Διεθνές Πανεπιστήμιο της Ελλάδος

Τμήμα Μηχανικών Πληροφορικής, Υπολογιστών και Τηλεπικοινωνιών

Τομέας Μηχανικών Λογισμικού

Εξόρυξη γνώσης από αποθετήρια κώδικα

Πτυχιακή εργασία

της

Μήτσου Παυλίνας

ΑΕΜ: 4139

Επιβλέπων: Πεταλίδης Νικόλαος

19 Νοεμβρίου 2019

Περίληψη

Ψηφιακή πληροφορία δημιουργείται καθημερινά από τους προγραμματιστές κατά τη διάρκεια της κατασκευής του λογισμικού, που κατά βάση δεν είναι κώδικας. Από τις συνολικές γραμμές κώδικα που γράφει ένας προγραμματιστής ενσωματώνονται μόνο 10 από αυτές στο έργο. [1] Ο προγραμματιστής δαπανά το χρόνο του επίσης στην επικοινωνία με τους συναδέλφους του μέσω ηλεκτρονικών μηνυμάτων, αναζητώντας λύσεις για πιθανά σφάλματα στον κώδικα σε φόρουμ ή σε ιστοσελίδες, βλέποντας οπτικοακουστικό υλικό στο διαδίκτυο κ.α. Ο όγκος αυτός της πληροφορίας που παράγει ο προγραμματιστής αποδεικνύεται ότι επικεντρώνεται στα περιφερειακά εργαλεία που χρησιμοποιεί παρά στο πραγματικό κώδικα του προγράμματος.

Ως εκ τούτου, γεννιέται η επιτακτική ανάγκη διαχείρισης αυτού του όγκου πληροφορίας, με τη μετατροπή των μέσων που αλληλεπιδρά από στατικά σε ενεργά από τα τα οποία υπάρχει η δυνατότητα καθοδήγησης στη λήψη αποφάσεων στα σύγχρονα έργα λογισμικού και στην βελτίωση εξέλιξης του προϊόντος. [1] Η Εξόρυξη γνώσης από αποθετήρια κώδικα (Mining on Software Repositories) είναι το πεδίο της τεχνολογίας του λογισμικού που επικεντρώνεται στην ανάκτηση δεδομένων από τα εργαλεία λογισμικού και κώδικα με σκοπό την εξόρυξη χρήσιμης πληροφορίας.

Abstract

Huge digital records of software-engineering work are left by software developers during the development process that is mostly not code. Each day, a developer writes 10 lines of code that they will totally be used in the main code base. [1] A software developer spent her time in communication by emails with colleagues, searching for fixes at bugs in forums or websites, writing answer to others questions, watching tutorials etc.

So that means that there is a lot of information that they produce and we can use it to extract valuable knowledge from it. We can convert this information from static to active that will help as to take decisions and improve the software procedure and quality. Mining software repositories (MSR) is a software engineering field where software practitioners and researchers use data mining techniques to analyze the data in software repositories to extract useful and actionable information produced by developers during the development process.

Ευχαριστίες

Η παρούσα πτυχιακή εργασία είναι το αποτέλεσμα μιας σειράς αλληλεπιδράσεων με διάφορα άτομα, καθένα από τα οποία έπαιξε σημαντικό ρόλο στην εξέλιξή της. Θα ήθελα στο σημείο αυτό να εκφράσω τις θερμές μου ευχαριστίες σε όλους όσους με στήριξαν στην προσπάθεια αυτή.

Πρώτο απ' όλους, στον επιβλέποντα καθηγητή της πτυχιακής μου εργασίας, κύριο Νικόλαο Πεταλίδη για την αμέριστη υποστήριξη, τις ουσιώδεις συμβουλές καθώς και την ενθάρρυνση που μου παρείχε σε όλη την διάρκεια της εργασίας αυτής. Επίσης για όλη αυτή την σημαντική γνώση που μου παρείχε μέσα από τα μαθήματά του που συντέλεσαν στο να είμαι στη θέση να μπορώ να διεκδικήσω σημαντικές θέσεις στην αγορά εργασίας.

Αισθάνομαι επίσης την ανάγκη να ευχαριστήσω τους γονείς μου γιατί με βοήθησαν να σπουδάσω και να βρίσκομαι στην ευχάριστη θέση αυτή τη στιγμή να γράφω την πτυχιακή μου εργασία.

Τέλος, θα επιθυμούσα να ευχαριστήσω όλους εκείνους που στάθηκαν στο πλευρό μου και με συμβούλευαν σε όλοι την διάρκεια των τεσσάρων αυτών ετών: φίλους και ορισμένους πάρα πολύ κοντινούς μου ανθρώπους, που χωρίς τους οποίους τίποτα από όσα έχω καταφέρει μέχρι σήμερα δεν θα ήταν πραγματικότητα.

Παυλίνα Μήτσου

Σέρρες, 2019

Εξόρυξη γνώσης από αποθετήρια κώδικα

Παυλίνα Μήτσου

pavlinamitsou@gmail.com

19 Νοεμβρίου 2019

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Ογκώδης ψηφιακή πληροφορία δημιουργείται καθημερινά από τους προγραμματιστές κατά τη διάρκεια της κατασκευής του λογισμικού, που κατά βάση δεν είναι κώδικας. Από τις συνολικές γραμμές κώδικα που γράφει ένας προγραμματιστής ενσωματώνονται μόνο 10 από αυτές στο έργο. [1] Ο προγραμματιστής δαπανά το χρόνο του επίσης στην επικοινωνία με τους συναδέλφους του μέσω ηλεκτρονικών μηνυμάτων, αναζητώντας λύσεις για πιθανά σφάλματα στον κώδικα σε φόρουμ ή σε ιστοσελίδες, βλέποντας οπτικοακουστικό υλικό στο διαδίκτυο κ.α. Ο όγκος αυτός της πληροφορίας που παράγει ο προγραμματιστής αποδεικνύεται ότι επικεντρώνεται στα περιφερειακά εργαλεία που χρησιμοποιεί παρά στο πραγματικό κώδικα του προγράμματος.

Ως εκ τούτου, γεννιέται η επιτακτική ανάγκη διαχείρισης αυτού του όγκου πληροφορίας, με τη μετατροπή των μέσων που αλληλεπιδρά από στατικά σε ενεργά από τα τα οποία υπάρχει η δυνατότητα καθοδήγησης στη λήψη αποφάσεων στα σύγχρονα έργα λογισμικού και στην βελτίωση εξέλιξης του προϊόντος. [1] Η Εξόρυξη γνώσης από αποθετήρια κώδικα (Mining on Software Repositories) είναι το πεδίο της τεχνολογίας του λογισμικού που επικεντρώνεται στην ανάκτηση δεδομένων από τα εργαλεία λογισμικού και κώδικα με σκοπό την εξόρυξη χρήσιμης πληροφορίας. Οι τεχνικές εξόρυξης γνώσης από αποθετήρια κώδικα έχουν χρησιμοποιηθεί για πολλαπλά εργαλεία έως σήμερα για την βελτίωση της προγραμματιστικής διαδικασίας μέσω χρήσιμων πληροφοριών που ανακαλύπτονται. Χρησιμοποιώντας ιστορικές πληροφορίες και τα μη εμφανή μοτίβα που ανακαλύπτονται, οι ενδιαφερόμενοι βασίζονται σε αυτές τις πληροφορίες και όχι στο προαίσθημα και την εμπειρία τους. [2]

1.1 Κίνητρα έρευνας

Ένα σημαντικό πεδίο που εφαρμόζονται αυτές οι τεχνικές ειναι στην τμηματοποίηση του κώδικα. Η ανάγκη για ομαδοποίηση κώδικα(modularity) εμφανίστηκε από τα πρώτα χρόνια διάδοσης του προγραμματισμού και της διόγκωσης του μεγέθους των εφαρμογών. Αλλαγές σε μικρότερα συστήματα εφαρμογών είναι πιο οικονομικές και ανώδυνες σε σχέση με τις αλλαγές σε μεγαλύτερα υποσυστήματα [3]. Η κεντρική ιδέα της ομαδοποίησης του κώδικα (modularity) είναι η απόκρυψη σημαντικών σχεδιαστικών αποφάσεων που μπορούν να αλλάξουν [2]. Ως μέρος της διαδικασίας, η συσταδοποίηση του κώδικα χωρίζει τον κώδικα σε υποσυστήματα τέτοια ώστε τα υποσυστήματα να είναι όσο πιο ανεξάρτητα γίνεται σεβόμενα την κατανόηση, τις αλλαγές, την επαναχρησιμοποίηση και άλλα κριτήρια. [3] Η καλή ομαδοποίηση του κώδικα σε υποσυστήματα συμβάλει στην συντήρηση, κατανόηση και αλλαγή του κώδικα ευκολότερα. Κατά καιρούς αρκετοί ερευνητές έχουν χρησιμοποιήσει τεχνικές και αλγορίθμους για την καλύτερη συμβούλευση της ομαδοποίησης του κώδικα. [4] [3]. Στη παρούσα πτυχιακή εργασία θα παρουσιαστεί η εφαρμογή Pink pony η οποία προτείνει υποσυστήματα και ομαδοποιήσεις του κώδικα με βάση τις συχνές αλλαγές που συμβαίνουν στο Git. Επίσης θα αναλυθεί ο τρόπος προσέγγισης του προβλήματος και οι τεχνικές που χρησιμοποιήθηκαν.

1.2 Ερευνητικές ερωτήσεις

Στόχος της παρούσας πτυχιακής εργασίας είναι να απαντηθούν ορισμένες ερευνητικές ερωτήσεις που θα μπορέσουν να βοηθήσουν την επιστημονική κοινότητα να τις αναπτύξει και να τις επεκτείνει. Το βασικό ερώτημα που θα απαντηθεί είναι : Ποιος αλγόριθμος είναι ο καλύτερος για την συσταδοποίηση αρχείων, πακέτων ή modules από το ιστορικό του Git ; Ένα επίσης πολύ σημαντικό ερώτημα που θα απαντηθεί είναι: Πόσο ικανοποιητική κρίνεται η συσταδοποίηση των αλγορίθμων και ποιο είναι το κέρδος της ; Ποιες κρυφές σχέσεις υπάρχουν μέσα στο πρόγραμμα ;

1.3 Σχετικό έργο

Οι συχνές αλλαγές (co-changes) εξορύσσονται από το ιστορικό εκδόσεων για να υποστηρίξουν διάφορες πτυχές στην εξέλιξη και τη συντήρηση του λογισμικού όπως, στην ανίχνευση κακής οσμής κώδικα [5] , τη βελτίωση τεχνικών ανίχνευσης ελαττωμάτων [6], τη κατανόηση του προγράμματος [6], τη λογική ανίχνευση εξάρτησης και τη παρότρυνση για αλλαγές [7] O Ball [8] εισήγαγε τον co-change γράφο και έπειτα [3] πρότειναν μια τεχνική απεικόνισης που αποκαλύπτει ομάδες co-change αντικειμένων.

1.4 Διάρθρωση

- Στο Κεφάλαιο 2 αναλύεται το θεωρητικό υπόβαθρο και οι τεχνικές που θεωρούνται απαραίτητες για τον αναγνώστη ώστε να κατανοήσει βασικές έννοιες που χρησιμοποιούνται στην εφαρμογή.
- Στο Κεφάλαιο 3 παρουσιάζεται η εφαρμογή και οι δυνατότητες που έχει. Επίσης περιέχονται οδηγίες για τη χρήση της.
- Στο Κεφάλαιο 4 αιτιολογούνται οι σχεδιαστικές αποφάσεις της εφαρμογής και η δομή της.
- Στο Κεφάλαιο 5 σε πρώτο στάδιο παρουσιάζονται τα πειράματα που εκτελέστηκαν πάνω στην εφαρμογή και στη συνέχεια η ανάλυση των αποτελεσμάτων.
- Στο Κεφάλαιο 6 γίνεται συζήτηση για τα γενικότερα συμπεράσματα από την περάτωση της παρούσας διπλωματικής εργασίας.
- Στο Κεφάλαιο 7 δίνονται προτάσεις για βελτιώσεις και μελλοντικές επεκτάσεις του συστήματος.

Κεφάλαιο 2

Θεωρητικό υπόβαθρο

2.1 Εξόρυξη γνώσης από αποθετήρια κώδικα

Η Εξόρυξη γνώσης από αποθετήρια κώδικα (Mining on Software Repositories) είναι το πεδίο της τεχνολογίας του λογισμικού που επικεντρώνεται στην ανάκτηση δεδομένων από τα εργαλεία λογισμικού και κώδικα με σκοπό την εξόρυξη χρήσιμης πληροφορίας. Οι τεχνικές εξόρυξης γνώσης από αποθετήρια κώδικα έχουν χρησιμοποιηθεί για πολλαπλά εργαλεία έως σήμερα για την βελτίωση της προγραμματιστικής διαδικασίας μέσω χρήσιμων πληροφοριών που ανακαλύπτονται. Χρησιμοποιώντας ιστορικές πληροφορίες και τα μη εμφανή μοτίβα που ανακαλύπτονται, βοηθά τους ενδιαφερόμενους του να βασίζονται σε αυτές τις πληροφορίες και όχι στο προαίσθημα τους και την εμπειρία τους. [2]

2.2 Πηγές πληροφοριών

Μπορούν να θεωρηθούν πηγές εξόρυξης γνώσης όλα τα εργαλεία που χρησιμοποιεί ένα μέλος μιας ομάδας λογισμικού. Τα πιο σημαντικά από αυτά που μπορούν να παρέχουν χρήσιμες πληροφορίες είναι:

1. **Πηγαίος κώδικας**: Ο βασικός κώδικας του προγράμματος περιέχει πολύ σημαντικές πληροφορίες. Υπάρχει η δυνατότητα μέσα από κομμάτια κώδικα να εντοπιστούν κοινά μοτίβα και συσχετίσεις που με τις κατάλληλες τροποποιήσεις

Σχήμα 2.1: Εξόρυξη από διάφορες πηγές αποθετηρίων κώδικα

να βελτιωθεί κατά μεγάλο βαθμό η ποιότητα του προγράμματος.

2. **Συστήματα ελέγχου έκδοσης (Git,SVN)**: Χρησιμοποιείται από τους προγραμματιστές για την διαχείριση εκδόσεων από αρχεία και βοηθάει στην παράλληλη συγγραφή κώδικα. Ουσιαστικά περιέχουν το ιστορικό του λογισμικού προγράμματος και αποτελούν κύρια πηγή γνώσης αφού μπορούν να απαντήσουν σημαντικές ερωτήσεις για την ανάπτυξη του προγράμματος. Ποιες είναι οι αλλαγές που προστέθηκαν στο πρόγραμμα· Ποιοι δημιούργησαν αυτές τις αλλαγές· Πότε έγιναν· Ποια αρχεία αλλάζουν συχνά μαζί· κ.α.

3. **Συστήματα διαχείρισης αλλαγών (Bugzilla, Redmine)**: Λογισμικό που χρησιμοποιείται για την αποθήκευση, διαχείριση και αναφορά σφαλμάτων σε ένα πρόγραμμα. Το υλικό που προσφέρεται μέσα από αυτά τα συστήματα με την εξόρυξη γνώσης από αποθετήρια κώδικα μπορεί να χρησιμοποιηθεί για τη δημιουργία αρχείων αναφορών των σφαλμάτων. Επίσης χρησιμοποιούνται ως αναφορές για συστήματα προτεινόμενων συμβουλών.

4. **Συστήματα διαχείρισης εργασιών (Jira, Asana)**: Είναι εφαρμογές λογισμικού που παρέχουν ένα σύστημα εργασιών για την καταγραφή και παρακολούθηση της εξέλιξης κάθε ζητήματος που προσδιορίζεται από έναν χρήστη μέχρι να επιλυθεί

το ζήτημα. Ένα ζήτημα μπορεί να είναι οτιδήποτε από μια αναφορά σφάλματος μέχρι την έρευνα ανάπτυξης.

5. **Ηλεκτρονικές επικοινωνίες (Email, Skype)**: Ηλεκτρονικά μηνύματα σχετικά με την εξέλιξη του έργου και τα σφάλματα που έχουν εμφανιστεί
6. **Deployment logs** : Περιέχουν πληροφορίες σχετικές με την εκτέλεση και τη χρήση της εφαρμογής σε διάφορες deployment ιστοσελίδες.
7. **Πήγες πληροφοριών (Confluence, SharePoint, WWW, Q&A sites)**: Ηλεκτρονικές σελίδες για την εύρεση χρήσιμων πληροφοριών σχετικά με το έργο εσωτερικά στην ομάδα αλλά και στο διαδίκτυο
8. **Οπτικοακουστικό υλικό (Youtube, tutorial websites)**: Οι προγραμματιστές δημιουργούν οπτικοακουστικό υλικό σχετικό με τη δημιουργία κώδικα , προσωπικές εμπειρίες, υλοποίηση νέων μεθόδων και παρουσίαση της εκτέλεσης του προγράμματος.

2.3 Πεδία της εξόρυξης γνώσης από αποθετήρια κώδικα

Τα επιτεύγματα της Εξόρυξης γνώσης από αποθετήρια κώδικα έως σήμερα είναι εντυπωσιακά και βοηθάνε καθημερινά όλους τους ενδιαφερόμενους του έργου στη λήψη αποφάσεων και στη βελτίωση της προγραμματιστικής διαδικασίας.

1. **Ανίχνευση διπλότυπων** : Η επαναχρησιμοποίηση του κώδικα είναι πολύ σημαντική στη σύγχρονη ανάπτυξη λογισμικού. Υπάρχουν ήδη τεχνικές στην εξόρυξη γνώσης από αποθετήρια κώδικα που εντοπίζουν διπλότυπο κώδικα ή επαναλαμβανόμενα μοτίβα κώδικα και προτείνουν αλλαγές στον κώδικα στους προγραμματιστές. Με αυτό τον τρόπο επαναχρησιμοποιείται όσο γίνεται μεγαλύτερο μέρος του κώδικα βοηθώντας τους προγραμματιστές να διατηρούν τον κωδικά τους όσο πιο τακτοποιημένο γίνεται.
2. **Πρόβλεψη και εντοπισμός σφαλμάτων**: Η πρόβλεψη σφαλμάτων αποτελεί ακόμα ένα από τα πιο ενδιαφέροντα θέματα στην ανάπτυξη λογισμικού. Τα ιστο-

ρικά αποθετήρια μπορούν να χρησιμοποιηθούν για την αποφυγή σφαλμάτων του παρελθόντος. Οι προγραμματιστές με την βοήθεια των εργαλείων που βασίζονται στην εξόρυξη γνώσης από αποθετήρια κώδικα , εστιάζουν την προσοχή τους στα πιο ριψοκίνδυνα κομμάτια του κώδικα όπως και οι οι προγραμματιστές που τον ελέγχουν δίνουν προτεραιότητα στα κομμάτια αυτά. Επίσης έχουν δημιουργηθεί εργαλεία που μέσα από τα αναφερόμενα σφάλματα χρηστών , εμφανίζουν προειδοποιητικά μηνύματα στο χρήστη για να αποφευχθεί η ξαφνική διακοπή του προγράμματος. Με αυτό τον τρόπο βελτιώνεται η συνολική εμπειρία του χρήστη.

