Επέκταση και αναμόρφωση εργαλείου οπτικοποίησης παράλληλα εξελισσόμενων χρονοσειρών

Μπατσίλας Χρήστος

Διπλωματική Εργασία

Επιβλέπων: Π. Βασιλειάδης

Ιωάννινα, Οκτώβριος 2022

ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Παναγιώτη Βασιλειάδη για την πολύτιμη βοήθεια και την καθοδήγηση του καθ' όλη τη διάρκεια αυτής της διπλωματικής εργασίας.

10/06/2022

Μπατσίλας Χρήστος

Περίληψη στα ελληνικά

Σκοπός την συγκεκριμένης διπλωματικής εργασίας είναι η επέκταση και αναμόρφωση ενός υπάρχον εργαλείου οπτικοποίησης παράλληλα εξελισσόμενων χρονοσειρών. Το εργαλείο θα αποτελέσει μια εξέλιξη των εργασιών [Giac15] και [Pant21] των φοιτητών Θ. Γιάχο και Ν. Παντελίδη αντίστοιχα. Το εργαλείο αυτό το ονομάζουμε «Πλουτάρχου Βίοι Παράληλλοι», και αυτό που προσφέρει είναι η απεικόνιση της εξέλιξης των πινάκων μιας βάσης δεδομένων σε παράλληλες γραμμές. Αυτό το γίνεται μέσω ενός διαγράμματος που το αποκαλούμε «Διάγραμμα Παράλληλων Ζωών» και αποτελεί έναν δισδιάστατο πίνακα, με γραμμές τις οντότητες της βάσης και στήλες τις χρονικές στιγμές. Στον πίνακα, τα κελιά είναι χρωματισμένα και η ένταση των χρωμάτων αναπαριστούν την δραστηριότητα που είχε η οντότητα για εκείνη την συγκεκριμένη χρονική στιγμή. Το εργαλείο προσφέρει πληθώρα δυνατοτήτων για την διαχείριση του διαγράμματος όπως ταξινόμηση με βάση κάποιων παραμέτρων, λήψη στιγμιότυπου οθόνης, εξαγωγή του διαγράμματος σε μορφή .tsv κ.α.

Η παρούσα διπλωματική διορθώνει το πρόβλημα αναπαράστασης συνόλων δεδομένων με μεγάλο όγκο.

Επίσης, προσθέτει την δυνατότητα παραγωγής μοτίβων από αλγορίθμους οι οποίοι με βάση την θέση των κελιών στον διάγραμμα κατασκευάζουν μοτίβα και τα χαρακτηρίζουν με ένα πρότυπο, αυτό μας έδωσε την δυνατότητα να παρατηρήσουμε φαινόμενα δημιουργίας, διαγραφής και ενημερώσεων πινάκων για την ίδια χρονική στιγμή και να βγουν κάποια συμπεράσματα για εκείνη την χρονική στιγμή.

Επίσης προσφέρεται η δυνατότητα παραγωγής μιας αναφοράς για το μοτίβα που βρέθηκαν σε ένα διάγραμμα. Με βάση αυτό παράγονται αναφορές για πολλά σύνολα δεδομένων, και πραγματοποιείται μια πειραματική διαδικασία η οποία έχεις ως σκοπό να βρεθεί ο παράγοντας που επηρεάζει περισσότερο τον χρόνο υπολογισμού των μοτίβων. Οι παράγοντες που ελέγχθηκαν είναι για κάθε dataset: το σύνολο των μοτίβων που βρέθηκαν, το σύνολο των στηλών του, το σύνολο των γραμμών του και το μέγεθος του.

Λέξεις Κλειδιά: χρονοσειρές, διάγραμμα, οντότητες, μοτίβα, βάσεις δεδομένων

Abstract

The purpose of this specific thesis is the extension and reformation of an existing visualization tool of parallel evolving time series. The tool will be a development of the works [Giac15] and [Pant21] of the students Th. Yacho and N. Pantelidis respectively. We call this tool "Plutarch's Parallel Lives", and what it offers is the visualization of the evolution of the tables of a database in parallel lines. This is done through a diagram we call the "Parallel Lives Diagram" and is a 2D table, with rows the base entities and columns the moments in time. In the table inside the cells inside are colored and the intensity of the colors represent the activity that the entity had for that specific moment in time. The tool offers many capabilities for managing the diagram such as sorting based on some parameters, taking a screenshot, exporting the diagram in .tsv format, etc. The capability of producing patterns that we can distinguish from the cells of the diagram will be added to this diploma.

Keywords: time-series, diagram, entities, patterns, databases

Πίνακας περιεχομένων

Κεφάλαιο 1. Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Στον σύγχρονο κόσμο που τα δεδομένα πολλαπλασιάζονται τα συστήματα και οι εφαρμογές αναβαθμίζονται, οι βάσεις δεδομένων δεν θα μπορούσαν να μείνουν στάσιμες. Έτσι και οι βάσεις δεδομένων εξελίσσονται στην πάροδο του χρόνου, η εξέλιξη αυτή είναι μείζονος σημασίας να μπορεί να καταγραφεί και να αναπαρασταθεί σε ένα εργαλείο, ώστε να μπορεί κάποιος να βγάλει εύκολα συμπεράσματα από αυτήν.

Για παράδειγμα η ιστορικότητα μιας βάσης δεδομένων και η πληροφορία που προσφέρει μπορεί να αποδειχθεί πολύ χρήσιμη μελλοντικά και να αποτρέψει να γίνουν λάθη που είχαν γίνει στο παρελθόν.

Το 2015 στο Πανεπιστήμιο Ιωαννίνων κατασκευάστηκε από τον Θ. Γιάχο κατά την διάρκεια του MSc του ένα εργαλείο με την ονομασία «Πλουτάρχου Βίοι Παράλληλοι»[Giac15] το οποίο απεικονίζει την εξέλιξη των πινάκων μιας βάσης δεδομένων σε παράλληλες γραμμές. Ο τρόπος αναπαραστάσεις αυτών των δεδομένων είναι ένα διάγραμμα(PLD) που οπτικοποιεί τις παράλληλα εξελισσόμενες χρονοσειρές, έχοντας για γραμμές τους πίνακες της βάσης δεδομένων και για γραμμές τις χρονικές στιγμές.

Το 2021 ο Ν. Παντελίδης [Pant21] πήρε αυτό το εργαλείο και το εξέλιξε, δίνοντας του την δυνατότητα να αναπαραστήσει οποιοδήποτε σύνολο δεδομένων όπως δεδομένα πληθυσμού, πίνακες σε σχεσιακές βάσεις δεδομένων κ.α. Επιπλέον προστέθηκαν πολλά χαρακτηριστικά για την καλύτερη ανάλυση των δεδομένων.

Η δικιά μας συνεισφορά στο συγκεκριμένο εργαλείο είναι η εξής :

- Στην προηγούμενη έκδοση του εργαλείου υπάρχει η αδυναμία να αναπαρασταθούν μεγάλα σύνολα δεδομένων, καθώς η εφαρμογή σε τέτοιες περιπτώσεις «παγώνει». Το πρόβλημα εντοπίζεται κατά την αναπαράσταση του διαγράμματος. Η βιβλιοθήκη που είναι υπεύθυνη για την αλληλεπίδραση του

χρήστη με την εφαρμογή, είναι η JavaFX, και το αντικείμενο της TableView αναπαριστά το διάγραμμα μας ως ένα δισδιάστατο πίνακα, παρόλα αυτά το συγκεκριμένο αντικείμενο έχει την αδυναμία να διαχειριστή μεγάλο όγκο δεδομένων. Στην δικιά μας προσέγγιση, αντικαταστήσαμε την βιβλιοθήκη JavaFX με την Java Swing και το αντικείμενο TableView με ένα JTable, αυτό είχε ως αποτέλεσμα η εφαρμογή να μπορεί να αναπαράγει διαγράμματα με μεγάλο όγκο δεδομένων.

- Θα προσθέσουμε αλγόριθμους που θα βρίσκουν μοτίβα από τα κελιά του διαγράμματος και θα τα τοποθετούν σε ένα πρότυπο. Επιπλέον, θα προσθέσουμε την δυνατότητα παραγωγής μίας αναφοράς αυτών των μοτίβων στο διάγραμμα, όπου θα μπορεί να χρησιμοποιηθεί για κάθε σύνολο δεδομένων.
- Προστέθηκε η δυνατότητα εύρεσης μοτίβων από την θέση των κελιών στο διάγραμμα. Η εύρεση γίνεται μέσω αλγορίθμων οι οποίοι με βάση την κατάσταση που έχει το κάθε κελί(κελί γέννησης, διαγραφής, ενημέρωσης οντότητας) «χτίζει» τα αντίστοιχα μοτίβα και τα χαρακτηρίζει με ένα πρότυπο. Τα πρότυπα αυτά είναι τα εξής: «Στήλη πολλαπλών γεννήσεων», «Στήλη πολλαπλών διαγραφών», «Στήλη πολλαπλών ενημερώσεων» και «Σκάλα πολλαπλών γεννήσεων». Στα 3 πρώτα πρότυπα βελτιστοποιήσαμε τον αλγόριθμο μας, καθώς η διαδικασία υπολογισμού τους ήταν ίδια, με σκοπό σε μία αναζήτηση του διαγράμματος να τα υπολογίζει όλα.
- Από τα μοτίβα που βρίσκει η εφαρμογή, δίνεται η δυνατότητα παραγωγής μιας αναφοράς που περιέχει γενικές πληροφορίες του dataset που χρησιμοποιούμε, γενικές πληροφορίες για τα μοτίβα που βρέθηκαν και το κάθε μοτίβο ξεχωριστά μαζί με τα κελία που συμμετέχουν σε αυτό.
- Μέσω μιας πειραματικής διαδικασίας διερευνήσαμε τον χρόνο υπολογισμού των μοτίβων για το κάθε dataset, και ποιος παράγοντας είναι αυτή που το επηρεάζει. Συναρτήσει του χρόνου υπολογισμού οι παράγοντες που ελέγξαμε είναι οι εξής:
    - Σύνολο μοτίβων που βρέθηκαν
    - Σύνολο στηλών του dataset
    - Σύνολο γραμμών του dataset
    - Μέγεθος του dataset

  Επειδή δεν υπάρχει μεγάλο σύνολο δεδομένων, δεν μπορούσαμε να βγάλουμε ξεκάθαρα αποτελέσματα, αλλά ανακαλύψαμε ότι οι παράγοντες που επηρεάζουν περισσότερο τον χρόνο υπολογισμού είναι ο αριθμός των στηλών και το μέγεθος του dataset.

1.2 Οργάνωση του τόμου

Στα επόμενα κεφάλαια θα αναλύσουμε την αναλυτικότερα την υλοποίηση της διπλωματικής. Πιο συγκεκριμένα:

Στο δεύτερο κεφάλαιο, γίνεται μια ανάλυση του στόχου, και δίνεται μια πιο σαφής εικόνα, της παρούσας διπλωματικής, καθώς επίσης γίνεται και αναφορά σε σχετικές με το θέμα εργασίες.

Στο τρίτο κεφάλαιο ορίζονται τα προβλήματα που καλούμαστε να αντιμετωπίσουμε στην διπλωματική μας, καθώς και οι λύσεις που υλοποιήθηκαν. Για τις λύσεις περιγράφεται ο σχεδιασμός και η αρχιτεκτονική του συστήματος μέσω διαγραμμάτων UML όπως επίσης γίνεται αναφορά και στα τεστ που δημιουργήθηκαν για τον έλεγχο του. Τέλος, δίνονται πληροφορίες των τεχνικών χαρακτηριστικών, των βημάτων εγκατάστασης καθώς επίσης και μελλοντικές επεκτάσεις που μπορούν να γίνουν.

Στο τέταρτο κεφάλαιο παρουσιάζεται η πειραματική διαδικασία που ακολουθήσαμε και τα αποτελέσματα της.

Στο πέμπτο και τελευταίο κεφάλαιο υπάρχει μια περίληψη της διπλωματικής μας, τα συμπεράσματα μας και κάποιες ιδέες για μελλοντικές επεκτάσεις του συστήματος.

Κεφάλαιο 2. Περιγραφή Θέματος

2.1 Στόχος της εργασίας

Στόχος της παρούσας Διπλωματικής Εργασίας είναι η επέκταση και αναμόρφωση του εργαλείου «Πλουτάρχου Βίοι Παράλληλοι». Ένα εργαλείο οπτικοποίησης παράλληλα εξελισσόμενων χρονοσειρών το οποίο διευκολύνει την ταυτόχρονη οπτική αναπαράσταση της ζωής των ομότιμων οντοτήτων, που συν-εξελίσσονται στο ίδιο χρονικό πλαίσιο.

Συγκεκριμένα, οι βελτιώσεις και προσθήκες του συστήματος οργανώνονται ως εξής:

1. Αλλαγή στο αντικείμενο απεικόνισης των δεδομένων μας, έτσι ώστε να μπορεί -με ένα απλό δισδιάστατο πίνακα- να χειριστεί μεγάλο αριθμό γραμμών και στηλών
2. Ορισμός μοτίβων με βάση την απεικόνιση των δεδομένων μας στο πίνακα, και διαχωρισμός τους με βάση κάποιο από τα πρότυπα που έχουμε ορίσει.
3. Κατασκευή μιας αναφοράς που περιγράφει τα ευρήματα από τα μοτίβα που ανακαλύφθηκαν.

