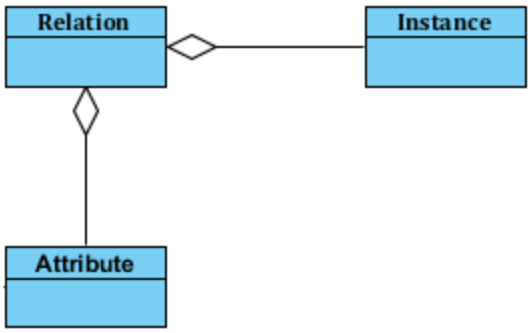
4.2 Πεδίο προβλήματος

Βασική απαίτηση είναι η αναπαράσταση του χώρου αναζήτησης με τη χρήση πραγματικών δυαδικών δένδρων [20]. Κατά συνέπεια, το πλέον βασικό δομικό στοιχείο είναι ο κόμβος (Node). Κάθε κόμβος ανήκει σε ένα δένδρο (Tree) και έχει ακριβώς δύο κόμβους – παιδιά (Εικόνα 19).



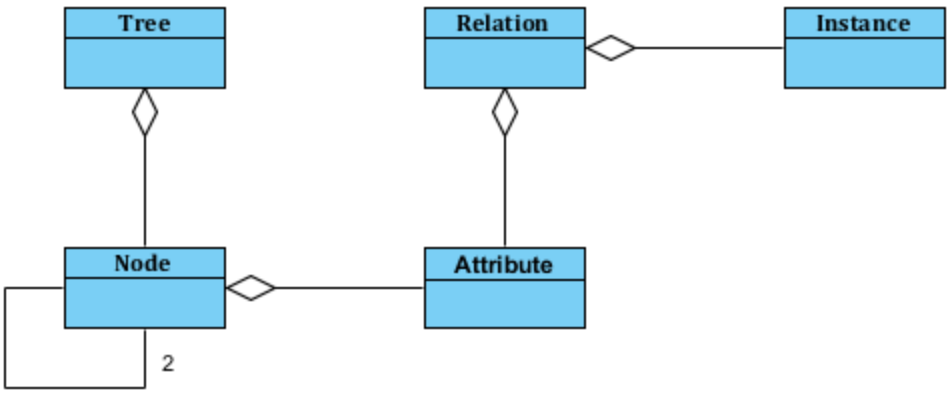
Εικόνα 19: Μοντελοποίηση του χρωμοσώματος.

Από την άλλη, μια συλλογή δεδομένων (Relation) αποτελείται από κάποιες ιδιότητες (Attribute) και φυσικά διαθέτει μια σειρά στιγμιότυπων (Instance), όπως φαίνεται στην Εικόνα 20.



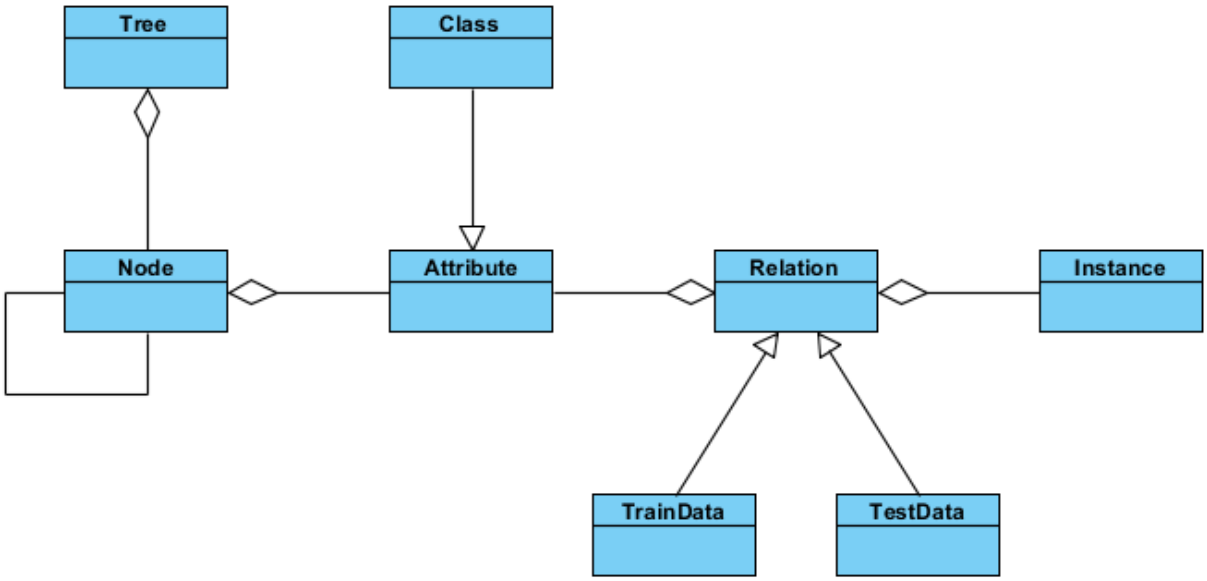
Εικόνα 20: Μοντελοποίηση της συλλογής δεδομένων (dataset).

Αυτό που συνδέει τις δύο προηγούμενες συστάδες είναι το γεγονός πως ένας κόμβος αντιστοιχίζεται με μια ιδιότητα (Εικόνα 21).



Εικόνα 21: Βασικό μοντέλο του πεδίου προβλήματος.

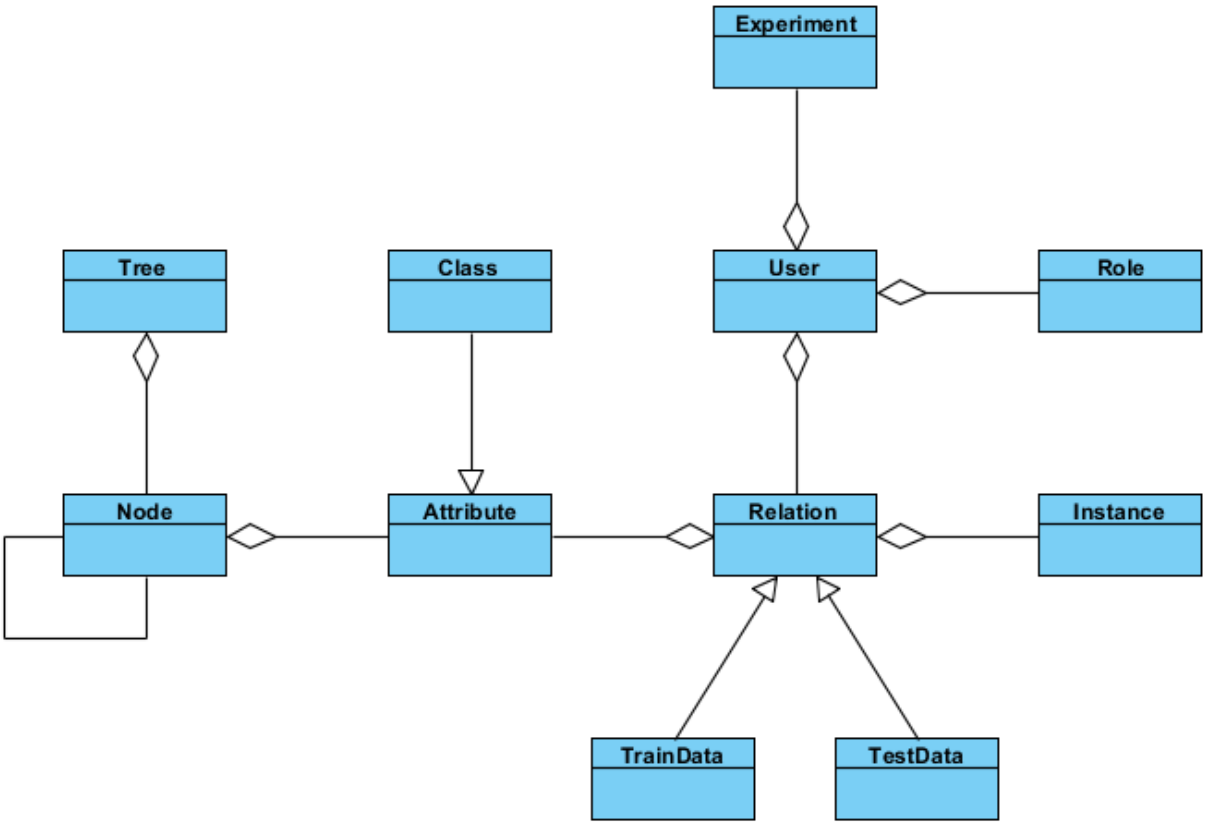
Στο σημείο αυτό συνεχίζεται η εκλέπτυνση, το αποτέλεσμα της οποίας παρουσιάζεται στην Εικόνα 22. Η τιμή μιας συγκεκριμένης ιδιότητας είναι αυτή που καθορίζει την κατάταξη του στιγμιότυπου. Πρόκειται για την ιδιότητα κατάταξης – κλάση (Class). Τέλος, μια συλλογή μπορεί να παρέχει δεδομένα προς εκπαίδευση (TrainData) ή προς αξιολόγηση (TestData).



Εικόνα 22: Μοντέλο του πεδίου προβλήματος για την υποδομή του γενετικού αλγορίθμου.

Για το διαδικτυακό περιβάλλον, οι βασικές λειτουργίες κινούνται γύρω από το πείραμα βελτιστοποίησης (Experiment) και τη συλλογή δεδομένων. Επιπλέον ορίζεται μια απλή υποδομή πιστοποίησης, η οποία ακολουθεί την προσέγγιση RBAC (Role-Based Access Control[5]), και αποτελείται από το χρήστη (User) και τους ρόλους που διαθέτει αυτός στο σύστημα (Role), όπως φαίνεται στην Εικόνα 23.

[5] http://en.wikipedia.org/wiki/Role-based_access_control



Εικόνα 23: Μοντέλο του πεδίου προβλήματος για το διαδικτυακό περιβάλλον.

Κεφάλαιο 5
Τεχνική αρχιτεκτονική

Στο κεφάλαιο αυτό παρουσιάζεται η αρχιτεκτονική του νέου συστήματος, γίνεται ο χωρισμός σε υποσυστήματα, και λαμβάνονται οι απαραίτητες τεχνολογικές αποφάσεις.

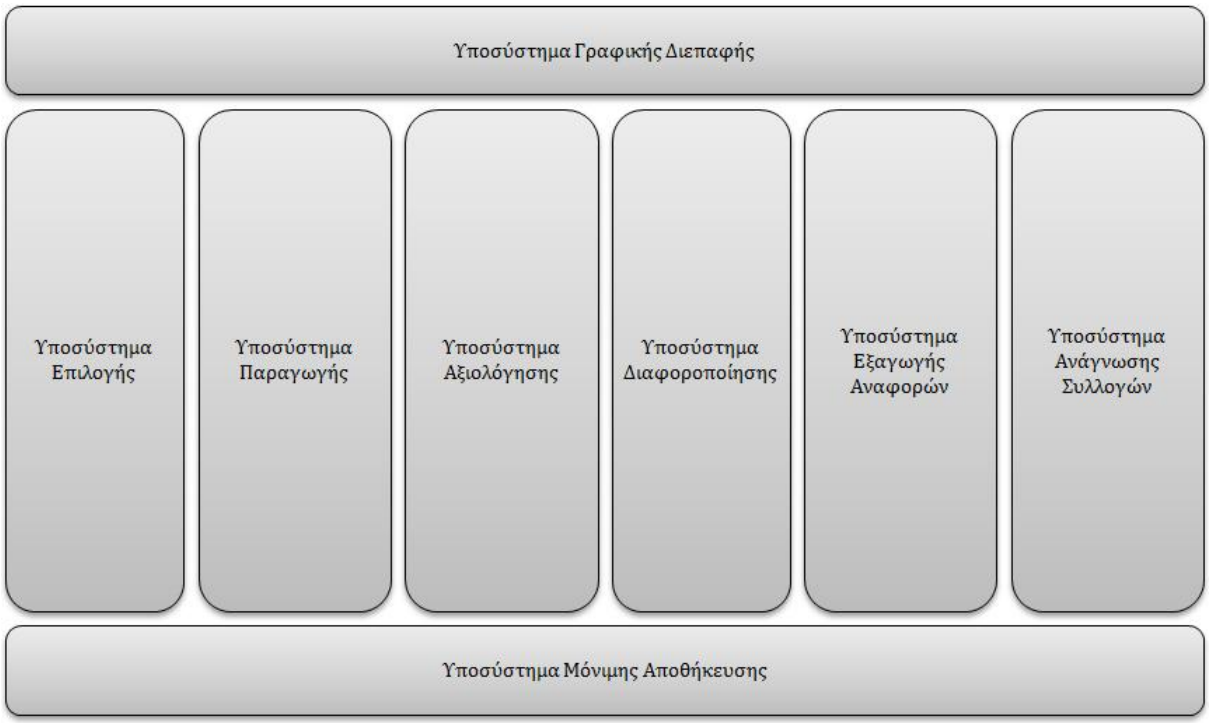
5.1 Υποσυστήματα

Ως βασικό συστατικό της αρχιτεκτονικής έχει επιλεγεί το υποσύστημα (λειτουργική ενότητα ή module). Πρόκειται για ένα αυτοτελές κομμάτι λογισμικού το οποίο επικοινωνεί με τα υπόλοιπα κομμάτια μέσω σαφώς ορισμένων διεπαφών, κρύβοντας τις λεπτομέρειες στο εσωτερικό του. Επιδίωξη είναι το υποσύστημα να εστιάζει σε ακριβώς μία βασική λειτουργία της εφαρμογής. Ενδεικτικά αναφέρονται τα χαρακτηριστικά ενός υποσυστήματος - η μοναδικότητα, η έκδοση, η διεπαφή επικοινωνίας, η ξεκάθαρα δηλωμένη τυχόν εξάρτησή του από άλλα υποσυστήματα και ο κύκλος ζωής του [21].

Καταρχήν, ο στοχευμένος περιορισμός της ευθύνης κάθε υποσυστήματος οδηγεί τελικά σε μια καθαρή σχεδίαση. Έπειτα, διευκολύνεται σε διαχειριστικό επίπεδο η επέκταση της εφαρμογής. Για παράδειγμα, ένας ερευνητής με εξαιρετικές ιδέες, θα μπορούσε να τις εφαρμόσει στο υποσύστημα ενδιαφέροντός του, αδιαφορώντας για τη δομή και την τυχόν πολυπλοκότητα των

υπολοίπων υποσυστημάτων. Επίσης, η χρήση διακριτών υποσυστημάτων προετοιμάζει την ομαλή ένταξη της εφαρμογής σε έναν οργανωμένο κύκλο ζωής λογισμικού. Αυτό κάνει δυνατό το άνοιγμα της εφαρμογής ως ΕΛ/ΛΑΚ (ελεύθερο λογισμικό / λογισμικό ανοιχτού κώδικα) είτε ως εμπορικό προϊόν. Τέλος, η χρήση διακριτών υποσυστημάτων διευκολύνει μελλοντικές προσαρμογές για την αξιοποίηση υποδομών πλέγματος, κατανεμημένων περιβαλλόντων ή άλλων τεχνικών παραλληλισμού.

Οι λειτουργικές ενότητες που έχουν οριστεί απεικονίζονται στην Εικόνα 24.



Εικόνα 24: Αρχιτεκτονική λειτουργικών ενοτήτων.

5.1.1 Υποσύστημα Επιλογής

Επιλέγει μέρος του πληθυσμού προς διαγραφή με βάση αντικειμενικά κριτήρια (π.χ. χαμηλό αποτέλεσμα) ή προς διαφοροποίηση με βάση την εκάστοτε πολιτική (π.χ. τεχνική ρουλέτας [32]).

5.1.2 Υποσύστημα Παραγωγής

Παράγει τον αρχικό πληθυσμό προς εξέλιξη. Επιπλέον, αναπληρώνει το μέρος του πληθυσμού που απορρίπτεται σε κάθε γενιά.

5.1.3 Υποσύστημα Αξιολόγησης

Αξιολογεί τα δένδρα ενός πληθυσμού με βάση την αντικειμενική συνάρτηση. Επιπλέον, λαμβάνει υπόψη του συγκεκριμένους τύπους κόμβων (π.χ. numeric), και εφαρμόζει την κατάλληλη πολιτική για τις τιμές που λείπουν (missing attributes).

5.1.4 Υποσύστημα Διαφοροποίησης

Διαφοροποιεί ένα δένδρο από τα υπόλοιπα του πληθυσμού, γεγονός καθοριστικό για την εξέλιξη. Περιλαμβάνει τόσο ενέργειες διασταύρωσης όσο και μετάλλαξης.