3. **Δημιουργία αρχείων αναφορών**: Πολλές φορές ογκώδεις βιβλιοθήκες δεν εμπεριέχουν καλογραμμένα εγχειρίδια ή δεν έχουν αντιπροσωπευτικά ονόματα στις μεθόδους τους με αποτελέσματα να μην είναι εύκολη η επαναχρησιμοποίηση τους. Μέσω των παραπάνω πηγών μπορούν να δημιουργηθούν πολλά διαφορετικά είδη αναφορών. Έχουν δημιουργηθεί εργαλεία που μπορούν να παράγουν αναγνώσιμο κείμενο από αλλαγές στη συμπεριφορά στο σώμα της μεθόδου μέσω τροποποιήσεων. Επίσης μπορούν να δημιουργηθούν αναφορές σφαλμάτων, οι οποίες θα έχουν μία σύνοψη των σφαλμάτων και πιθανούς λόγους πρόκλησης.

4. **Συμβουλευτικά συστήματα**: Μπορούν να δημιουργηθούν διαφόρων ειδών συμβουλευτικά συστήματα με τις τεχνικές Εξόρυξης γνώσης από αποθετήρια κώδικα. Μία αρκετά χρήσιμη συμβουλή που θα μπορούσε να μας δώσει ένα τέτοιο σύστημα είναι ο διαχωρισμός του κώδικα σε κομμάτια. Επίσης προτροπές για καλύτερη ονοματοθεσία στα διάφορα μέλη τού κώδικα με προτάσεις ονομάτων σχετικότερων με την ιδιότητα του κάθε μέλους.

5. **Συστήματα πρόβλεψης ποιότητας λογισμικού**: Στα σύγχρονα λογισμικά προϊόντα, η ποιότητα έχει πρωταρχική σημασία. Η εξόρυξη γνώσης από αποθετήρια κώδικα έχει τη δυνατότητα να αναγνωρίσει κατά πόσο η ποιότητα ενός κώδικα είναι η επιθυμητή, εντοπίζοντας τα σφάλματα στον πηγαίο κώδικα που μπορεί να έχουν επίπτωση στην εμπειρία του χρήστη κατά τη χρήση της εφαρμογής. Από τα ιστορικά δεδομένα μπορούν επίσης να προβλεφθούν πιθανά σφάλματα που θα μπορούσαν να παρουσιαστούν στο μέλλον. Μεγάλο ενδιαφέρον παρουσιάζει η πρόβλεψη της ποιότητας διασταυρωμένων προϊόντων (cross product) στην οποία το ένα πρόγραμμα χρησιμοποιείται για την πρόβλεψη σφαλμάτων σε ένα

παρόμοιο πρόγραμμα.

2.4 Modularity

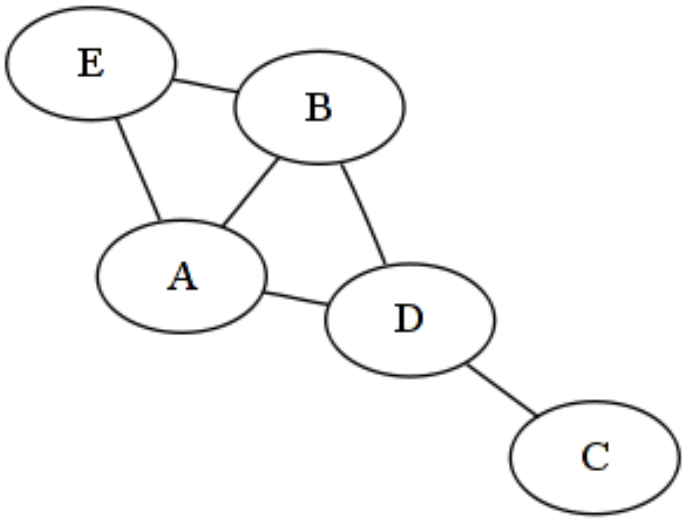
Η κεντρική ιδέα πίσω από τον όρο Modularity είναι η απόκρυψη σημαντικών σχεδιαστικών αποφάσεων ή των αποφάσεων που πιθανόν να αλλάξουν στο μέλλον. [9] Με αυτό τον τρόπο, το modularity συνεισφέρει στη βελτίωση την αποδοτικότητάς των προγραμματιστών στη φάση της ανάπτυξης του λογισμικού αλλά και στη συνέχεια της συντήρησης. Συνεπώς, ένα καλά τμηματοποιημένο σύστημα είναι ευκολότερο να το συντηρήσεις και να το επεκτείνεις αφού τα modules μπορούν να κατανοηθούν και να αλλάξουν ανεξάρτητα το ένα από το άλλο. [4] Για την καλύτερη όμως τμηματοποίηση του κώδικα σημαντικό ρόλο λαμβάνουν η σύζευξη (coupling) και η συνεκτικότητα (cohesion). Αυτές υπολογίζονται χρησιμοποιώντας τις δομικές εξαρτήσεις μεταξύ των modules του συστήματος (σύζευξη) και μεταξύ των εσωτερικών στοιχείων του κάθε module (συνεκτικότητα). [4] Στόχος είναι η χαμηλή σύζευξη μεταξύ των modules και η υψηλή συνεκτικότητα εσωτερικά.

2.5 Γράφοι

Στα διακριτά μαθηματικά, ένας γράφος ή ένα γράφημα είναι μια αφηρημένη αναπαράσταση ενός συνόλου στοιχείων, όπου μερικά ζευγάρια στοιχείων συνδέονται μεταξύ τους με δεσμούς. Τα διασυνδεδεμένα στοιχεία αναπαριστώνται με μαθηματικές έννοιες οι οποίες ονομάζονται κορυφές ενώ οι δεσμοί που συνδέουν τα ζευγάρια των κορυφών ονομάζονται ακμές. [10]

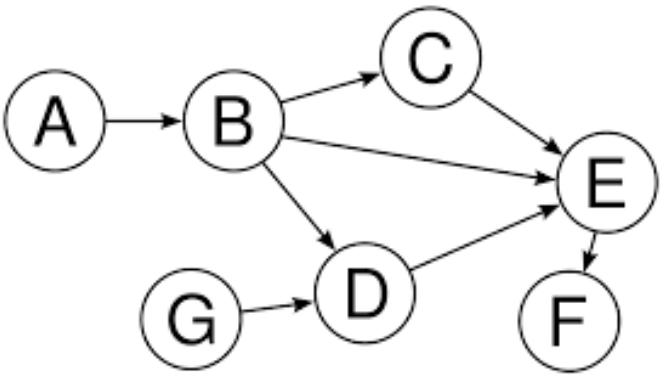
2.5.1 Τύποι γράφων

1. **Μη κατευθυνόμενοι γράφοι** : Ένα μη-κατευθυνόμενος γράφος είναι εκείνος στον οποίο οι ακμές δεν έχουν προσανατολισμό. Η ακμή (α, β) είναι ταυτόσημη με την άκρη (β, α), δηλαδή, δεν υπάρχουν διατεταγμένα ζεύγη, αλλά σύνολα u, v (ή 2-multisets) των κορυφών. [10]



Σχήμα 2.2: Μη κατευθυνόμενος γράφος χωρίς βάρη.

2. **Κατευθυνόμενοι γράφοι**: Ένα κατευθυνόμενο γράφημα ή διγράφημα είναι ένα διατεταγμένο ζεύγος D = (V, A) όπου V, είναι ένα σύνολο του οποίου τα στοιχεία λέγονται κορυφές ή κόμβοι και A, είναι μια σειρά από διατεταγμένα ζεύγη κορυφών, τα οποία ονομάζονται τόξα, διατεταγμένες ακμές ή βέλη. Ένα τόξο a = (x, y) θεωρείται ότι κατευθύνεται από το x στο y · y ονομάζεται η αρχή και x το τέλος του τόξου y λέγεται ότι είναι άμεσος διάδοχος του x, και το x λέγεται ότι είναι ο άμεσος προκάτοχός του y. Αν ένα μονοπάτι οδηγεί από το x στο y, τότε το y λέγεται ότι είναι διάδοχος του x και προσβάσιμο από το x, και το x λέγεται ότι είναι ο προκάτοχος του y. Το τόξο (y, x) ονομάζεται το τόξο (x, y) ανεστραμμένο.

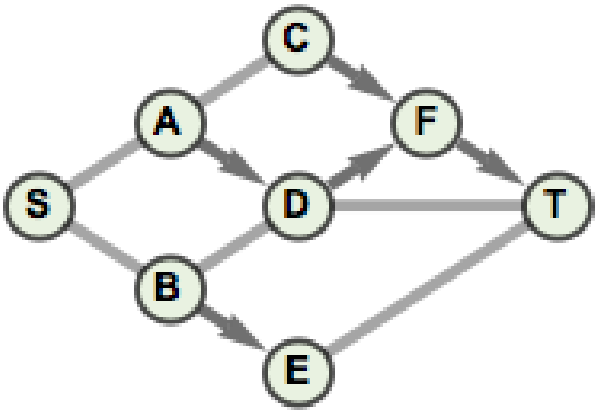


Σχήμα 2.3: Κατευθυνόμενος γράφος χωρίς βάρη.

Ένας κατευθυνόμενος γράφος D λέγεται συμμετρικός αν για κάθε τόξο στο D, το αντίστοιχο αντεστραμμένο τόξο ανήκει και αυτό στο D. Ένα συμμετρικό χωρίς

επαναλήψεις κατευθυνόμενο γράφημα D = (V, A) είναι ισοδύναμο με ένα απλό μη-κατευθυνόμενο γράφημα G = (V, E)| , όπου τα ζευγάρια του αντιστρόφου τόξου στο A αντιστοιχούν 1-προς-1 με τις ακμές στο E· έτσι ο αριθμός |E| = |A|/2, ή το μισό του αριθμού των τόξων στο D. Μια παραλλαγή αυτού του ορισμού είναι το προσανατολισμένο γράφημα, στο οποίο δεν μπορούν να υπάρχουν παραπάνω από ένα από τα (x, y) και (y, x) τα οποία μπορούν να είναι τόξα. [10]

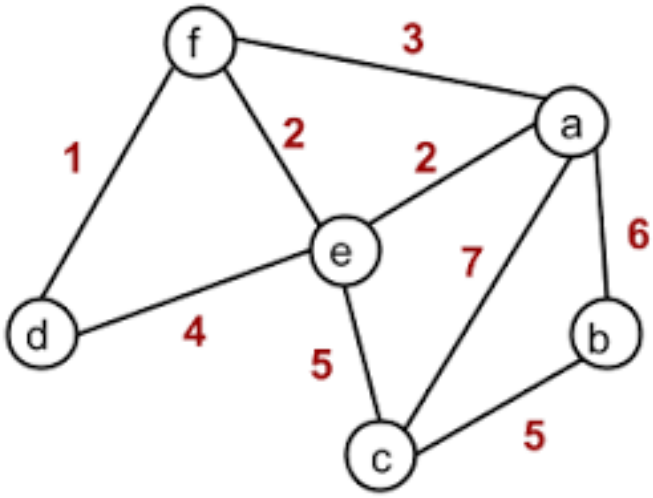
3. **Μεικτοί γράφοι**: Ένα μικτό γράφημα G είναι ένα γράφημα στο οποίο μερικές ακμές μπορεί να είναι κατευθυνόμενες και κάποιες μπορεί να είναι μη κατευθυνόμενες. Το γράφημα είναι γραμμένο ως διατεταγμένο τριπλό G = (V, E, A) με V, E και A όπως ορίζεται ανωτέρω. Τα κατευθυνόμενα και μη κατευθυνόμενα γραφήματα είναι ειδικές περιπτώσεις. [10]



Σχήμα 2.4: Μεικτός γράφος χωρίς βάρη.

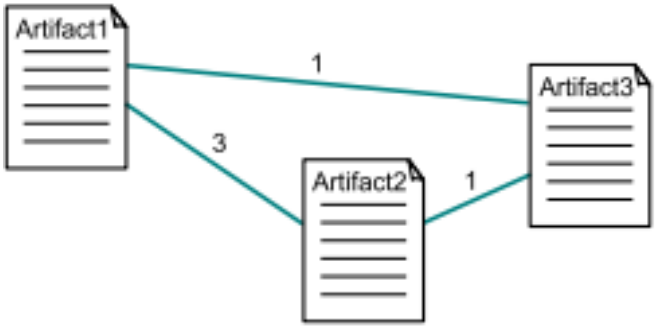
4. **Γράφοι με βάρος**: Ένας γράφος με βάρος ή αλλιώς ένα δίκτυο είναι ένας γράφος ο οποίος σε κάθε ακμή ορίζεται ένα βάρος. Τέτοια βάρη μπορεί να αναπαριστούν ορισμένα κόστη, μήκη ή δυνατότητες ανάλογα το πρόβλημα. [10] Μπορεί παράλληλα να υφίσταται ένας γράφος που είναι συνδυασμός των παραπάνω, π.χ. Μη κατευθυνόμενος γράφος με βάρη.

5. **Co-change γράφος** Όπως προτείνεται από τους Beyer και Noack [3] ο co-change γράφος είναι μία αναπαράσταση ενός version control συστήματος((VCS)). Ας υποθέσουμε ότι ένα σύνολο commmits σε ένα VCS που ορίζεται ως T = {T1, T2,. . . , Tn}, όπου κάθε συναλλαγή Ti αλλάζει ένα σύνολο κλάσεων. Ουσιαστικά ενάς



Σχήμα 2.5: Μη κατευθυνόμενος γράφος με βάρη.

co-change γράφος είναι είναι ένα μη κατευθυνόμενο γράφημα G = {V, E}, όπου V είναι το σύνολο των κλάσεων και το E είναι ένα σύνολο ακμών. Προσδιορίζεται ένα άκρο (Ci, Cj) μεταξύ των κλάσεων (κορυφές) Ci και Cj κάθε φορά που υπάρχει μια συναλλαγή Tk, έτσι ώστε Ci, Cj ∈ Tk, για i ≠ j . Τέλος, κάθε ακμή έχει ένα βάρος που αντιπροσωπεύει τον αριθμό των συναλλαγών μεταξύ των συνδεδεμένων κλάσεων.



Σχήμα 2.6: Co-change γράφος

2.6 Συσταδοποίηση γράφων

Η συσταδοποίηση γράφων είναι μια διαδικασία εύρεσης ομάδων ισχυρά συσχετισμένων κορυφών μεταξύ τους από έναν γράφο. Το αποτέλεσμα είναι υπογράφοι με ισχυρούς δεσμούς ανάμεσα τους.

2.6.1 Είδη συσταδοποίσης γράφων

Υπάρχουν δύο κατηγορίες που διαχωρίζουν τις συσταδοποιήσεις των γράφων σχετικά με το είδος των συστάδων που αναπαράγουν.

1. **Σκληροί(hard)**: Η σκληρή συσταδοποίση αφορά την ομαδοποίηση των στοιχείων δεδομένων έτσι ώστε κάθε στοιχείο να αντιστοιχεί μόνο σε μία συστάδα. Για παράδειγμα, θέλουμε τον αλγόριθμο να διαβάσει όλα τα tweets και να καθορίσει εάν ένα tweet είναι θετικό ή αρνητικό tweet. Ο K-Means είναι ένας διάσημος αλγόριθμος σκληρού τύπου ομαδοποίησης, οπότε τα στοιχεία δεδομένων συγκεντρώνονται σε συστάδες K έτσι ώστε κάθε στοιχείο να ανήκει μόνο σε μία συστάδα.

2. **Μαλακοί(softή fuzzy)** Η μαλακή συσταδοποίση αφορά την ομαδοποίηση των στοιχείων δεδομένων, έτσι ώστε ένα στοιχείο να μπορεί να υπάρχει σε πολλαπλές συστάδες. Στην ουσία κάθε στοιχείο αναθέτεται σε μία συστάδα μαζί με ένα συντελεστή πιθανότητας ή ένα βάρος.