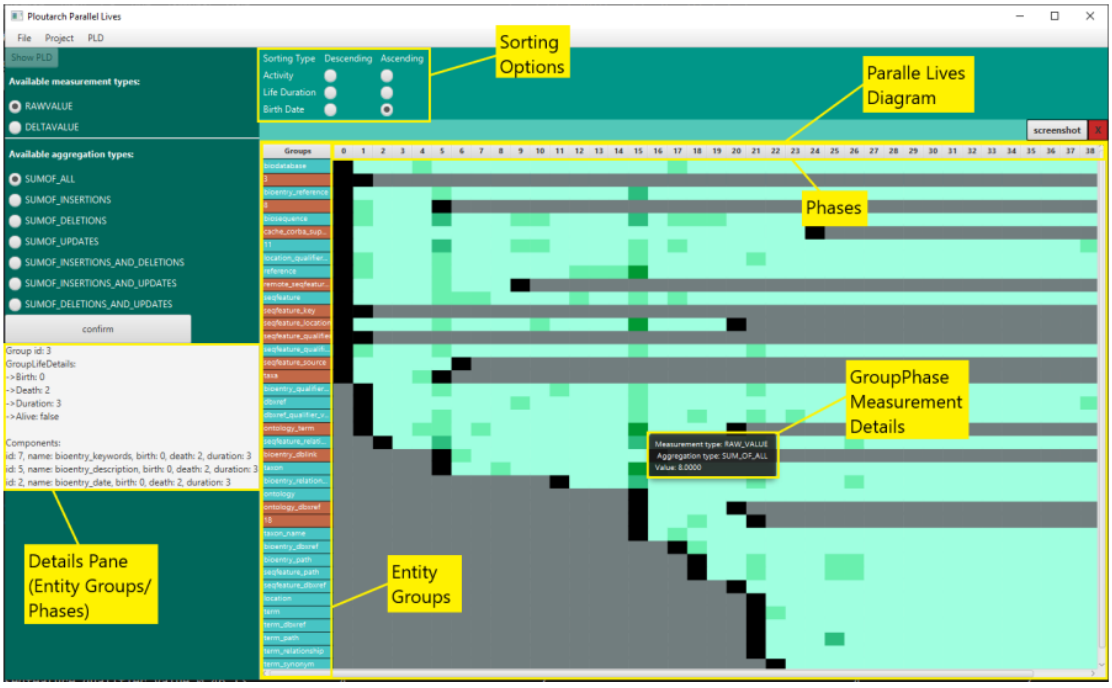
2.2 Σχετικές εργασίες και τεχνολογίες

Σε αυτή την ενότητα περιγράφουμε το εργαλείο «Πλουτάρχου Βίοι Παράλληλοι» . Είναι το εργαλείο που θέλουμε να αναμορφώσουμε και να επεκτείνουμε, ένα εργαλείο (από τον Θ. Γιάχο [Giac15] και μετέπειτα το Ν. Παντελίδη [Pant21]) το οποίο απεικονίζει την εξέλιξη των πινάκων μιας βάσης δεδομένων σε παράλληλες γραμμές. Επιπλέον θα περιγράψουμε και άλλα εργαλεία που βοήθησαν στην υλοποίηση των λειτουργιών.

2.2.1 Πλουτάρχου Βίοι Παράλληλοι

Πρόκειται για ένα εργαλείο το οποίο απεικονίζει την εξέλιξη των πινάκων μιας βάσης δεδομένων σε παράλληλες γραμμές. Η απεικόνιση γίνεται σε έναν δισδιάστατο πίνακα,

όπου, όπου κάθε πίνακας της βάσης(οντότητα) αντιστοιχεί σε μία γραμμή και κάθε χρονική στιγμή αντιστοιχεί σε μία στήλη. Στο παρακάτω διάγραμμα[Εικόνα 1], εύκολα αντικρίζει κανείς πως κάθε κελί είναι χρωματισμένο έτσι ώστε να αναπαριστά την δραστηριότητα της οντότητας εκείνη την χρονική στιγμή. Η πρώτη έκδοση του εργαλείου δημιουργήθηκε από τον Θ. Γιάχο. Σε αυτή την πρώτη έκδοση ο Γιάχος χρησιμοποίησε δυο αλγορίθμους ομαδοποίησης. Έναν για την ομαδοποίηση χρονικών στιγμών και έναν για την ομαδοποίηση οντοτήτων και οι δυο ομαδοποιήσεις γίνονται με βάση την ομοιότητα των στοιχείων που αναφέραμε. Δημιουργώντας μια σύνοψη των δεδομένων, κατάφερε να αναπαραστήσει τα δεδομένα στην οθόνη του εργαλείου χωρίς να χάσει σημαντικό όγκο πληροφοριών. Στη πορεία ο Παντελίδης δημιούργησε από την αρχή μια πιο γενικευμένη έκδοση του εργαλείου.

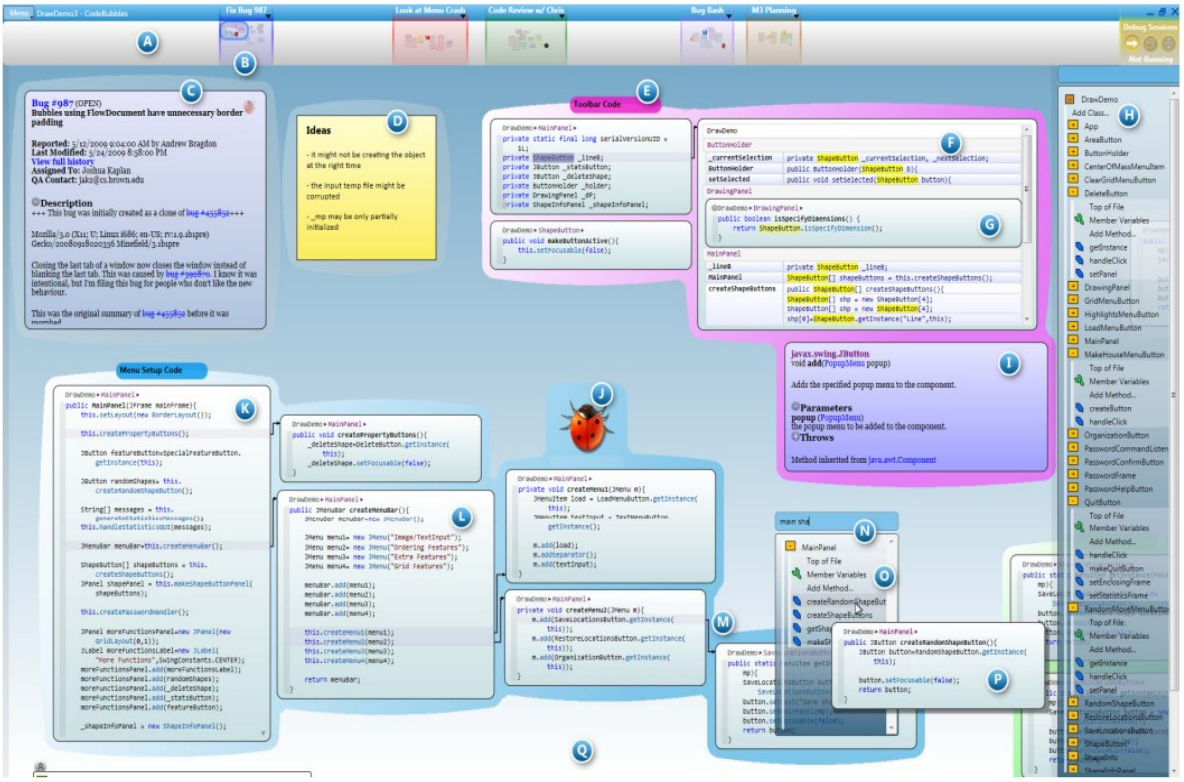


Εικόνα 1: Πλουτάρχου Βίοι Παράλληλοι

2.2.2 Code Bubbles

Το Code Bubbles [BrZK10] είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) που σχεδιάστηκε για να απλοποιήσει την ανάπτυξη κώδικα, διευκολύνοντας την εμφάνιση του κώδικα και να κάνοντας πιο απλή την πλοήγηση του, μέσω της δυνατότητας ο προγραμματιστής να ορίσει και να χρησιμοποιήσει σετ εργασίας, τα οποία είναι κλάσεις, εγχειρίδια, σημειώσεις κ.ά. Αυτό επιτυγχάνετε παρέχοντας συμπαγείς

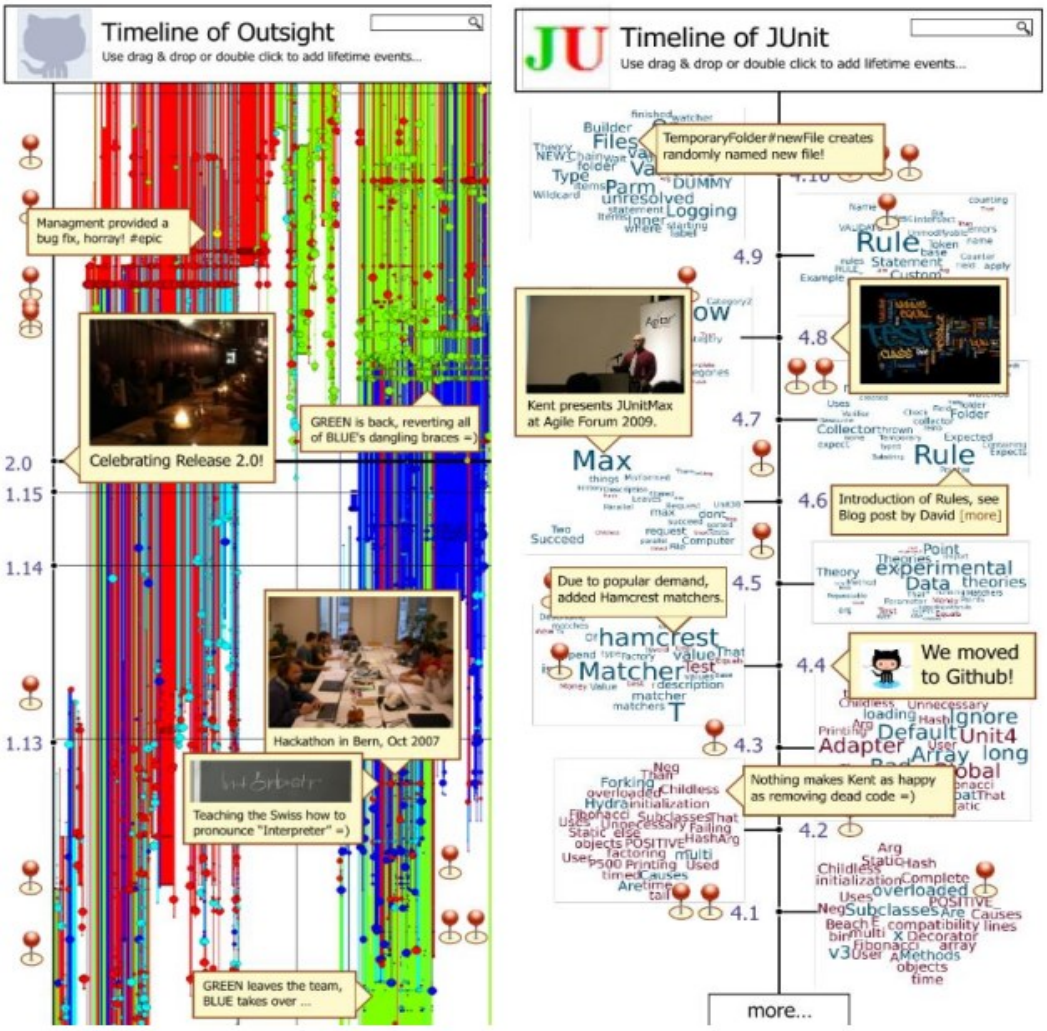
προβολές του κώδικα που εμφανίζονται σε «Bubbles». Σε μία τέτοια περιοχή ο προγραμματιστής μπορεί να έχει πολλά σετ εργασίας του και να τα διαρρυθμίσει. Τέλος το Code Bubbles προσφέρει τις κλασικές λειτουργίες των άλλων ολοκληρωμένων περιβαλλόντων ανάπτυξης όπως σύνταξη κώδικα, debugging και testing.



Εικόνα 2: [BrZK10] Code Bubbles

2.2.3 CodeTimeline

Το CodeTimeline [KuSt12] είναι ένα εργαλείο που θέλει να οπτικοποιήσει την ιστορία και την εξέλιξη ενός συστήματος λογισμικού. Για να γίνει αυτό οι συγγραφείς προτείνουν ένα μοντέλο «Πίνακα ανακοινώσεων» στον οποίο απεικονίζονται δύο διαφορετικές τεχνικές οπτικοποίησης της ιστορίας του συστήματος. Η μία τεχνική είναι η «Collaboration View» που εμφανίζει τους προγραμματιστές και την έκδοση του συστήματος στην οποία έχουν συνεισφέρει. Η δεύτερη τεχνική ονομάζεται «Sourcecloud Flow View» που εμφανίζει τις αλλαγές που έχουν γίνει στο κομμάτι του κώδικα του συστήματος.