5.1.5 Υποσύστημα Γραφικής Διεπαφής

Περιλαμβάνει τη γραφική διεπαφή προς το χρήστη, καθώς και τις απαραίτητες οριζόντιες λειτουργίες (ασφάλεια, καταγραφή, πιστοποίηση).

5.1.6 Υποσύστημα Εξαγωγής Αναφορών

Μορφοποιεί το αποτέλεσμα της εξέλιξης στους απαιτούμενους τύπους αρχείων αλλά και με οπτικά εύληπτο τρόπο.

5.1.7 Υποσύστημα Ανάγνωσης Συλλογών

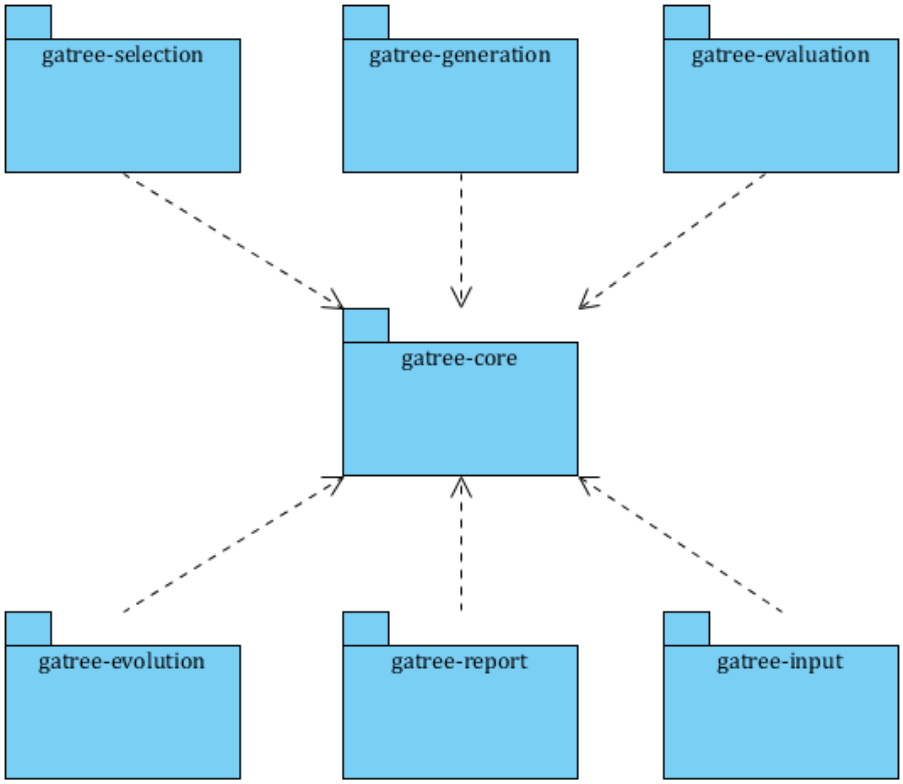
Ανοίγει αρχεία συλλογών, τα ελέγχει για την ορθότητά τους και διαβάζει τα δεδομένα αποτυπώνοντάς τα στο αντικειμενοστραφές μοντέλο.

5.1.8 Υποσύστημα Μόνιμης Αποθήκευσης

Παρέχει λειτουργίες μόνιμης αποθήκευσης περιεχομένου (persistence). Οι λειτουργίες αυτές δεν είναι απαραίτητες για την εκτέλεση του αλγορίθμου. Σκοπός τους είναι να συμβάλλουν στη μελλοντική ανάπτυξη υπηρεσιών προστιθέμενης αξίας γύρω από το GATree.

5.1.9 Διάγραμμα πακέτων

Στην Εικόνα 25 παρατίθεται το διάγραμμα των βασικών πακέτων [04]. Ως μοναδικό προσδιοριστικό της εφαρμογής έχει οριστεί η λέξη gatree, η οποία αναφέρεται στο **G**enetic **A**lgorithm **Tree**.

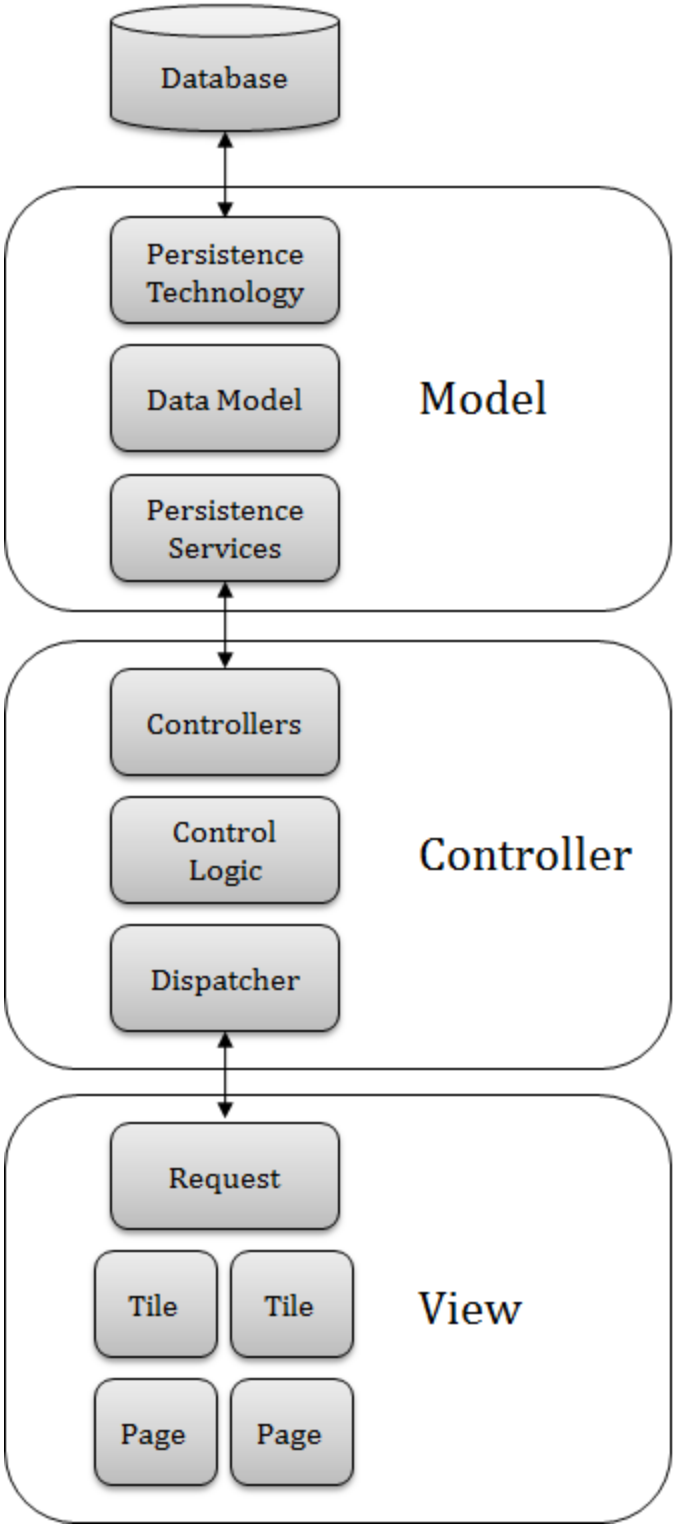


Εικόνα 25: Διάγραμμα βασικών πακέτων.

Παρατηρούμε πως τα βασικά υποσυστήματα είναι ανεξάρτητα μεταξύ τους. Καθένα μπορεί να επιτελέσει τη δουλειά του έχοντας μόνο μια εξάρτηση – το επαναχρησιμοποιήσιμο υποσύστημα στο οποίο ορίζονται το μοντέλο δεδομένων και οι παράμετροι της εφαρμογής. Αυτό βοηθάει την ανάπτυξη κατανεμημένων τοπολογιών. Για παράδειγμα ένας υπολογιστικός κόμβος μπορεί να εκτελεί αξιολογήσεις τρέχοντας μόνο το υποσύστημα αξιολόγησης, ένας άλλος να εξελίσσει τον πληθυσμό που του δίνεται κοκ.

5.2 Επίπεδα

Η τριών επιπέδων αρχιτεκτονική βασίζεται στη σχεδιαστική προσέγγιση Model – View – Controller. Τα επίπεδα είναι σαφώς ορισμένα, έχουν χαλαρή σύνδεση μεταξύ τους και επικοινωνούν μέσω καθορισμένων διεπαφών (Εικόνα 26).



Εικόνα 26: Γενική αρχιτεκτονική τριών επιπέδων.

Τα δεδομένα αποθηκεύονται σε μια βάση δεδομένων, η οποία είναι σχεσιακή. Το σύστημα επικοινωνεί με τη βάση μέσω μιας τεχνολογίας μόνιμης αποθήκευσης και ανάκτησης δεδομένων. Στη συνέχεια ορίζεται το μοντέλο των δεδομένων, που απεικονίζει τα αντικείμενα του πραγματικού κόσμου που αφορούν το σύστημα. Τέλος, το πρώτο επίπεδο παρέχει κάποιες

υπηρεσίες στο επόμενο επίπεδο. Οι υπηρεσίες αυτές περιλαμβάνουν ενδεικτικά την ανάκτηση, αναζήτηση και αποθήκευση δεδομένων, και ολοκληρώνουν την αφαίρεση των λειτουργιών του **Model**.

Οι υπηρεσίες του πρώτου επιπέδου χρησιμοποιούνται από κάποιους ελεγκτές, δουλειά των οποίων είναι να ελέγχουν τα δεδομένα του χρήστη και να πυροδοτούν τη ροή των ενεργειών μέσα στο σύστημα. Οι κινήσεις των ελεγκτών βασίζονται στην επιχειρησιακή λογική που διέπει το σύστημα. Τέλος, ορίζεται ένας αποστολέας ο οποίος διατηρεί τη ροή των οθονών που εμφανίζονται στον τελικό χρήστη. Ο αποστολέας αποτελεί το συνδετικό κρίκο του **Controller** με το επόμενο επίπεδο.

Η πληροφορία που καταγράφει την αλληλεπίδραση και τις ενέργειες του χρήστη αποτυπώνεται σε ένα αντικείμενο με το γενικό όνομα Request. Από εκεί και πέρα, διάφορα επαναχρησιμοποιήσιμα κομμάτια συνθέτουν τις οθόνες της εφαρμογής. Με τον τρόπο αυτό ολοκληρώνεται το επίπεδο **View**.

5.3 Τεχνολογίες

Χρησιμοποιείται ελεύθερο λογισμικό / λογισμικό ανοιχτού κώδικα[6], το οποίο έχει μηδενικό κόστος πρόσκτησης και επ' αόριστον χρήσης.

5.3.1 Java

Ως βασική τεχνολογία ανάπτυξης έχει επιλεγεί η Java Enterprise Edition, η οποία έχει να επιδείξει μια μακρά πορεία αξιοσημείωτης ωρίμανσης σε εταιρικές εφαρμογές ευρείας κλίμακας. Επιπλέον, έχει χρησιμοποιηθεί επαγγελματικά από το γράφοντα με επιτυχία σε μεγάλα έργα. Ενδεικτικά αναφέρονται τα:

- Forms Filler & Designer suite για τον Εκτελεστικό Οργανισμό Εκπαίδευσης, Οπτικοακουστικών Θεμάτων και Πολιτισμού (EACEA[7]).

- Symmetry για τη Γενική Διεύθυνση Εκπαίδευσης και Πολιτισμού (DG EAC[8]).

[6] http://www.ellak.gr/
[7] http://eacea.ec.europa.eu/

- Electronic Proposal Submission Service για την Υπηρεσία Πληροφόρησης για την Κοινοτική Έρευνα και Ανάπτυξη (CORDIS[9]).

- RCD Systems Portfolio για το Γραφείο Εναρμόνισης στο πλαίσιο της Εσωτερικής Αγοράς (OHIM[10]).

- Εθνικός Συλλογικός Κατάλογος Επιστημονικών Περιοδικών – Νέο Σύστημα Διαδανεισμού Εθνικού Δικτύου Επιστημονικών & Τεχνολογικών Βιβλιοθηκών για το Εθνικό Κέντρο Τεκμηρίωσης (ΕΚΤ[11]).

5.3.2 Spring

Το Spring είναι μια συλλογή καλογραμμένων διεπαφών Java που απλοποιεί την ανάπτυξη εταιρικών εφαρμογών. Με τις καλές πρακτικές που ορίζει, συμβάλλει στην απλοποίηση του κώδικα, την ευκολότερη επαναχρησιμοποίηση και συντήρησή του, τον αποτελεσματικότερο έλεγχο, τη μείωση του χρόνου ανάπτυξης με ταυτόχρονη αύξηση της παραγωγικότητας. Η δυναμική του με τα χρόνια έγινε τόσο σημαντική, που οι ιδέες του επηρεάζουν την ίδια την εξέλιξη της Java Enterprise Edition.

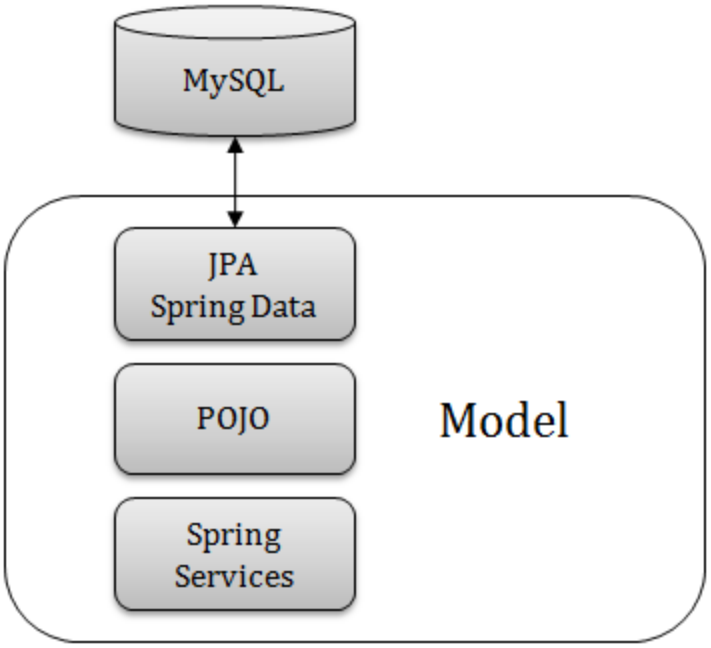
Ο πυρήνας του Spring στηρίζεται στις αρχές των Dependency Injection (DI) και Aspect Oriented Programming (AOP) [29], πάνω στις οποίες χτίζει ένα πλήθος εργαλείων για την κάλυψη οποιασδήποτε ανάγκης σε μια εταιρική εφαρμογή.

5.3.3 Maven

Το Maven είναι εργαλείο διαχείρισης έργων λογισμικού, φτιαγμένο για μηχανικούς λογισμικού. Εγγυάται διαφανή διαχείριση του κύκλου ζωής ανεξάρτητα από το εργαλείο ανάπτυξης [12]. Επιπλέον εισάγει την έννοια του αποθετηρίου παραγομένων, το οποίο εντάσσεται εύκολα σε έναν οργανωμένο κύκλο ζωής λογισμικού. Περισσότερες πληροφορίες παρουσιάζονται στο Παράρτημα Α .

[8] http://ec.europa.eu/dgs/education_culture/
[9] http://cordis.europa.eu/
[10] http://oami.europa.eu/
[11] http://www.ekt.gr/

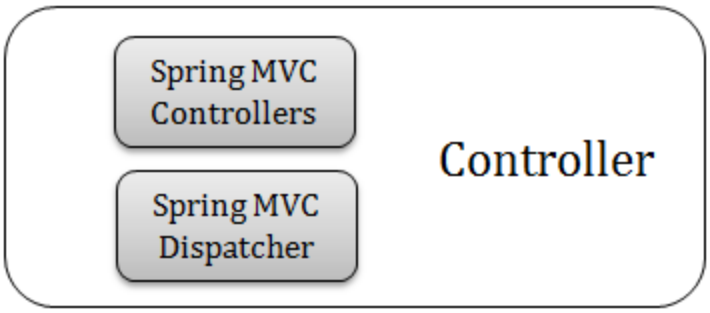
Στο σημείο αυτό θα εκλεπτύνουμε την αρχιτεκτονική της προηγούμενης ενότητας, με βάση τις τεχνολογίες που έχουν επιλεγεί.



Εικόνα 27: Η τεχνολογική εκλέπτυνση του επιπέδου Model.