2.6.2 Αλγόριθμοι συσταδοποίησης γράφων

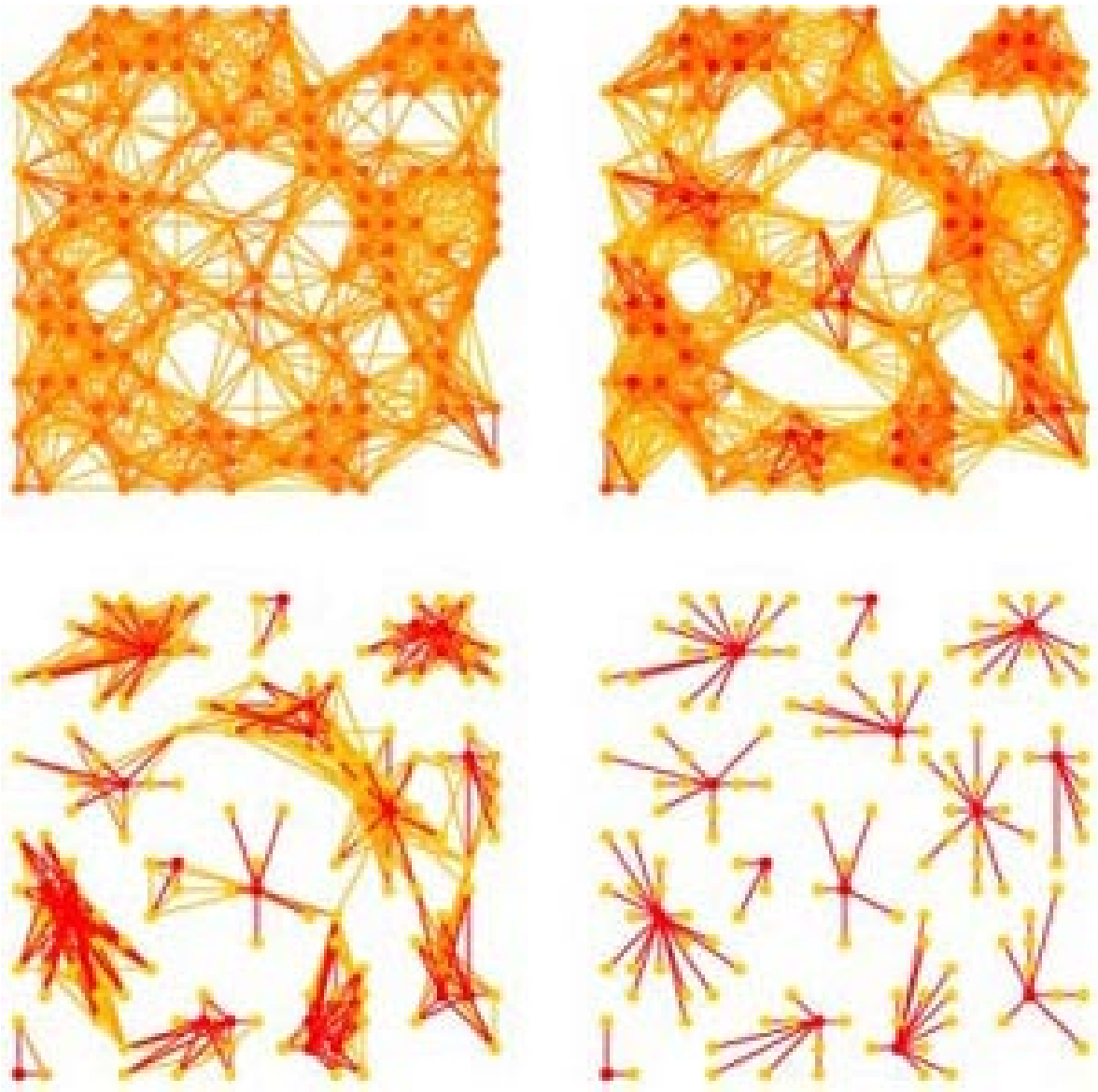
1. Η **Μαρκοβιανή συσταδοποίηση** (MCL) [11] είναι ένας σκληρός αλγόριθμος συσταδοποίησης και αποτελεί έναν από τους πιο αξιόπιστους αλγορίθμους για την ταυτοποίηση συστάδων σε βιολογικά δίκτυα. Βασίζεται στην εξομοίωση της ροής της πληροφορίας μέσα σε ένα γράφο για την ταυτοποίηση συστάδων. Για την αναγνώριση των συστάδων λαμβάνονται υπό όψη οι περιοχές με υψηλή ροή. Με απλά λόγια τα τυχαία μονοπάτια πολλαπλών βημάτων σε ένα γράφο που θα ξεκινήσουν και θα καταλήξουν στην ίδια πυκνή περιοχή (συστάδα) έχουν μεγαλύτερη πιθανότητα από τα υπόλοιπα μονοπάτια. Έχει αποδειχθεί ότι ο αλγόριθμος αυτός είναι ιδανικός για αραιούς γράφους, δηλαδή γράφους για τους οποίους ο μέσος βαθμός των κόμβων είναι κατά μια τάξη μικρότερος από τον αριθμό των κόμβων.

   Αρχικά, σε ένα γράφο G ο αλγόριθμος MCL παράγει ένα πίνακα Markov TG από τον πίνακα γειτνίασης M του γράφου, με τον ακόλουθο τρόπο: Κάθε στήλη του TG προκύπτει από κανονικοποίηση της αντίστοιχης στήλης του M. Ο πίνακας TG είναι ένας στοχαστικός πίνακας που περιέχει τις πιθανότητες μετάβασης

από ένα κόμβο σε ένα άλλο και αντιστοιχεί σε ένα συσχετισμένο γράφο Markov G'. Έπειτα το σύνολο των πιθανοτήτων μετάβασης υπολογίζεται εκ νέου σε επαναλαμβανόμενα βήματα εκτόνωσης (expansion) και εμφύσησης (inflation). Η εκτόνωση υλοποιείται με πολλαπλασιασμό του TG με τον εαυτό του και επιτρέπει σε κόμβους να εντοπίσουν γειτονικούς κόμβους. Η εμφύσηση υλοποιείται με πολλαπλασιασμό στοιχείο προς στοιχείο του πίνακα TG με τον εαυτό του, ακολουθούμενο από κανονικοποίηση ώστε ο παραγόμενος πίνακας να παραμένει στοχαστικός. Μέσω της εμφύσησης οι προτιμώμενοι γείτονες αναβαθμίζονται και οι μη προτιμώμενοι υποβιβάζονται. Ανάμεσα στα βήματα εκτόνωσης και εμφύσησης εφαρμόζεται και μια μέθοδος κλαδέματος του γράφου για να βελτιώσει την απόδοση του αλγορίθμου. Τελικά με τα επαναλαμβανόμενα βήματα εκτόνωσης και εμφύσησης αυξάνεται η αντίθεση μεταξύ περιοχών ισχυρής και αδύναμης ροής. Ο αλγόριθμος τερματίζεται με το διαμερισμό του γράφου σε συστάδες.

2. Ο **Chinese Whispers** [12] είναι σκληρός hard αλγόριθμος συσταδοποιήσης για γράφους με βάρη. Μπορεί να θεωρηθεί μία ειδική περίπτωση του MCL με ένα απλουστευμένο βήμα για την ενημέρωση της τάξης. Σε κάθε επανάληψη, οι ετικέτες όλων των κόμβων ενημερώνονται σύμφωνα με το την πλειοψηφία των ετικετών μεταξύ των γειτονικών κόμβων. Ο αλγόριθμος έχει μια υπερ-παράμετρο που ελέγχει τα βάρη γραφημάτων που μπορούν να οριστούν σε τρεις τιμές: (1) CW_{top} αθροίζει στις τάξεις της γειτονιάς · (2) CW_{lin} υποβαθμίζει την επίδραση ενός γειτονικού κόμβου με τον βαθμό του · (3) CW_{log} υποβαθμίζει την επίδραση ενός γειτονικού κόμβου με τον λογάριθμο από τον βαθμό του. [13] Η πολυπλοκότητα του αλγορίθμου είναι γραμμική (O(n)/) όπου αποτελεί και ισχυρό πλεονέκτημα για αυτόν. Για αυτό το λόγο θεωρείται ιδανικός για την ανάλυση κοινοτήτων σε έναν γράφο με πολλούς κόμβους. Από την άλλη πλευρά, επειδή ο αλγόριθμος δεν είναι ντετερμινιστικός στην περίπτωση του μικρού αριθμού κόμβου, οι συστάδες που προκύπτουν συχνά διαφέρουν σημαντικά μεταξύ τους που έχει ως συνέπεια να μην συνιστάται για μικρού μεγέθους γράφους. [14]

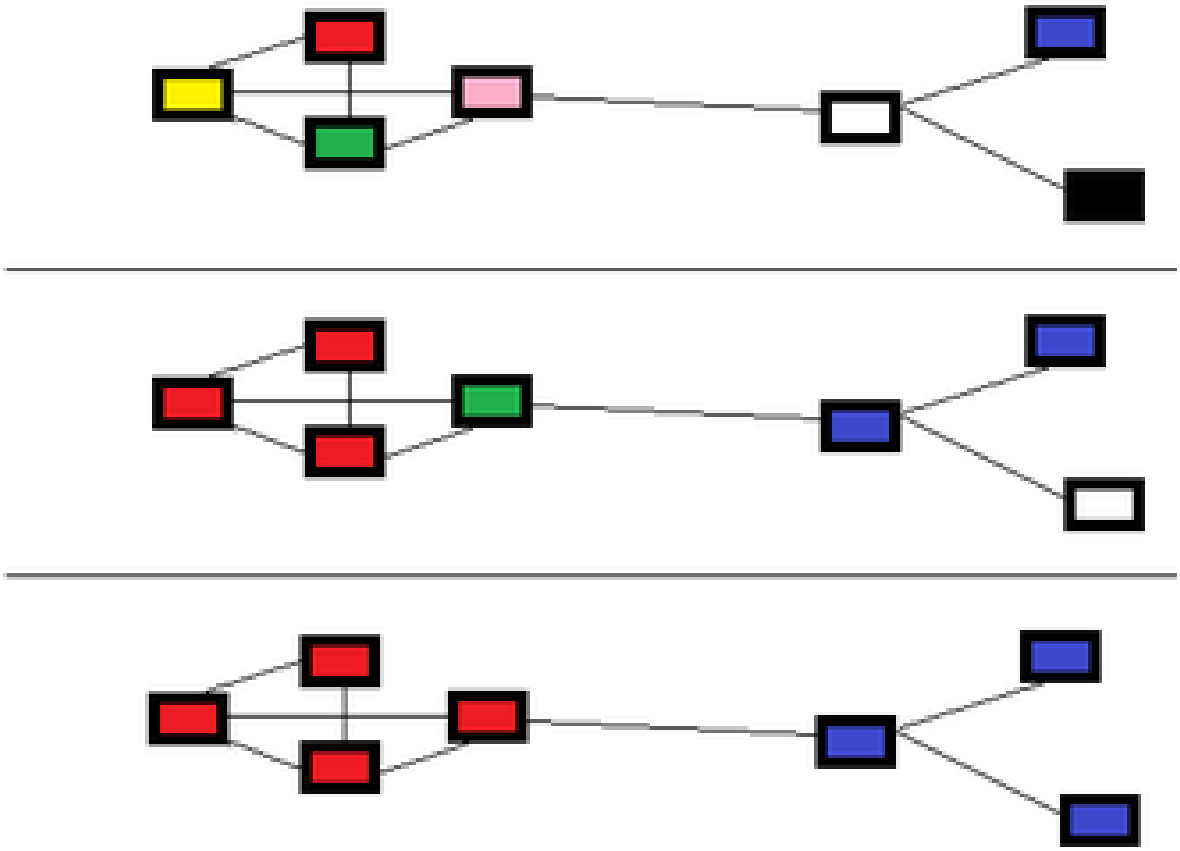
3. Ο **MaxMax** [15] είναι ένας μαλακός (soft ή fuzzy) αλγόριθμος που σχεδιάστηκε για την επαγωγική έννοια της λέξης (word sense induction) . Με λίγα λόγια, τα ζεύγη κόμβων ομαδοποιούνται εάν έχουν μια μέγιστη αμοιβαία συγγένεια. Ο αλγόριθμος αρχίζει με τη μετατροπή του αρχικού μη κατευθυνόμενου γράφου σε



Σχήμα 2.7: Στάδια της συσταδοποίησης MCL

κατευθυνόμενο διατηρώντας τους μέγιστους κόμβους συγγένειας του κάθε κόμβου. Στη συνέχεια, όλοι οι κόμβοι επισημαίνονται ως κόμβοι ¨ρίζες¨. Τέλος, για κάθε κόμβο ¨ρίζας¨, η ακόλουθη διαδικασία επαναλαμβάνεται: Όλα τα μεταβατικά ¨παιδιά¨ αυτής της ¨ρίζας¨ σχηματίζουν ένα συστάδα και οι ¨ρίζες¨ που υπάρχουν στη συστάδα παύουν να είναι. Ένας κόμβος ¨ρίζας¨ μαζί με όλα τα μεταβατικά ¨παιδιά¨ του σχηματίζουν μία συστάδα. [13] Η πολυπλοκότητα του αλγορίθμου είναι γραμμική (O(n))

4. Ο **Watset** [13] είναι ένας μετα-αλγόριθμος για μαλακή (fuzzy) συσταδοποίση γράφων. Για παράδειγμα σε ένα γράφο που συνδέονται πιθανά διφορούμενα

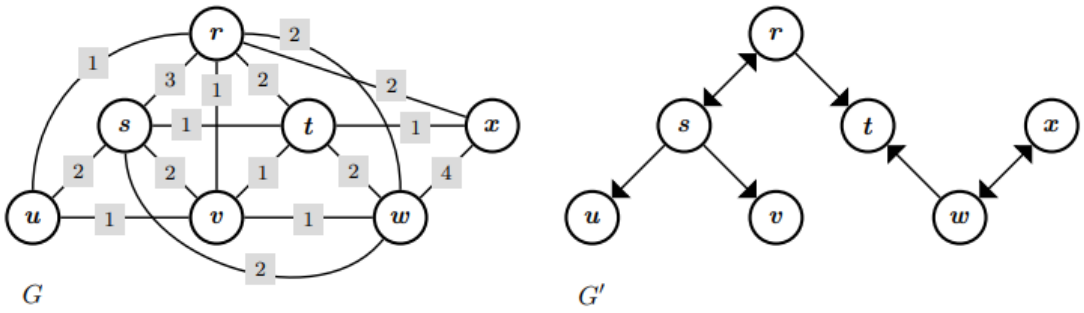


Σχήμα 2.8: Ένα παράδειγμα για το πώς λειτουργεί ο Chinese whispers. Τα διαφορετικά χρώματα αντιπροσωπεύουν διαφορετικές συστάδες.

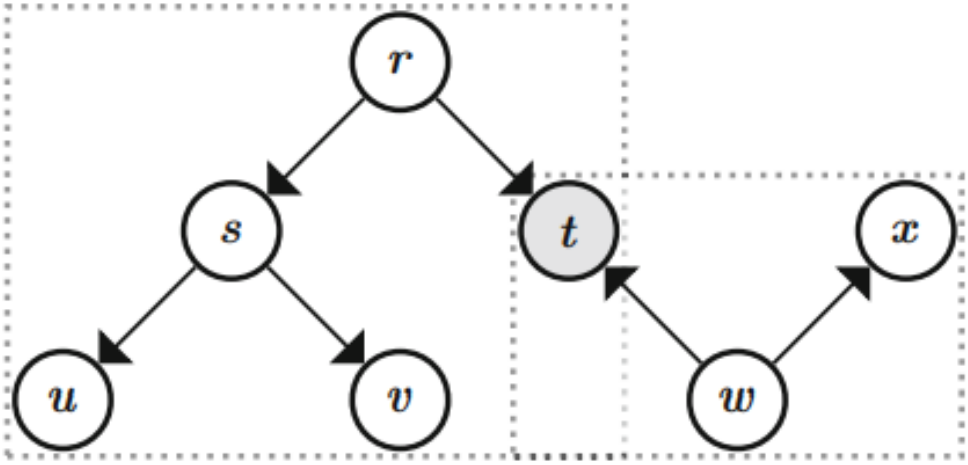
αντικείμενα , ο Watset παράγει ένα σύνολο από σαφείς αλληλεπικαλυπτόμενες συστάδες με αποσαφήνιση και ομαδοποιώντας τα διφορούμενα αυτά αντικείμενα. Ουσιαστικά χρησιμοποιεί την υπάρχοντες αλγορίθμους σκληρής ομαδοποίησης για γράφους για να ληφθεί μια μαλακή (fuzzy) συσταδοποίηση.

2.7 Git

Το Git είναι ένα σύστημα ελέγχου εκδόσεων (λέγεται και σύστημα ελέγχου αναθεωρήσεων ή σύστημα ελέγχου πηγαίου κώδικα) με έμφαση στην ταχύτητα, στην ακεραιότητα των δεδομένων και στην υποστήριξη για κατανεμημένες μη γραμμικές ροές εργασίας. Το Git σχεδιάστηκε και αναπτύχθηκε αρχικά από τον Λίνους Τόρβαλντς για τη ανάπτυξη του πυρήνα Linux το 2005 και έχει γίνει από τότε το πιο διαδεδομένο σύστημα ελέγχου εκδόσεων για ανάπτυξη λογισμικού.

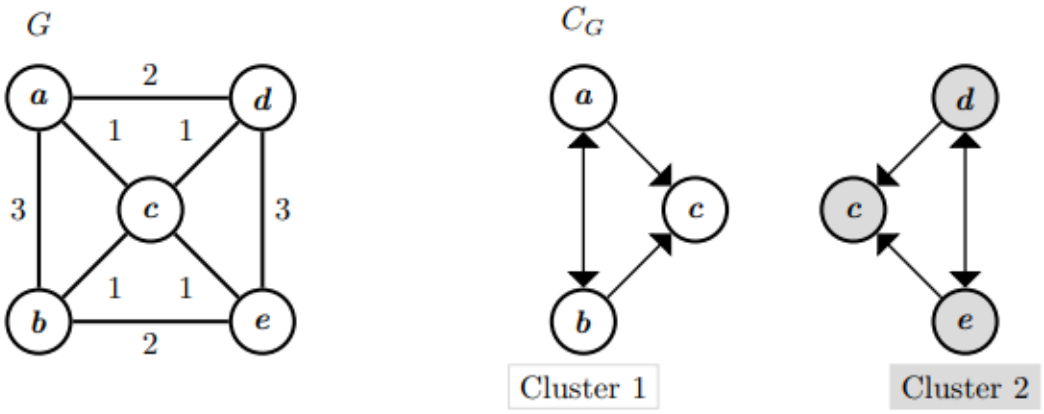


Σχήμα 2.9: G και μετατρέπεται σε κατευθυνόμενο γράφο με βάρη G'

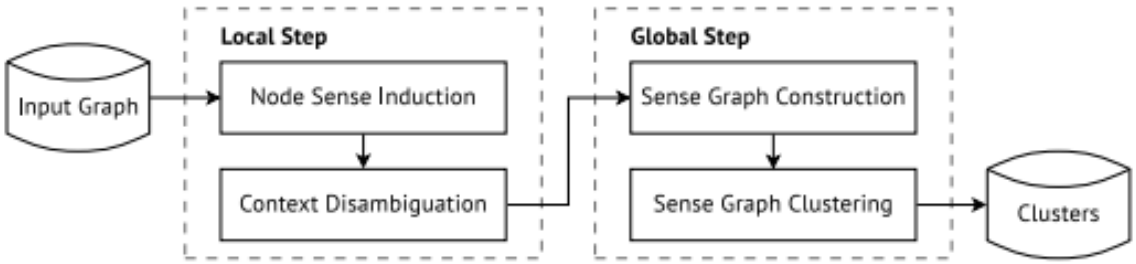


Σχήμα 2.10: Παράδειγμα 1: δύο συστάδων μετά τη μετατροπή

Όπως τα περισσότερα άλλα κατανεμημένα συστήματα ελέγχου εκδόσεων/αναθεωρήσεων και αντίθετα με τα περισσότερα συστήματα πελάτη-διακομιστή, κάθε κατάλογος εργασίας του Git είναι ένα ολοκληρωμένο αποθετήριο λογισμικού με πλήρες ιστορικό και δυνατότητες πλήρους παρακολούθησης της έκδοσης, ανεξάρτητα από την πρόσβαση δικτύου ή ενός κεντρικού διακομιστή. Όπως ο πυρήνας Linux, το Git είναι Ελεύθερο λογισμικό που διανέμεται κάτω από τους όρους της έκδοσης 2 της Γενικής Άδειας Δημόσιας Χρήσης GNU.