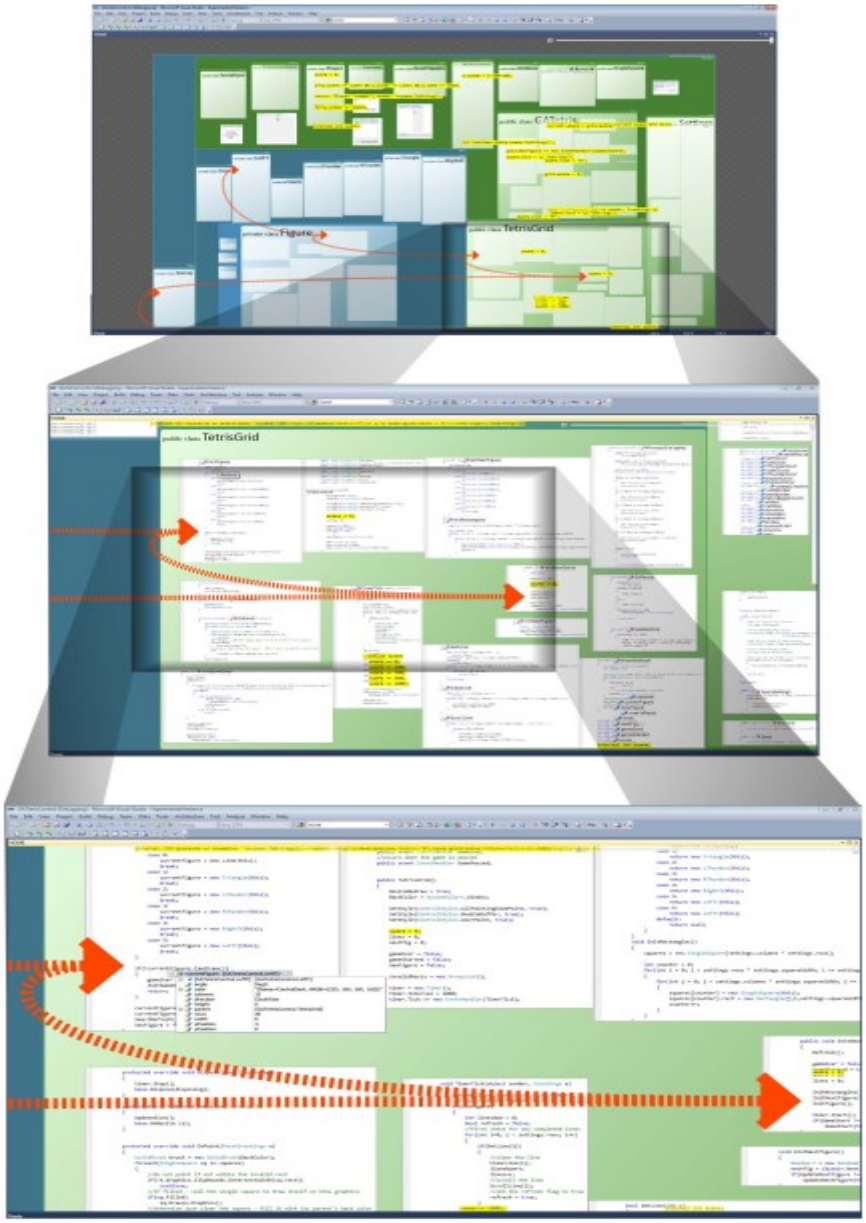


Εικόνα 3: [KuSt10] CodeTimeline. Αριστερά το «Collaboration View» και δεξιά το «Sourcecloud Flow View»

2.2.4 Code Canvas

Το Code Canvas [DeRo10] πρόκειται για ένα περιβάλλον διεπαφής χρήστη. Σε αυτό το περιβάλλον οι συγγραφείς δοκιμάζουν ένα νέο μοντέλο το οποίο αντί να βασίζετε σε καρτέλες και παράθυρα για να πλοηγηθεί ο χρήστης στον κώδικα, το Code Canvas τοποθετεί όλα τα αρχεία ενός project (πχ αρχεία κώδικα, σχέδια, εικόνες κ.ά.) σε ένα, άπειρο από άποψη χωρητικότητας, κεντρικό παράθυρο στο οποίο υπάρχει η δυνατότητα για μεγέθυνση και σμίκρυνση έτσι ώστε να γίνεται πιο άμεση η επισκόπηση και η επεξεργασία των αρχείων. Αυτός ο σχεδιασμός έχει σκοπό να μειώσει

τον αποπροσανατολισμό του προγραμματιστή τις στιγμές επεξεργασίας κώδικα καθώς και να επωφεληθεί κάνοντας ταυτόχρονη εκτέλεση πολλών ενεργειών.



Εικόνα 4: [DeRo10] Code Canvas

2.3 Ανάλυση απαιτήσεων

Το παρόν εργαλείο θα αναπτύξει και θα προσφέρει στο ήδη υπάρχον εργαλείο τις παρακάτω δυνατότητες :

1. Ορισμός μοτίβων σε δισδιάστατα παράθυρα του διαγράμματος παράλληλων ζωών, αν σε αυτά τα παράθυρα υπάρχει όμοια λογική μεταξύ των οντοτήτων κατά την εξέλιξη τους
2. Κατασκευή διαγράμματος που θα περιέχει χρωματισμένα τα κελιά των μοτίβων που βρέθηκαν.
3. Αφού προδιαγράψουμε τα μοτίβα που βρήκαμε, να παράγεται μια αναφορά που να περιγράφει τα μοτίβα για την ιστορία του πληθυσμού των παράλληλων ζωών.

Use Case: Εμφάνιση διαγράμματος με ειδικά χρωματισμένα τα κελιά που συμμετέχουν στα μοτίβα
ID: UC1
Actors: Χρήστης
Preconditions: 1. Έχει φορτωθεί στο σύστημα ένα σύνολο δεδομένων
Flow of events: 1. Η περίπτωση ξεκινά όταν ο χρήστης επιλέγει από το μενού το πλήκτρο «Patterns» και έπειτα το κουμπί «Show patterns» 2. Στην συνέχεια ο χρήστης επιλέγει το πρότυπο των μοτίβων που θέλει να δει 3. Το σύστημα εμφανίζει σε ξεχωριστό παράθυρο το διάγραμμα με ειδικά χρωματισμένα τα κελιά που συμμετέχουν στα μοτίβα
Postconditions: Το σύστημα δείχνει τα δεδομένα των μοτίβων χρησιμοποιώντας το PLD

Use Case: Δημιουργία αναφοράς των μοτίβων για την ιστορία των παράλληλων ζωών
ID: UC2
Actors: Χρήστης
Preconditions: 1. Έχει φορτωθεί στο σύστημα ένα σύνολο δεδομένων 2. Έχει φορτωθεί στο σύστημα το διάγραμμά με τα μοτίβα
Flow of events: 1. Η περίπτωση ξεκινά όταν ο χρήστης επιλέγει από το μενού το πλήκτρο «Patterns» και έπειτα το κουμπί «Save report of loaded patterns» 2. Το σύστημα ζητάει από τον χρήστη την τοποθεσία και το όνομα του αρχείου(.txt) που θέλει να αποθηκευτεί
Postconditions: Το σύστημα αποθηκεύει την αναφορά στο path και με το όνομα που του έχει ορίσει ο χρήστης

Κεφάλαιο 3. Σχεδίαση & Υλοποίηση

3.1 Ορισμός προβλήματος και αλγόριθμοι επίλυσης

Για την ολοκλήρωση της παρούσας διπλωματικής θα πρέπει να γίνουν οι παρακάτω αλλαγές/υλοποιήσεις:

1. Το πρώτο πρόβλημα που έπρεπε να διορθωθεί, κατά την αναμόρφωση του εργαλείου, ήταν πως σε περιπτώσεις με dataset που περιείχαν μεγάλο όγκο δεδομένων το εργαλείο κόλλαγε και αναγκαζόταν να τερματίσει. Αυτό το πρόβλημα προκαλείται από την υλοποίηση του JavaFX TableView που αντιμετωπίζει θέμα ως προς την οπτικοποίηση για μεγάλο πλήθος δεδομένων. Στο Repository του JavaFX στο Github, ανακαλύψαμε ότι πρόκειται για ένα διαδεδομένο πρόβλημα (Issue: [https://github.com/javafxports/openjdk-jfx/issues/409]) και πως άλλοι χρήστες έχουν αντιμετωπίσει παρόμοιο ζήτημα. Αυτό που προκύπτει από το issue στο GitHub είναι πως το TableView έχει περιορισμό στην αναπαράσταση των αριθμών των στηλών.
2. Το επόμενο ζητούμενο που καλούμαστε να υλοποιήσουμε είναι η κατασκευή αλγορίθμων με σκοπό την εύρεση μοτίβων, και την απεικόνιση τους σε ένα διάγραμμα, από τα κελιά που βρίσκονται σε κοινή κατάσταση(κελιά γέννησης, ενημέρωσης και διαγραφής της οντότητας) στην ίδια χρονική στιγμή, καθώς και μοτίβα που η απεικόνιση τους είναι ένα σχήμα, όπως σκάλα πολλαπλών γεννήσεων.
3. Αφού ολοκληρωθεί η διαδικασία της εύρεση των μοτίβων θα πρέπει να υπάρχει η δυνατότητα παράγωγής μιας αναφοράς που θα περιέχει ένα σύνολο γενικών πληροφορίων για το dataset και για τα μοτίβα που βρέθηκαν, καθώς και τα δεδομένα του κάθε μοτίβου ξεχωριστά.

Our GitHub Repository: https://github.com/DAINTINESS-Group/PlutarchParallelLives

3.2 Σχεδίαση και αρχιτεκτονική λογισμικού

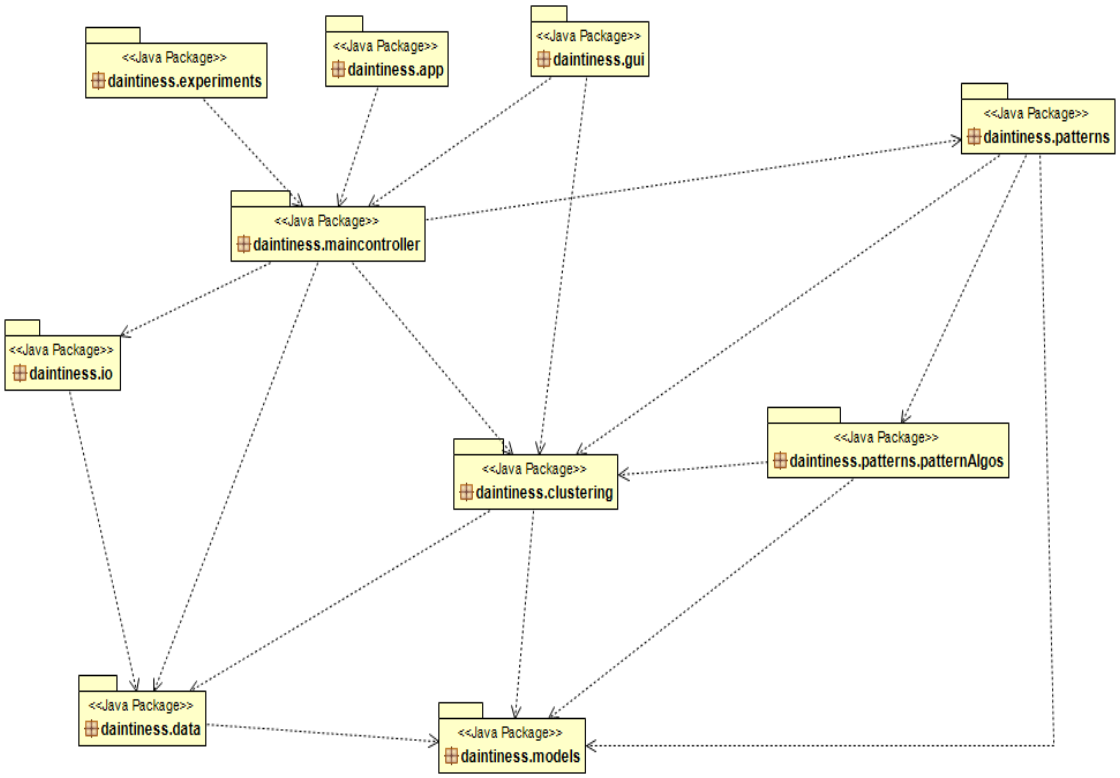
3.2.1 Νέα αρχιτεκτονική

Στο εργαλείο υπάρχουν 9 κύρια πακέτα, τα οποία είναι υπεύθυνα για τις εξής λειτουργίες:

- **Experiments**: Πακέτο που χρησιμοποιείται για την μέτρηση χρόνων της εφαρμογής.
- **App**: Πακέτο που δημιουργεί και εξάγει την ενδιάμεση αναπαράσταση του δοθέντος συνόλου δεδομένων.
- **GUI**: Αυτό το πακέτο είναι υπεύθυνο για την αλληλεπίδραση του χρήστη με την εφαρμογή.
- **MainController**: Πακέτο στο οποίο βρίσκεται το API που κάνει τις κλήσεις για την λειτουργικότητα της εφαρμογής.
- **IO**: Πακέτο υπεύθυνο για την διαχείριση των αρχείων.
- **Clustering**: Πακέτο υπεύθυνο για την λειτουργικότητα των ομαδοποιήσεων των δεδομένων που χρησιμοποιεί η εφαρμογή.
- **Data**: Πακέτο στο οποίο περιέχεται η διαχείριση της ενδιάμεσης αναπαράστασης.
- **Models**: Πακέτο στο οποίο βρίσκονται τα αντικείμενα που χρειάζεται η εφαρμογή.
- **Utilities**: Πακέτο όπου σε αυτό περιέχονται Enums που χρησιμοποιούνται από όλα τα υπόλοιπα πακέτα.

Στην νέα αρχιτεκτονική του εργαλείου προστέθηκαν δύο νέα πακέτα

- **Patterns**: Πακέτο υπεύθυνο για την επικοινωνία των αλγορίθμων με την υπόλοιπη εφαρμογή και την κατασκευή της αναφοράς εύρεσης μοτίβων.
- **PatternAlgos**: Πακέτο στο οποίο βρίσκονται οι αλγόριθμοι που περιέχουν όλη την λογική της εύρεσης των μοτίβων και ο χαρακτηρισμός τους με ένα πρότυπο.



Εικόνα 5: Διάγραμμα πακέτων

3.2.2 Refactoring του υπάρχοντος front end

Σε αυτό το κομμάτι θα αναλύσουμε της διαφοροποιήσεις που πρέπει να γίνουν για να βελτιωθεί το εργαλείο.

3.2.2.1 Προηγούμενη έκδοση αρχιτεκτονικής λογισμικού στο πακέτο GUI

PLDiagram: Εδώ αρχικοποιείται το διάγραμμα μας και πιο συγκεκριμένα το Swing JTable το αντικείμενο που χρησιμοποιούμε για την αναπαράσταση των δεδομένων των παράλληλων ζωών, καθώς και ο χρωματισμός των κελιών του με βάση την πληροφορία που έχουμε.

Controller: Σε αυτή την κλάση γίνεται η επικοινωνία του Front-End με το API της εφαρμογής (MainController) και κατ' επέκταση με τις λειτουργίες που χρειαζόμαστε, καθώς και η άντληση της πληροφορίας που χρησιμοποιεί το TableView.

ClusteringProfileDialogController: Σε αυτή την κλάση γίνεται η διαχείριση του αρχείου ClusteringProfileDialog.fxml όπου με βάση τις τιμές που ο χρήστης περνάει φτιάχνεται ένα profile για το πως θα φαίνεται το TableView.