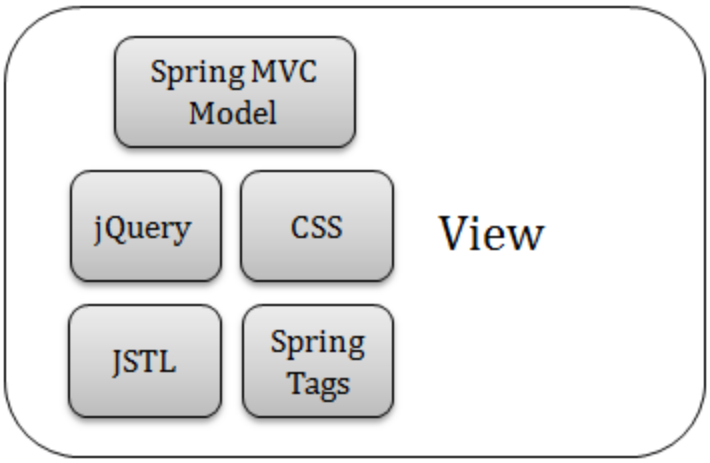
Καταρχήν, η βάση δεδομένων είναι η MySQL (Εικόνα 27). Η εφαρμογή επικοινωνεί μαζί της μέσω του προτύπου JPA (Java Persistence API), το οποίο στη συγκεκριμένη περίπτωση υλοποιείται από το Hibernate. Επιπρόσθετα, επιστρατεύονται οι ευκολίες που παρέχει το Spring Data. Το μοντέλο δεδομένων αναπαρίσταται με απλά αντικείμενα POJO (Plain Old Java Objects). Τέλος, το επίπεδο Model παρέχει υπηρεσίες στο επόμενο επίπεδο, ο κύκλος ζωής των οποίων διαχειρίζεται από τον container του Spring.

Το JPA [10] είναι το αποτέλεσμα μακροχρόνιας συνεργασίας φορέων της βιομηχανίας και κοινοτήτων ελεύθερου λογισμικού. Επιτρέπει τη σχεδίαση και υλοποίηση του συστήματος αποκλειστικά στα πλαίσια του πεδίου προβλήματος, αφαιρώντας την όποια πολυπλοκότητα προκύπτει από την εξάρτηση από τη βάση δεδομένων. Το Spring Data προσθέτει ένα επιπλέον επίπεδο αφαίρεσης πάνω από το JPA. Παρέχει έτοιμη λειτουργικότητα για κοινές ανάγκες μόνιμης αποθήκευσης και προάγει την καθαρή σχεδίαση.



Εικόνα 28: Η τεχνολογική εκλέπτυνση του επιπέδου Controller.

Οι Controllers του Spring MVC [29] χρησιμοποιούν τις παραπάνω υπηρεσίες και καθορίζουν τις περαιτέρω ενέργειες μέσα στο σύστημα (Εικόνα 28). Στο τέλος του επιπέδου Controller βρίσκεται ο Dispatcher του Spring MVC, ο οποίος υλοποιεί το σχεδιαστικό πρότυπο Front Controller, και διαχειρίζεται τις οθόνες που βλέπει ο χρήστης.



Εικόνα 29: Η τεχνολογική εκλέπτυνση του επιπέδου View.

Η πληροφορία που αποτυπώνει τις ενέργειες και τα δεδομένα του χρήστη κατά την αλληλεπίδρασή του με την εφαρμογή, μοντελοποιούνται εύκολα στο Spring MVC (Εικόνα 29). Οι οθόνες είναι JSP (JavaServer Pages), συμπληρωμένες με τις διακοσμητικές ικανότητες της CSS και το αποτελεσματικότατο jQuery [30]. Επίσης, επιστρατεύεται επικουρικά το JSTL (JavaServer Pages Standard Tag Library) και μερικά Spring tags.

Τέλος, οι οριζόντιες λειτουργίες της πιστοποίησης και της εξουσιοδότησης των χρηστών επιτελούνται με τη βοήθεια του Spring Security [17].

Ο τρόπος με τον οποίο φτιάχτηκε, λοιπόν, το διαδικτυακό περιβάλλον ακολουθεί την προσέγγιση Model-View-Controller. Τα τρία επίπεδα παρέχουν υπηρεσίες το ένα στο άλλο, υπηρεσίες οι οποίες διαχειρίζονται από τον Spring container. Αυτό σημαίνει ότι το Spring αρχικοποιεί, διασυνδέει και εξασφαλίζει την διαθεσιμότητα των υπηρεσιών κατά το χρόνο εκτέλεσης.

Στο επίπεδο Model ορίζουμε ακριβώς μία υπηρεσία αποθήκευσης για κάθε οντότητα του πεδίου προβλήματος. Η κάθε υπηρεσία παρέχει βασικές λειτουργίες μόνιμης αποθήκευσης, όπως ανάκτηση, αναζήτηση, εισαγωγή, διαγραφή κλπ.

Στο επίπεδο Controller ορίζουμε ακριβώς έναν Spring controller για κάθε κύρια οντότητα της διαδικτυακής εφαρμογής (πείραμα βελτιστοποίησης και συλλογή δεδομένων). Ο κάθε controller διαχειρίζεται τη ροή των οθονών που αφορούν την οντότητα αυτή και καλεί την αντίστοιχη υπηρεσία αποθήκευσης από το επίπεδο Model.

Στο επίπεδο View ορίζονται οι οθόνες του χρήστη. Οι σελίδες JSP είναι ομαδοποιημένες σε φακέλους, έναν για κάθε κύρια οντότητα της διαδικτυακής εφαρμογής. Έτσι, για την οντότητα Πείραμα δημιουργήθηκε ο φάκελος experiment, ενώ για τη συλλογή δεδομένων ο φάκελος dataset. Επιπλέον, το όνομα του αρχείου κάθε σελίδας απεικονίζει την ενέργεια που επιτελεί ο χρήστης στην εκάστοτε οντότητα. Για παράδειγμα το αρχείο της σελίδας, μέσω της οποίας ο χρήστης μπορεί να δει τις λεπτομέρειες ενός πειράματος, ονομάζεται απλά view.jsp.

Τέλος, ακολουθείται το σχήμα διευθυνσιοδότησης REST (REpresentational State Transfer[12]), το οποίο υποστηρίζεται άμεσα από το Spring MVC. Ως βάση της διεύθυνσης ορίζεται το λεκτικό gatree. Έπειτα ακολουθεί το όνομα της βασικής οντότητας (experiment ή dataset) και τέλος η ενέργεια επί της βασικής οντότητας (create, view, list κλπ). Για παράδειγμα, ο χρήστης μπορεί να δει μια λίστα με τα πειράματα που εκτέλεσε στη διεύθυνση gatree/experiment/list.

[12] http://en.wikipedia.org/wiki/Representational_state_transfer

Κεφάλαιο 6
Βελτιώσεις σε σχέση με την παλαιά εφαρμογή

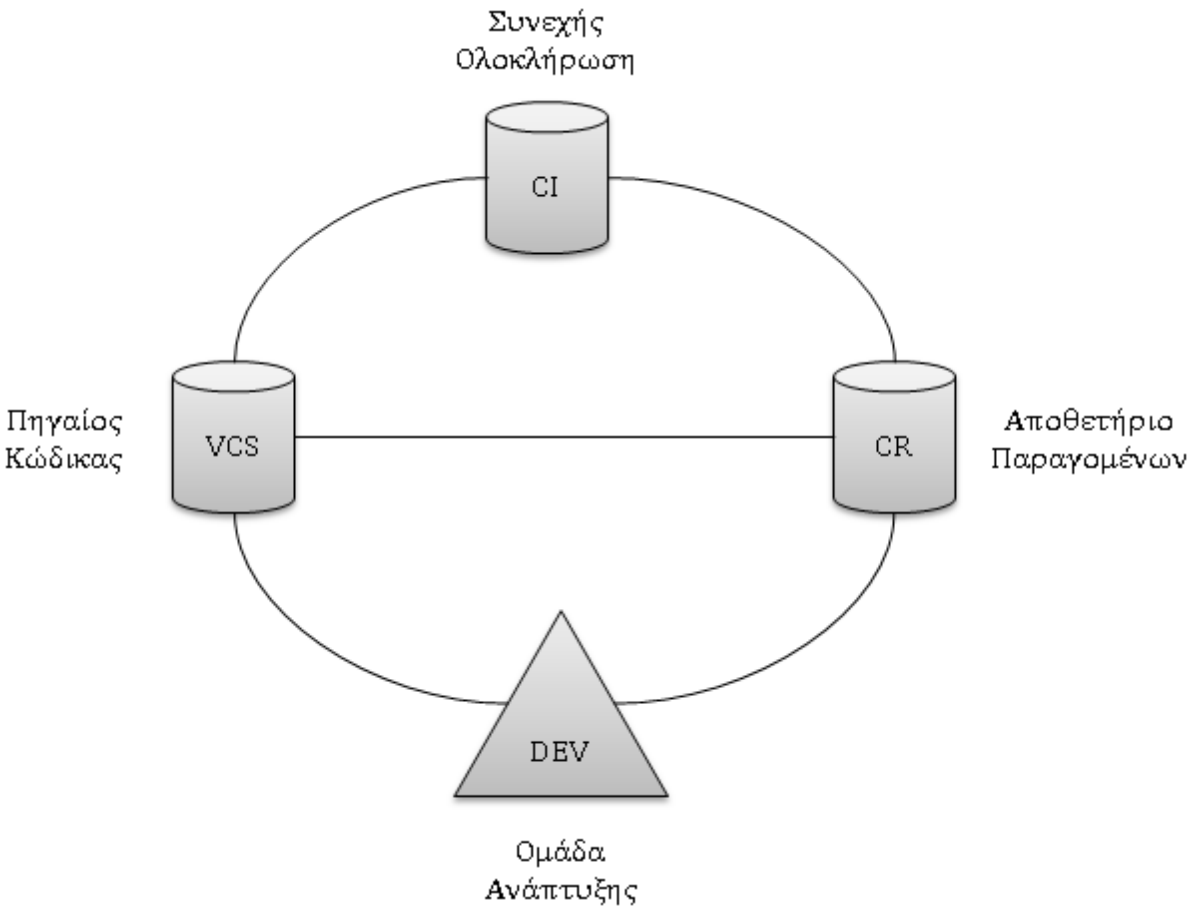
Στο κεφάλαιο αυτό γίνεται σύγκριση του νέου συστήματος με το παλαιό, παρουσιάζοντας τα πλεονεκτήματα και τις βελτιώσεις που έγιναν σε σχέση με το παλιό σύστημα.

6.1 Πρόσβαση μέσω διαδικτύου

Η μεταγραφή στη διαδικτυακή μορφή έχει προφανή οφέλη για την περαιτέρω επέκταση χρήσης της εφαρμογής. Θέτει τις βάσεις για την ανάπτυξη υπηρεσιών και την παραγωγή υπεραξίας. Μια πρώτη ενδεικτική λειτουργία είναι η δυνατότητα εγγραφής και πιστοποίησης των χρηστών. Βέβαια, η εμπορική αξιοποίηση απαιτεί την οργάνωση ενός κύκλου ζωής για το λογισμικό, όπως παρουσιάζεται στην επόμενη ενότητα.

6.2 Κύκλος ζωής

Έχει οριστεί ένας οργανωμένος κύκλος ζωής για την εφαρμογή, η οποία πλέον μπορεί να δοθεί στην κοινότητα ως λογισμικό ανοιχτού κώδικα ή να ενταχθεί εύκολα στις παραγωγικές δομές μιας εταιρίας. Η οργάνωση, που παρουσιάζεται στην Εικόνα 30, οφείλεται κατά ένα σημαντικό μέρος στη χρήση του Maven, το οποίο απλοποιεί την έκδοση μιας εφαρμογής (snapshots, releases) και συνεργάζεται άψογα με πολλά διαφορετικά συστήματα. Τα τελευταία χρόνια μάλιστα, είναι ιδιαίτερα αγαπητό στις ευέλικτες επαναληπτικές μεθοδολογίες [07], όπως η SCRUM [22].



Εικόνα 30: Διάταξη εργαλείων για τη διαχείριση του κύκλου ζωής του λογισμικού [07].

Η ομάδα ανάπτυξης οργανώνει τον πηγαίο κώδικα με τη βοήθεια ενός κατάλληλου συστήματος διαχείρισης εκδόσεων (VCS – Version Control System), όπως τα ευρέως διαδεδομένα Subversion, Git, Mercurial, Veracity [26]. Επιπλέον, χρησιμοποιεί το αποθετήριο (CR – Component Repository) για τη διαχείριση των παραγόμενων αποτελεσμάτων και τις αμέτρητες ευκολίες που παρέχει κατά την ανάπτυξη.

Βασικό μέλημα του συστήματος συνεχούς ολοκλήρωσης (CI – Continuous Integration) είναι να σιγουρευτεί πως το έργο χτίζεται σωστά. Αφού τραβήξει τον πηγαίο κώδικα, επιβεβαιώνει τις

εξαρτήσεις του προς τα υπόλοιπα υποσυστήματα, χτίζει το έργο και στέλνει e-mail (SMTP – Simple Mail Transfer Protocol) σε περίπτωση αποτυχίας. Προαιρετικά, εκτελεί στατική ανάλυση και βγάζει αναφορές για την υγεία του κώδικα [12]. Η δουλειά του ποιοτικού ελέγχου είναι δυνατόν να δοθεί και σε ένα εξειδικευμένο στον τομέα αυτό σύστημα, όπως το Sonar, που συμπληρώνει με ιδανικό τρόπο τη συνεχή ολοκλήρωση. Ο διαχειριστής του έργου μπορεί να συμβουλευτεί τις αναλυτικότατες αναφορές ποιότητας του έργου, ώστε να σιγουρευτεί ότι βρίσκονται εντός των στόχων που έχουν τεθεί στο συμβόλαιο.

Το αποθετήριο παραγόμενων επιτρέπει τη συνεργασία μεταξύ των μελών μιας ομάδας, αλλά και των διαφορετικών ομάδων ενός έργου. Για παράδειγμα, η ομάδα ανάπτυξης στην Αθήνα χρησιμοποιεί εύκολα το API (Application Programming Interface) που ανέπτυξε μια άλλη ομάδα στη Μαδρίτη. Ακόμα, η ομάδα ελέγχου (Testing Team ή Quality Assurance) μπορεί να πάρει από εδώ την εκάστοτε σταθερή έκδοση του λογισμικού για ενδελεχή έλεγχο. Με τον τρόπο αυτό εξασφαλίζεται πως ακριβώς η ίδια έκδοση του λογισμικού ελέγχεται και παραδίδεται αργότερα στον πελάτη.

6.3 Αρχικοποίηση πληθυσμού

Κάθε δένδρο του αρχικού πληθυσμού είναι μοναδικό. Αυτό γίνεται με τη χρήση ενός πίνακα κατακερματισμού (hash table). Κάθε φορά που δημιουργείται ένα νέο δένδρο, γίνεται έλεγχος εάν υπάρχει ήδη στον πίνακα ως κλειδί. Εάν όχι, τότε το δένδρο γίνεται μέρος του πληθυσμού και προστίθεται στον πίνακα. Στην αντίθετη περίπτωση, το δένδρο απορρίπτεται και στη θέση του δημιουργείται ένα καινούργιο [12].

Επιπλέον, εξασφαλίζεται η διαφορετικότητα των φύλλων σε κάθε δένδρο του αρχικού πληθυσμού. Όπως έχουμε πει, ο πληθυσμός αρχικά αποτελείται από ελάχιστα δυαδικά δένδρα. Κάθε φορά που παράγεται ο δεύτερος κόμβος, γίνεται έλεγχος εάν έχει την ίδια ιδιότητα και τιμή με τον άλλο. Εάν όχι, τότε ο κόμβος γίνεται το δεύτερο φύλλο του δένδρου. Στην αντίθετη περίπτωση, κόμβος απορρίπτεται και παράγεται, με τυχαίο τρόπο, ένας καινούργιος.