Σχήμα 2.11: Παράδειγμα 2: δύο συστάδων μετά τη μετατροπή



Σχήμα 2.12: Το περίγραμμα του αλγορίθμου WATSET που δείχνει το τοπικό βήμα της επαφής και του πλαισίου της αίσθησης λέξεων την αποσαφήνιση και το συνολικό βήμα της κατασκευής και της συσταδοποίησης του γράφου

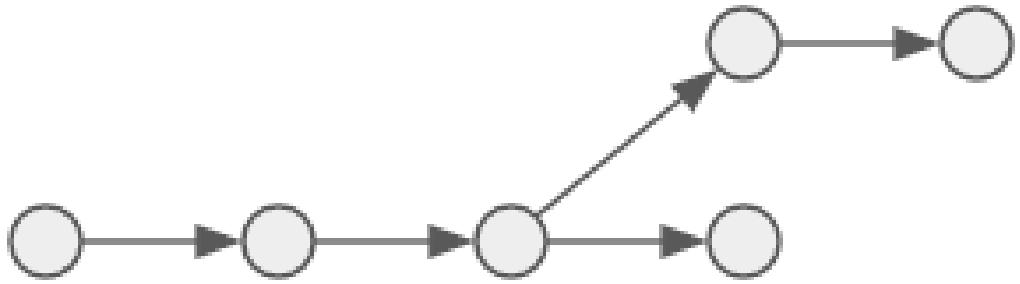
2.7.1 Λεξιλόγιο

Παρακάτω παρατίθεται το βασικό λεξιλόγιο σχετικό με το Git για την κατανόηση ορισμένων εννοιών που θα ακολουθήσουν στη συνέχεια της πτυχιακής εργασίας.

- **Bare repository** είναι κανονικά ένας κατάλληλος κατάλογος με ένα κατάληξη .git που δεν διαθέτει ένα τοπικά ελεγμένο αντίγραφο οποιουδήποτε από τα αρχεία που βρίσκονται υπό τον έλεγχο αναθεώρησης. Δηλαδή, όλα τα αρχεία διαχείρισης και ελέγχου του Git που κανονικά θα υπήρχαν στον κρυμμένο υποφάκελο .git είναι απευθείας παρόντα στον κατάλογο repository.git και δεν υπάρχουν άλλα αρχεία. Συνήθως οι εκδότες δημόσιων χώρων αποθήκευσης διαθέτουν διαθέσιμα γυμνά αποθετήρια. [16]
- **Repository** είναι μία συλλογή από αναφορές μαζί με μια βάση από αντικείμενα

που είναι προσβάσιμα από τις αναφορές. Πιθανόν συνοδεύονται από μεταδεδομένα από ένα ή πολλά αποθετήρια. [16]

- **DAG (Directed acyclic graph)** Τα commits objects κατατάσσονται σε ένα κατευθυνόμενο απεριοδικό γράφο, αφού έχουν γονείς (κατευθυνόμενος) και ο γράφος των commit objects είναι απεριοδικός. (δεν υπάρχει αλυσίδα που αρχίζει και τελειώνει με το ίδιο αντικείμενο) [16]



Σχήμα 2.13: DAG (Directed acyclic graph)

- **Clone** είναι ένα αντίγραφο ενός repository που ζει στον υπολογιστή, αντί του διακομιστή ενός δικτυακού τόπου κάπου, ή η πράξη δημιουργίας αυτού του αντιγράφου. Με τον κλώνο μπορούμε να επεξεργαστούμε τα αρχεία στον υπολογιστή μας τοπικά και να χρησιμοποιούμε το Git για να παρακολουθούμε τις αλλαγές χωρίς να χρειάζεται να είμαστε συνδεδεμένοι στο διαδίκτυο. Συνδέεται, ωστόσο, με την απομακρυσμένη έκδοση, έτσι ώστε οι αλλαγές να μπορούν να συγχρονιστούν μεταξύ των δύο. Μπορούμε να ¨σπρώξουμε¨ (push) τις τοπικές αλλαγές στο remote repository για να υπάρχει συγχρονισμός μεταξύ τους. [17]

- **Branch** (κλαδί) είναι μια παράλληλη έκδοση ενός repository (αποθετηρίου). Περιλαμβάνεται μέσα στο repository, αλλά δεν επηρεάζει κύρια ή το κύριο repository ώστε να εργαζόμαστε ελεύθερα χωρίς να διακόπτεται η ΄κύρια΄ έκδοση. Όταν δημιουργούνται αλλαγές οι οποίες θέλουμε να συγχωνεύτουν με το κύριο πρόγραμμα μπορούμε να κάνουμε merge τις αλλαγές και να δημοσιεύτουν μαζί με το κύριο πρόγραμμα. Το κύριο κλαδί του προγράμματος ονομάζεται master. [17]

- **Commit** ή revision, είναι μια μεμονωμένη αλλαγή σε ένα αρχείο (ή ένα σύνολο αρχείων). Είναι παρόμοια διαδικασία με την αποθήκευση ενός αρχείου, κάθε φορά δημιουργείται ένα μοναδικό ID(SHA-1 ή hash) που επιτρέπει να καταγράφουμε τις αλλαγές που έγιναν, την ώρα που έγιναν και από ποιον. Τα commits συνήθως περιέχουν ένα μήνυμα που αποτελεί μία σύντομη περιγραφή των αλλαγών που έγιναν. [17]

- **Committer** είναι ένα άτομο που είναι σε θέση να τροποποιήσει τον πηγαίο κώδικα ενός συγκεκριμένου λογισμικού ανοιχτού κώδικα και να τον ανεβάσει. [18]

- **Author** είναι ο χρήστης που έγραψε τις αλλαγές, ενώ ο Committer είναι ο χρήστης που τον ανέβασε στο αποθετήριο. Υπάρχουν αρκετές περιπτώσεις που αυτά τα δύο πρόσωπα δεν είναι ίδια.

- **Merge** το χρησιμοποιούμε για να μεταφέρουμε τα περιεχόμενα ενός άλλου branch (πιθανώς από ένα εξωτερικό repository) στον τρέχον branch. Το Merging πραγματοποιείται με μια αυτόματη διαδικασία που προσδιορίζει τις αλλαγές που έγιναν από τότε που τα branches αποκλίνουν και στη συνέχεια εφαρμόζει όλες αυτές τις αλλαγές μαζί. Σε περιπτώσεις που οι αλλαγές έρχονται σε σύγκρουση, μπορεί να χρειαστεί χειροκίνητη παρέμβαση για να ολοκληρωθεί της συγχώνευσης. [16]

- **Pull request** είναι προτεινόμενες αλλαγές ή προσθήκες σε ένα αποθετήριο που υποβάλλονται από ένα χρήστη και εγκρίνονται ή απορρίπτονται από τους βασικούς δημιουργούς και συντηρητές του αποθετηρίου. [17]

- **SHA-1 (Secure Hash Algorithm 1)** είναι ένας αλγόριθμος κρυπτογραφίας που στο πλαίσιο του Git χρησιμοποιείται ως συνώνυμο για το όνομα του αντικειμένου. [16]

Κεφάλαιο 3

Συμβουλευτική εφαρμογή για την ομαδοποίηση κώδικα από το ιστορικό του git

Για το σκοπό της εργασίας αυτής δημιουργήθηκε μία εφαρμογή η οποία βασισμένη στο ιστορικό των αλλαγών ενός αποθετηρίου Γιτ και με τη χρήση ορισμένων αλγορίθμων συσταδοποίησης προτείνει ομαδοποιήσεις στο κώδικα. Η εφαρμογή ονομάστηκε Pink pony και μπορείτε να την κατεβάσετε και να την χρησιμοποιήσετε από το GitHub στο https://www.github.com/Pavlmits/PinkPony. Η εφαρμογή Pink pony βασίζεται στην ιδέα του ¨**Ο,τι αλλάζει μαζί, θα πρέπει να βρίσκεται και μαζί.**¨ Η εφαρμογή έχει τη δυνατότητα, ανάλογα με τις παραμέτρους της, να λειτουργήσει με δύο τρόπους. Ο πρώτος τρόπος που διαθέτει για να προτείνει τις συστάδες που χωρίζονται ανάλογα με τη λειτουργία είναι σε βαθμό αρχείου. Ο δεύτερος τρόπος από αυτούς είναι σε βαθμό module.

3.1 Σημαντικότητα Version Control ως πηγή γνώσεων

Καθημερινά οι προγραμματιστές χρησιμοποιούν ένα εργαλείο για την διαχείριση των αλλαγών στο πέρασμα του χρόνου. Δηλαδή υπάρχει η δυνατότητα παρακολούθησης

όλης της ζωής του έργου. Με τη χρήση ενός Version Control συστήματος βελτιώνεται η διαφάνεια στο έργο και επιταχύνει τη διαδικασία για την παράδοση του έργου. Η τυπική διαδικασία που ακολουθεί μία/ένας προγραμματίστρια/ης για να προσθέσει ή να διορθώσει ένα κομμάτι κώδικα ξεκινά με την δημιουργία ενός κλαδιού(branch) για να μπορέσει να πάρει ένα αντίγραφο του βασικού κώδικα που υπάρχει στο αποθετήριο. Έπειτα ακολουθεί η συγγραφή του κώδικα για την προσθήκη ή τη διόρθωση που καλείται να κάνει. Στο τέλος της συγγραφής η/ο προγραμματίστρια/ης καλείται να κάνει commit τις αλλαγές της/του για την αποθήκευση τους στο τοπικό αποθετήριο. Για να μπορέσουν αυτές οι αλλαγές να μοιραστούν με την υπόλοιπη ομάδα γίνεται και push, δηλαδή στέλνονται οι αλλαγές στο απομακρυσμένο αποθετήριο της ομάδας. Με αυτό τον τρόπο το απομακρυσμένο αποθετήριο έχει αποθηκευμένες όλες τις αλλαγές, τις προσθήκες και τις διορθώσεις και μπορούν να χρησιμοποιηθούν ως ιστορικό του έργου.

3.2 Σε βαθμό αρχείου

Με τη σωστή επίκληση της εφαρμογής, έχει τη δυνατότητα ο χρήστης να ομαδοποιήσει τα αρχεία του κωδικά του σε συστάδες. Δηλαδή η εφαρμογή θα προτείνει ομάδες κώδικα που είναι λογικά συνδεδεμένες μεταξύ τους ώστε να υπάρχει καλύτερη ομαδοποίηση του κώδικα. Το πρόγραμμα αρχικά φιλτράρει τα αρχεία ώστε να επεξεργαστεί μόνο αυτά που χρειάζονται. Τα αρχεία αυτά που απομονώνονται από το πρόγραμμα είναι αρχεία .txt που περιέχουν δεδομένα για τα tests, αρχεία που σχετίζονται με τη διαχείριση του version control, αρχεία .xml που χρησιμοποιούνται ως configuration files και ό,τι θεωρηθεί ότι δεν μας απασχολεί για την ερευνά μας μπορούμε να το απομονώσουμε και αυτό. Οι ομάδες που θα παραχθούν από το πρόγραμμα θα μπορούσαμε να πούμε ότι είναι υποψήφια modules. Επεξηγηματικά τα αρχεία που βρίσκονται μέσα σε κάθε μία από τις ομάδες έχουν αλλάξει αρκετές φορές μαζί που σημαίνει ότι είναι στενά συνδεδεμένα μεταξύ τους και για αυτό θα μπορούσαν να αποτελούν ένα module στο προγράμματά μας.

3.3 Σε βαθμό module

Είναι αρκετά σύνηθες να συναντάμε παλαιότερα συστήματα λογισμικού που να περιέχουν αρκετά modules. Δηλαδή να αποτελούν ένα υπέρογκο μονολιθικό σύστημα που χρειάζεται αδικαιολόγητο χρόνο για ένα απλό build. Στις μέρες μας μεγάλες εταιρίες αποφασίζουν να δημιουργήσουν ή να προσπαθήσουν να μετατρέψουν τα συστήματα τους σε κατανεμημένα συστήματα. Για τη μετατροπή των ήδη υπαρχόντων μονολιθικών προγραμμάτων σε σε κατανεμημένα συστήματα η εφαρμογή Pink pony μπορεί να βοηθήσει στην ανίχνευση των modules που θα μπορούσαν να αποκοπούν. Με την ίδια λογική όπως και με τα αρχεία, τα modules που έχουν αλλάξει πολλαπλές φορές μαζί θα μπορούσαν να μελετηθούν εκτενέστερα για να εξεταστούν τα ενδεχόμενα να αποτελέσουν ένα απομονωμένο υποσύστημα.

3.4 Τρόπος χρήσης της εφαρμογής

Βαθμός Συσταδοποίησης

Ανάλογα το βαθμο που θέλουμε να κάνουμε την ερευνά μας, υπάρχουν δύο επιλογές.

- `file` : Σε βαθμό αρχείου οι ομάδες κώδικα
- `module` : Σε βαθμό module οι ομάδες κώδικα
- Η παρακάτω εντολή θα προτείνει ομάδες στο κώδικα σε βαθμό αρχείου.

```
$ java -jar pinkpony.jar path\to\.git file max
```

- Η παρακάτω εντολή θα προτείνει modules κάτω από ένα συγκεκριμένο module.

Παράδειγμα

Δομή πρότζεκτ εισόδου

```
moduleName
    file1
    file2
    ...
    fileN
```

output

```
    cluster1
        file1
        file3
    cluster2
        file6
        file89
        ...
    clusterN
        file2
        file15
```

```
$ java -jar pinkpony.jar path\to\.git file max moduleName
```

- Η ακόλουθη εντολή θα προτείνει νέα co-change modules από τα submodules : mod1, mod2, mod3 με τον Markov αλγόριθμο συσταδοποίησης.

Παράδειγμα

Δομή πρότζεκτ εισόδου

```
mod1
    subMod1
    sudMod2
    subMod3
mod2
    subMod4
    sudMod5
mod3
    subMod6
    sudMod7
    subMod8
    subMod9
```

output

```
cluster1
    subMod3
    subMod9
    subMod8
cluster2
    subMod4
    subMod5
    subMod1
...
```

```
$ java -jar pinkpony.jar path\to\.git module mr mod1 mod1 mod2
```

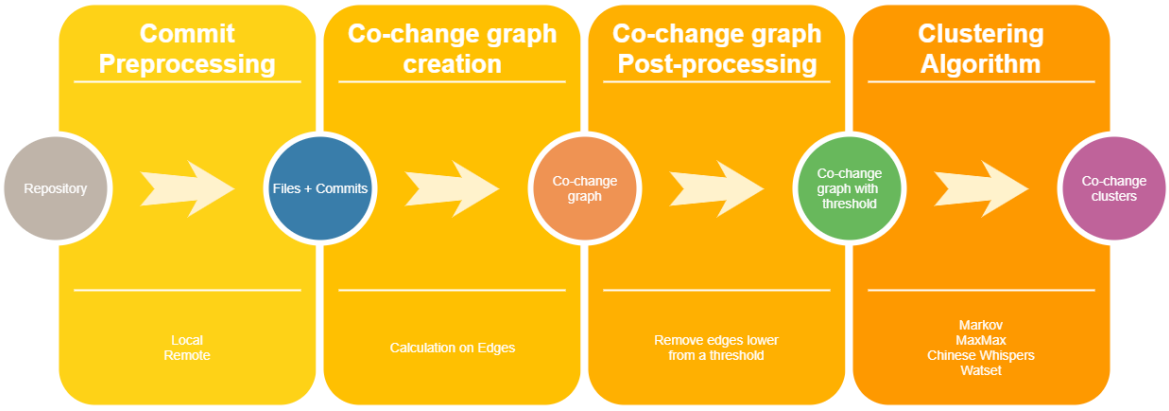
Κεφάλαιο 4

Δομή της εφαρμογής Pink pony

Η εφαρμογή Pink pony είναι γραμμένη σε Java και έχουν χρησιμοποιηθεί αρκετές βιβλιοθήκες της επίσης, όπως η Guava της Google, το Lombok, το jGit και άλλες. Επίσης περιλαμβάνει πολλαπλά τεστ ώστε να μπορεί να εγγυηθεί για τη ποιότητα των αποτελεσμάτων με τις παρούσες συνθήκες.

4.1 Βήματα υλοποίησης

Για την υλοποίηση της εφαρμογής Pink pony ακολουθήθηκαν πολλαπλά βήματα βασισμένα σε μεθόδους που εφάρμοσαν ερευνητές στα πρόβλημα της σωστής ομαδοποίησης του κώδικα.



Σχήμα 4.1: Βήματα υλοποίησης

4.1.1 Προεπεξεργασία δεδομένων

Καθώς διαθέτουμε ολόκληρο το ιστορικό του Git από αυτό η εφαρμογή αποσπά μόνο τα commits τα οποία περιέχουν τη χρήσιμη πληροφορία που την αφορούν. Σε αυτό το βήμα επίσης η εφαρμογή εφαρμόζει φίλτρα αφού μερικά commits μπορεί να αλλοιώσουν το αποτέλεσμα που θέλουμε. Όταν εξάγουμε το co-change γράφο είναι θεμελιώδης η προεπεξεργασία των commits αφού ενδέχεται να μολύνουν το γράφημα με θόρυβο.