GuiCondition: Έδω υπάρχουν κάποια Enums που χρησιμοποιεί ο Controller.

Το υπάρχον πρόβλημα με την συγκεκριμένη αρχιτεκτονική είναι, όπως αναφέρθηκε και πιο πάνω, η δυσκολία διαχείρισης μεγάλου όγκου δεδομένων από το JavaFX TableView.

Εικόνα 6: Διάγραμμα κλάσεων για το πακέτο GUI με την παλιά αρχιτεκτονική

3.2.2.2 Νέα αρχιτεκτονική λογισμικού στο πακέτο GUI

Το πρόβλημα που είχε η προηγούμενη αρχιτεκτονική αντιμετωπίστηκε με την μετατροπή του JavaFX TableView σε Swing JTable που καταφέρνει να οπτικοποιεί χωρίς πρόβλημα μεγάλα dataset. Η πρώτη προσέγγιση ήταν απλά να αντικατασταθεί το TableView από ένα JTable και τα υπόλοιπα να παραμείνουν ως είχαν. Με αυτό το σενάριο η διαχείριση του κώδικα έγινε δυσκολότερη καθώς υπήρχαν κομμάτια κώδικα σε JavaFX και σε Swing, επιπλέον χρησιμοποιώντας τις δύο βιβλιοθήκες παράλληλα, η εφαρμογή μπερδευόταν με τα threads και δημιουργούσε θέματα στην εμφάνιση του table. Έτσι έγινε ένα ολοκληρωτικό refactor στο package του Gui, καθώς όλο το Front-End μετατράπηκε από JavaFX σε αποκλειστικά Swing.

Στην προηγούμενη εκδοχή του κώδικα, ο χρήστης για να μπορέσει να οπτικοποιήσει το τα δεδομένα μέσω του TableView, έτρεχε ένα από τα εξής δύο σενάρια:

- Φόρτωση του αρχείου -> Επιλογή του επιθυμητού προφίλ ομαδοποίηση -> Πάτημα κουμπιού «Show PLD» πάνω από της διαθέσιμες ομαδοποιήσεις δεδομένων
- Φόρτωση του αρχείου -> Επιλογή του επιθυμητού προφίλ ομαδοποίηση -> Κουμπί στο menu «PLD» -> Πάτημα κουμπιού «Show PLD»

Και στις δύο περιπτώσεις πλέον εμφανίζεται το αντικείμενο του JTable. Επίσης όποιες λειτουργίες περιείχε η προηγούμενη δομή (sort, αποθήκευση του πίνακα σε εικόνα, αλλαγή στην ομαδοποίηση των δεδομένων, εξαγωγή–εισαγωγή project, zoom in/zoom out) αντικαταστάθηκαν εξίσου. Οι νέες κλάσεις που προστέθηκαν είναι οι εξής:

PLDiagramSwing: Σε αυτή την κλάση χτίζεται το Swing JTable από τα δεδομένα που λαμβάνει.

PaintCellTableRenderer: Αυτή η κλάση κληρονομεί από την κλάση DefaultTableCellRenderer και είναι υπεύθυνη για τον χρωματισμό των κελιών του JTable με βάση την πληροφορία που έχουμε.

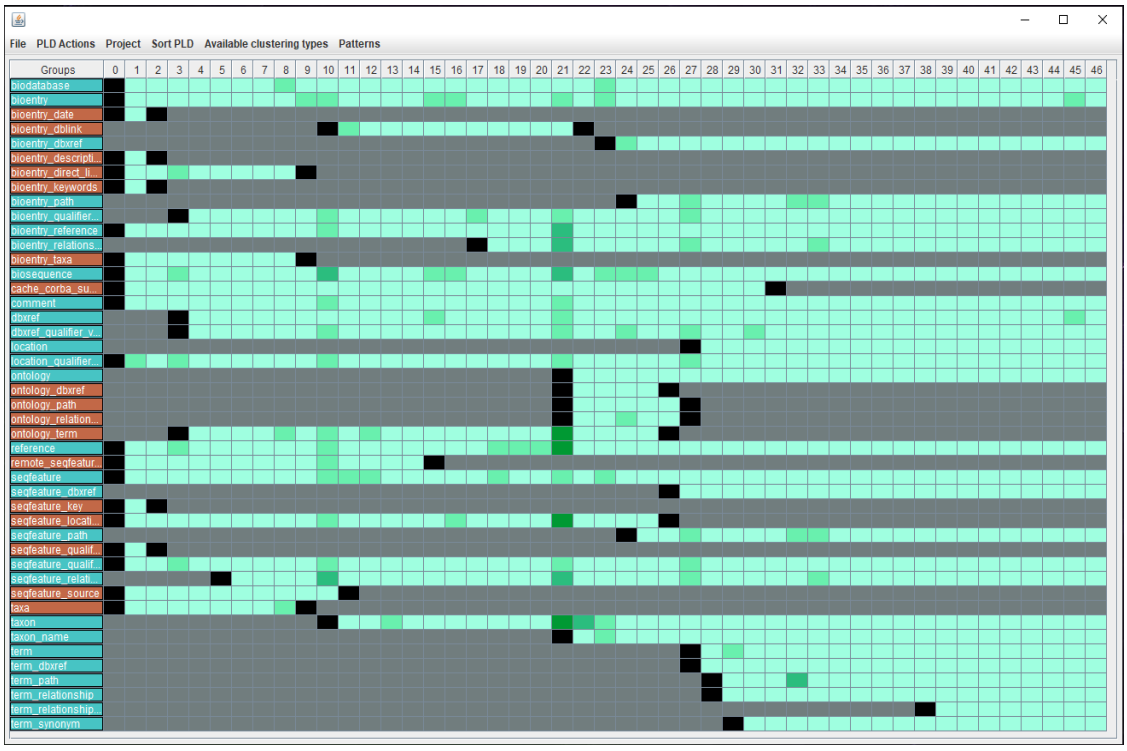
MainAppSwing: Αποτελεί την κλάση που κάνει το User Interaction της εφαρμογής με τον χρήστη. Είναι ένα JFrame το οποίο έχει ένα menu με λειτουργίες και ένα χώρο στον οποίο εμφανίζεται το table(Εικόνα 7).

ClusteringProfileJDialog: Πρόκειται για ένα JDialog Pop-Up που ο χρήστης μπορεί να κάνει cluster τα δεδομένα (Εικόνα 8). Με βάση τις τιμές που δίνει φτιάχνει ένα profile και από αυτό παράγει την μορφή που θα έχουν τα δεδομένα.

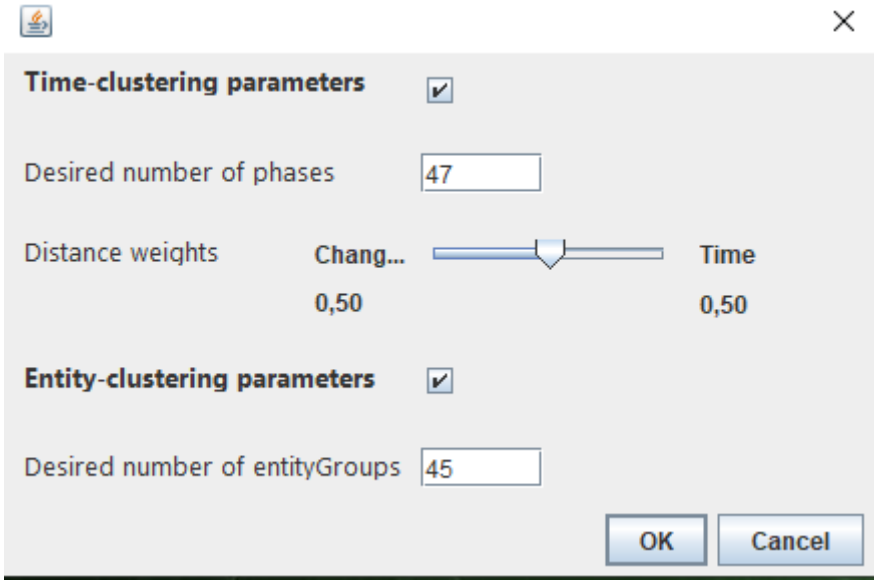
ControllerSwing: Σε αυτή την κλάση γίνεται η όλη διαχείριση του Front-End της εφαρμογής.

Η κλάση αυτή είναι υπεύθυνη για την επικοινωνία του MainAppSwing με το Back-End κομμάτι της εφαρμογής.

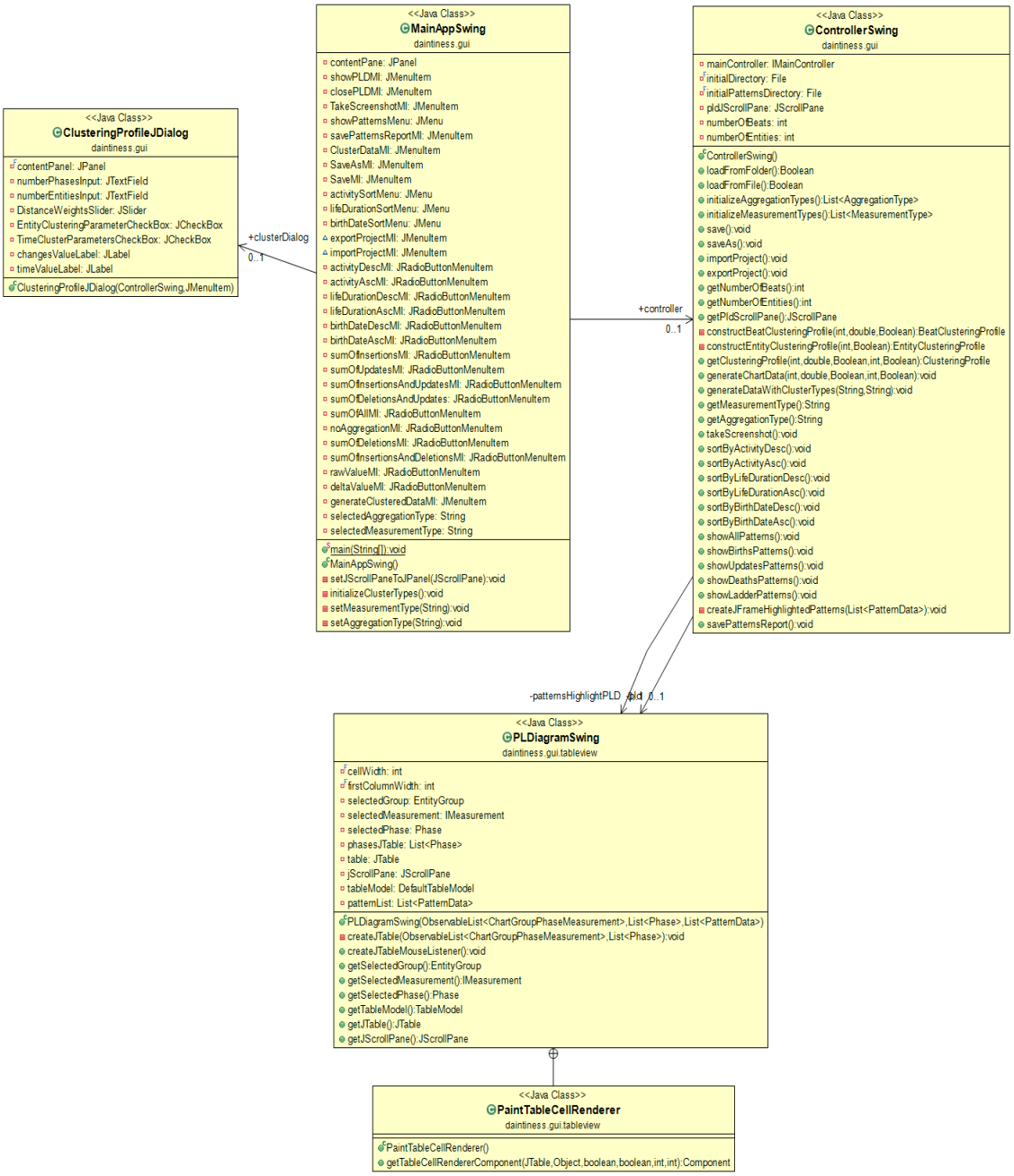
Επιπλέον, εδώ αρχικοποιείται το PLDiagramSwing, για τις περιπτώσεις της απλής εμφάνισης του διαγράμματος, και του διαγράμματος με τα μοτίβα.



Εικόνα 7: Το κύριο παράθυρο της εφαρμογής



Εικόνα 8: Το Pop-Up dialog που ο χρήστης μπορεί να κάνει cluster τα δεδομένα



Εικόνα 9: Διάγραμμα κλάσεων για το πακέτο GUI με την καινούργια αρχιτεκτονική

3.2.3 Κατασκευή αλγορίθμων για δημιουργία μοτίβων στο διάγραμμα Παράλληλων Ζωών

Σε αυτή την ενότητα θα παρουσιαστεί η υλοποίηση για την παραγωγή των μοτίβων στο διάγραμμα.

3.2.3.1 Πακέτο Models

Στο συγκεκριμένο πακέτο προστέθηκαν δύο αντικείμενα για την διαχείριση της πληροφορίας των δεδομένων μας. Αρχικά, το κάθε κελί αποτελείτε από μια οντότητα και μία χρονική. Οι οντότητες είναι οι πίνακες μιας βάσης δεδομένων και στην περίπτωση του διαγράμματος μας αποτελούν τις γραμμές του, και οι χρονικές στιγμές είναι οι στήλες του.

CellInfo: Αυτό το αντικείμενο περιέχει την πληροφορία για την οντότητα και μια συγκεκριμένη χρονική στιγμή. Αυτό μας εξυπηρετεί στο να διαχειριζόμαστε καλύτερα τα κελιά που θα συμμετάσχουν στα μοτίβα

PatternData: Σε αυτό το αντικείμενο βρίσκεται όλη η πληροφορία που θα περιέχει ένα μοτίβο, μια λίστα από κελιά που συμμετέχουν σε αυτό και το πρότυπο που το χαρακτηρίζουμε. Το πρότυπο το αναλύουμε παρακάτω στο 3.2.3.4

Για λόγους εξυπηρέτησης στην κατασκευή των αλγορίθμων, φτιάχτηκαν επιπλέον μέθοδοι για να παίρνουμε μια συγκεκριμένη πληροφορία. Όπως για παράδειγμα η getFirstCellOfPattern() που επιστρέφει το πρώτο κελί στο μοτίβο.