6.4 Μορφές μετάλλαξης

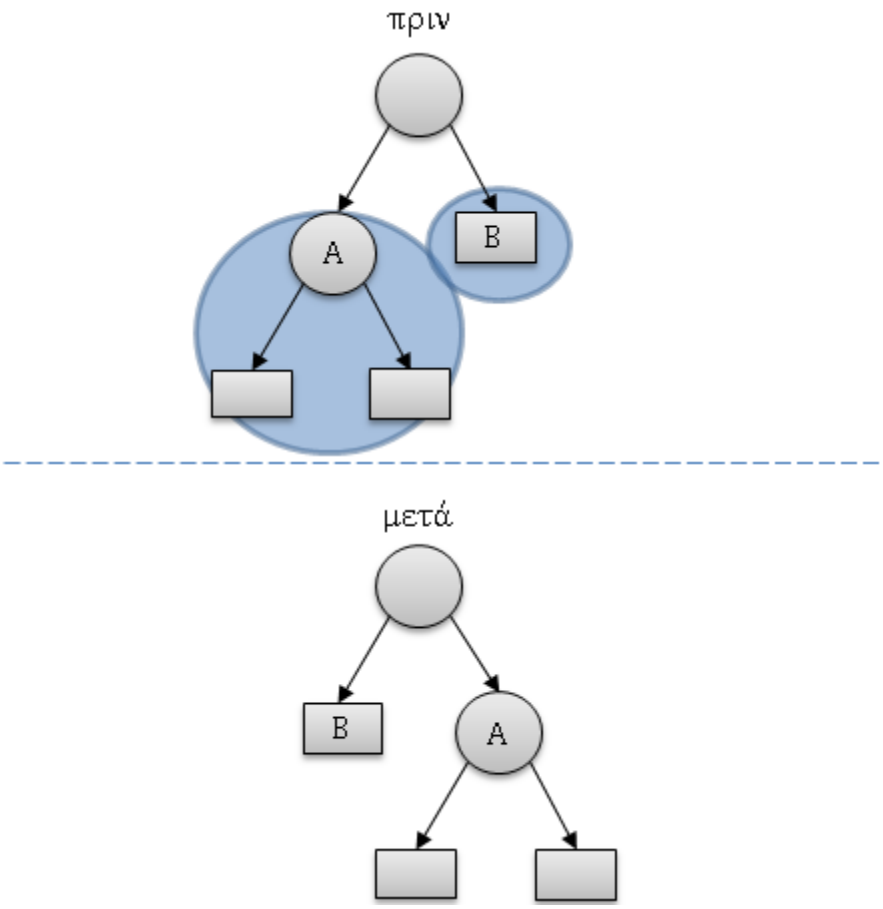
Αρχικά, η πιθανότητα μετάλλαξης αναφέρεται σε κάθε κόμβο. Με άλλα λόγια, κάθε κόμβος οποιουδήποτε δένδρου μπορεί να μεταλλαχθεί με την ίδια πιθανότητα. Αυτό που έχει προστεθεί

είναι πως μπορεί να δοθεί ποσοστό προτίμησης στα φύλλα σε σχέση με τους εσωτερικούς κόμβους και αντίστροφα.

Έπειτα, η πιθανότητα μετάλλαξης μπορεί να αναφέρεται πλέον και σε ένα ολόκληρο δένδρο. Εάν ένα δένδρο οφείλει να μεταλλαχθεί, τότε επιλέγεται ένας κόμβος του δένδρου προς μετάλλαξη. Καταρχήν, παράγεται τυχαία ένας θετικός ακέραιος αριθμός μέχρι το μέγεθος του δένδρου. Ύστερα, διασχίζεται το δένδρο με έναν τυχαίο τρόπο (προδιατεταγμένα, μεταδιατεταγμένα, ενδοδιατεταγμένα ή κατά επίπεδα) μέχρι τον κόμβο που προσδιορίζει ο αριθμός. Ο κόμβος αυτός μεταλλάσσεται. Κι εδώ μπορεί να δοθεί προτίμηση στα φύλλα σε σχέση με τους εσωτερικούς κόμβους και αντίστροφα. Επιπλέον, μπορεί να προκαθοριστεί ο τρόπος διάσχισης του δένδρου, ώστε να αποφεύγεται ο υπολογισμός ενός επιπλέον τυχαίου αριθμού.

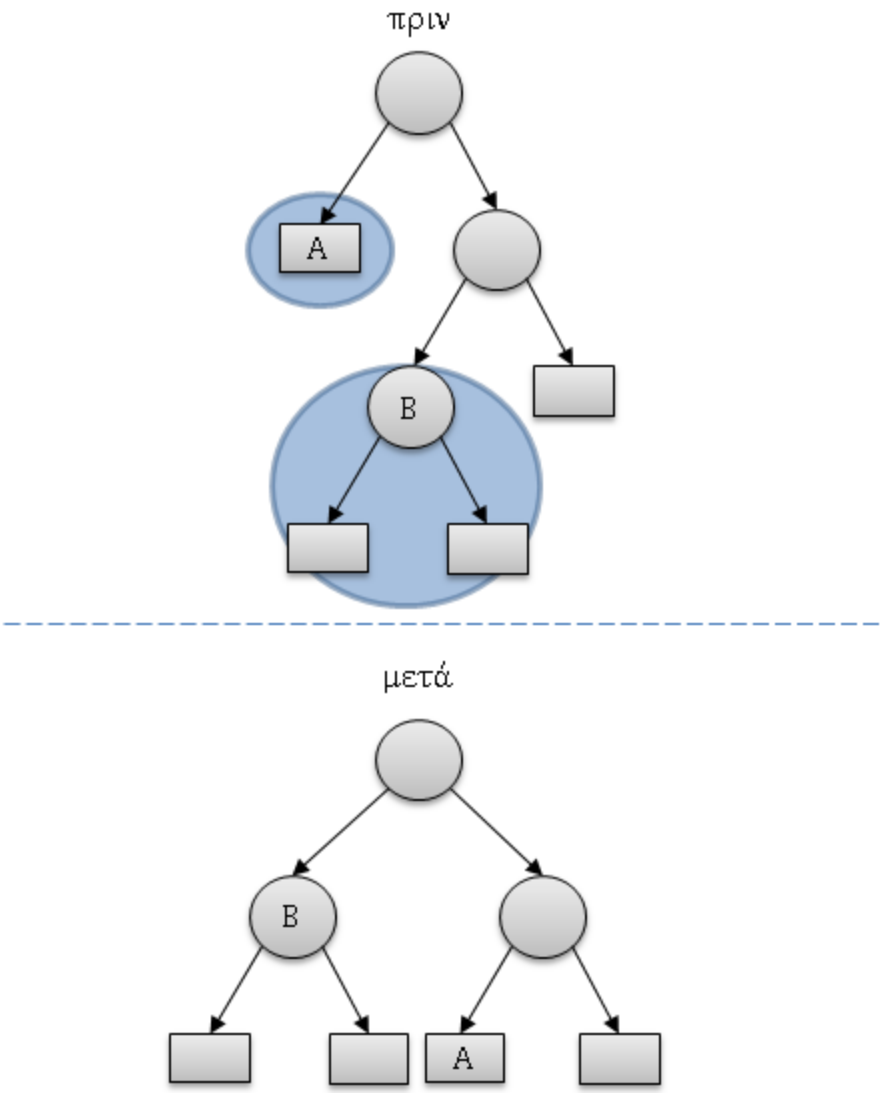
Στη συνέχεια γίνεται διαθέσιμη μια σειρά από νέες μορφές μετάλλαξης [12].

- Η αλλαγή θέσεων μεταξύ των υποδένδρων ενός εσωτερικού κόμβου. Το αριστερό υποδένδρο περνάει δεξιά και το δεξί έρχεται στη θέση του αριστερού (Εικόνα 31).



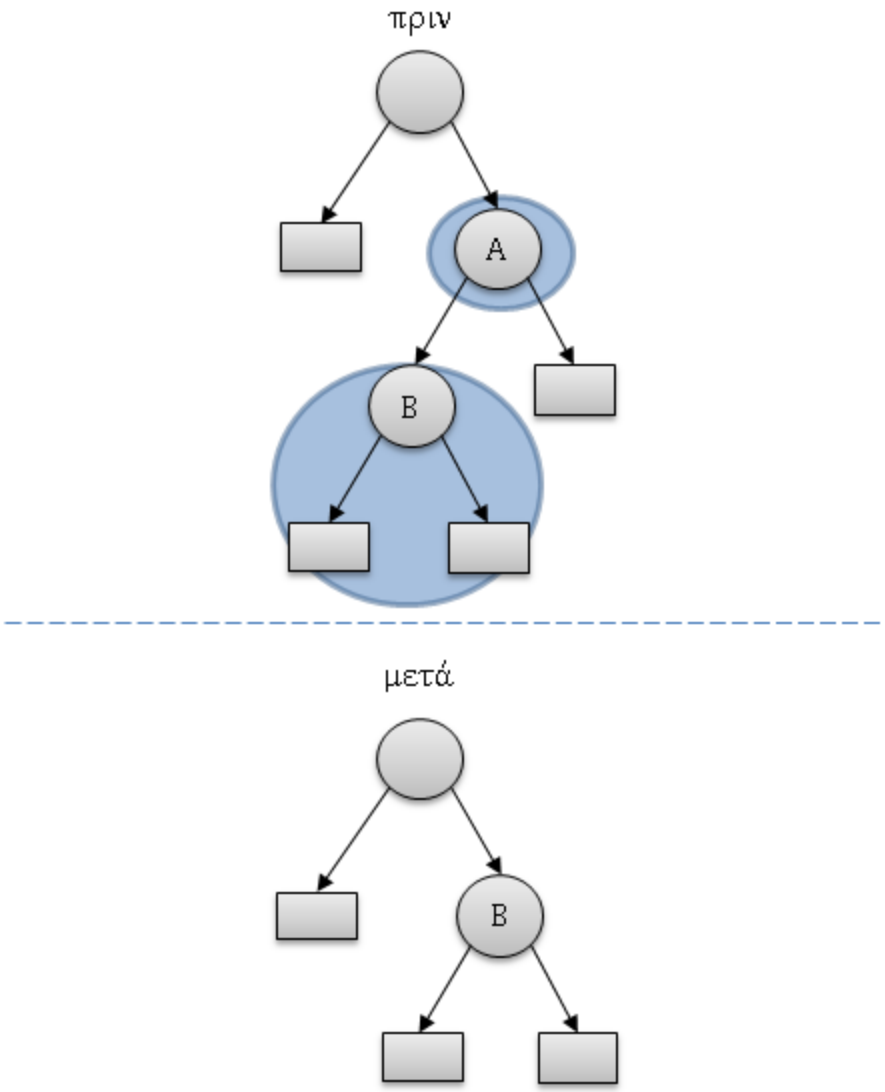
Εικόνα 31: Μετάλλαξη με ανταλλαγή των υποδένδρων ενός εσωτερικού κόμβου.

- Η ανταλλαγή δυο τυχαίων μη επικαλυπτόμενων υποδένδρων του ίδιου δένδρου (Εικόνα 32).



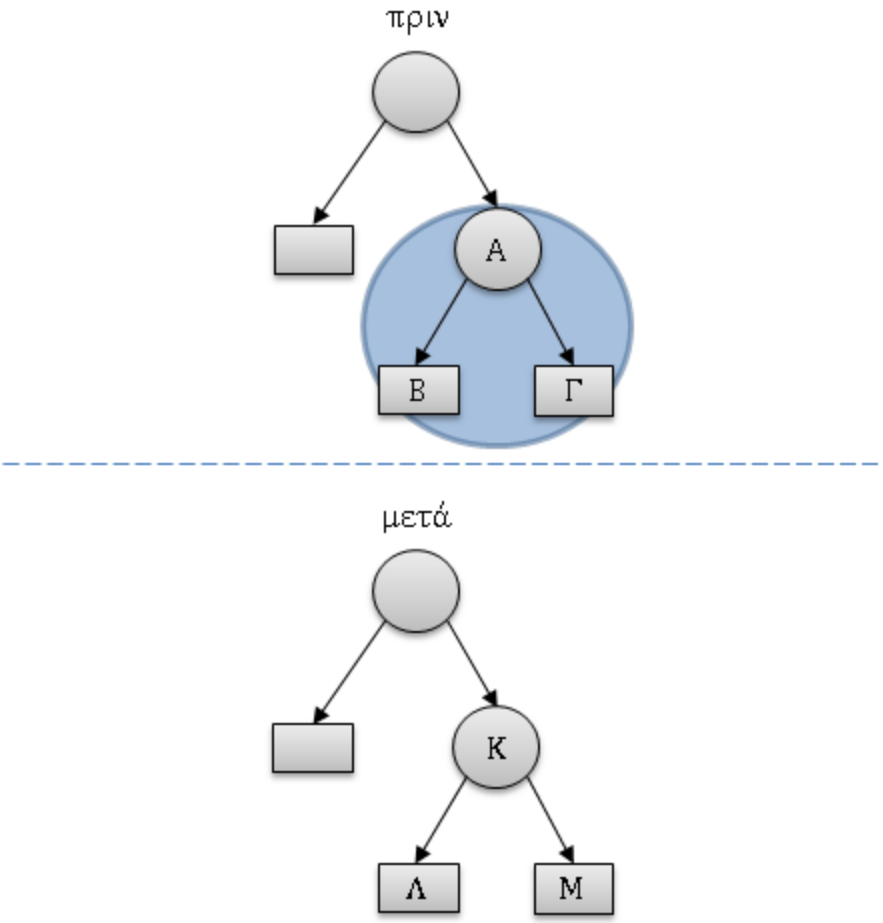
Εικόνα 32: Μετάλλαξη με εσωτερική ανταλλαγή μη επικαλυπτόμενων υποδένδρων.

- Η αντικατάσταση ενός εσωτερικού κόμβου από το ένα από τα δύο υποδένδρα του (Εικόνα 33).



Εικόνα 33: Μετάλλαξη με την αντικατάσταση εσωτερικού κόμβου από υποδένδρο του.

- Η αντικατάσταση ενός υποδένδρου με ένα τυχαία παραγόμενο υποδένδρο (Εικόνα 34).

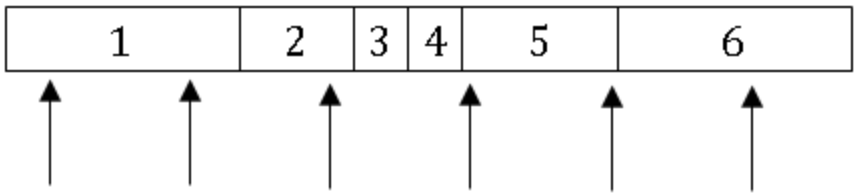


Εικόνα 34: Μετάλλαξη με την αντικατάσταση υποδένδρου από τυχαία παραγόμενο υποδένδρο.

6.6 Τεχνικές επιλογής

Πέρα από τη ρουλέτα γίνονται διαθέσιμες δύο νέες τεχνικές επιλογής.

Η στοχαστική γενική δειγματοληψία (stochastic universal sampling) είναι μια ενδιαφέρουσα παραλλαγή της ρουλέτας. Στην Εικόνα 35, έστω s το άθροισμα των τιμών της αντικειμενικής συνάρτησης και n είναι ο αριθμός των επιλογών που θα να γίνουν. Τότε, ο πρώτος δείκτης επιλέγεται τυχαία μεταξύ 0 και s/n. Οι υπόλοιποι δείκτες απέχουν σταθερά μεταξύ τους κατά s/n [33].



Εικόνα 35: Παράδειγμα στοχαστικής γενικής δειγματοληψίας, όπου επιλέγονται οι υποψήφιοι 1, 2, 5, 6.

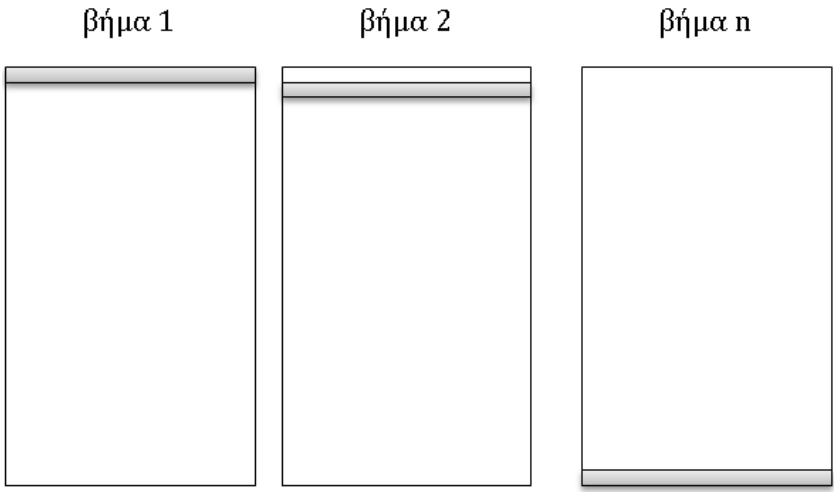
Η παραλλαγή αυτή παρουσιάζει δύο πλεονεκτήματα σε σχέση με τη ρουλέτα. Καταρχήν είναι ελαφρώς πιο γρήγορη, καθώς τρέχει σε O(n) σε σχέση με το O(nlgn) της ρουλέτας. Επίσης, σίγουρα επιλέγονται οι υποψήφιοι με καταλληλότητα μεγαλύτερη από s/n [12].