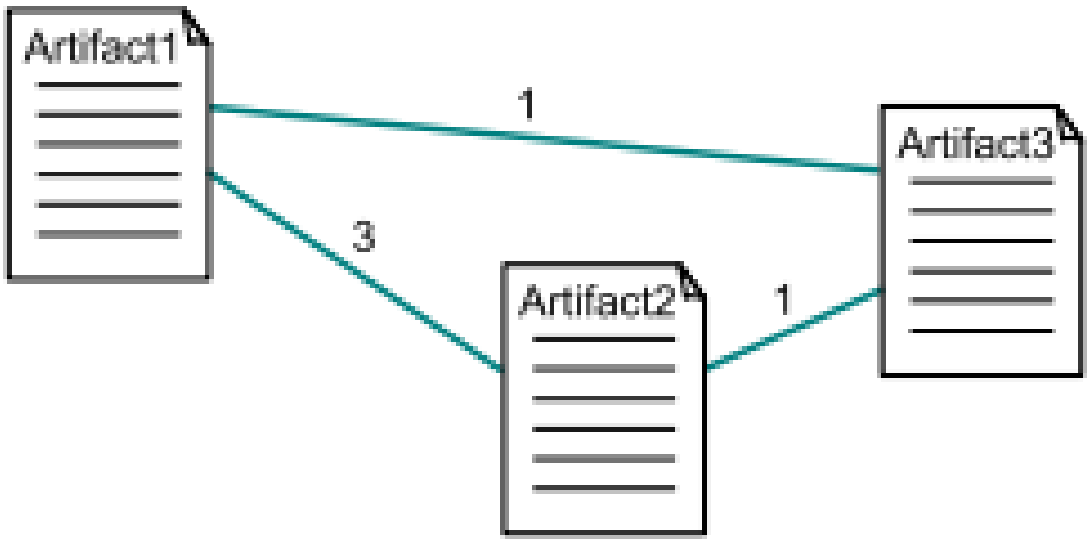
1. **Αφαίρεση commits που δεν σχετίζονται με ζητήματα συντήρησης του κώδικα** . Στα αρχικά στάδια υλοποίησης του συστήματος θα υπάρχουν commits που μπορεί να υποδηλώσουν μερικές υλοποιήσεις των εργασιών προγραμματισμού, δεδομένου ότι το σύστημα βρίσκεται υπό κατασκευή [19]. Όταν τέτοια commits εκτελούνται πολλές φορές, δημιουργούν θόρυβο στα βάρη των ακμών.

2. **Αφαίρεση commits που δεν αλλάζουν κλάσεις**: Οι συναλλαγές που εξετάζονται από την προσέγγισή αύτη καθορίζονται κατά κύριο παράγοντα από τις κλάσεις και τα modules. Ωστόσο, υπάρχουν commits που αλλάζουν μόνο αντικείμενα όπως αρχεία διαμόρφωσης, τεκμηρίωση, script files κλπ. Επομένως, απορρίπτουμε τέτοια commits προκειμένου να εξεταστούν μόνο commits που αλλάζουν τουλάχιστον μία κλάση ή ένα module.

3. **Αφαίρεση εκτεταμένων commits** : Αφαιρούμε commits που αντιπροσωπεύουν εξαιρετικά διάσπαρτες αλλαγές κώδικα, δηλαδή commits που τροποποιούν ένα τεράστιο αριθμό κλάσεων. Συνήθως, τέτοια commits σχετίζονται με refactorings (όπως τη μέθοδο μετονομασίας), βελτίωσης ποιότητας (όπως απομάκρυνση νεκρού κώδικα), υλοποίηση νέων χαρακτηριστικών ή μικρές συντακτικές διορθώσεις (όπως αλλαγές στα σχόλια). Τέτοια είδους commits ενδέχεται να έχουν σημαντικό αντίκτυπο όταν λαμβάνονται υπόψη στον co-change γράφο, λόγω της πολύ μεγάλης απόκλισης μεταξύ του αριθμού των κλάσεων που έχουν αλλάξει από αυτούς και από τα υπόλοιπα co-change στο σύστημα. Για αυτό το λόγο αποφασίστηκε να αφαιρεθούν κατά τη διάρκεια δημιουργίας του γράφου. Χρησιμοποιώντας αυτόν τον ορισμό, αγνοούμε commits που αλλάζουν περισσότερα από MAX_SCATTERING_FILES

4.1.2 Δημιουργία του Co-change graph

Όπως προτείνεται από τους Beyer και Noack [3] ο co-change γράφος είναι μία αναπαράσταση ενός version control συστήματος((VCS)). Στη περίπτωση του Pink pony κάθε φορά οι κορυφές του γράφου αλλάζουν ανάλογα με την έρευνα που πραγματοποιούμε. Αν ενδιαφερόμαστε να ομαδοποιήσουμε κλάσεις ή τα αρχεία τότε όπως και παραπάνω θα χρησιμοποιήσουμε ως κορυφές τις κλάσεις και θα ενώνονται με τις ακμές αν κάποια στιγμή έχουν αλλάξει μαζί. Αν όμως θέλουμε να μελετήσουμε τις σχέσεις ανάμεσα στα modules και να τα ομαδοποιήσουμε θα δημιουργηθεί ο co-change γράφος που θα έχει ως κορυφές τα διαφορετικά modules και θα συνδέονται με το ίδιο τρόπο με τις ακμές.



Σχήμα 4.2: Co-change γράφος

4.1.3 Υπολογισμός βαρών που co-change γράφου

Για τον υπολογισμό των βαρών του co-change γράφου χρησιμοποιήθηκε ο αριθμός για το πόσες φορές άλλαξαν μαζί τα αρχεία. Από το ιστορικό του Git λαμβάνουμε τα commits και υπολογίζουμε πόσες φορές ένα ζεύγος αρχείων έχει αλλάξει μαζί. Τα αποτελέσματά από όλους αυτούς τους υπολογισμούς, καθώς πραγματοποιείται ο υπολογισμός, αποθηκεύονται σε έναν άνω τριγωνικό πίνακα. Χρησιμοποιείται ο άνω τριγωνικός πίνακας για την εξοικονόμηση χρόνου ώστε να μην υπολογίζονται τα ίδια

αποτελέσματα σε επανάληψη.

4.1.4 Συσταδοποίηση του co-change γράφου

Για την συσταδοποίηση του co-change γράφου χρησιμοποιήθηκαν τέσσερις διαφορετικοί αλγόριθμοι συσταδοποίησης. Ο χρήστης έχει τη δυνατότητα να επιλέξει με ποιόν αλγόριθμο θα θέλει να υπολογίσει το αποτέλεσμα για την ομαδοποίηση των αρχείων ή των modules του συστηματός του. Οι επιλογές στους αλγόριθμους είναι ανάμεσα στο Markov [11], στο MaxMax [15], στο Watset [13] και το Chinese Whispers [14].

4.2 Σχεδιαστικές αποφάσεις

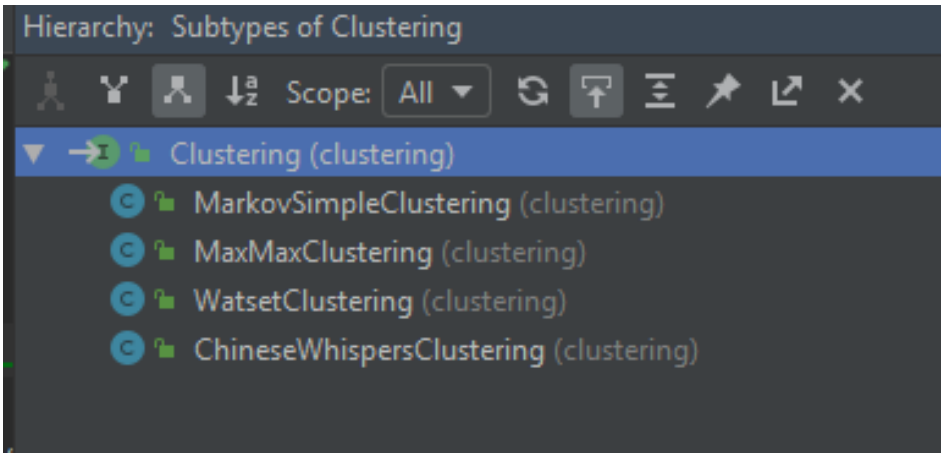
1. Αρχικά για να μπορέσουμε να μελετήσουμε το ιστορικό του Git χρησιμοποιήθηκε η βιβλιοθήκη JGit. Η βιβλιοθήκη JGit έχει πολύ λίγες εξαρτήσεις, καθιστώντας την κατάλληλη για ενσωμάτωση σε οποιαδήποτε εφαρμογή Java. Διαθέτει EDL (new-style BSD) άδεια χρήσης και βρίσκεται κάτω από την ομπρέλα των εφαρμογών του Eclipse fountation. Διαθέτει επίσης ρουτίνες πρόσβασης στο αποθετήριο, πρωτόκολλα δικτύου και βασικούς αλγόριθμους για τον ελέγχο της έκδοσης λογισμικού. version control.

2. Στην εφαρμογή ως build tool χρησιμοποιήθηκε το Maven. Το Maven εξετάζει δύο πτυχές του λογισμικού κατασκευής: πρώτον, περιγράφει τον τρόπο κατασκευής του λογισμικού και δεύτερον, περιγράφει τις εξαρτήσεις του. Ένα αρχείο XML περιγράφει το έργο λογισμικού που κατασκευάζεται, τις εξαρτήσεις του σε άλλες εξωτερικές μονάδες και εξαρτήματα, την τάξη κατασκευής, τους καταλόγους και τις απαραίτητες προσθήκες. Ως προσθήκη στο Maven χρησιμοποιήθηκε το maven-assembly-plugin για την δημιουργία ενός εκτελέσιμου αρχείου jar.

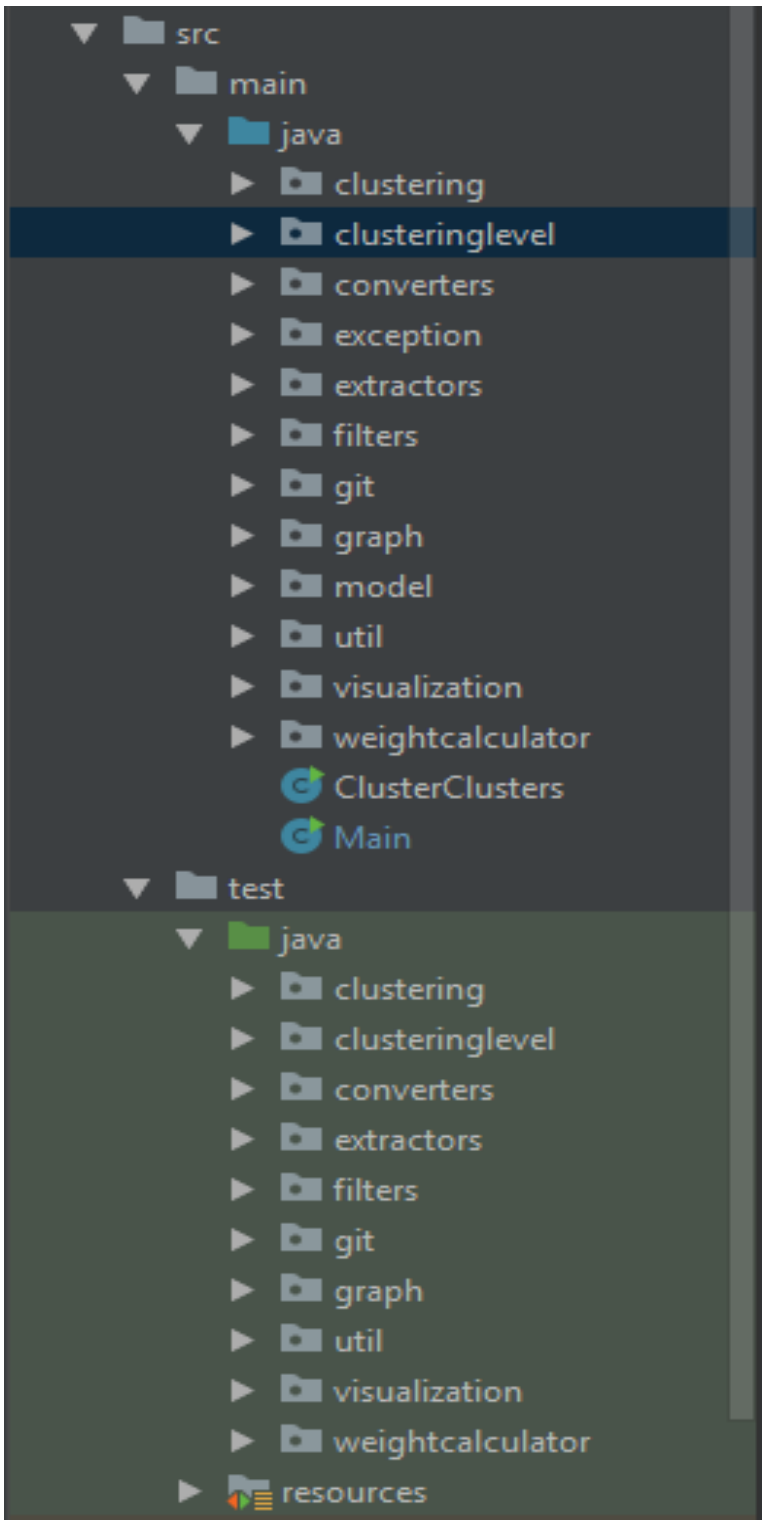
3. Για την αποφυγη boilerplate κώδικα χρησιμοποιείται εκτεταμένα το Lombok. Το Lombok είναι μία βιβλιοθήκη στη Java που δημιουργεί αυτόματα την στιγμή του build αρκετά βασικά στοιχεία των κλάσεων. Στην εφαρμογή του Pink pony χρησιμοποιήθηκαν για την αυτόματη δημιουργία ορισμένων setters-getters και constructors.

4. Ο διαχωρισμός των κλάσεων της εφαρμογής σε πακέτα πραγματοποιήθηκε με βάση τα λειτουργικά χαρακτηριστικά της εφαρμογής και την λειτουργική σύνδεση μεταξύ των components. Όπως είναι εμφανές από το Σχήμα 4.4 για κάθε πακέτο υπάρχει και το αντίστοιχο πακέτο στο πακέτο των τεστ αφού η εφαρμογή περιέχει τεστ για πολλαπλά σενάρια και λειτουργίες.

5. Η εφαρμογή έχει χτιστεί με τέτοιο τρόπο έτσι ώστε να μπορεί να υιοθετήσει αρκετά ομαλά πολλές μελλοντικές προσθήκες όπως για παράδειγμα αλγόριθμοι συσταδοποίησης, διαφορετική προσέγγιση στον υπολογισμό των βαρών του γράφου αλλά και καινούργια επίπεδα στο βαθμό της συσταδοποίησης. Για παράδειγμα για τους διαφορετικούς αλγόριθμους συσταδοποίησης υπάρχει ένα interface Clustering που υλοποιείται από τους αλγόριθμους και μια κλάση ClusteringFactory που ανάλογα με τις απαιτήσεις την παρούσα φάση επιστρέφει το σωστό αντικείμενο.



Σχήμα 4.3: Interface Clustering μαζί με υλοποιήσεις.



Σχήμα 4.4: Δομή των πακέτων του προγράμματος

Κεφάλαιο 5

Πειράματα

5.1 Σε βαθμό αρχείου

5.1.1 OjAlgo

Το ojAlgo είναι μια βιβλιοθήκη της Java όπου ο κώδικας είναι ανοιχτού λογισμικού που έχει σκοπό να κάνει τα μαθηματικά, τη γραμμική άλγεβρα και τη βελτιστοποίηση. Η συγκεκριμένη βιβλιοθήκη περιέχει περίπου 5 εκατομμύρια γραμμές κώδικα και έχει περίπου 10 contributors. Περιέχει επίσης περίπου 1,400 commits και 136 pull requests.

Στη βιβλιοθήκη OjAlgo πραγματοποιήθηκε η ανάλυση του Pink pony και παρουσίασε ορισμένα πολύ σημαντικά αποτελέσματα. Στο πείραμα αυτό χρησιμοποιήθηκαν και οι τέσσερις αλγόριθμοι. Όπως είναι εμφανές από τα διαγράμματα 5.2 και 5.2 οι αλγόριθμοι MaxMax και Watset που είναι αλγόριθμοι μαλακού τύπου συσταδοποίησης διαθέτουν τα ίδια αρχεία σε παραπάνω από μία κλάσεις. Σε αντίθεση με τους Markov και Chinese Whispers από τα διαγράμματά 5.4 και 5.3, τα αρχεία ανήκουν αυστηρά μόνο σε μία συστάδα. Επίσης καθώς παρατηρούμαι τα διαγράμματα μπορούμε να κάνουμε μια χοντρική εκτίμηση ότι τα αποτελέσματα είναι πολύ κοντά στο ιδανικό καθώς τα ονόματα των κλάσεων που βρίσκονται στην ίδια συστάδα είναι παρόμοια.

5.2 Σε βαθμό module

5.2.1 Spring Framework

Το Spring Framework είναι ένα application framework και inversion of control container για την πλατφόρμα της Java. Τα χαρακτηριστικά του πυρήνα του framework μπορούν να χρησιμοποιηθούν από οποιαδήποτε Java εφαρμογή, αλλά υπάρχουν και κάποιες επεκτάσεις για την δυνατότητα χτισίματος web εφαρμογών πάνω από τη Java EE (Enterprise Edition) . Έχει γίνει δημοφιλής στο Java community ως μία προσθήκη ή μια αντικατάσταση για τα Enterprise JavaBeans (EJB). Το Spring Framework είναι λογισμικό ανοιχτού κώδικα. Στο GitHub περιέχει 100 contributors και 1,353,274 γραμμές κώδικα. Περιέχει 43,271 commits και το πρώτο commit δημιουργήθηκε το Σεπτέμβριο του 2006.