Εικόνα 10: Τα αντικείμενα που προστέθηκαν στο πακέτο Models

3.2.3.2 Πακέτο Patterns

Αυτό το πακέτο αποτελεί ένα ενδιάμεσο στάδιο μεταξύ των αλγορίθμων και της υπόλοιπης εφαρμογής.

PatternManager: Πρόκειται για τον διαχειριστή των μοτίβων, καλεί την μέθοδο για να «τρέξουν» οι αλγόριθμοι και να υπολογιστούν τα μοτίβα. Επιπλέον, εδώ γίνεται η κατασκευή της αναφοράς των μοτίβων για την ιστορία των παράλληλων ζωών. Η αναφορά είναι κατασκευασμένη έτσι ώστε στις πρώτες της γραμμές να υπάρχουν γενικές πληροφορίες για τα μοτίβα και το επιλεγμένο dataset πχ όνομα dataset, πλήθος μοτίβων κ.α. (Εικόνα 11). Στη συνέχεια, καταγράφονται τα μοτίβα που βρέθηκαν κάθε ένα ξεχωριστά, μαζί με τα κελιά που συμμετέχουν σε αυτό (Εικόνα 12).

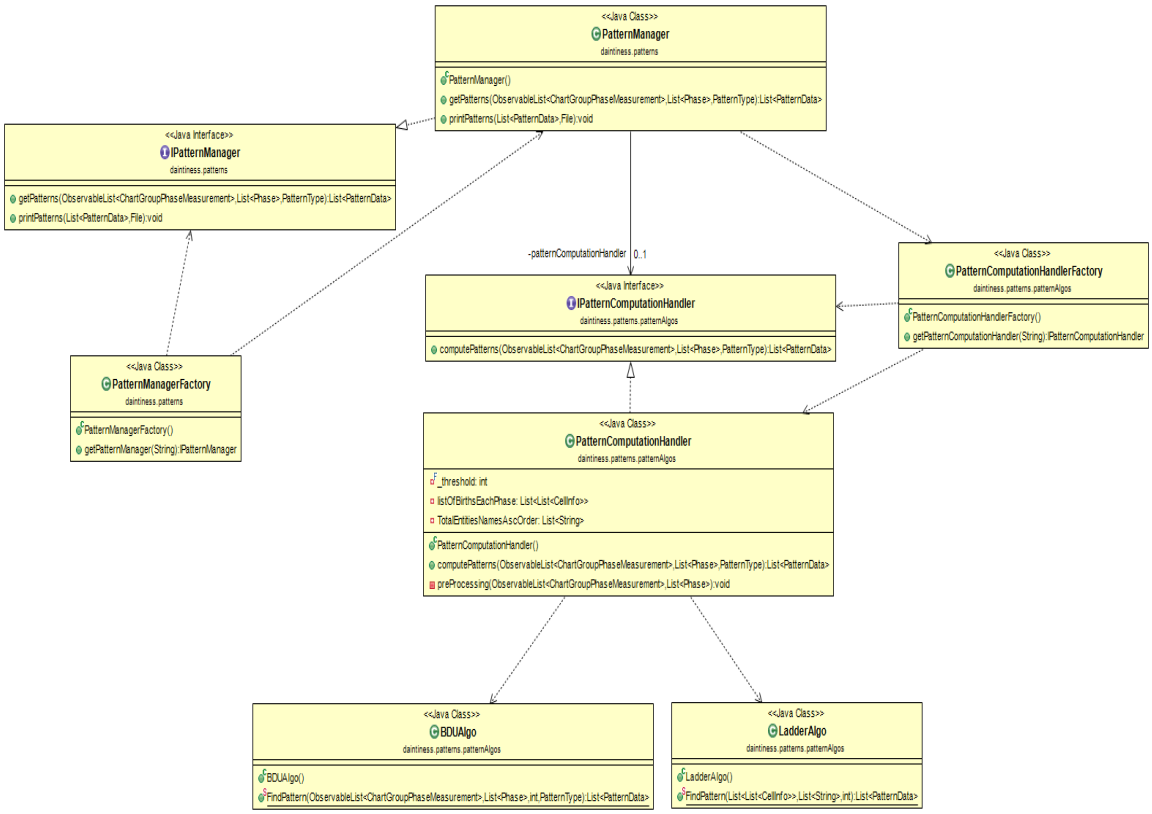
3.2.3.3 Πακέτο PatternAlgos

Αποτελεί το πακέτο που περιέχει τους αλγορίθμους για την κατασκευή των μοτίβων.

PatternComputationHandler: Αυτή η κλάση καλεί τους αλγόριθμους που πρέπει να τρέξουν με βάση την παράμετρο που του έχουμε περάσει. Επιπλέον, υπάρχει μια μέθοδος που κάνει μια προ-επεξεργασία για τα δεδομένα που θα χρειαστούν όλοι οι αλγόριθμοι. Τέλος, εδώ ορίζεται μια σταθερά, ένας αριθμός για τον ελάχιστο αριθμό κελιών που θα αποτελέσουν ένα μοτίβο.

BDUAlgo: Σε αυτή την κλάση περιέχεται ο αλγόριθμος για την εύρεση των μοτίβων με τα πρότυπα «Στήλη πολλαπλών γεννήσεων», «Στήλη πολλαπλών διαγραφών», «Στήλη πολλαπλών ενημερώσεων» και «Σκάλα πολλαπλών γεννήσεων».

LadderAlgo: Σε αυτή την κλάση περιέχεται ο αλγόριθμος για την εύρεση του μοτίβου με το πρότυπο «Σκάλα πολλαπλών γεννήσεων».



Εικόνα 11: Διάγραμμα κλάσεων για το πακέτο Patterns

3.2.3.4 Επεξήγηση των προτύπων

Στην εικόνα 12 διακρίνουμε ότι στην στήλη 1 υπάρχουν μαζικές γεννήσεις για πολλές οντότητες. Επιπλέον, διακρίνουμε ένα παράδειγμα κλιμακωτής αύξησης όπου σε διαδοχικές χρονικές στιγμές γεννιούνται πίνακες. Επίσης, στην στήλη 2 πολλές οντότητες διαγράφονται. Τέλος, σε πολλές χρονικές στιγμές υπάρχουν μαζικές ενημερώσεις των οντοτήτων, όπως στην στήλη 10.

Για τον χαρακτηρισμό των μοτίβων έχουμε ορίσει 4 πρότυπα, τα οποία είναι τα εξής:

- Στήλη πολλαπλών γεννήσεων
- Στήλη πολλαπλών διαγραφών
- Στήλη πολλαπλών ενημερώσεων
- Σκάλα πολλαπλών γεννήσεων

Στην εικόνα 12 υπάρχουν μοτίβα που χαρακτηρίζονται με το κάθε πρότυπο. Από την στήλη 0 έως 5 και 20 έως 29 υπάρχουν 2 μοτίβα με το πρότυπο της «Σκάλας πολλαπλών γεννήσεων», καθώς σε διαδοχικές στήλες υπάρχουν κελιά γεννήσεων και το

άθροισμα τους είναι μεγαλύτερο του 3 που έχουμε ορίσει ως ελάχιστο αριθμό κελιών που μπορεί να αποτελέσει ένα πρότυπο. Στην στήλη 2 υπάρχουν μαζικές διαγραφές οντοτήτων με αριθμό μεγαλύτερο του 3, οπότε μπορούμε αυτά τα κελιά να τα εντάξουμε σε ένα μοτίβο με το πρότυπο του «Στήλη πολλαπλών διαγραφών». Αντίστοιχα, στην στήλη 10 υπάρχει μοτίβο με το πρότυπο του «Στήλη πολλαπλών ενημερώσεων». Τέλος στην στήλη 1 γίνεται εύκολα αντιληπτό ότι υπάρχει μοτίβο με το πρότυπο του «Στήλη πολλαπλών γεννήσεων» αλλά στην εικόνα κρύβεται από το μοτίβο του προτύπου «Σκάλα πολλαπλών γεννήσεων»

3.2.3.5 Επεξήγηση των αλγόριθμων

Στην διπλωματική μας έχουμε υλοποιήσει δύο αλγόριθμους εύρεσης μοτίβων και είναι οι εξής:

- **BDU Algorithm**

  Σε αυτόν τον αλγόριθμο διατρέχουμε το διάγραμμα μας και βρίσκουμε μοτίβα τα οποία τα χαρακτηρίζουμε με ένα από τα εξής πρότυπα «Στήλη πολλαπλών γεννήσεων», «Στήλη πολλαπλών διαγραφών», «Στήλη πολλαπλών ενημερώσεων» Για λόγους βελτιστοποίησης ο αλγόριθμος υπολογίζει και τα μοτίβα και για τα 3 πρότυπα, καθώς η διαδικασία υπολογισμού τους είναι ίδια. Παρακάτω παραθέτουμε ένα ψευδοκώδικα για τον αλγόριθμο.

1. **Είσοδος**: Λίστα οντοτήτων(γραμμές), λίστα χρονικών στιγμών(στήλες), threshold, επιλεγμένο πρότυπο(περιέχεται και η επιλογή για όλα τα πρότυπα)
2. **Έξοδος**: Λίστα με τα μοτίβα που βρέθηκαν
3. **Begin**

```
Για κάθε στήλη i
        Αρχικοποίηση λιστών
        Για κάθε στήλη j
            Αν(κατάσταση κελί(i,j) == Γέννηση)
                    Προσθήκη κελιού στη λίστα γεννήσεων
            Αλλιώς αν(κατάσταση κελί(i,j) == ενημέρωση)
                    Προσθήκη κελιού στη λίστα ενημερώσεων
                Αλλιώς αν(κατάσταση κελί(i,j) == διαγραφή)
                    Προσθήκη κελιού στη λίστα διαγραφών
        Αν μέγεθος λίστας επιλεγμένου προτύπου > threshold
            Προσθήκη μοτίβου στην λίστα εξόδου
    Επιστροφή λίστας εξόδου
End
```

- **Ladder Algorithm**

  Σε αυτόν τον αλγόριθμο διατρέχουμε το διάγραμμα μας και βρίσκουμε μοτίβα τα οποία τα χαρακτηρίζουμε με το πρότυπο «Σκάλα πολλαπλών γεννήσεων». Για να μπορέσει να τρέξει ο συγκεκριμένος αλγόριθμος πρέπει σε προηγούμενο βήμα να γίνει ένα σορτάρισμα του διαγράμματος με βάση την αύξουσα σειρά της γέννησης των οντοτήτων. Επιπλέον χρειάζεται να γίνει μια προ-επεξεργασία όπου θα κρατάμε όλα τα κελιά με κατάσταση γέννησης για κάθε στήλη ώστε να τα λαμβάνουμε όλα υπόψιν και όχι μόνο τα τελευταία. Παρακάτω παραθέτουμε ένα ψευδοκώδικα για τον αλγόριθμο.

1. **Είσοδος**: Λίστα από λίστες των κελιών γέννησης κάθε στήλης, λίστα οντοτήτων με βάση την ταξινόμηση αύξουσας σειράς γεννήσεων
2. **Έξοδος**: Λίστα με τα μοτίβα που βρέθηκαν
3. **Begin**

```
Αρχικοποίηση λίστας προτύπου σκάλας
    Για κάθε στήλη i
    Πάρε το τελευταίο κελί γέννησης
        Αν(το επόμενο κελί γέννησης βρίσκεται στις επόμενες 3 στήλες η στις επόμενες 3 γραμμές)
            Πρόσθεσε το κελί(i) στη λίστα και όλα τα κελιά γέννησης που βρίσκονται στην ίδια στήλη
    Αλλιώς
            Αν(μέγεθος λίστας επιλεγμένου προτύπου > threshold)
                Προσθήκη μοτίβου στην λίστα εξόδου

            Αρχικοποίηση λίστας προτύπου σκάλας

    Αν(μέγεθος λίστας επιλεγμένου προτύπου > threshold)
        Προσθήκη μοτίβου στην λίστα εξόδου
Επιστροφή λίστας εξόδου
```

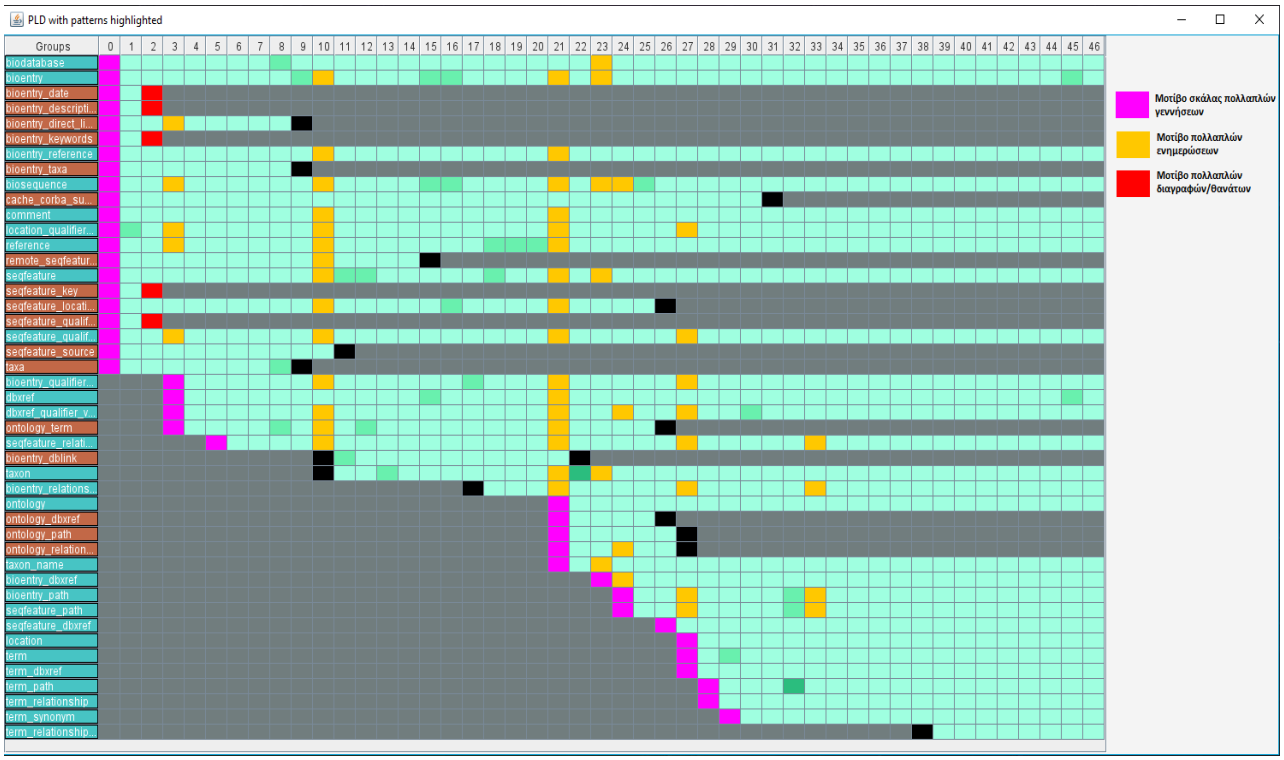
End

3.2.3.6 Πακέτο Gui

Στην εικόνα 9 βλέπουμε ότι στην νέα αρχιτεκτονική του πακέτου Gui προστέθηκαν πεδία και μέθοδοι που είναι υπεύθυνοι για τις διαδικασίες της προβολής των μοτίβων.