Στην επιλογή πρωταθλήματος (tournament selection) επιλέγεται τυχαία ένας μικρός αριθμός υποψηφίων, κι από αυτούς προχωράει ο καλύτερος. Αυτό επαναλαμβάνεται μέχρι να συμπληρωθεί ο επιθυμητός αριθμός. Η τεχνική αυτή υλοποιείται εύκολα σε παράλληλα περιβάλλοντα [32].

6.7 Επικύρωση με διασταύρωση

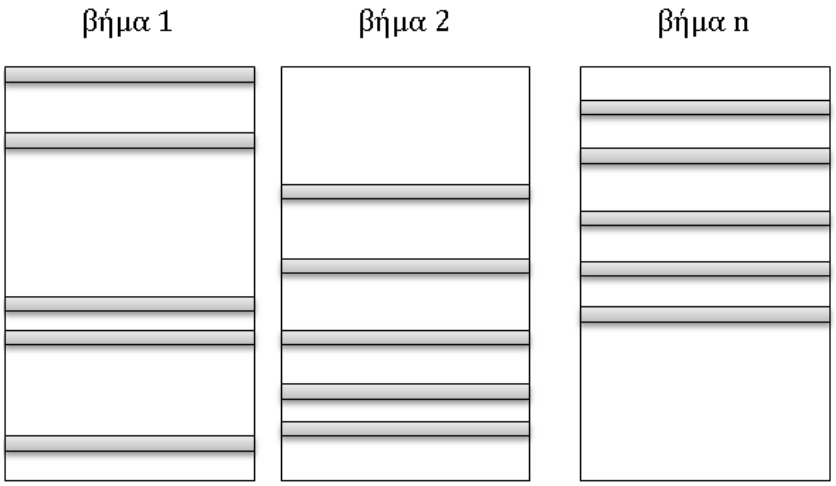
Πέρα από τη διασταυρωμένη επικύρωση σε Κ μέρη, υποστηρίζονται η επικύρωση leave-one-out και η τυχαία δειγματοληψία.

Η επικύρωση με διασταύρωση leave-one-out είναι ειδική περίπτωση της επικύρωσης σε Κ μέρη. Πιο συγκεκριμένα το Κ επιλέγεται ίσο με το πλήθος των δειγμάτων και χρησιμοποιείται ακριβώς ένα (1) δείγμα κάθε φορά για έλεγχο (testing). Στην Εικόνα 36, τα δείγματα προς έλεγχο απεικονίζονται με σκούρο χρώμα και τα προς εκπαίδευση με λευκό. Επίσης, n είναι το πλήθος των δειγμάτων.



Εικόνα 36: Τα βήματα επικύρωσης με διασταύρωση leave-one-out.

Όσον αφορά την τυχαία δειγματοληψία, επιλέγεται σε κάθε βήμα τυχαία ένα πλήθος δειγμάτων (Εικόνα 37). Η χαρακτηριστική διαφορά με τις προηγούμενες επικυρώσεις έγκειται στο ότι δεν είναι απαραίτητο να χρησιμοποιηθούν όλα τα στιγμιότυπα στην εκπαίδευση και τον έλεγχο.



Εικόνα 37: Τα βήματα επικύρωσης με τυχαία δειγματοληψία.

Κεφάλαιο 7
Κατευθύνσεις παραλληλισμού

Η ανοιχτή αρχιτεκτονική υποσυστημάτων που υιοθετήθηκε, ευνοεί την αξιοποίηση υποδομών παράλληλου προγραμματισμού. Το παρόν κεφάλαιο προτείνει συγκεκριμένες τεχνικές που θα μπορούσαν να φανούν χρήσιμες για το σκοπό αυτό.

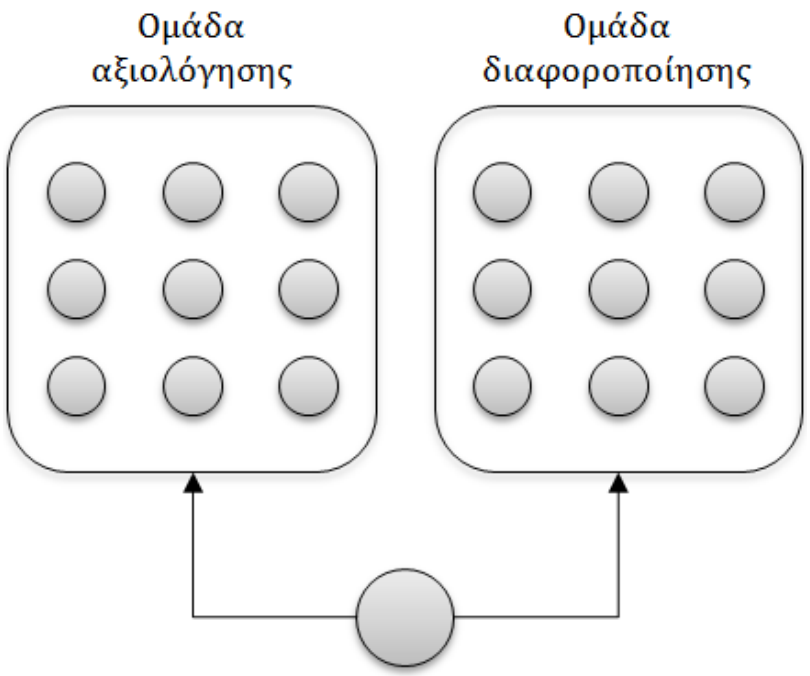
7.1 Δημιουργία peer-to-peer δικτύου

Το JXTA (Juxtapose) είναι ένα σύνολο από ανοιχτά, γενικευμένα Peer-to-Peer (P2P) πρωτόκολλα που επιτρέπουν σε οποιεσδήποτε συσκευές σε ένα δίκτυο να επικοινωνούν και να συνεργάζονται σαν ισότιμες οντότητες. Μάλιστα τα πρωτόκολλα αυτά είναι ανεξάρτητα από τη γλώσσα υλοποίησης, καθώς βασίζονται στο XML. Στόχος του JXTA είναι να δώσει μια πλατφόρμα με τις βασικές λειτουργίες πάνω στις οποίες μπορεί να χτιστεί ένα νοητό P2P δίκτυο υπερκείμενο σε οποιαδήποτε τοπολογία [37].

Ένα δίκτυο που βασίζεται στο JXTA αποτελείται από δια-συνδεδεμένους κόμβους οι οποίοι μπορούν να οργανώνονται σε ομάδες. Η ομάδα είναι ένας οριοθετημένος χώρος στον οποίο εκτελείται ένα συμφωνημένο πλαίσιο υπηρεσιών. Η επικοινωνία εντός της ομάδας λαμβάνει χώρα με την ανταλλαγή μηνυμάτων και μπορεί να είναι απλού προορισμού (unicast) ή πολλαπλών προορισμών (multicast), ασύγχρονη ή σύγχρονη [37].

Στην περίπτωση του GATree θα μπορούσαμε να οργανώσουμε ομάδες κόμβων, κάθε μία από τις οποίες να είναι εστιασμένη σε μία συγκεκριμένη υπηρεσία (π.χ. αξιολόγηση δένδρων). Ο συντονισμός θα γίνεται από έναν (κεντρική οργάνωση) ή περισσότερους (ημι-κεντρική οργάνωση) υπερ-κόμβους.

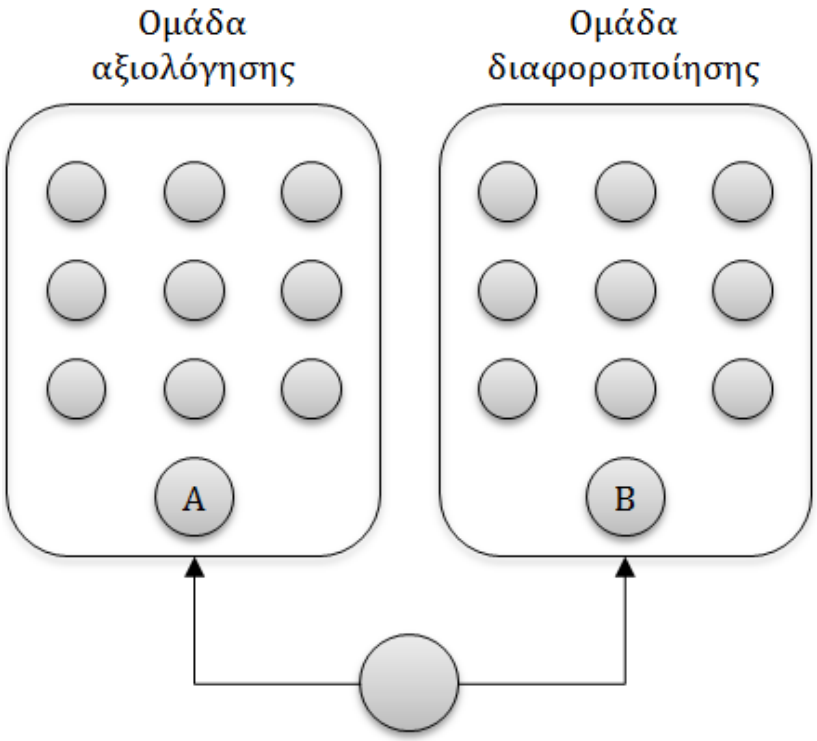
Στην κεντρική οργάνωση ένας μόνο κόμβος είναι υπεύθυνος για την επίβλεψη της διαδικασίας, τη διαχείριση των αιτημάτων, τη λήψη των αποτελεσμάτων κοκ (Εικόνα 38).



Εικόνα 38: Κεντρική οργάνωση peer-to-peer δικτύου με χρήση του JXTA, για την παράλληλη εκτέλεση του νέου συστήματος GATree.

Στην ημι-κεντρική οργάνωση προστίθεται ένας επιπλέον κόμβος ανά ομάδα, στον οποίο μεταβιβάζεται η διαχείριση της επικοινωνίας εντός της ομάδας. Στην Εικόνα 39, ο κόμβος Α τίθεται υπεύθυνος της ομάδας αξιολόγησης, ενώ ο κόμβος Β της ομάδας διαφοροποίησης. Για παράδειγμα, ο κόμβος Α στέλνει στους κόμβους διαφορετικά κομμάτια του πληθυσμού προς

αξιολόγηση, περιμένει τη λήψη της απάντησης και ενεργεί κατάλληλα σε περίπτωση που η απάντηση αργήσει να φθάσει.



Εικόνα 39: Ημι-κεντρική οργάνωση peer-to-peer δικτύου με χρήση του JXTA, για την παράλληλη εκτέλεση του νέου συστήματος GATree.

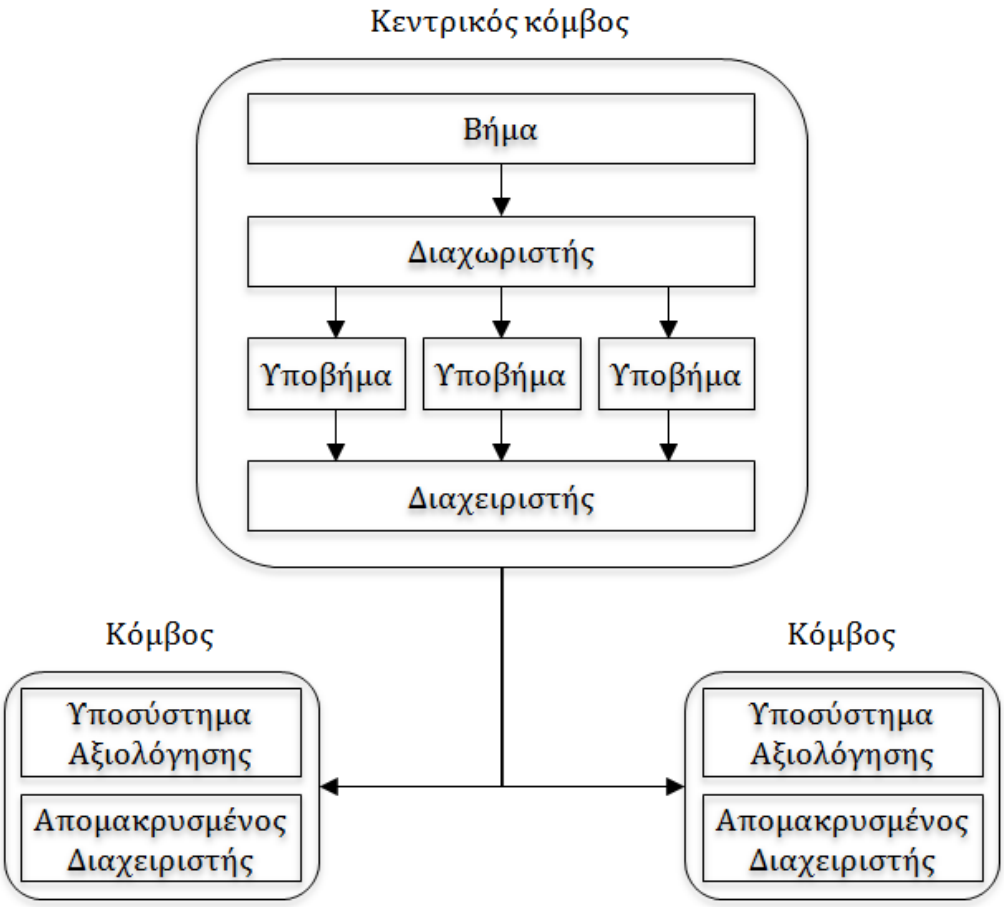
Παρατηρούμε πως το JXTA δίνει τη δυνατότητα σχεδιασμού κατανεμημένων τοπολογιών, με ιδιαίτερα ευέλικτους μηχανισμούς επικοινωνίας – οι οποίοι κρύβουν την πολυπλοκότητα απευθείας χρήσης των Sockets ή των πρωτοκόλλων TCP/UDP. Παρόλαυτα, απαιτείται η ανάπτυξη ενός στοιχειώδους ενδιάμεσου λογισμικού στους υπερ-κόμβους, ώστε να υπάρχει έλεγχος της διαδικασίας.

7.2 Διαχείριση κατανεμημένων εργασιών

Το Spring Batch έχει ως κεντρική έννοια την εργασία (job, batch ή process). Μια εργασία αποτελείται από έναν αριθμό βημάτων. Ένα βήμα μπορεί να εκτελείται παράλληλα με κάποιο άλλο, να εκμεταλλεύεται τους πολυπύρηνους επεξεργαστές, να μπορεί να εκτελεστεί ξανά σε περίπτωση αποτυχίας ή να εκτελείται μόνο υπό συγκεκριμένες προϋποθέσεις. Σε σχέση με την προηγούμενη ενότητα, το σημαντικό είναι πως παρέχεται ένας αρκετά πλήρης μηχανισμός ελέγχου των εργασιών, που περιλαμβάνει τη γνώση της τρέχουσας κατάστασης, τη δυνατότητα

επανέναρξης, τη διαχείριση σφαλμάτων και την καταγραφή στατιστικών επίδοσης. Επιπλέον, χρησιμοποιείται συχνά σε εφαρμογές οι οποίες θέλουν να κάνουν χρήση παράλληλης επεξεργασίας, χωρίς να χρειαστεί να προσαρμόσουν τον κώδικά τους σε συγκεκριμένους αλγόριθμους ή δομές δεδομένων [02].

Ας εξετάσουμε πώς θα μπορούσε να γίνει παράλληλη η αξιολόγηση δένδρων από το GATree. Για το σκοπό αυτό, η πλέον κατάλληλη τεχνική που προσφέρει το Spring Batch, είναι ο διαχωρισμός (partitioning)· ένα βήμα εκλεπτύνεται σε περαιτέρω υποβήματα, καθένα από τα οποία αναλαμβάνει ένα κομμάτι των δεδομένων. Κάθε βήμα μπορεί να εκτελεστεί τοπικά ή απομακρυσμένα, κι επιπλέον έχει τη δυνατότητα πολυνηματικής εκτέλεσης [15].



Εικόνα 40: Διαχωρισμός με χρήση του ενδιάμεσου λογισμικού Spring Batch, για την παράλληλη εκτέλεση του νέου συστήματος GATree [15].