Το Pink pony χρησιμοποιήθηκε στο Spring Framework για την ομαδοποίηση των modules που διαθέτει. Τα αποτελέσματα που φαίνονται στα διαγράμματα 5.7, 5.10 και 5.9 δείχνουν ακριβώς τις ίδιες τρεις ομαδοποιήσεις μεταξύ των modules εκτός από το διάγραμμα 5.8 που δείχνει πολλαπλές ομαδοποιήσεις. Ουσιαστικά οι τρεις αλγόριθμοι προτείνουν να ομαδοποιηθούν όλα τα modules μαζί και ξεχωριστά να είναι τα integration-tests και το JCL ου είναι το Jakarta Commons Logging API. Για την παραγωγή του αποτελέσματος της ανάλυσης της εφαρμογής χρειάστηκαν μόλις 7-10 δευτερόλεπτα, ανάλογα τον αλγόριθμο συσταδοποίησης. Ο υπολογιστής, στον οποίο έτρεξαν τα αποτελέσματα, διαθέτει 16GB RAM και ο επεξεργαστής του είναι ο Intel i5-8400 με 6 πυρήνες.

5.3 Σε ρεαλιστικό σύστημα

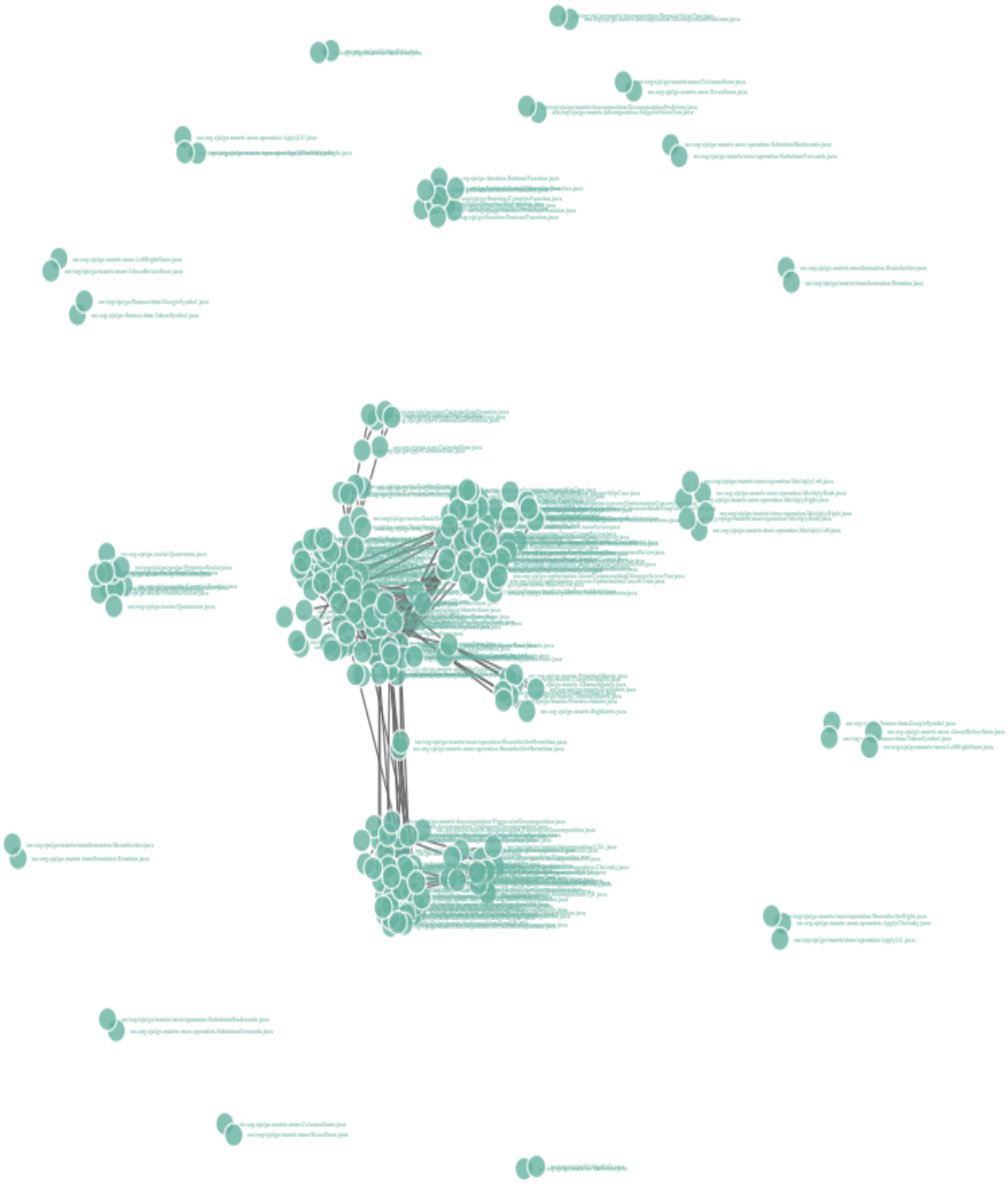
Η εφαρμογή Pink pony χρησιμοποιήθηκε ως βοηθητικό και για την διάσπαση ενός ρεαλιστικού συστήματος σε μία ολλανδική πολυεθνική εταιρεία.

5.3.1 Το Πρόβλημα

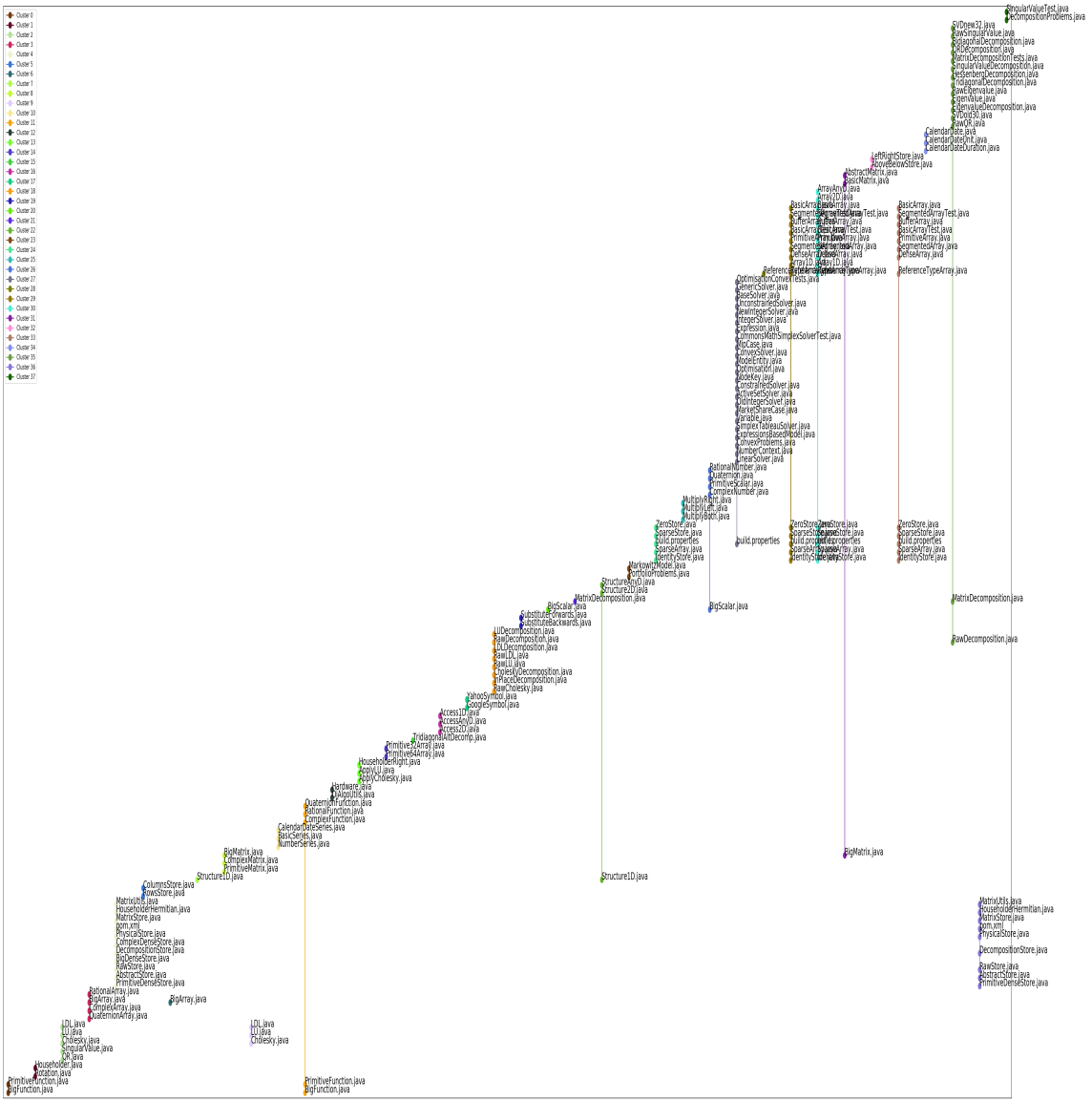
Το πρόβλημα που υφίσταται στο σύστημα τη συγκεκριμένης πολυεθνικής είναι ο μεγάλος αριθμός των modules στο σύστημα. Όλες οι ομάδες δουλεύουν σε αυτό το σύστημα με αποτέλεσμα να έχει μεγαλώσει σε τέτοιο βαθμό το σύστημα που καθυστερεί αρκετά την ανάπτυξη του κώδικα στις ομάδες. Η εταιρεία χρησιμοποιούσε το Code Scene [20] που είναι ένα εργαλείο για την ανακάλυψη μοτίβων στον κώδικα και ήθελε να δοκιμάσει και το Pink pony για να συνδυάσει τις πληροφορίες και από τα δύο ώστε να είναι σε θέση να πάρει κάποιες αποφάσεις για την διάσπαση του μονολιθικού συστήματος.

5.3.2 Χρήση του Pink pony στο πρόβλημα

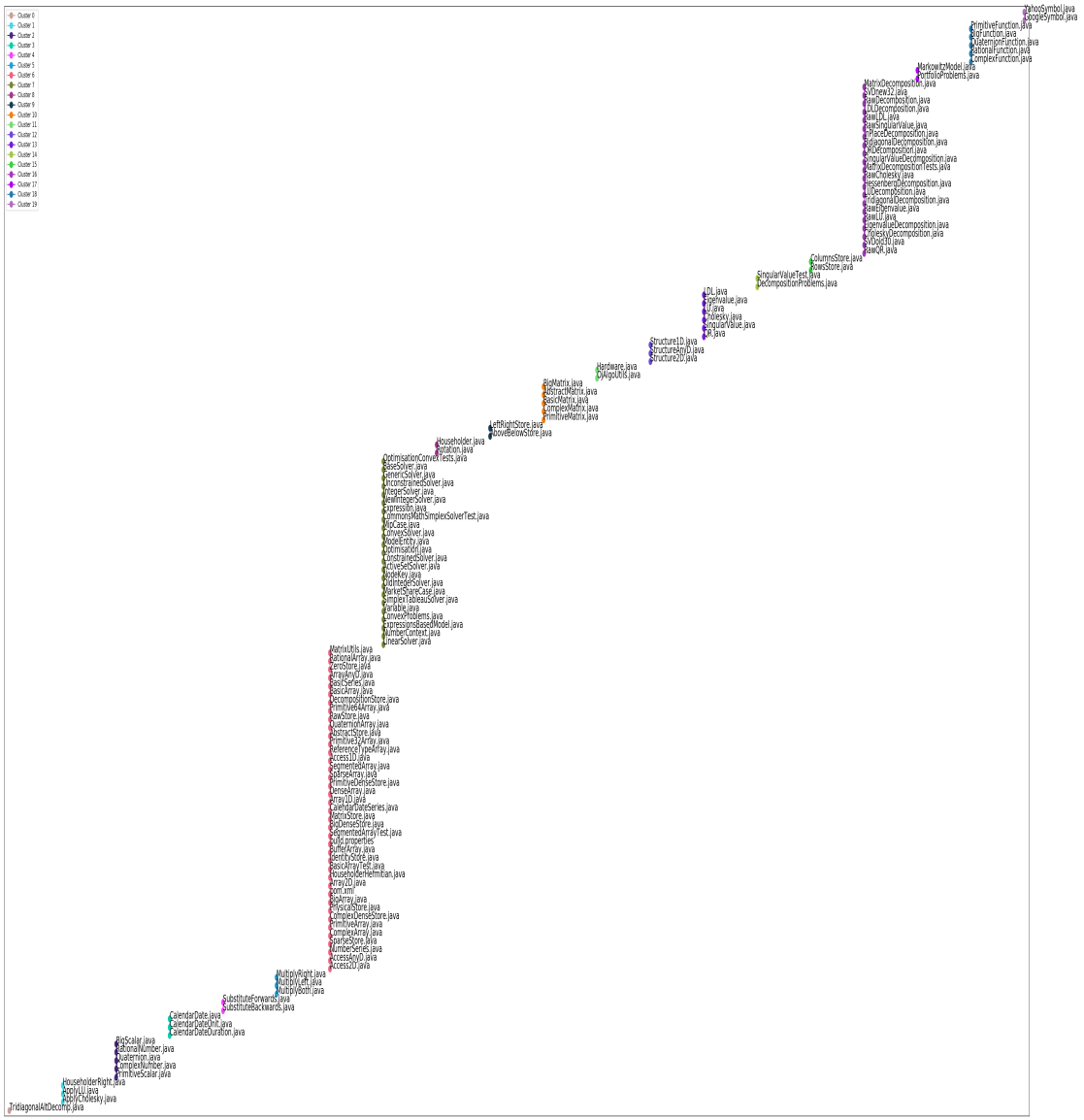
Στα πλαίσια και της ερευνάς, δημιουργήσαμε ένα ηλεκτρονικής μορφής ερωτηματολόγιο για να μπορέσουμε να έχουμε μία ρεαλιστική εκτίμηση από επαγγελματίες που γνωρίζουν το σύστημα. Το ερωτηματολόγιο περιελάμβανε ερωτήσεις για την ποιότητα,τη σημαντικότητα και το σκοπό της εφαρμογής Pink pony.Το ερωτηματολόγιο δόθηκε στους αρχιτέκτονες του συστήματος όπου το συμπλήρωσαν από κοινού. Στο Σχήμα 5.11 παρατίθενται οι απαντήσεις από το ερωτηματολόγιο που δόθηκε στους αρχιτέκτονες του συστήματος. Το σχόλιο που υπήρχε στο τέλος του ερωτηματολογίου ήταν ¨Η αξιολόγηση βασίζεται σε αξιολόγηση υψηλού επιπέδου και κατά πόσο οι προτεινόμενες ομάδες έχουν νόημα από λειτουργική / οργανωτική άποψη¨. Από τις απαντήσεις που ερωτηματολογίου μπορούμε να συμπεράνουμε ότι η συνολική εργασία αποδείχθηκε αρκετά σημαντική σε ένα υπάρχων πρόβλημα. Επίσης για την παρούσα εφαρμογή σε αυτά τα δεδομένα φαίνεται πως οι αλγόριθμοι Markov και Watset έχουν αρκετά φτωχά αποτελέσματα συγκριτικά με τα αποτελέσματα του MaxMax και του Chinese Wispers. Ο λόγος που μπορεί να κρύβεται πίσω από αυτά τα αποτελέσματα είναι ότι ο MaxMax και ο Chinese Wispers είναι μαλακοί αλγόριθμοι συσταδοποίησης, που σημαίνει ότι προτείνουν πολλαπλές ομαδοποιήσεις για ένα αρχείο ή module. Σε αντίθεση με το Markov που τοποθετεί ένα αρχείο ή module μόνο σε μία συστάδα. Από την άλλη η προσπαθεία του Watset να μετατρέψει έναν αλγόριθμο σκληρής συσταδοποίησης σε αλγόριθμο μαλακής φαίνεται να μην αρκετά επιτυχής.



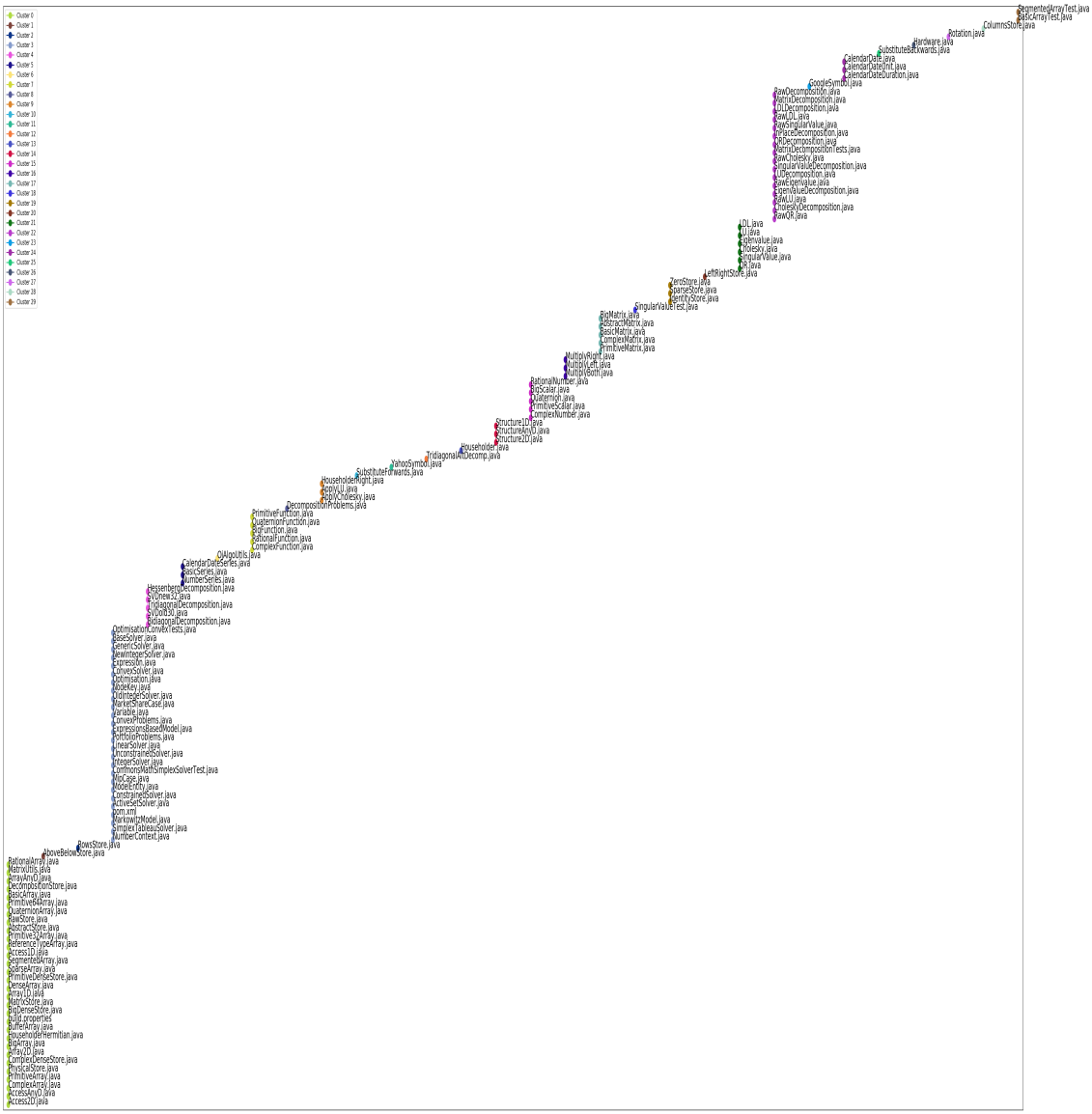
Σχήμα 5.1: Co-change από αρχεία στο OjAlgo