Στη μπάρα πλοήγησης της εφαρμογής προστέθηκε το menu «Patterns». Σε αυτό ο χρήστης επιλέγοντας την επιλογή «Show Patterns» και το πρότυπο των μοτίβων που θέλει να δει, θα του εμφανιστεί ένα pop-up παράθυρο το οποίο θα περιέχει το PLD με χρωματισμένα τα κελιά που συμμετέχουν στα μοτίβα, όπως βλέπουμε στην εικόνα 12.

Επίσης, στο menu υπάρχει η επιλογή «Save report of loaded patterns», που κλικάροντας την ο χρήστης έχει την δυνατότητα να αποθηκεύσει την αναφορά των μοτίβων σε μορφή .txt.



Εικόνα 12: Το PLD της Biosql με χρωματισμένα τα κελιά που συμμετέχουν στα μοτίβα.

```
Project Name:   biosql
biosql  Number of columns:  47
biosql  Number of rows: 45
biosql  Number of columns that participate in patterns: 13
biosql  Number of rows that participate in patterns:    43
biosql  Number of total patterns:   13
biosql  Number of births patterns:  3
biosql  Number of deaths patterns:  1
biosql  Number of updates patterns: 7
biosql  Number of ladder patterns:  2
biosql  Patterns computation(sec)   0.0073371
```

Εικόνα 13: Παράδειγμα γενικών πληροφορίων των μοτίβων στην εξαγόμενη αναφορά

```
MULTIPLE_DEATHS
The pattern consists of 5 cells
Entity Name : bioentry_date PhaseId: 2
Entity Name : bioentry_description PhaseId: 2
Entity Name : bioentry_keywords PhaseId: 2
Entity Name : seqfeature_key PhaseId: 2
Entity Name : seqfeature_qualifier PhaseId: 2

MULTIPLE_BIRTHS
The pattern consists of 4 cells
Entity Name : bioentry_qualifier_value PhaseId: 3
Entity Name : dbxref PhaseId: 3
Entity Name : dbxref_qualifier_value PhaseId: 3
Entity Name : ontology_term PhaseId: 3

MULTIPLE_UPDATES
The pattern consists of 5 cells
Entity Name : bioentry_direct_links PhaseId: 3
Entity Name : biosequence PhaseId: 3
Entity Name : location_qualifier_value PhaseId: 3
Entity Name : reference PhaseId: 3
Entity Name : seqfeature_qualifier_value PhaseId: 3
```

Εικόνα 14: Παράδειγμα κάποιων μοτίβων που καταγράφηκαν κατά την εξαγωγή της αναφοράς

3.3 Σχεδίαση και αποτελέσματα ελέγχου του λογισμικού

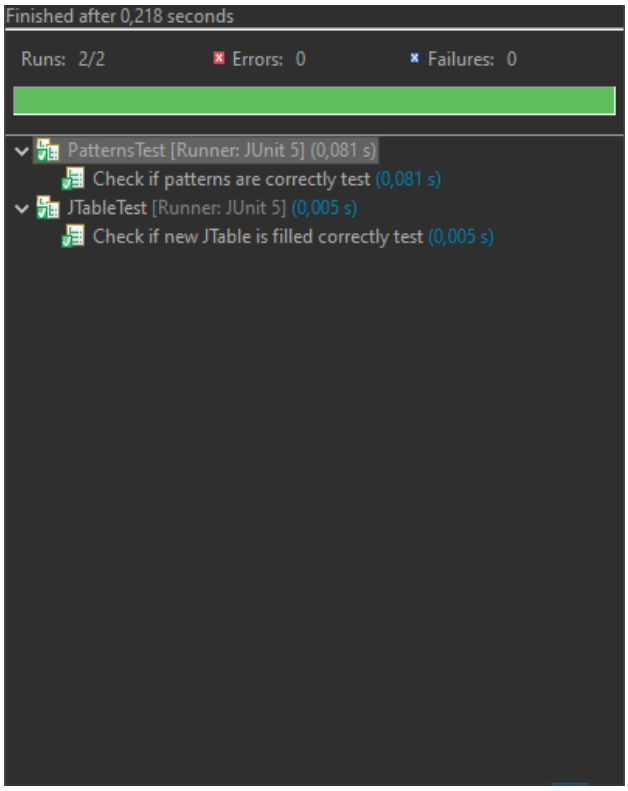
Σε αυτό το κεφάλαιο θα περιγράψουμε τις διενέργειες ελέγχου της σωστής λειτουργίας του λογισμικού.

3.3.1 Test που υλοποιήθηκαν

Για την δημιουργία των test χρησιμοποιούμε το framework JUnit 5 και ελέγχουμε τα εξής test cases:

- **Έλεγχος αν το καινούργιο JTable γεμίζει σωστά.**
  Για τον έλεγχο αυτού του σεναρίου παίρνουμε όλα τα δεδομένα που αναπαριστά το JTable(Οντότητες, χρονολογίες, μετρήσεις μέσα στα κελιά) και τα συγκρίναμε με τα δεδομένα του «keybase_node-client» dataset.

- **Έλεγχος αν οι αλγόριθμοι για τα μοτίβα δουλεύουν σωστά.**
  Για τον έλεγχο αυτού του σεναρίου χτίζουμε τα κελιά με βάση τα μοτίβα που υπάρχουν στο dataset και τα συγκρίνουμε με τα παραγόμενα μοτίβα των αλγορίθμων. Χρησιμοποιήθηκε το «biosql» dataset.



Εικόνα 15: Unit tests της εφαρμογής

3.4 Λεπτομέρειες εγκατάστασης και υλοποίησης

Στην ενότητα αυτή περιγράφονται τα χαρακτηριστικά της συγκεκριμένης υλοποίησης, όπως η πλατφόρμα ανάπτυξης και εκτέλεσης, τα προγραμματιστικά εργαλεία, οι απαιτήσεις της εφαρμογής σε hardware, κ.λ.π.

3.4.1 Τεχνικά Χαρακτηριστικά

Το συγκεκριμένο εργαλείο αναπτύχθηκε με την γλώσσα προγραμματισμού Java(https://www.java.com/), και χρησιμοποιήθηκε η Java 11.

Για την συγγραφή του κώδικα χρησιμοποιήθηκε το περιβάλλον ανάπτυξης του Eclipse(https://www.eclipse.org/) και συγκεκριμένα το Eclipse IDE (2020-12).

Για την υλοποίηση του γραφικού περιβάλλοντος χρησιμοποιήθηκε η βιβλιοθήκη Swing.

Οι δοκιμές του κώδικα έγιναν με την διαδικασία του «Unit Testing», αυτή υλοποιήθηκε με το JUnit 5 framework.

Επιπλέον, χρησιμοποιήθηκε το εργαλείο Apache Maven 5 για την διαχείριση των dependencies της εφαρμογής.

Τέλος, υπάρχουν κομμάτια κώδικα που χρησιμοποιούν την JavaFX 11.

3.4.2 Εγκατάσταση

Για την εγκατάσταση της εφαρμογής θα πρέπει να γίνουν τα εξής βήματα:

1. Άνοιγμα του Eclipse
2. Επιλογή File -> Import
3. Επιλογή Git -> Projects from Git(with smart import)
4. Συμπληρώστε την διεύθυνση του repository (Η οποία είναι αυτή: https://github.com/DAINTINESS-Group/PlutarchParallelLives)
5. Εάν όλα έχουν γίνει επιτυχώς το Eclipse θα πρέπει να αναγνωρίσει το project.
6. Κλικ στο Finish
7. Δεξί κλικ πάνω στο Project
8. Επιλέγουμε Maven -> Update Project
9. Σε αυτό το παράθυρο θα πρέπει να επιλεχθούν τα εξής checkboxes:
    - Να επιλεχθεί το Project κάτω από την κατηγορία «Available Maven Codebases»
    - Η επιλογή «Update dependencies»

- Η επιλογή «Update project configuration using from pom.xml»
- Η επιλογή «Refresh workspace resources from local filesystem»
- Η επιλογή «Clean projects»

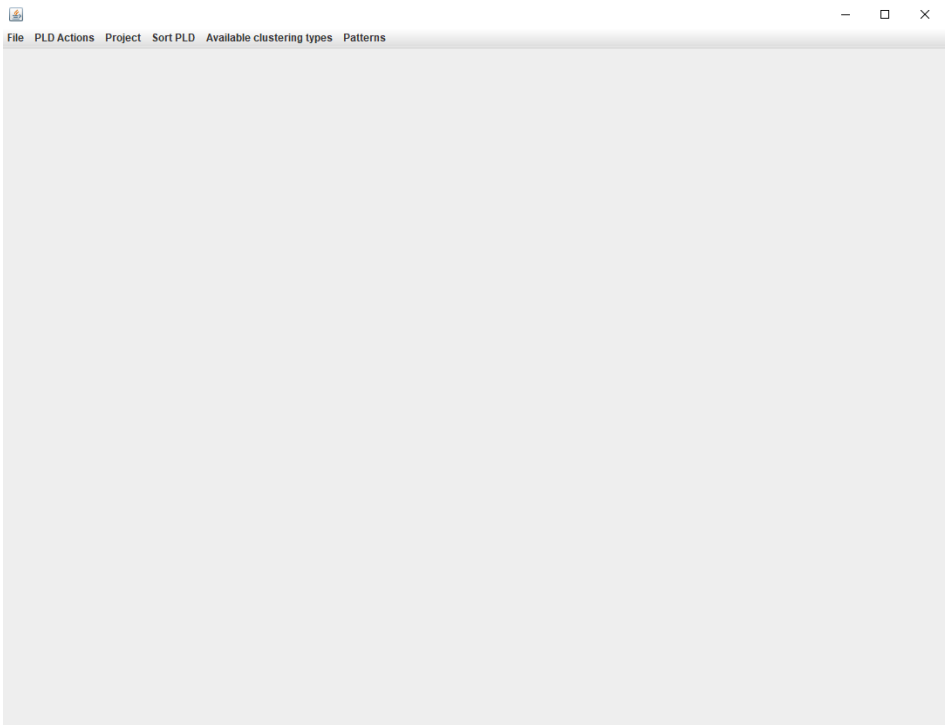
10. Κλικ στο OK και η εγκατάσταση είναι έτοιμη

Σημείωση: Πρέπει να έχει εγκατασταθεί στο σύστημά ένα jdk 11+ με το σωστό configuration του JAVA_HOME. Εάν χρησιμοποιηθεί διαφορετικός IDE από αυτόν του Eclipse, θα πρέπει να γίνει κάποια διαφορετική προεργασία για την εισαγωγή του project από το Github και την απόκτηση των dependencies μέσω Maven καθώς ορισμένα IDE δεν έχουν ενσωματωμένη προσθήκη του Maven και ίσως χρειαστεί να την εγκατασταθεί ξεχωριστά.