Στην Εικόνα 40, προτείνεται η διάταξη για την κατανομή της αξιολόγησης των μελών του πληθυσμού σε περισσότερους κόμβους. Η προς αξιολόγηση γενιά χωρίζεται σε μικρότερα κομμάτια. Κάθε κομμάτι αναλαμβάνεται από ένα διαφορετικό υποβήμα. Ο διαχειριστής του ενδιάμεσου λογισμικού, ο οποίος παρέχεται έτοιμος από το Spring Batch, διανέμει τα υποβήματα

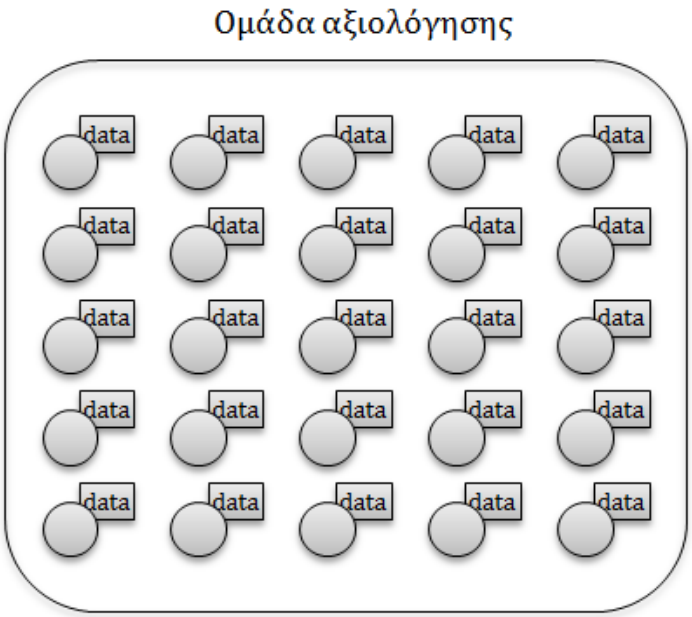
στους απομακρυσμένους κόμβους. Από την πλευρά του GATree, σε κάθε κόμβο είναι απαραίτητο μόνο το υποσύστημα αξιολόγησης. Μόλις οι κόμβοι ολοκληρώσουν την αξιολόγηση των δένδρων, στέλνουν το αποτέλεσμα πίσω στον κεντρικό κόμβο.

Ένα προφανές πλεονέκτημα του Spring Batch είναι πως χρησιμοποιεί το Spring. Όπως είδαμε, το GATree παρέχει τις βασικές λειτουργίες του ως Spring services. Συνεπώς, αναμένεται εύκολη ολοκλήρωση (integration) μεταξύ Spring Batch και GATree. Από την άλλη, το Spring Batch απαιτεί λίγο παραπάνω καιρό εκμάθησης για όποιον δεν είναι εξοικειωμένος με το πεδίο εφαρμογής των εργασιών (εργασία, βήμα, έλεγχος, παραλληλισμός).

7.3 Ενδιάμεσο λογισμικό υπολογιστικού πλέγματος

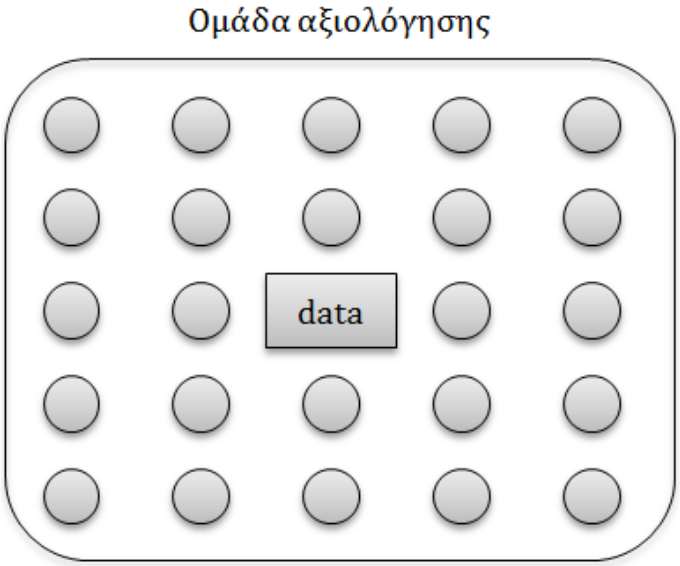
Δύο έτοιμες αλληλοσυμπληρούμενες λύσεις ενδιάμεσου λογισμικού, οι οποίες συνεργάζονται με το Spring, είναι οι Terracotta και GridGain.

Ας εξετάσουμε ένα πρακτικό πρόβλημα. Το GATree χρειάζεται πρόσβαση στη συλλογή δεδομένων (dataset) ανά πάσα στιγμή, για την αξιολόγηση κάθε γενιάς,. Ένας τρόπος είναι να μοιραστούν τα στιγμιότυπα μία φορά σε όλους τους κόμβους (Εικόνα 41). Ενδεχομένως, η πρακτική αυτή μειονεκτεί σε συλλογές μεγάλου μεγέθους, οι οποίες έχουν ιδιαίτερες απαιτήσεις σε μνήμη.



Εικόνα 41: Μοίρασμα των δεδομένων σε κάθε κόμβο, για την παράλληλη εκτέλεση αξιολόγησης πληθυσμών από το νέο σύστημα GATree.

Μια εναλλακτική πρόταση είναι να τοποθετηθούν τα δεδομένα σε ένα σημείο κοινής πρόσβασης, όπως στην Εικόνα 42. Το Terracotta[13] καλύπτει ακριβώς αυτή την ανάγκη· προσφέρει μια μνήμη γρήγορης πρόσβασης, η οποία είναι κοινή μεταξύ διαφορετικών κόμβων. Η μνήμη αυτή είναι πιο ευέλικτη από μια βάση δεδομένων, επειδή αποθηκεύει αντικείμενα. Επιπλέον, είναι πιο γρήγορη από το μηχανισμό της σειριοποίησης (serialization) [13].



Εικόνα 42: Τοποθέτηση των δεδομένων σε μια κοινή μνήμη γρήγορης πρόσβασης, με χρήση του Terracotta, για την παράλληλη εκτέλεση αξιολόγησης πληθυσμών από το νέο σύστημα GATree.

[13] http://terracotta.org/

Το GridGain[14] είναι ένα ενδιάμεσο λογισμικό υπολογιστικού πλέγματος, το οποίο είναι υλοποιημένο εξολοκλήρου σε Java και Spring. Δίνει τη δυνατότητα δημιουργίας και πλήρους παραμετροποίησης κόμβων, στους οποίους μπορεί να μοιραστεί το φορτίο απαιτητικών εργασιών. Τα αποτελέσματα των υπολογισμών συγκεντρώνονται και επιστρέφονται με διάφανο για το χρήστη τρόπο. Ένα αξιοσημείωτο χαρακτηριστικό του είναι πως βασίζεται στο προγραμματιστικό μοντέλο MapReduce [03]. Με πολύ απλό κώδικα, βασισμένο σε annotations, επιτρέπει τη διανομή εργασιών σε κατανεμημένα περιβάλλοντα.

7.4 Ροές εργασίας σε υπολογιστικό πλέγμα

Υπολογιστικό πλέγμα είναι μια συλλογή συνδεδεμένων ετερογενών συστημάτων τα οποία μοιράζονται κυρίως επεξεργαστική ισχύ και αποθηκευτικό χώρο [35]. Ενδεικτικά αναφέρονται το Hellas Grid[15] στην Ελλάδα και το LinkSCEEM-2[16] στην Κύπρο.

Η παραδοσιακή χρήση των υποδομών πλέγματος απαιτούσε εξειδικευμένες γνώσεις, καθώς γινόταν συνήθως μέσω γραμμής εντολών και unix scripts. Το 2012 ο Γεώργας [35] δοκίμασε πιλοτικά, στα πλαίσια της διατριβής του, τη σύνδεση της εξειδικευμένης διαδικτυακής πύλης WS-PGrade[17] με το Hellas Grid, και συνεισέφερε στη σωστή ρύθμιση και αποσφαλμάτωση αυτής. Με τον τρόπο αυτό άνοιξε ο δρόμος για ευκολότερη χρήση του πλέγματος. Πλέον η σχεδίαση ροών, ο συντονισμός των εργασιών και ο έλεγχος εκτέλεσης γίνεται μέσω μιας φιλικής προς το χρήστη γραφικής διεπαφής.

[14] http://gridgain.com/
[15] http://www.hellasgrid.gr
[16] http://www.linksceem.eu/ls2
[17] https://guse.sztaki.hu

Κεφάλαιο 8
Επίλογος

Οι γενετικοί αλγόριθμοι έχουν μια κοινή υποδομή με επιρροές από συγκεκριμένα φαινόμενα της γενετικής, όπως η διασταύρωση, η μετάλλαξη και η επιλογή (Ενότητα 2.3). Από εκεί και πέρα, τα κυριότερα σημεία διαφοροποίησης είναι ο τρόπος με τον οποίο απεικονίζεται (Ενότητα 2.1), δημιουργείται (Ενότητα 2.2) και αξιολογείται (Ενότητα 2.4) κάθε γενιά.

Ο αλγόριθμος GATree (Κεφάλαιο 2) εστιάζει αποκλειστικά σε δυαδικά δένδρα, καθώς τα θεωρεί μια φυσική μορφή αναπαράστασης για συγκεκριμένα προβλήματα (Ενότητα 2.1). Πρόκειται για ένα σημαντικό πλεονέκτημα, καθώς τα δυαδικά δένδρα είναι δομές δεδομένων πολύ φιλικές προς τον αντικειμενοστρεφή τρόπο σκέψης. Αυτό κάνει την απεικόνιση στο πεδίο προβλήματος απλή κι εύληπτη (Ενότητα 4.2). Έπειτα, η αξιολόγηση γίνεται με μια ισορροπημένη συνάρτηση η οποία διαθέτει καλά ποιοτικά χαρακτηριστικά (συνέχεια, μονοτονία, απλούς παράγοντες). Παρόλο που υπάρχει σαφής εύνοια προς τα μικρότερα σε μέγεθος δένδρα [20], δίνεται η δυνατότητα κατεύθυνσης του πειράματος προς μεγαλύτερα, με την επιλογή της τιμής μιας συγκεκριμένης παραμέτρου (Ενότητα 2.4).

Ως μέλος της οικογένειας των γενετικών αλγορίθμων, ο GATree μπορεί να χαρακτηριστεί «αχόρταγος» ως προς τους υπολογιστικούς πόρους. Είναι προφανές πως αν δεν υπήρχε περιορισμός σε χρόνο, επεξεργαστική ισχύ και μνήμη, θα μπορούσαμε να εκτελούμε πειράματα με ολοένα μεγαλύτερο πληθυσμό και ολοένα μεγαλύτερο αριθμό γενεών. Η σχεδίαση του συστήματος έγινε έχοντας κατά νου αυτή την ανάγκη για πόρους, σε μια εποχή όπου η ανάπτυξη κατανεμημένων συστημάτων διατηρεί αμείωτο ερευνητικό ενδιαφέρον (Κεφάλαιο 8).

Αναπτύχθηκε διαδικτυακή εφαρμογή που κατασκευάζει δένδρα απόφασης με τον αλγόριθμο GATree, με βάση συγκεκριμένες προδιαγραφές (Κεφάλαιο 3). Σε σχεδιαστικό επίπεδο, επιλέχθηκε η οργάνωση γύρω από μικρά διακριτά και αυτοτελή υποσυστήματα. Η προσέγγιση αυτή ευνοεί τόσο την επεκτασιμότητα όσο και τη συντηρησιμότητα (Ενότητα 5.1). Η αρχιτεκτονική βασίζεται σε καλές πρακτικές (Ενότητα 5.2) κι εκλεπτύνεται με γνωστά τεχνολογικά εργαλεία (Ενότητα 5.3): Java, Spring και Maven (Παράρτημα Α).

Η συνεισφορά του συστήματος είναι η άμεση πρόσβαση που παρέχει πλέον μέσω διαδικτύου, η ενσωμάτωσή του σε έναν σαφή κύκλο ζωής που επιτρέπει την περαιτέρω ανάπτυξή του εμπορικά ή εντός μιας κοινότητας ανοιχτού λογισμικού, και η προσφορά μιας σειράς από νέες μορφές μετάλλαξης, τεχνικές επιλογής και μεθόδους επικύρωσης με διασταύρωση (Κεφάλαιο 6). Μια προφανής προοπτική επέκτασης είναι η ανάπτυξη διαδικτυακών υπηρεσιών προστιθέμενης αξίας προς τους ερευνητές και προς εξωτερικά συστήματα. Επιπλέον, στα πλαίσια μιας μεταπτυχιακής διατριβής, θα μπορούσε να γίνει τεκμηριωμένη αξιολόγηση των νέων επιλογών παραμετροποίησης και της συμβολής τους στο επιστημονικό αποτέλεσμα.

Έπειτα, εξετάστηκε η κατανομή και ο συντονισμός παράλληλης εκτέλεσης των υπηρεσιών του συστήματος με σκοπό την άρση του περιορισμού των υπολογιστικών πόρων. Προτείνεται η δημιουργία ενός peer-to-peer συνεργατικού δικτύου με το JXTA (Ενότητα 7.1), η διαχείριση εργασιών με το Spring Batch (Ενότητα 7.2), η ολοκλήρωση με τα ενδιάμεσα λογισμικά Terracotta και GridGain (Ενότητα 7.3), και τέλος ο ορισμός υπολογιστικών ροών σε υπάρχουσες υποδομές πλέγματος (Ενότητα 7.4). Κάθε μία από τις τεχνικές αυτές, ή ανά ζεύγη, θα μπορούσε να αποτελέσει πρόσφορο έδαφος για μεταπτυχιακές διατριβές στο άμεσο μέλλον.

Τέλος, μια ενδιαφέρουσα πτυχή αποτέλεσε η προσπάθεια προσαρμογής της μεθοδολογίας SCRUM στα πλαίσια μιας μεταπτυχιακής διατριβής (Ενότητα Β.5). Οι ευέλικτες μεθοδολογίες είναι πλέον δημοφιλείς στο χώρο του Λογισμικού και άρχισαν ήδη να διδάσκονται σε

πανεπιστημιακό επίπεδο[18]. Στην Ενότητα Β.6 αξιολογούνται οι βασικοί λόγοι για τους οποίους απέτυχε εν τέλει η εφαρμογή στην παρούσα διατριβή· αναφέρεται επίσης κι ένα παράδειγμα επιτυχίας. Η αυτο-οργάνωση, που καλλιεργεί η συγκεκριμένη μεθοδολογία, ταιριάζει με τις ιδιαίτερες συνθήκες μια μεταπτυχιακής διατριβής, όπου ουσιαστικά ο φοιτητής φέρει την ευθύνη εκπόνησης. Επίσης, οι τακτικές προκαθορισμένες συναντήσεις (σύντομες, ανασκόπησης, αξιοποίησης εμπειρίας) συμβάλλουν στη διατήρηση ανοιχτών καναλιών επικοινωνίας. Ο καθηγητής (product owner) δίνει τις απαραίτητες κατευθύνσεις, έχοντας μια σαφέστερη εικόνα του ευρύτερου χώρου. Ο εμπλεκόμενος ερευνητής (ScrumMaster) συμβάλλει επικουρικά στην ομαλή προσαρμογή του φοιτητή. Ο τελευταίος μαθαίνει να συν–εργάζεται με βάση καλά δοκιμασμένες πρακτικές. Τέλος, είναι ιδιαίτερα σημαντικό να αναπτύσσεται νωρίς ένα πρωτότυπο του συστήματος, καθώς, μεταξύ άλλων, εκλεπτύνεται γρηγορότερα ο στόχος της μεταπτυχιακής διατριβής.

[18] https://www.coursera.org/course/saas

Βιβλιογραφία

[01] G. D. Battista, P. Eades, R. Tamassia, I. G. Tollis. «Algorithms for Drawing Graphs: an Annotated Bibliography». Computational Geometry: Theory and Applications, Number 4, 235-282, 1994.

[02] A. Cogoluègnes, T. Templier, O. Bazoud, G. Gregory. «Spring Batch in Action». Manning Publications, NY, 2012.

[03] J. Dean, S. Ghemawat. «MapReduce: Simplified Data Processing on Large Clusters». Google, Inc, 2004.

[04] M. Fowler. «UML Distilled: A Brief Guide to the Standard Object Modeling Language». 3rd Edition, Addison Wesley, USA, 2004.

[05] A. Fox, D. Patterson. «Engineering Long-Lasting Software: An Agile Approach Using SaaS and Cloud Computing». Alpha Edition, Strawberry Canyon LLC, 2012.

[06] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten. «The Weka Data Mining Software: An Update». SIGKDD Explorations, Volume 11, Number 1, 10-18, 2009.

[07] M. Huttermann. «Agile ALM - Lightweight tools and Agile strategies». Manning Publications, NY, 2012.

[08] D. Kalles, A. Papagelis. «Lossless fitness inheritance in genetic algorithms for decision trees». Soft Computing - A Fusion of Foundations, Methodologies and Applications, Volume 14, Number 9, 973-993, 2010.

[09] M. Kantardzic. «Data Mining: Concepts, Models, Methods, and Algorithms». Wiley, 2nd edition, New Jersey, 2011.

[10] M. Keith, M. Schnicariol. «Pro JPA 2 - Mastering the Java Persistence API». Apress, USA, 2009.