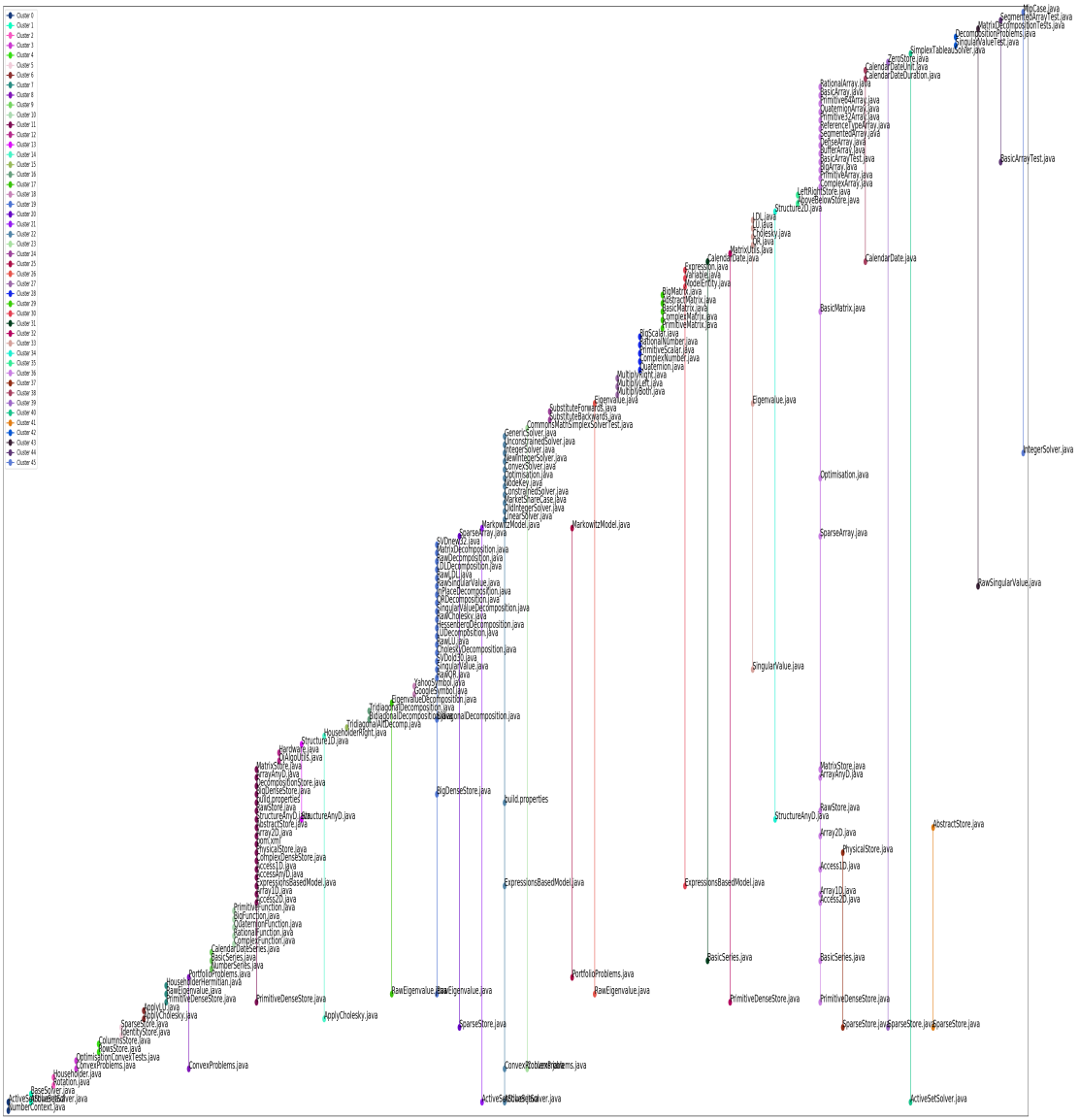
Σχήμα 5.2: MaxMax clustering στα αρχεία στο OjAlgo



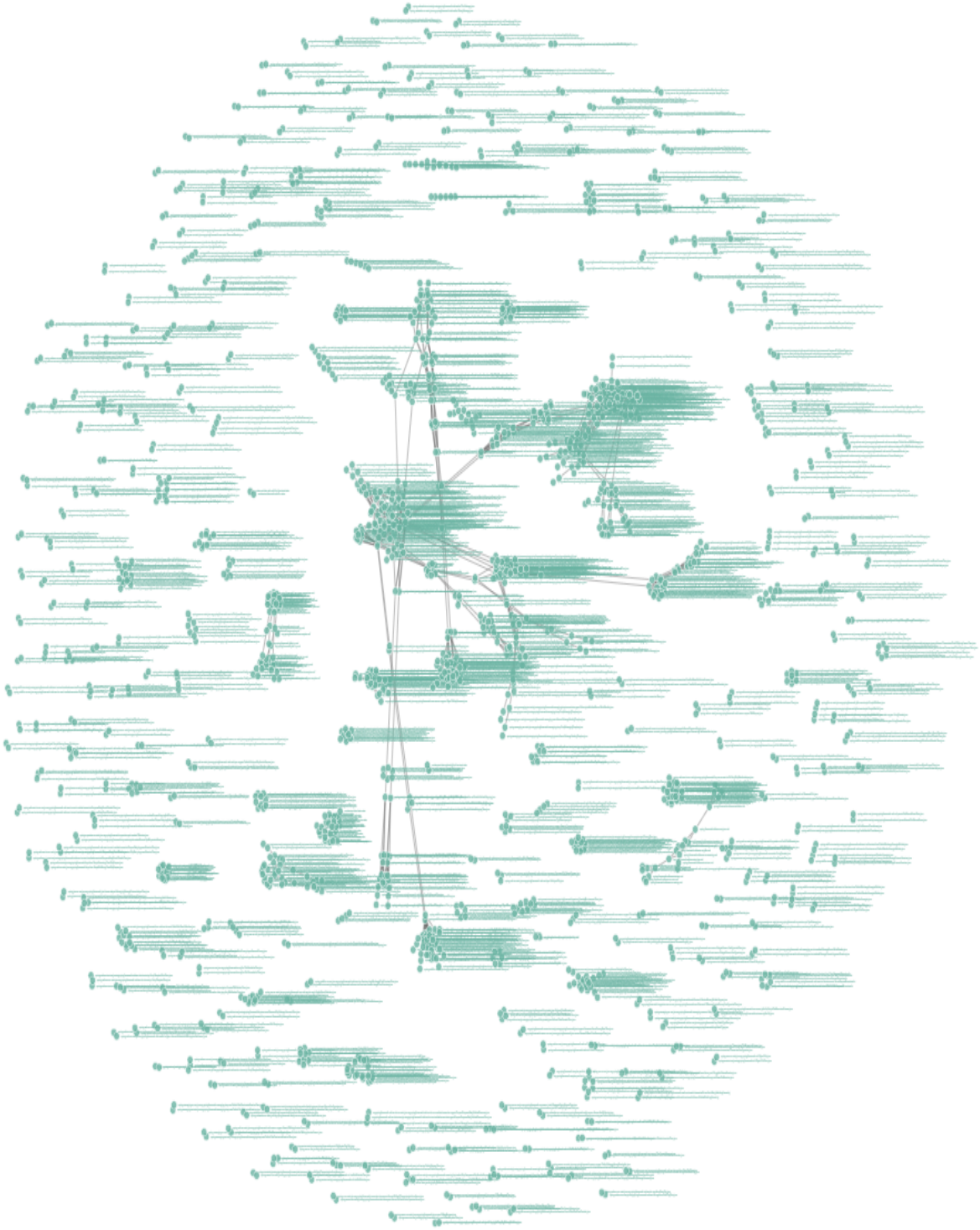
Σχήμα 5.3: Chinese clustering στα αρχεία στο OjAlgo



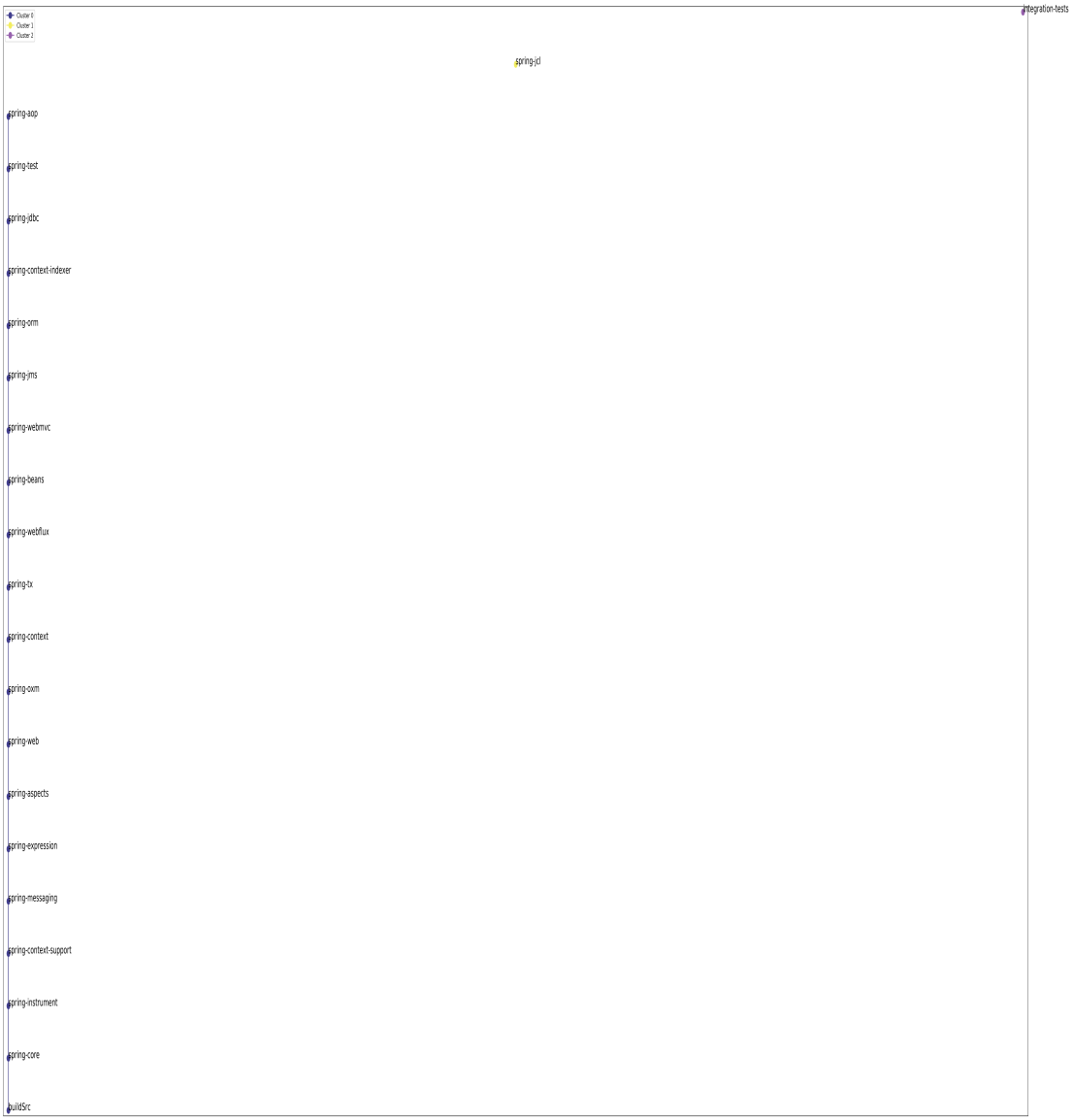
Σχήμα 5.4: Markov clustering στα αρχεία στο OjAlgo



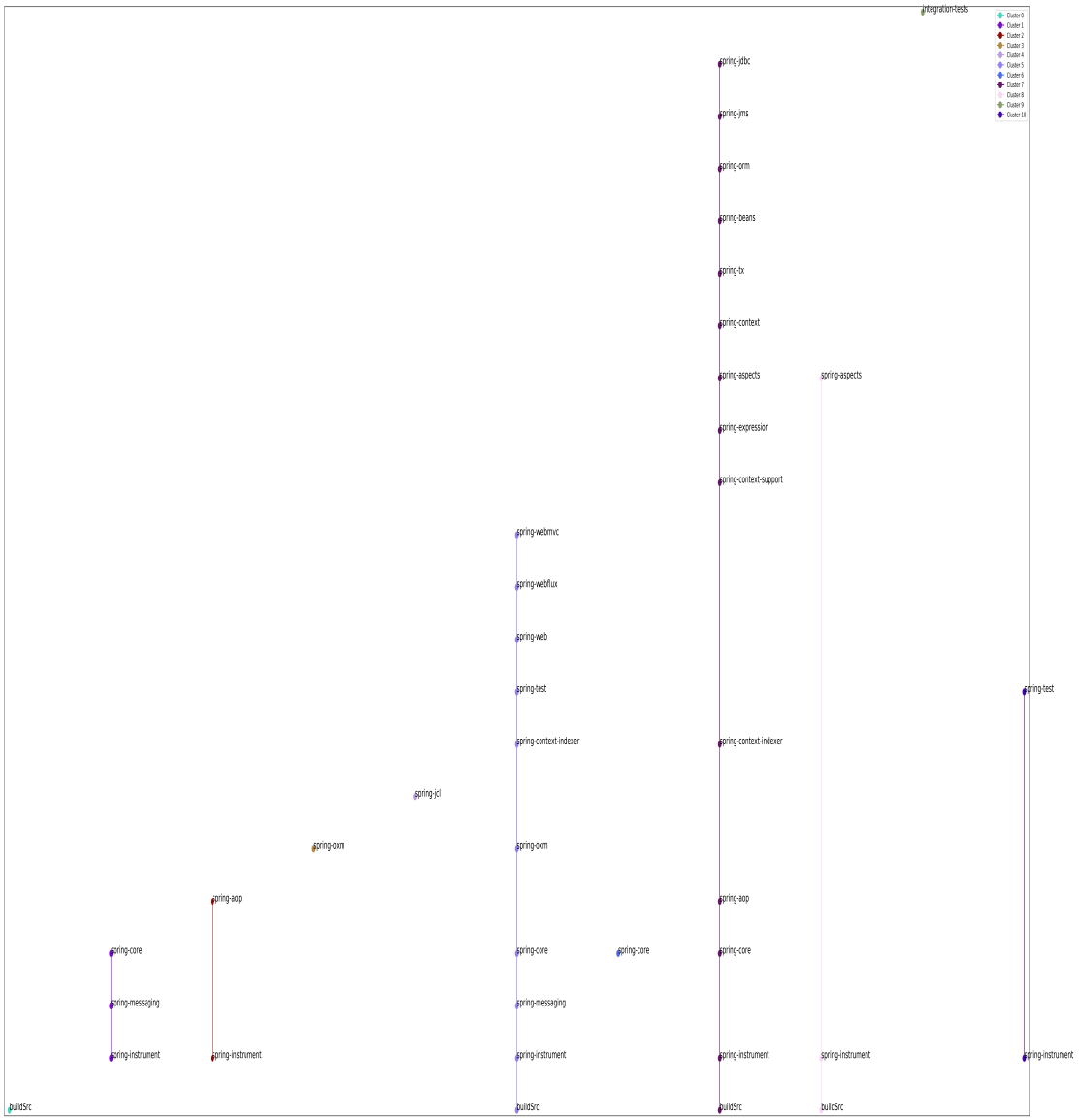
Σχήμα 5.5: Watset clustering στα αρχεία στο OjAlgo