3.4.3 Χρήση Εργαλείου

Παρακάτω θα περιγράψουμε την διαδικασία χρήσης του εργαλείου:

1. Άνοιγμα του MainAppSwing
2. Θα εμφανιστεί η κύρια οθόνη



Εικόνα 16: Κεντρική οθόνη

3. Φορτώνουμε στο σύστημα μας τα δεδομένα από κάποιες από τις διαθέσιμες μορφές αρχείων:

a. Load from Folder: Φόρτωση δεδομένων από φάκελο. Για την επιλογή αυτή το εργαλείο απαιτεί μια συγκεκριμένη δομή φακέλου, η δομή αυτή είναι η εξής:

   i. Ένα αρχείο dataset_DetailedStats.tsv που περιέχει τα δεδομένα για τις οντότητες του dataset

Table	Duration	Birth	Death	LastKnownVersion	BirthDate	LKVDate	YearOfBirth	YearOfLKV	DurationDays	SchemaSize@Birth	SchemaSize@LKV	SchemaSizeAvg	SchemaSizeResizeRatio
biodatabase	47	0	-	46	2002-01-28 01:30:31	2012-09-10 10:18:40	0	11	3878	2	4	3.34	2 4 2 0.09 0.04 2 20 1 21
bioentry	47	0	-	46	2002-01-28 01:30:31	2012-09-10 10:18:40	0	11	3878	6	9	8.23	1.5 12 7 0.26 0.15 1.71 20 2 22

Εικόνα 17: Μορφή του dataset_DetailedStats.tsv 1

SumUpd	CountVwUpd	ATU UpdRate	AvgUpdVolume	SurvivalClass	ActivityClass	LADClass

Εικόνα 18: Μορφή του dataset_DetailedStats.tsv 2

   ii. Ένα αρχείο SchemaHeartBeat.tsv που περιέχει τα δεδομένα για τις χρονικές στιγμές του dataset

trID	epochTime	oldVer	newVer	humanTime	distFromV0InDays	runningYearFromV0	runningMonthFromV0	#numOldTables	#numNewTables	#numOldAttrs	#numNewAttrs
0	1012181431		1012181431.sql	2002-01-28 01:30:31	0	0	0	21	74	21	0 74 0 0 0 0 0	74	0 0 74	0 74
1	1014631726	1012181431.sql	1014631726.sql	2002-02-25 10:08:46	28	1	1	21	21	74	74 0 0 0 0 1 1 0 0 0 0 1 1 0 1 1 2			

Εικόνα 19: Μορφή του SchemaHeartBeat.tsv 1

tablesIns	tablesDel	attrsInsWithTableIns	attrsbDelWithTableDel	attrsInjected	attrsEjected	attrsWithTypeUpd	attrsInPKUpd

Εικόνα 20: Μορφή του SchemaHeartBeat.tsv 2

tableDelta	attrDelta	attrBirthsSum	attrDeathsSum	attrUpdsSum	Expansion	Maintenance	TotalAttrActivity

Εικόνα 21: Μορφή του SchemaHeartBeat.tsv 3

   iii. Ένα αρχείο transitions.csv που περιέχει τα δεδομένα για τις μεταβάσεις μιας οντότητας από μια χρονική στιγμή σε μία άλλη

```
trID;oldVer;newVer;Table;EventType;attrName;attrType;iskey;pkey;fkey
1;1012181431.sql;1014631726.sql;location_qualifier_value;Insertion:UpdateTable;qualifier_int_value;INT(10);false;0;-
1;1012181431.sql;1014631726.sql;location_qualifier_value;Deletion:UpdateTable;slot_value;INT(10);false;0;-
```

Εικόνα 21: Μορφή του transitions.csv

   Και τα τρία αρχεία πρέπει να βρίσκονται σε ένα φάκελο με ονομασία results.

b. Load from File: Φόρτωση δεδομένων αρχείου που μπορεί να είναι είτε .tsv είτε .csv. Η μορφή πρέπει να είναι η εξής: στην πρώτη γραμμή να είναι οι κεφαλίδες της κάθε κολώνας, και οι επόμενες γραμμές στην πρώτη στήλη το όνομα του dataset και έπειτα τα δεδομένα για την κάθε κεφαλίδα.
c. Import from Project: Φόρτωση δεδομένων από τα εξαγόμενα αρχεία που έχει κατασκευάσει η εφαρμογή.

4. Καταχωρούμε την ομαδοποίηση που θέλουμε να έχουν τα δεδομένα μας(Εικόνα 8)
5. Έχουν φορτωθεί τα δεδομένα και έχει δημιουργηθεί ένα διάγραμμα PLD που τα εμφανίζει(Εικόνα 22)
6. Πλέον έχουμε την δυνατότητα να εκτελέσουμε κάποιες ενέργειες
    a. Εξαγωγή του project σε ενδιάμεση μορφή
    b. Λήψη στιγμιότυπου οθόνης
    c. Ταξινόμηση του πίνακα από με φίλτρα ένα σύνολο πληροφοριών
    d. Άνοιγμα ενός διαγράμματος για zoom in/out
    e. Επιλογή οντότητας και εμφάνιση πληροφοριών για αυτή
    f. Αλλαγή στην ομαδοποίηση των δεδομένων με βάση κάποια παράμετρο
    g. Εύρεση μοτίβων με βάση τα κελιά του διαγράμματος και εμφάνιση νέου διαγράμματος με χρωματισμένα τα κελιά των μοτίβων(Εικόνα 13)
    h. Παραγωγή μιας αναφοράς που μας δίνει πληροφορίες για τα μοτίβα που βρέθηκαν(Εικόνα 14,15)

Εικόνα 22: Κεντρική οθόνη με φορτωμένα τα δεδομένα

3.5 Επεκτασιμότητα του λογισμικού

Η εφαρμογή είναι υλοποιημένη έτσι ώστε να μπορούν μελλοντικά να γίνουν επεκτάσεις με βάση τις ανάγκες του χρήστη. Παρακάτω θα παραθέσουμε κάποιες από αυτές.

Επέκταση προτύπων μοτίβων: Αυτή την στιγμή υπάρχουν 4 είδη μοτίβων στην εφαρμογή, πολλαπλές γεννήσεις, διαγραφές, ενημερώσεις πινάκων και σκάλα πολλαπλών γεννήσεων. Μελλοντικά θα μπορούν να φτιαχτούν και άλλοι αλγόριθμοι που θα ψάχνουν στο διάγραμμα να βρουν ένα άλλο πρότυπο μοτίβου, «όπως πρότυπο από διάστημα μηδενικών αλλαγών».

Στην εφαρμογή ο ελάχιστος αριθμός των κελιών που μπορούν να αποτελέσουν ένα μοτίβο είναι hard-coded. Στην εφαρμογή μας η τιμή αυτή είναι 3. Αν κάποιος μελλοντικά θέλει αυτό να αλλάξει, θα πρέπει να πάει στην κλάση PatternComputationHandler και να αλλάξει την τιμή του πεδίου _threshold.

Κεφάλαιο 4. Πειραματική Αξιολόγηση

Στην ενότητα αυτή παρουσιάζουμε αναλυτικά την πειραματική αξιολόγηση της μεθόδου μας.

4.1 Μεθοδολογία πειραματισμού

Για την πειραματική διαδικασία αυτό που θέλαμε να εξετάσουμε ήταν ο χρόνος που χρειάζεται η εφαρμογή μας για υπολογίσει όλα τα μοτίβα που περιέχονται σε ένα dataset. Αυτή η πληροφορία υπάρχει στην αναφορά που κατασκευάζουμε κατά την διάρκεια της εύρεσης τους. Για την συγκεκριμένη διαδικασία είχαμε στην διάθεση μας 195 dataset.

Για την παραγωγή της αναφοράς φτιάξαμε ένα εκτελέσιμο αρχείο το οποίο λαμβάνει παραμετρικά ένα path ενός φακέλου από datasets και ένα path για το που να παράγει τις αναφορές. Στη συνέχεια, φορτώνει όλα τα dataset, κάνει το κάθε ένα sort με βάση την αύξουσα στο χρόνο δημιουργία των πινάκων, ώστε να μπορούν να παραχθούν μοτίβα για την σκάλα πολλαπλών γεννήσεων, και στο επόμενο βήμα παράγει τα .txt αρχεία για το dataset.

Έπειτα, από τα αρχεία που παράχθηκαν πήραμε τις γενικές πληροφορίες που βρίσκονται στην αρχή του αρχείου και δημιουργήσαμε αρχεία tsv που περιέχουν τις παρακάτω πληροφορίες:

- Όνομα βάσης δεδομένων
- Σύνολο χρονικών στιγμών στο διάγραμμα
- Σύνολο οντοτήτων στο διάγραμμα
- Σύνολο χρονικών στιγμών που συμμετέχουν στα μοτίβα
- Σύνολο οντοτήτων που συμμετέχουν στα μοτίβα
- Σύνολο μοτίβων που βρέθηκαν
- Σύνολο μοτίβων πολλαπλών γεννήσεων πινάκων
- Σύνολο μοτίβων πολλαπλών διαγραφών πινάκων
- Σύνολο μοτίβων πολλαπλών ενημερώσεων πινάκων
- Σύνολο μοτίβων σκάλας πολλαπλών γεννήσεων
- Χρόνος που χρειάστηκε για να υπολογιστούν τα μοτίβα

Στην εικόνα 23 βλέπουμε ένα κομμάτι από το αρχείο που παράχθηκε για τον φάκελο από dataset «4_SCHEMA_EVO_2019_v04_withHeraResults»

Για τα πειράματα χρησιμοποιήσαμε ένα HP Pavilion 15 Laptop με τις εξής προδιαγραφές:

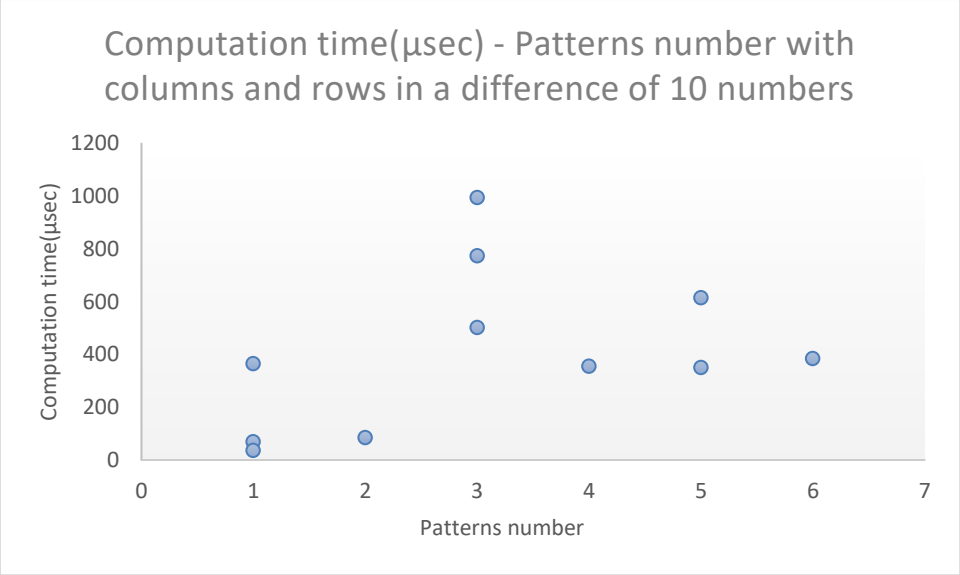
- CPU: AMD RYZEN 4800H
- STORAGE: 512 SSD
- RAM: 16GB

	Project	TotalNumberOfColumns	TotalNumberOfRows	NumberO	NumberO	NumberO	NumberO	NumberO	NumberO	NumberO	ComputationTime(sec)
2	3ev__tev_label	4	2	0	0	0	0	0	0	0	0
3	AA-ALERT__frbcatdb	16	16	3	16	3	1	0	2	0	9.961E-4
4	accgit__acl	17	7	3	7	3	1	0	2	0	3.237E-4
5	aimeos__aimeos-typo3	6	3	0	0	0	0	0	0	0	0
6	aiyi__go-user	10	6	6	6	6	1	0	5	0	1.575E-4
7	alextselegidis__easyappointments	19	11	2	9	2	1	0	1	0	3.011E-4
8	anchorcms__anchor-cms	17	20	3	20	3	2	1	0	0	5.025E-4
9	ankitjain28may__registration-module	2	2	0	0	0	0	0	0	0	0
10	APTrust__exchange	2	3	0	0	0	0	0	0	0	0
11	archan937__cached_record	5	5	0	0	0	0	0	0	0	0
12	arnoldasgudas__Hangfire.MySqlStorage	15	24	3	24	3	2	1	0	0	7.751E-4
13	atomjump__loop-server	8	10	1	10	1	1	0	0	0	1.465E-4
14	Attendly__maillist	9	7	2	6	2	1	0	1	0	1.694E-4
15	azzlack__Sentinel.OAuth	3	3	0	0	0	0	0	0	0	0
16	benoitletondor__TwitterBot	10	4	4	4	1	0	0	0	1	1.14E-4
17	bgentry__que-go	3	1	0	0	0	0	0	0	0	0
18	blabla1337__skf-flask	45	25	6	20	6	2	1	3	0	0.0015406
19	blueriver__MuraCMS	5	41	1	41	1	1	0	0	0	0.0077126
20	BotBotMe__botbot-bot	2	3	0	0	0	0	0	0	0	0
21	brettkromkamp__topic_db	16	15	5	15	5	2	1	2	0	3.512E-4
22	builderscon__octav	74	27	13	16	4	1	0	0	3	0.002973
23	byteball__byteballcore	12	68	4	68	3	1	0	1	1	0.0012479
24	cartalyst__sentry	13	5	2	4	2	1	0	1	0	1.315E-4
25	cgrates__cgrates	190	40	26	29	15	1	0	11	3	0.0084592

Εικόνα 23: Παράδειγμα από το excel για το «4_SCHEMA_EVO_2019_v04_withHeraResults»

4.2 Αναλυτική παρουσίαση αποτελεσμάτων

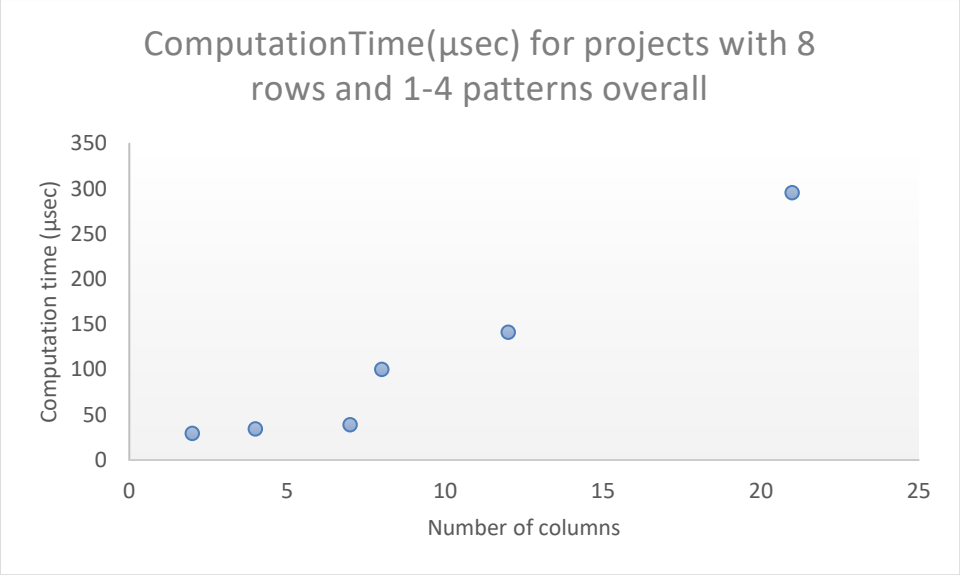
Για την ανάλυση των αποτελεσμάτων χρησιμοποιούμε scatter plots που τιμές τους θα είναι τα dataset, στον Y άξονα θα έχουμε τον χρόνο που χρειάστηκε για την εύρεση των μοτίβων και στον X θα αλλάζουμε τις παραμέτρους, για να δούμε αν μπορούμε να βγάλουμε κάποια πιο γενικευμένα αποτελέσματα. Βέβαια, τα dataset που είχαμε στην διάθεση μας δεν ήταν πολλά και τα περισσότερα είχαν αριθμό από οντότητες και χρονικές στιγμές <100 με αποτέλεσμα να μην βρίσκουμε αρκετά μοτίβα.