[11] S. Luke, L. Panait, G. Balan, S. Paus, Z. Skolicki, E. Popovici, K. Sullivan, J. Harrison, J. Bassett, R. Hubley, A. Chircop, J. Compton, W. Haddon, S. Donnelly, B. Jamil, J. Zelibor, E. Kangas, F. Abidi, H. Mooers, J. O'Beirne. «ECJ 20: A Java-based Evolutionary Computation Research System». http://cs.gmu.edu/~eclab/projects/ecj, Evolutionary Computation Laboratory, George Mason University.

[12] S. Luke. «Essentials of Metaheuristics». http://cs.gmu.edu/~sean/book/metaheuristics/, Lulu, 2009.

[13] G. Mak, J. Long, D. Rubio. «Spring Recipes». Apress, Second Edition, NY, 2010.

[14] K. Meffert, J. Meseguer, E. D. Martv, A. Meskauskas, J. Vos, N. Rotstan. «JGAP - Java Genetic Algorithms and Genetic Programming Package». http://jgap.sf.net.

[15] M. T. Minella. «Pro Spring Batch». Apress, NY, 2011.

[16] M. Moser, T. O'Brien. «The Hudson Book». Oracle, Edition 1.0, 2011.

[17] P. Mularien. «Spring Security 3». Packt Publishing, UK, 2010.

[18] T. O'Brien, M. Moser, J. Casey, B. Fox, J. V. Zyl, E. Redmond, L. Shatzer. «Maven: The Complete Reference». Sonatype, Edition 1.0, 2011.

[19] A. Papagelis, D. Kalles. «GATree: Genetically Evolved Decision Trees». IEEE International Conference on Tools with Artificial Intelligence, Vancouver, Canada, 2000.

[20] A. Papagelis, D. Kalles. «Breeding Decision Trees Using Evolutionary Techniques». International Conference on Machine Learning, Williamstown, Massachusetts, June-July 2001.

[21] J. Petri. «NetBeans Platform 6.9 Developer's Guide». Packt Publishing, UK, 2010.

[22] A. Pham. «Scrum in Action: Agile Software Project Management and Development». Course Technology PTR, USA, 2011.

[23] D. Rosenberg, M. Stephens. «Use Case Driven Object Modeling with UML: Theory and Practice». Apress, New York, 2007.

[24] D. Rosenberg, K. Scott. «Applying Use Case Driven Object Modeling with UML: An Annotated e-Commerce Example». Addison Wesley, New Jersey, 2001.

[25] G. Sander. «VCG: Visualization of Compiler Graphs». Technical Report A01-95, Universität des Saarlandes, FB 14 Informatik, Edition 1.30, 1995.

[26] E. Sink. «Version Control by Example». Pyrenean Gold Press, USA, 2011.

[27] P. N. Tan, M. Steinbach, V. Kumar. «Introduction to Data Mining». Pearson Education, 2006.

[28] M. Wall. «GAlib: A C++ Library of Genetic Algorithm Components». M.I.T., 1996.

[29] C. Walls. «Spring in Action». Manning Publications, 3rd Edition, USA, 2011.

[30] D. Wellman. «jQuery UI 1.8 - The User Interface Library for jQuery». Packt Publishing, UK, 2011.

[31] I. H. Witten, E. Frank, M. A. Hall. «Data Mining: Practical Machine Learning Tools and Techniques». Morgan Kaufmann, 3rd edition, Morgan Kaufmann, 2011.

[32] Y. Xinjie, G. Mitsuo. «Introduction to Evolutionary Algorithms». Springer, 2010.

[33] Ν. Αννίνου. «Γενετικοί και Μετά-Γενετικοί Αλγόριθμοι και η Εφαρμογή τους στην Εκτίμηση ARMA Μοντέλων». Πανεπιστήμιο Πατρών, 2009.

[34] Ι. Βλαχάβας, Π. Κεφαλάς, Ν. Βασιλειάδης, Ι. Ρεφανίδης, Φ. Κόκκορας, Η. Σακελλαρίου. «Τεχνητή Νοημοσύνη». Έκδοση 1η, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, 2002.

[35] Α. Γεώργας. «Εκτέλεση ροών εργασίας στο υπολογιστικό πλέγμα μέσω διαδικτυακής πύλης». Ελληνικό Ανοικτό Πανεπιστήμιο, 2012.

[36] Β. Μπερουκλή. «Νοημοσύνη Σμήνους και Γενετικός Προγραμματισμός». HOU-CS-UGP-2010-03, Ελληνικό Ανοικτό Πανεπιστήμιο, 2010.

[37] Ν. Πουγούνιας. «Δημιουργία peer-to-peer δικτύου συνεργασίας, με σκοπό την αποτελεσματική αντιμετώπιση των Κατανεμημένων Επιθέσεων Άρνησης Υπηρεσίας». Εθνικό Μετσόβιο Πολυτεχνείο, 2004.

[38] Ε. Τσεκρέκου. «Υπηρεσία Ανάκτησης Γεωπληροφοριών με βάση τη Θέση (Location Based Services) - Εφαρμογή AthensShops». Εθνικό Μετσόβιο Πολυτεχνείο, 2012.

Πίνακας ακρωνυμίων

AOP	Aspect Oriented Programming
API	Application Programming Interface
ARFF	Attribute – Relation File Format
CI	Continuous Integration
CORDIS	Community Research and Development Information Service
CR	Component Repository
DG EAC	Directorate-General for Education and Culture
DI	Dependency Injection
EACEA	Education, Audiovisual and Culture Executive Agency
ECJ	Java-based Evolutionary Computation Research System
GATree	Genetic Algorithm Tree
GAlib	Genetic Algorithms Library
GDL	Graph Description Language
IDE	Integrated development environment
JGAP	Java Genetic Algorithms Package

JPA	Java Persistence API
JSP	JavaServer Pages
JSTL	JavaServer Pages Standard Tag Library
JXTA	Juxtapose
OHIM	Office of Harmonization for the Internal Market
P2P	Peer-to-Peer
POJO	Plain Old Java Objects
REST	REpresentational State Transfer
TDD	Test Driven Development
VCG	Visualization of Compiler Graphs
VCS	Version Control System
XML	Extensible Markup Language
ΕΚΤ	Εθνικό Κέντρο Τεκμηρίωσης
ΕΛ/ΛΑΚ	Ελεύθερο Λογισμικό / Λογισμικού Ανοιχτού Κώδικα

Παράρτημα Α
Διαχείριση έργου με το Maven

Το Maven είναι εργαλείο διαχείρισης έργων λογισμικού, φτιαγμένο για μηχανικούς λογισμικού. Εγγυάται διαφανή διαχείριση του κύκλου ζωής ανεξάρτητα από το εργαλείο ανάπτυξης [18]. Για να δοθεί μια εικόνα εξ αρχής, ας δούμε μερικές ενδεικτικές ενέργειες στις οποίες προβαίνει το Maven όταν χτίζεται ένα έργο:

1. Κατέβασμα των εξαρτήσεων.
2. Μεταγλώττιση του πηγαίου κώδικα.
3. Εκτέλεση προγραμματιστικών ελέγχων.
4. Σχηματισμός της παραγόμενης εφαρμογής.
5. Εγκατάσταση της εφαρμογής στο τοπικό περιβάλλον.
6. Εκτέλεση ελέγχων γραφικής διεπαφής.
7. Μεταφορά των παραγόμενων στο κεντρικό σύστημα ποιοτικού ελέγχου.

8. Εγκατάσταση των παραγόμενων στο τοπικό αποθετήριο.

A.1 Τι προσφέρει

Εύκολο χτίσιμο. Ένας προγραμματιστής μπορεί να πάρει οποιοδήποτε έργο και να το χτίσει αμέσως. Τα σημαντικότερα IDEs (Integrated Development Environments) υποστηρίζουν άμεσα το Maven (Eclipse, NetBeans, Intellij).

Διαχείριση δομής. Ορίζεται μια απλή, ενιαία δομή στον κώδικα, όπου διαχωρίζεται σαφώς η υλοποίηση των λειτουργικών απαιτήσεων από τους προγραμματιστικούς ελέγχους. Αυτή η σταθερή δομή συμβάλλει στην ταχύτερη κατανόηση ενός υπάρχοντος έργου.

Γρήγορη εκκίνηση. Παρέχονται αρχέτυπα για γρήγορη εκκίνηση ενός έργου με βάση αναγνωρισμένες καλές πρακτικές. Πρόκειται για βασικά έργα διαφόρων τύπων με έτοιμες ρυθμίσεις και ενδεικτική λειτουργικότητα, που υποβάλλονται από εταιρείες και κοινότητες ανοιχτού λογισμικού.

Διαχείριση παραγόμενων. Ο προγραμματιστής δεν χρειάζεται πια να κατεβάζει παραγόμενα και να τις προσθέτει με το χέρι στο classpath. Ούτε να τα αποθηκεύει στο σύστημα διαχείρισης πηγαίου κώδικα μαζί με το εκάστοτε έργο. Αρκεί να δηλώσει την εξάρτηση (π.χ. commons-logging) και το Maven θα αναλάβει τα υπόλοιπα. Πέρα από την ευκολία του ενός, αυτό είναι χρήσιμο για το συγχρονισμό των μελών μιας ομάδας, αλλά και ομάδων ανάπτυξης εντός του ίδιου οργανισμού. Έχοντας το δικό του αποθετήριο, ο οργανισμός έχει τη δυνατότητα εσωτερικής διαχείρισης των παραγομένων που αναπτύσσει με έναν αποδοτικό τρόπο.

Συνεργασία με συστήματα συνεχούς ολοκλήρωσης (Hudson [12], Jenkins, TeamCity, Bamboo), αυτοματοποιημένου ελέγχου (Selenium), διαχείρισης ποιότητας (Sonar), παλαιότερα εργαλεία χτισίματος (Ant) και άλλες γλώσσες προγραμματισμού (Ruby, Scala, Groovy).

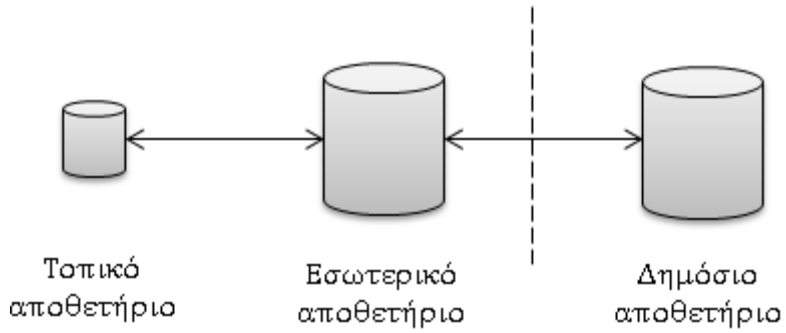
A.2 Αποθετήρια

Το Maven διατηρεί ένα τοπικό αποθετήριο παραγόμενων στον υπολογιστή του προγραμματιστή. Το αποθετήριο αυτό είναι συνδεδεμένο με ένα ή περισσότερα δημόσια αποθετήρια, όπως στην Εικόνα 43.

Εικόνα 43: Τυπική διάταξη αποθετηρίων παραγομένων για έναν προγραμματιστή.

Όταν ο προγραμματιστής ζητήσει ένα παραγόμενο, γίνεται πρώτα αναζήτηση στο τοπικό αποθετήριο. Εάν δεν υπάρχει εκεί, τότε η αναζήτηση διευρύνεται στο δημόσιο αποθετήριο. Σε κάθε περίπτωση το παραγόμενο καταλήγει τελικά εγκατεστημένο στο τοπικό αποθετήριο, σε ακριβώς ένα (1) σημείο στον υπολογιστή του προγραμματιστή. Από αυτό το σημείο, το Maven ρυθμίζει το classpath των έργων με αυτόματο τρόπο.

Μια τυπική διάταξη εντός ενός οργανισμού παρουσιάζεται στην Εικόνα 44.

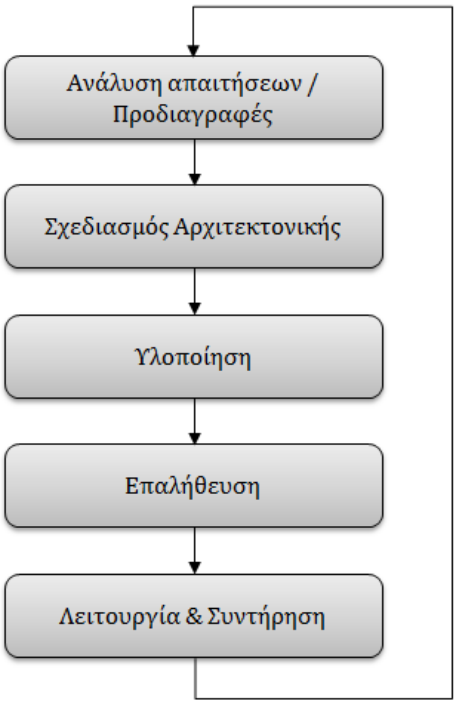


Εικόνα 44: Τυπική διάταξη αποθετηρίων παραγομένων εντός ενός οργανισμού.

Τα τοπικά αποθετήρια των προγραμματιστών συνδέονται με το εσωτερικό αποθετήριο του οργανισμού. Το τελευταίο δεν είναι προσβάσιμο απ' έξω, αλλά διατηρεί σύνδεση με ένα ή περισσότερα δημόσια αποθετήρια. Με τον τρόπο αυτό δίνεται η δυνατότητα στον οργανισμό να διαχειριστεί τα παραγόμενα που αναπτύσσει με βάση την εσωτερική του πολιτική.

Παράρτημα Β
Η ευέλικτη μεθοδολογία SCRUM

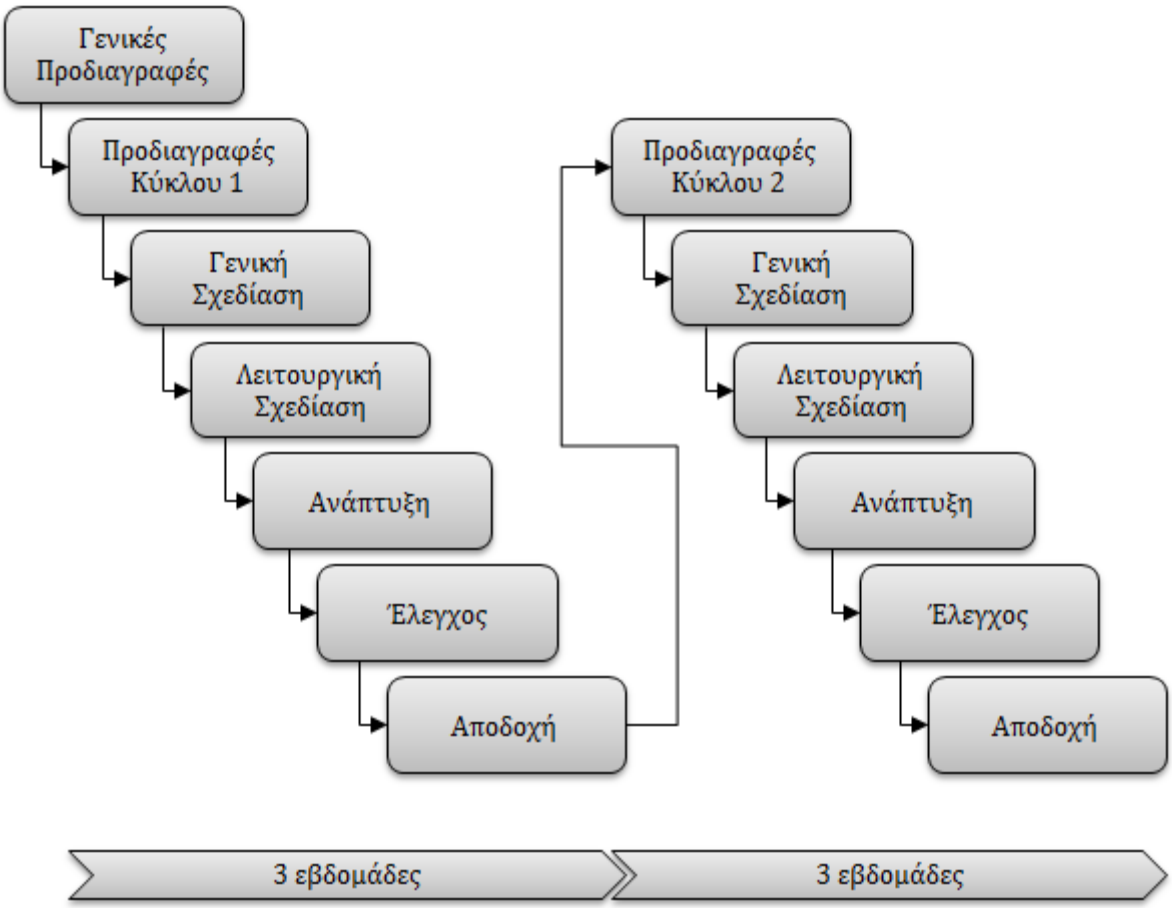
Η κλασική διαδικασία ανάπτυξης με βάση το μοντέλο του «καταρράκτη» είναι χωρισμένη σε διακριτές φάσεις μεγάλης διάρκειας, όπου ο σχεδιασμός ολοκληρώνεται και αποτυπώνεται πλήρως προτού ξεκινήσει οποιαδήποτε υλοποίηση (Εικόνα 45). Έχοντας καταβολές στην παραγωγή βιομηχανικών προϊόντων υλικού, επιδιώκει τον εντοπισμό των σφαλμάτων όσο το δυνατόν νωρίτερα. Αφετέρου, η αναλυτική τεκμηρίωση βοηθάει να προσαρμοστούν ευκολότερα τα άτομα που εισέρχονται αργότερα στο έργο. Αξίζει να σημειωθεί πως έχει καλή εφαρμογή σε λογισμικό των οποίων οι προδιαγραφές δεν αλλάζουν, όπως συστήματα εναέριου ελέγχου ή αεροσκαφών [05].