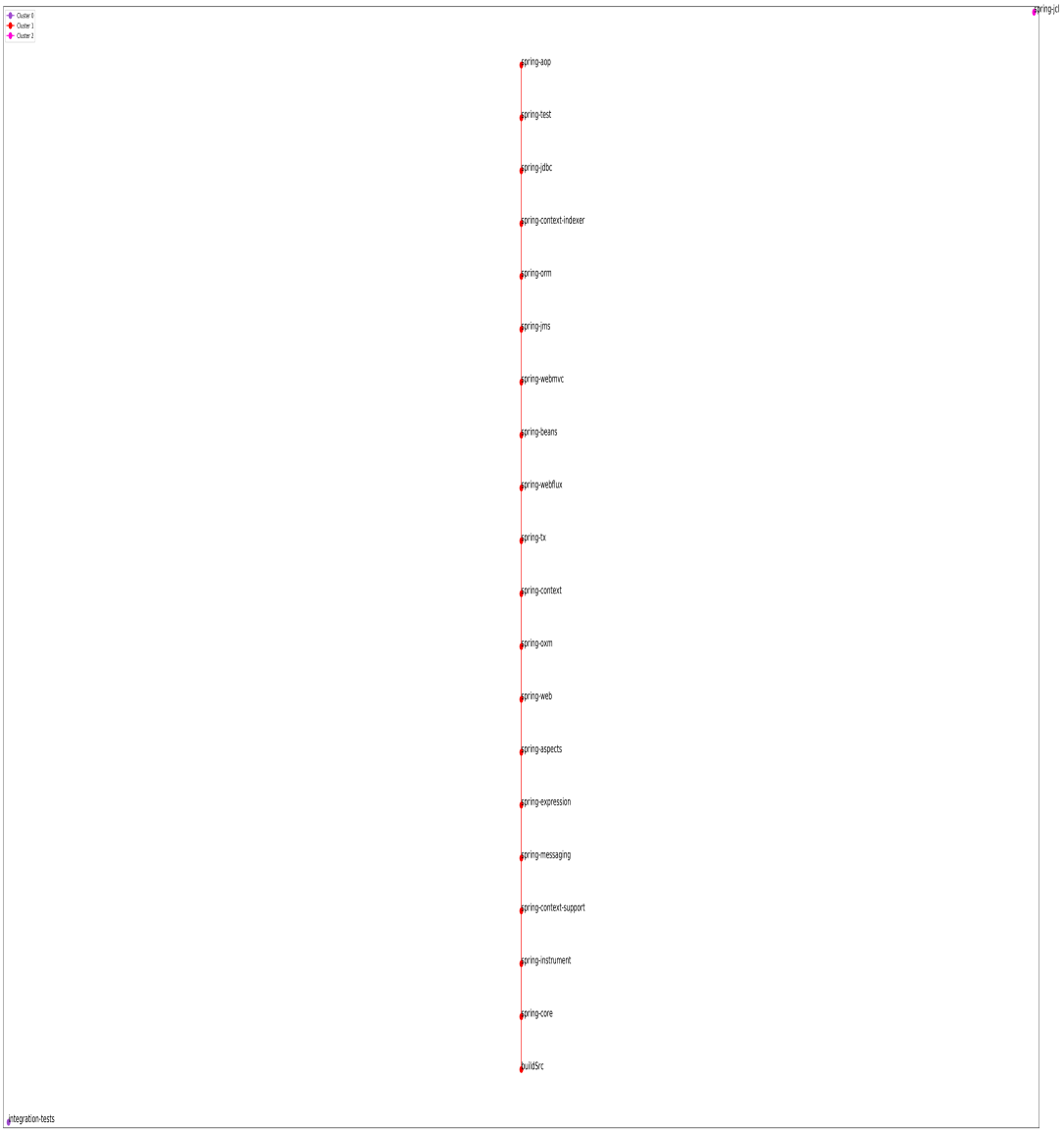
Σχήμα 5.6: Co-change από αρχεία στο Spring Framework



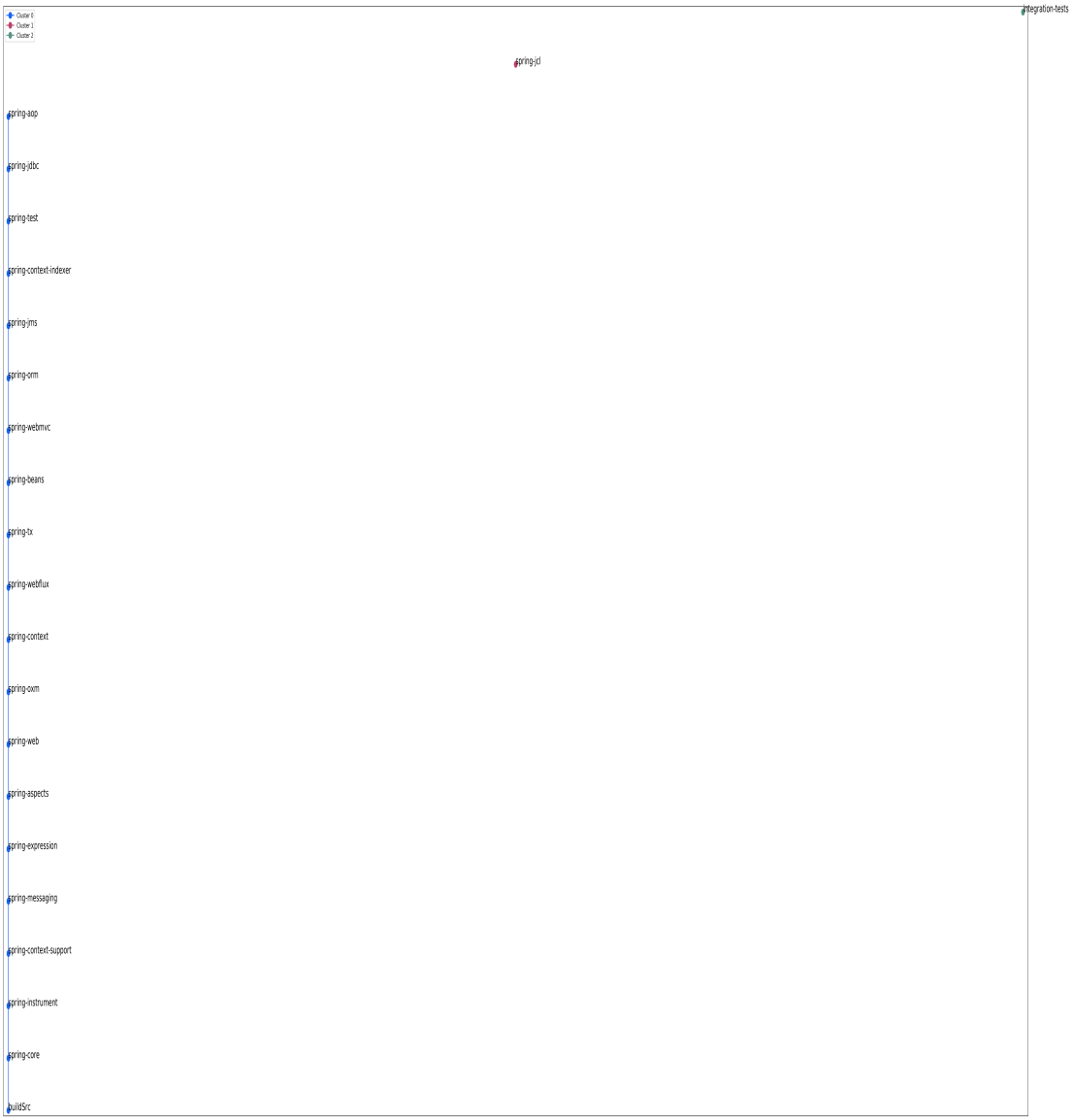
Σχήμα 5.7: Markov clustering στα αρχεία στο Spring Framework



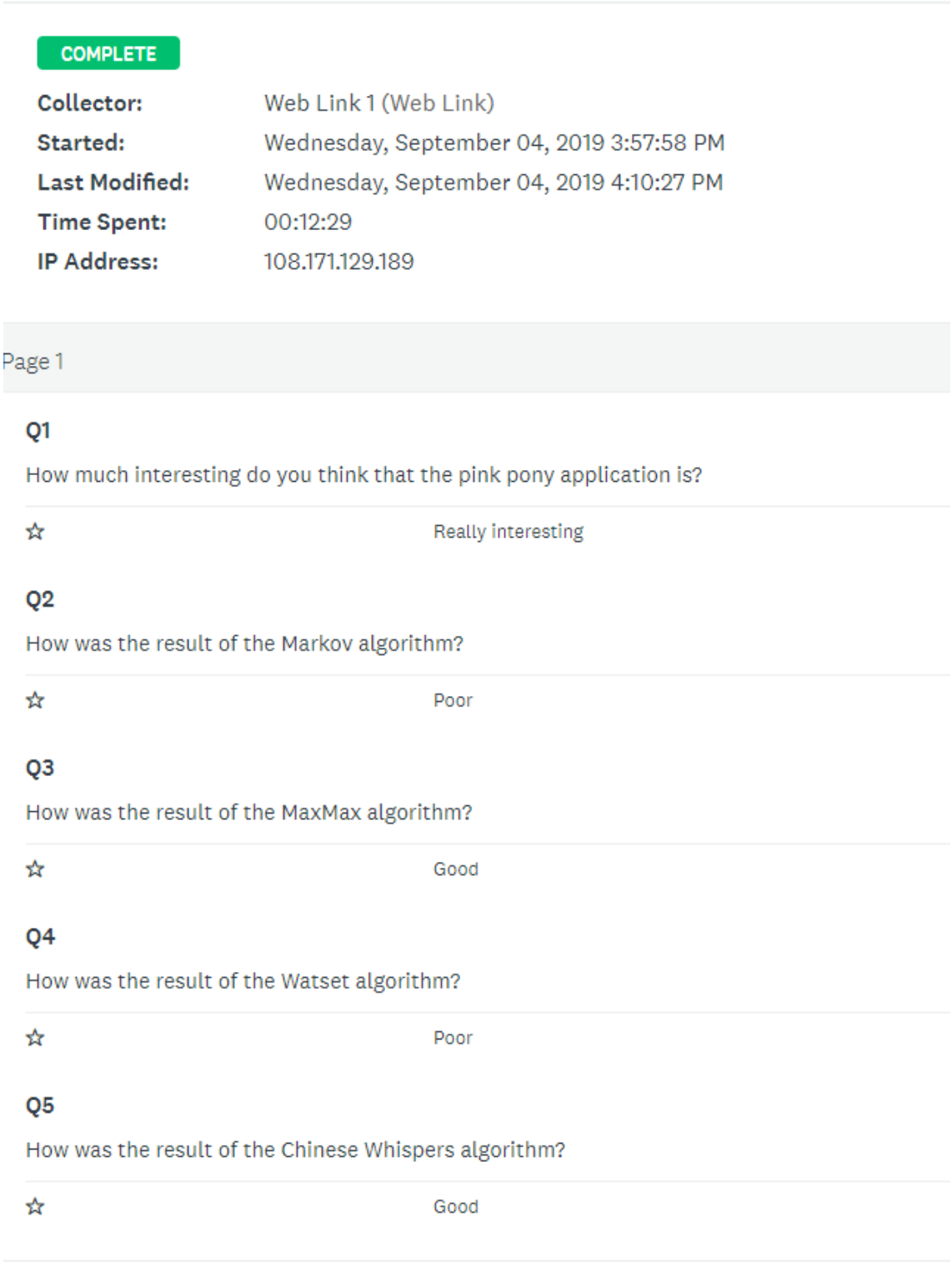
Σχήμα 5.8: MaxMax clustering στα αρχεία στο Spring Framework



Σχήμα 5.9: Chinese clustering στα αρχεία στο Spring Framework



Σχήμα 5.10: Watset clustering στα αρχεία στο Spring Framework



Σχήμα 5.11: Απαντήσεις του ερωτηματολογίου από την ολλανδική εταιρεία

Κεφάλαιο 6

Συμπεράσματα και μελλοντικά πλάνα

6.1 Συμπεράσματα

Συμπερασματικά, αλλαγές σε μικρότερα συστήματα εφαρμογών είναι πιο οικονομικές και ανώδυνες σε σχέση με τις αλλαγές σε μεγαλύτερα υποσυστήματα. Είναι βασική ιδέα της ομαδοποίησης του κώδικα (modularity) η απόκρυψη σημαντικών σχεδιαστικών αποφάσεων που μπορούν να αλλάξουν. Ως μέρος της διαδικασίας, η συσταδοποίηση του κώδικα χωρίζει τον κώδικα σε υποσυστήματα τέτοια ώστε τα υποσυστήματα να είναι όσο πιο ανεξάρτητα γίνεται σεβόμενα την κατανόηση, τις αλλαγές, την επαναχρησιμοποίηση και άλλα κριτήρια. Η καλή ομαδοποίηση του κώδικα σε υποσυστήματα συμβάλει στην συντήρηση, κατανόηση και αλλαγή του κώδικα ευκολότερα. Πλέον τα έργα λογισμικού είναι αρκετά περίπλοκα και δεν υπάρχουν πολλοί εμφανείς δεσμοί μεταξύ των μεμονωμένων στοιχείων. Πολλές φορές είναι δύσκολο να πάρεις αποφάσεις μόνο από την γνώση και την εμπειρία των ενδιαφερομένων. Για αυτό θεωρούνται αρκετά σημαντικά τα εργαλεία αυτά που προτείνουν αλλαγές και ομαδοποιήσεις με τη χρήση των πληροφοριών του παρελθόντος ώστε να μπορούν οι αποφάσεις να βασίζονται πάνω σε αυτές τις πληροφορίες. Η επαλήθευση των αποτελεσμάτων αποτελεί μία αρκετά δύσκολη και μπορεί και όχι τόσο αντικειμενική διαδικασία αφού ή θα πρέπει να γνωρίζεις αρκετά καλά το project που θα πραγματοποιηθεί το πείραμα ή να είσαι σε επικοινωνία με τους ανθρώπους που γνωρίζουν την αρχιτεκτονική του προγράμματος.

6.2 Μελλοντικά πλάνα

Η εφαρμογή Pink pony έχει πολλαπλές προοπτικές που με κάποιες βελτιώσεις και προσθήκες θα μπορούσε να έχει αποτελέσματα που να είναι ανταγωνιστικά με προϊόντα από εταιρείες που ειδικεύονται στο συγκεκριμένο πεδίο. Το επόμενο βήμα στην εφαρμογή είναι η επέκταση της κλίμακας της εφαρμογής από μία απλή εφαρμογή σε ένα σύστημα που κάθε διαφορετικό component εκτελεί ένα διαφορετικό κομμάτι.

6.2.1 Γραφικό περιβάλλον

Ουσιαστικά θα προστεθεί ένα γραφικό περιβάλλον ώστε η εφαρμογή να είναι πιο φιλική στους χρήστες και ιδιαίτερα σε περιπτώσεις που θα χρησιμοποιηθεί και από managers ή ανθρώπους που δεν είναι στενά συνδεδεμένοι με command line εφαρμογές. Ο τύπος της εφαρμογής θα είναι web based και έκτος από τις ομαδοποιήσεις του κώδικα θα εξάγει και κάποιες πληροφορίες στατικής ανάλυσης.Υπάρχει και η δυνατότητά αργότερα να δημιουργηθεί και ένα plug-in σε ένα από τα περιβάλλοντα ανάπτυξης λογισμικού, που θα προτείνει σε πραγματικό χρόνο τις αλλαγές στον κώδικά και θα εμπλουτίζει τη γνώση του απευθείας από το περιβάλλον.

6.2.2 Επαλήθευση των αποτελεσμάτων

Μία πολύ σημαντική προσθήκη στο μέλλον της έρευνας είναι η επαλήθευση των αποτελεσμάτων. Καθώς το πρόβλημα που προσπαθεί να λύσει η παρούσα έρευνα δεν έχει κάποιον εμφανή τρόπο επαλήθευσης. Έχουν πραγματοποιηθεί κάποιες αξιοσέβαστες προσπάθειές όπως η Mojo 2.0 [21] που προσπαθεί να συγκρίνει τις ομαδοποιήσεις με κάποιες ιδανικές που τις έχουν προτείνει οι δημιουργοί και οι συντηρητές των συστημάτων και των εφαρμογών.

Bibliography

[1] F. Brooks, *The Mythical Man-Month*. Addison-Wesley, 1975, 1995.

[2] A. E. Hassan, "The road ahead for mining software repositories," in *2008 Frontiers of Software Maintenance*, Sep. 2008, pp. 48–57.

[3] D. Beyer and A. Noack, "Clustering software artifacts based on frequent common changes," in *13th International Workshop on Program Comprehension (IWPC'05)*, May 2005, pp. 259–268.

[4] L. L. Silva, M. T. Valente, and M. d. A. Maia, "Assessing modularity using co-change clusters," in *Proceedings of the 13th International Conference on Modularity*, ser. MODULARITY '14. New York, NY, USA: ACM, 2014, pp. 49–60. [Online]. Available: http://doi.acm.org/10.1145/2577080.2577086

[5] F. Palomba, G. Bavota, M. Di Penta, R. Oliveto, A. De Lucia, and D. Poshyvanyk, "Detecting bad smells in source code using change history information," in *Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering*. IEEE Press, 2013, pp. 268–278.

[6] M. D'Ambros, M. Lanza, and R. Robbes, "On the relationship between change coupling and software defects," in *2009 16th Working Conference on Reverse Engineering*. IEEE, 2009, pp. 135–144.

[7] S. Breu and T. Zimmermann, "Mining aspects from version history," in *21st IEEE/ACM International Conference on Automated Software Engineering (ASE'06)*, Sep. 2006, pp. 221–230.

[8] V. E. Ball, J.-C. Bureau, R. Nehring, and A. Somwaru, "Agricultural Productivity Revisited," *American Journal of Agricultural Economics*, vol. 79, no. 4, pp.

1045–1063, 11 1997. [Online]. Available: https://doi.org/10.2307/1244263

[9] D.L.Parnas, “On the criteria to be used in decomposing systems into modules,” *Communications of the ACM, 15(12):1053-2018*, 1972.

[10] Wikipedia. Γράφοι. [Online]. Available: https://el.wikipedia.org/wiki/Grafos

[11] S. van Dongen, “A cluster algorithm for graphs.” *Technical Report INSR0010, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam*, 2000.

[12] C. Biemann, “Chinese whispers - an efficient graph clustering algorithm and its application to natural language processing problems,” in *Proceedings of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing*. New York City: Association for Computational Linguistics, Jun. 2006, pp. 73–80. [Online]. Available: https://www.aclweb.org/anthology/W06-3812

[13] D. Ustalov, A. Panchenko, C. Biemann, and S. P. Ponzetto, “Watset: Local-Global Graph Clustering with Applications in Sense and Frame Induction,” *Computational Linguistics*, vol. 45, no. 3, 2019.

[14] Wikipedia. Chinese whispers (clustering method). [Online]. Available: https://en.wikipedia.org/wiki/Chinese_Whispers_(clustering_method)

[15] D. Hope and B. Keller, “Maxmax: A graph-based soft clustering algorithm applied to word sense induction,” vol. 7816, 03 2013, pp. 368–381.

[16] Git. Gitglossary. [Online]. Available: https://git-scm.com/docs/gitglossary

[17] GitHub. Github glossary. [Online]. Available: https://help.github.com/en/articles/github-glossary

[18] Wikipedia. Committer. [Online]. Available: https://en.wikipedia.org/wiki/Committer

[19] S. Negara, M. Vakilian, N. Chen, R. E. Johnson, and D. Dig, “Is it dangerous to use version control histories to study source code evolution?” in *ECOOP 2012 - Object-Oriented Programming*, J. Noble, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 79–103.

[20] Code scene. [Online]. Available: https://codescene.io/

[21] Zhihua Wen and V. Tzerpos, "An effectiveness measure for software clustering algorithms," in *Proceedings. 12th IEEE International Workshop on Program Comprehension, 2004.*, June 2004, pp. 194–203.