Εικόνα 24: Scatter plot Patterns computation time - Patterns

Στην εικόνα 24 βλέπουμε τον χρόνο που χρειάστηκε να υπολογιστούν τα μοτίβα συναρτήσει με τον αριθμό των μοτίβων, για dataset που έχουν μεταξύ τους ίδιες στήλες και γραμμές η μια διαφορά της τάξης των 10. Τα αποτελέσματα που λαμβάνουμε δεν είναι γραμμικά και διακρίνουμε στο διάγραμμα ότι υπάρχουν περιπτώσεις που ο χρόνος υπολογισμού 3 μοτίβων είναι μεγαλύτερος από αυτόν για 6. Τα dataset που συμμετέχουν είναι:

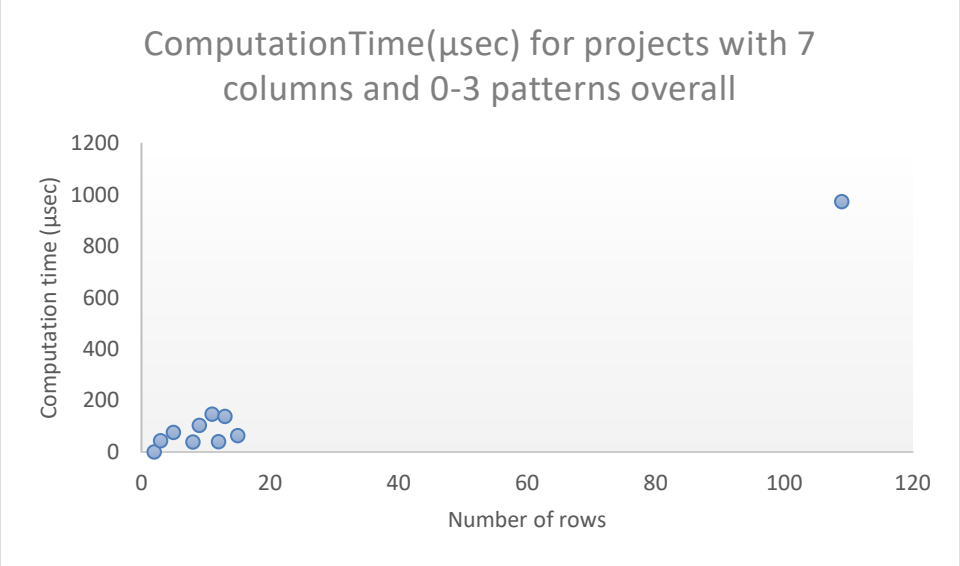
1. anchorcms__anchor-cms
2. AA-ALERT__frbcatdb
3. brettkromkamp__topic_db
4. MorpheusXAUT__eveauth
5. hurad__hurad
6. arnoldasgudas__Hangfire.MySqlStorage
7. gousiosg__github-mirror
8. gugoan__economizzer
9. pw-press__web-project
10. h2oai__steam
11. wskm__deruv



Εικόνα 25: Scatter plot Patterns computation time – Number of columns

Στην εικόνα 25 βλέπουμε ένα scatter plot μεταξύ χρόνου υπολογισμού των μοτίβων και τον αριθμό των συνολικών στηλών για dataset με 8 γραμμές και 0 έως 3 μοτίβα. Επειδή είναι μικρός ο αριθμός των dataset δεν μπορούμε να βγάλουμε ακριβή αποτελέσματα, παρόλα αυτά βλέπουμε πως με μια αύξηση στο σύνολο των στηλών αυξάνετε και ο χρόνος υπολογισμού. Τα dataset που συμμετέχουν είναι:

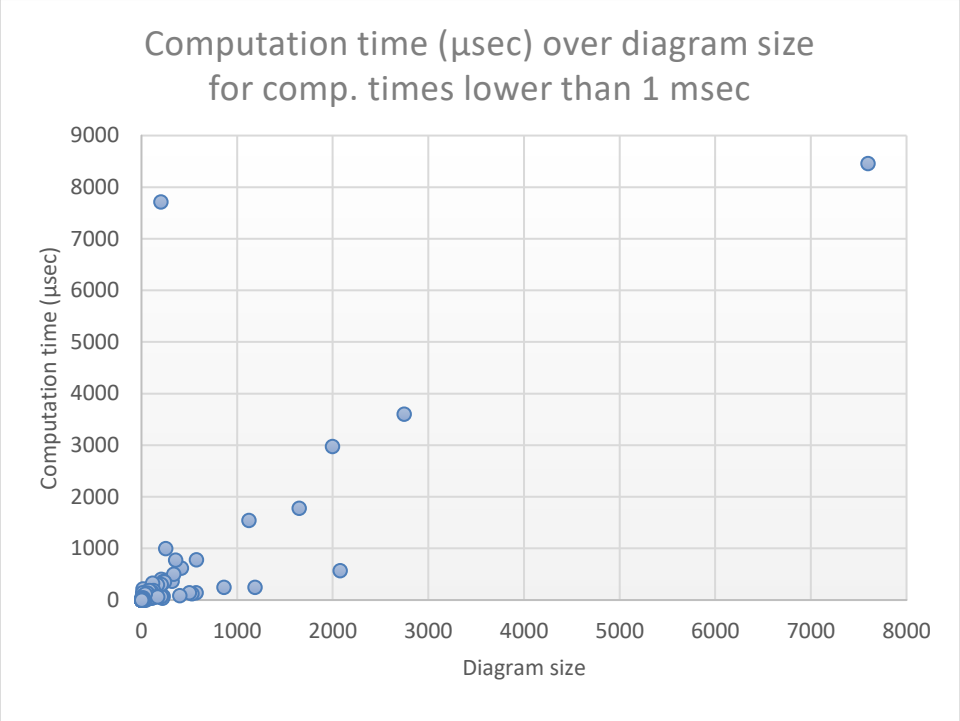
1. saltzm__yadi
2. SalesforceEng__cucumber-metrics
3. n2n__page
4. jadekler__git-go-d3-concertsap
5. jasdel__harvester
6. comforme__comforme



Εικόνα 26: Scatter plot Patterns computation time – Number of rows

Στην εικόνα 25 βλέπουμε ένα scatter plot μεταξύ χρόνου υπολογισμού των μοτίβων και τον αριθμό των συνολικών γραμμών για dataset με 7 στήλες και 0 έως 3 μοτίβα. Τα αποτελέσματα μας δείχνουν ότι δεν μπορεί να καθοριστεί αποκλειστικά από τις γραμμές η πολυπλοκότητα του χρόνου υπολογισμού του εκάστοτε dataset.

1. schimmy__shorty
2. scorelab__Bassa
3. keybase__node-client
4. n2n__page
5. jaybennett89__thorium-go
6. curt-labs__GoSurvey
7. protosam__hostcontrol
8. lisong__code-push-server
9. n2n__rocket
10. damnpoet__yiicart



Εικόνα 27: Scatter plot Patterns computation time – Number of rows

Στην εικόνα 27 βλέπουμε το διάγραμμα μεταξύ χρόνου υπολογισμού των μοτίβων και το μέγεθος του κάθε dataset(υπολογίζεται αριθμός στηλών * αριθμός γραμμών) για όλα τα dataset που έχουμε στη διάθεση μας. Μπορούμε να διακρίνουμε γενικά υπάρχει μια συσχέτιση χρόνου υπολογισμού μοτίβων και μέγεθος dataset και παρατηρείτε μια γραμμική σχέση στα περισσότερα dataset αλλά υπάρχουν και περιπτώσεις που δεν ισχύει αυτό. Σε διερεύνηση που κάναμε για να βγάλουμε κάποιο συμπέρασμα γιατί συμβαίνει αυτό δεν βρήκαμε κάποιο καθοριστικό παράγοντα και θεωρούμε πως οι χρόνοι αυτοί οφείλονται στην τυχαιότητα των μικρών χρόνων καθώς πρόκειται υπολογισμούς κάτω του 1 msec.

Κεφάλαιο 5. Επίλογος

Στην ενότητα αυτή συνοψίζουμε τη συνεισφορά και τα αποτελέσματα της εργασίας και παραθέτουμε σκέψεις για μελλοντικές επεκτάσεις της.

5.1 Σύνοψη και συμπεράσματα

Στον Πλούταρχο, υπάρχει ένα εργαλείο με την ονομασία «Πλουτάρχου Βίοι Παράλληλοι» όπου χρησιμοποιείτε για να απεικονίσει την εξέλιξη των πινάκων μιας βάσης δεδομένων σε παράλληλα εξελισσόμενες χρονοσειρές. Το εργαλείο χρησιμοποιεί έναν δισδιάστατο πίνακα που τον καλούμε «Διάγραμμα Παράλληλων Ζωών», όπου οι γραμμές του αποτελούν του πίνακες μια βάσης δεδομένων και οι κολώνες τι χρονικές στιγμές. Αυτό το εργαλείο αρχικά κατασκευάστηκε από τον Θ. Γιάχο[Giac15] και μετέπειτα συνεχίστηκε από τον Ν. Παντελίδη[Pant21].

Το πρόβλημα που κληθήκαμε να αντιμετωπίσουμε ήταν η δυσκολία του εργαλείου να αναπαραστήσει μεγάλα σύνολα δεδομένων, και συχνά σε τέτοιες περιπτώσεις να προκαλείτε τερματισμός. Για να το διορθώσουμε αυτό αντικαταστήσαμε την βιβλιοθήκη(JavaFX) που ήταν υπεύθυνη για το κομμάτι της αλληλεπίδρασης της εφαρμογής με τον χρήστη και την αντικαταστήσαμε με μία άλλη(Java Swing) που το αντικείμενο της JTable έχει την ικανότητα να αναπαριστά μεγάλο πλήθος δεδομένων, χωρίς να δημιουργεί πρόβλημα. Επιπλέον, πολλές λειτουργίες του εργαλείου που είχαν δημιουργηθεί με βάση την JavaFX προστέθηκαν εξίσου και στην νέα υλοποίηση.

Έπειτα, προσθέσαμε αλγορίθμους οι οποίοι με βάση την θέση των κελιών στον διάγραμμα κατασκευάζουν μοτίβα και τα χαρακτηρίζουν με ένα πρότυπο, αυτό μας έδωσε την δυνατότητα να παρατηρήσουμε φαινόμενα δημιουργίας, διαγραφής και ενημερώσεων πινάκων για την ίδια χρονική στιγμή και να βγουν κάποια συμπεράσματα για εκείνη την χρονική στιγμή.

Τέλος, προσφέρθηκε η δυνατότητα παραγωγής μια αναφοράς για το μοτίβα που βρέθηκαν σε ένα διάγραμμα. Με βάση αυτό, παραγάγαμε αναφορές από πολλά σύνολα δεδομένων και ακολουθήσαμε μια πειραματική διαδικασία, στην οποία υπολογίσαμε τον χρόνο που χρειάζεται το σύστημα για τον υπολογισμό των μοτίβων συναρτήσει παραμέτρους όπως ο αριθμός των μοτίβων, το σύνολο των οντοτήτων, το σύνολο των χρονικών στιγμών των μοτίβων κ.α.

5.2 Μελλοντικές επεκτάσεις

Το εργαλείο μπορεί μελλοντικά να βελτιωθεί και να επεκταθεί.

Αυτή τη στιγμή η δυνατότητα zoom in/zoom out του διαγράμματος γίνεται σε μια εικόνα που έχει κατασκευαστεί από αυτό, μελλοντικά θα μπορούσε η εικόνα να αντικατασταθεί από ένα Swing αντικείμενο η το ίδιο το JTable ώστε να μπορούν παράλληλα να χρησιμοποιηθούν και οι υπόλοιπες δυνατότητες που προσφέρει το εργαλείο.

Επιπλέον, μια μελλοντική προσθήκη είναι η δυνατότητα να τραβηχτεί ένα στιγμιότυπο οθόνης μέσα στο αναδυόμενο παράθυρο που περιέχει το διάγραμμα με χρωματισμένα τα κελιά που συμμετέχουν στα μοτίβα.

Τέλος, θα μπορούσαν να γίνουν βελτιώσεις στο εμφανισιακό κομμάτι του εργαλείου.

Βιβλιογραφία

[BrZK10] Andrew Bragdon, Steven Reiss, Robert Zeleznik, Suman Karumuri, William Chang, Joshua Kaplan, Christopher Coleman, Ferdi Adeputra, and Joseph J. LaViola Jr. Code Bubbles: Rethinking the User Interface Paradigm of Integrated Development Environments, 2010 ACM/IEEE 32nd International Conference on Software Engineering, May 2010

[DeRo10] Robert Deline, Kael Rowan. Code Canvas: Zooming towards Better Development Environments, 2010 ACM/IEEE 32nd International Conference on Software Engineering, May 2010

[Giac15] T. Giachos. Biography Synopses for Evolving Relational Database Schemata. MSc Thesis, Dept. of Comp. Sc. and Eng., 2015, Available at https://www.cs.uoi.gr/wp-content/uploads/publications/MT-2015-18.pdf

[KuSt12] A. Kuhn and M. Stocker, "CodeTimeline: Storytelling with versioning data," 2012 34th International Conference on Software Engineering (ICSE), Zurich, 2012, pp. 1333-1336, doi: 10.1109/ICSE.2012.6227086

[Pant21] N. Pantelidis. A visualization system for the study of parallelly evolving time-series. Diploma Thesis, Dept. of Comp. Sc. and Eng., 2021