Εικόνα 45: Κύκλος ζωής λογισμικού κατά το μοντέλο του «καταρράκτη».

Όταν ο πελάτης βλέπει ένα λειτουργικό αποτέλεσμα, θέλει συνήθως μεγάλες αλλαγές. Το ίδιο συμβαίνει και με τους προγραμματιστές, οι οποίοι αντιλαμβάνονται πώς θα μπορούσαν να δομήσουν καλύτερα ένα κομμάτι του λογισμικού, μόνο αφού το φτιάξουν. Εδώ ακριβώς εισέρχονται οι ευέλικτες μεθοδολογίες· θεωρούν τις συχνές αλλαγές δεδομένες επειδή οδηγούν σε μια λύση με μεγαλύτερη ακρίβεια. Γι' αυτό εστιάζουν στη διαρκή βελτίωση μιας λειτουργικής έκδοσης της εφαρμογής υπό την εποπτεία των πελατών.

Η SCRUM είναι μια διαδικασία σταδιακής και ελεγχόμενης ανάπτυξης λογισμικού, η οποία εντάσσεται στην οικογένεια των ευέλικτων μεθοδολογιών. Το έργο χωρίζεται σε μικρούς επαναληπτικούς κύκλους, όπως φαίνεται στην Εικόνα 46. Σε κάθε κύκλο αναπτύσσεται ένα συγκεκριμένο κομμάτι της λειτουργικότητας του συστήματος, το οποίο είναι πάντοτε διαθέσιμο για χρήση. Κύριο χαρακτηριστικό της μεθοδολογίας είναι η ανθρώπινη αλληλεπίδραση εντός της ομάδας, αλλά και η ουσιαστική συνεργασία – σύμπραξη μεταξύ της ομάδας και του πελάτη [22].



Εικόνα 46: Σταδιακή ανάπτυξη λογισμικού με βάση τη μεθοδολογία SCRUM.

Β.1 Ομάδα

Η ομάδα που αναλαμβάνει να φέρει σε πέρας το έργο αποτελείται από τον ιδιοκτήτη (product owner), ένα άτομο που γνωρίζει καλά τη μεθοδολογία (ScrumMaster), και την καθαυτή ομάδα ανάπτυξης.

Ο ιδιοκτήτης καλλιεργεί το όραμα του έργου, όντας στην ουσία η «φωνή» των πελατών. Εκπροσωπεί τους χρήστες και τα ενδιαφερόμενα μέρη, διατηρεί ανοιχτά κανάλια επικοινωνίας μαζί τους και λαμβάνει ανάδραση από αυτούς. Επίσης, καθορίζει τα παραδοτέα, την προτεραιότητά τους και αποφασίζει πότε θα δοθούν οι επίσημες εκδόσεις του λογισμικού.

Ο ScrumMaster προστατεύει την ομάδα και εξασφαλίζει την καλή λειτουργία της. Για το λόγο αυτό καλό είναι να διαθέτει καλή θεωρητική και πρακτική γνώση της μεθοδολογίας. Πέρα από την επιδίωξη καλής εμπέδωσης των διαδικασιών, βοηθάει επικουρικά στην καταγραφή της

προόδου του έργου και συμβάλλει στην αξιοποίηση της εμπειρίας που αποκτάται κατά το πέρασμα από τον ένα κύκλο στον επόμενο.

Η ομάδα ανάπτυξης είναι συνήθως μικρή σε μέγεθος (έως 10 άτομα) και αυτόνομη, δηλαδή γνωρίζει πώς να αυτο-διαχειρίζεται και να αυτο-οργανώνεται. Μερικές από τις εργασίες που επιτελεί είναι η εκλέπτυνση των παραδοτέων σε μικρότερες εργασίες, η καταγραφή της προόδου κάθε κύκλου, η υλοποίηση, η βελτίωση και η παρουσίαση της δουλειάς που επιτελεί σε όλα τα εμπλεκόμενα μέρη.

Β.2 Πρώιμη φάση

Όλα ξεκινούν όταν ο ιδιοκτήτης καταγράψει ένα αρχικό πλάνο απαιτήσεων (product backlog). Πρόκειται για μια λίστα με το «τί» θα γίνει, συνοδευόμενη από μια πρόχειρη εκτίμηση της προσπάθειας. Επίσης, στο σημείο αυτό συντάσσεται μια πρώτη αρχιτεκτονική – ένα πρώτο σχέδιο του συστήματος σε υψηλό επίπεδο, χωρίς λεπτομέρειες.

Β.3 Φάση ανάπτυξης

Το έργο χωρίζεται σε επαναληπτικούς κύκλους διάρκειας από μία εβδομάδα έως ένα μήνα ο καθένας. Το πλάνο και η αρχιτεκτονική εξελίσσονται σε κάθε κύκλο. Ταυτόχρονα πραγματοποιούνται προγραμματισμένες συναντήσεις με πολλαπλά οφέλη.

Β.3.1 Συναντήσεις στην αρχή του κύκλου

Στην αρχή κάθε κύκλου λαμβάνει χώρα η **συνάντηση σχεδιασμού**. Παρευρίσκονται όλοι οι εμπλεκόμενοι καθώς και η ομάδα ανάπτυξης. Ο ιδιοκτήτης επιλέγει τις απαιτήσεις που θα υλοποιηθούν και προχωράει σε μια επισκόπησή τους, εξηγώντας την εκάστοτε ανάγκη των χρηστών. Επιπρόσθετα, ορίζει προτεραιότητες διεκπεραίωσης (υψηλή, μεσαία, χαμηλή). Η ομάδα ανάπτυξης παρακολουθεί φροντίζοντας ο όγκος δουλειάς που θα επιλεγεί να χωράει σε έναν κύκλο.

Μετά την προηγούμενη συνάντηση, την ίδια ή την επόμενη ημέρα, λαμβάνει χώρα η **συνάντηση παραδοτέων**, στην οποία παρευρίσκεται αποκλειστικά η ομάδα ανάπτυξης. Πρώτα υπολογίζονται οι διαθέσιμες ανθρωποημέρες του κύκλου με βάση ένα εύκολο σύστημα

υπολογισμού πόντων και τη διαθεσιμότητα των μελών της ομάδας (τυχόν άδειες, ταυτόχρονη συμμετοχή σε κάποιο άλλο έργο κλπ). Στη συνέχεια, η ομάδα αναλύει τις ιστορίες χρήστη και τις σπάει σε σαφώς ορισμένες εργασίες. Σε κάθε εργασία δίνεται ένας τίτλος και μια αρχική εκτίμηση χρόνου που απαιτείται για την ολοκλήρωσή της.

B.3.2 Καθημερινές συναντήσεις

Η καθημερινή συνάντηση (scrum) λαμβάνει χώρα κάθε μέρα, κατά τη διάρκεια του κύκλου. Όλοι μπορούν να προσέλθουν, αλλά μόνο η ομάδα ανάπτυξης και ο ScrumMaster έχουν ενεργό συμμετοχή. Είναι προγραμματισμένη στο ίδιο μέρος κάθε μέρα, αρχίζει πάντοτε στην ώρα της και περιορίζεται αυστηρά στα 15 λεπτά.

Κατά τη διάρκεια της καθημερινής συνάντησης κάθε μέλος της ομάδας ανάπτυξης απαντάει στα εξής ερωτήματα:

- Τι έκανε εχθές.
- Τι σκοπεύει να κάνει σήμερα.
- Εάν αντιμετωπίζει κάποιο πρόβλημα σχετικά με την επίτευξη του στόχου του.

B.3.3 Συναντήσεις στο τέλος του κύκλου

Την τελευταία ημέρα κάθε κύκλου λαμβάνει χώρα η **συνάντηση ανασκόπησης**, στην οποία συμμετέχουν όλοι. Αρχικά, η ομάδα κάνει μια ανασκόπηση της δουλειάς που ολοκληρώθηκε και που τελικά δεν ολοκληρώθηκε. Έπειτα, γίνεται παρουσίαση της ολοκληρωμένης δουλειάς στα ενδιαφερόμενα μέρη. Έπειτα πραγματοποιείται η συνάντηση **αξιοποίησης εμπειρίας**, όπου τα μέλη της ομάδας αξιολογούν τον κύκλο που πέρασε, διατυπώνουν βελτιώσεις επί της διαδικασίας και συζητούν τί πήγε καλά κατά τη διάρκεια του κύκλου και τι χρήζει βελτίωσης ενόψει του επόμενου κύκλου.

B.4 Φάση κλεισίματος

Η φάση κλεισίματος ξεκινάει όταν συμφωνηθεί πως όλες οι απαιτήσεις έχουν ικανοποιηθεί. Το σύστημα θεωρείται πια ολοκληρωμένο. Δεν μπορούν να ανοίξουν νέα θέματα ούτε νέες απαιτήσεις.

Επίσης, ολοκληρώνεται ο ποιοτικός έλεγχος που αποτελείται από τους εξής τομείς:

- Έλεγχοι από τους χρήστες
- Προγραμματιστικοί έλεγχοι
- Έλεγχοι εντός της ομάδας ανάπτυξης

Όσον αφορά τους προγραμματιστικούς ελέγχους, προτείνεται ανεπιφύλακτα η πρακτική του Test Driven Development (TDD), κατά την οποία ο μηχανικός πρώτα δημιουργεί τους προγραμματιστικούς ελέγχους κι έπειτα αναπτύσσει τη λειτουργικότητα που αφορούν. Συμπληρωματικά έρχεται η χρήση υποδομής συνεχούς ολοκλήρωσης, που αυτοματοποιεί την εκτέλεση των προγραμματιστικών ελέγχων κι εξασφαλίζει πως το σύστημα είναι διαρκώς έτοιμο για πραγματική αξιολόγηση από τους χρήστες. Ταυτόχρονα, ο μηχανικός ελέγχου που είναι μέρος της ομάδας ανάπτυξης, πραγματοποιεί έλεγχο και αποτύπωση των λειτουργιών κάθε κύκλου. Μεγαλύτερη βαρύτητα όλων έχει φυσικά ο έλεγχος και η ανάδραση από τους τελικούς χρήστες.

Έπειτα, ολοκληρώνεται η εγκατάσταση στην παραγωγή. Το γεγονός πως η εφαρμογή έχει εγκατασταθεί πολλές φορές σε διάφορα περιβάλλοντα, ώστε να είναι συνεχώς διαθέσιμη για αξιολόγηση, μειώνει σημαντικά τον κίνδυνο κατά τη διαδικασία αυτή. Η ομάδα διαθέτει πλέον μια παρατεταμένη εμπειρία.

Τέλος, καταγράφονται τα μαθήματα και οι εμπειρίες που αποκόμισε η ομάδα, ώστε να τα μοιραστούν με άλλες ομάδες και να συμβάλλουν στη βελτίωση των διαδικασιών για μελλοντικά έργα.

B.5 Προτάσεις προσαρμογής σε διατριβή

Ας εξετάσουμε πώς θα μπορούσε να προσαρμοστεί η μεθοδολογία στα πλαίσια μιας μεταπτυχιακής διατριβής.

Η ομάδα ανάπτυξης αποτελείται από το φοιτητή μόνο. Το γεγονός αυτό έχει ως άμεση συνέπεια την αδυναμία πραγματοποίησης των καθημερινών συναντήσεων. Βέβαια, θα μπορούσε ο φοιτητής να αξιολογεί μόνος του κάθε μέρα το τί έκανε την προηγούμενη ημέρα, τί θα κάνει σήμερα και τί προβλήματα αντιμετωπίζει για την επίτευξη των στόχων του κύκλου. Με τον τρόπο αυτό όμως αναιρείται η χρησιμότητα της συνεργασίας. Λύση στο θέμα δίνει η εκπόνηση της διατριβής από δύο φοιτητές.

Ο ιδιοκτήτης είναι σαφώς ο καθηγητής. Δίνει τα κατάλληλα ερευνητικά ερεθίσματα στο φοιτητή, έχοντας ταυτόχρονα μια σαφέστερη εικόνα του χώρου μέσα στον οποίο θα κινηθεί η έρευνα και τα αποτελέσματά της. Επιπλέον, ουσιαστικό ρόλο θα μπορούσε να παίξει ένας συνεργαζόμενος με τον καθηγητή ερευνητής, υπό το ρόλο του ScrumMaster ή απλώς ενός εμπλεκόμενου φορέα που θα μπορούσε να παρέχει επικουρική καθοδήγηση.

Η αυτο-οργάνωση και αυτο-διαχείριση της ομάδας ανάπτυξης ταιριάζει γενικά με τις απαιτήσεις μιας μεταπτυχιακής διατριβής, όπου ουσιαστικά ο φοιτητής είναι αυτός που φέρει την ευθύνη για τη διεκπεραίωσή της.

Επίσης, με θετικό τρόπο φαίνεται να εντάσσονται οι τακτικές συναντήσεις, δια ζώσης ή ηλεκτρονικές, με σκοπό το σχεδιασμό και την παρουσίαση της επιτελούμενης δουλειάς. Προσοχή χρειάζεται εδώ η πλευρά του φοιτητή, ο οποίος ενδέχεται να μην μπορεί να ανταποκριθεί ικανοποιητικά σε έναν ευέλικτο μεν, αλλά συνεπή προγραμματισμό επαγγελματικών προδιαγραφών.

Β.6 Σκέψεις επιτυχίας

Ένας από τους χρυσούς κανόνες για επιτυχημένα ευέλικτα έργα, σύμφωνα με τον Keith Richards[19], είναι ο εξής: «Ξεκίνα αργά, επιτάχυνε αργότερα». Διευκρινίζει πως με την προσεκτική εκκίνηση, πατάμε σε γερές βάσεις. Αποφεύγεται έτσι η περιβόητη «παράλυση» της ανάλυσης, στην οποία κάνει ιδιαίτερη μνεία ο Rosenberg [23].

Ας εξετάσουμε τους λόγους για τους οποίους απέτυχε η εφαρμογή της μεθόδου στην παρούσα μεταπτυχιακή διατριβή. Καταρχήν, υπήρξε καθυστέρηση στην εκπόνησή της κατά το κρίσιμο πρώτο τρίμηνο του ακαδημαϊκού έτους. Αυτό καθυστέρησε την ανάλυση και τη διαμόρφωση

[19] www.keithrichardsconsulting.co.uk

του τελικού σκοπού της διατριβής – του πού ακριβώς πηγαίνει. Ένας δεύτερος λόγος είναι το γεγονός πως δεν αναπτύχθηκε γρήγορα, από την αρχή, ένα πρωτότυπο της εφαρμογής. Αυτό θα επέτρεπε την έναρξη ενός συνεπέστερου σταδιακού προγραμματισμού. Πολλές φορές όταν υπάρχει κάτι απτό, κι ας έχει περιορισμένες λειτουργίες, μορφοποιείται καλύτερα το τέλος – ο στόχος στο νου μας.

Ένα επιτυχημένο παράδειγμα προσαρμογής της SCRUM σε μεταπτυχιακή διατριβή είναι της Τσεκρέκου [38]. Η ερευνήτρια εφάρμοσε την επαναληπτική μεθοδολογία κατά την ανάπτυξη του γεωγραφικού συστήματος πληροφοριών «AthensShops». Η εμπέδωση της μεθοδολογίας έγινε με τη συμβολή του γράφοντος υπό το ρόλο του ScrumMaster. Το εγχείρημα ήταν ιδιαίτερα ενδιαφέρον και χρήζει περαιτέρω αξιοποίησης.

■