A Method to Establish Taxa of Schema Evolution

A Thesis

submitted to the designated
by the Assembly
of the Department of Computer Science and Engineering
Examination Committee

by

Georgios Theodoros Kalampokis

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN DATA AND COMPUTER SYSTEMS ENGINEERING

WITH SPECIALIZATION
IN ADVANCED COMPUTER SYSTEMS

University of Ioannina
School of Engineering
Ioannina 2021

Examining Committee:

- **Panos Vassiliadis,** Professor, Department of Computer Science and Engineering, University of Ioannina (Advisor)

- **Nikolaos Mamoulis,** Professor, Department of Computer Science and Engineering, University of Ioannina

- **Apostolos Zarras,** Associate Professor, Department of Computer Science and Engineering, University of Ioannina

DEDICATION

To my family.

ACKNOWLEDGMENTS

I would first love to thank the supervisor of my master thesis, Professor Panos Vassiliadis. His door office was always open whenever I ran into a trouble spot or had a question about my research or writing.
I would also like to thank my close friends for the cooperation that we had as well as for their support in a lot of situations.
Finally, most of all, I would love to express my gratitude and my love for my parents, Lampros and Elisavet, for the unconditional love and support they have showed me throughout my studies.

Ioannina, July 2021
Georgios Theodoros Kalampokis

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Georgios Theodoros Kalampokis, M.Sc. in Data and Computer Systems Engineering, Department of Computer Science and Engineering, School of Engineering, University of Ioannina, Greece, July 2021
Thesis Title: A Method to Establish Taxa of Schema Evolution
Advisor: Panos Vassiliadis, Professor

Software evolution is related to either the fix of any errors in the original or previous design of the software or the demand of the users to have additional features in the software. However, for the software to continue to be functional and viable, it needs to keep track of the new requirements. To achieve this, mostly, apart from the updates in the software, new information needs to be added. Schema evolution refers to the change of the internal structure of the database, either in terms of changes of the tables, or the attributes of the schema. The impact of this evolution on the entire software, that is built around the schema, is very big which makes it very important to find out how schemata evolve over time, as well as to extract some patterns related to their evolution.
Historically, due to the absence of datasets, only with the appearance of open-source software, was, the conduction of studies on schema evolution, made feasible. Recently, in bibliography the biggest study related to the schema evolution, has been conducted, with 195 schemata that were studied, in which, families of schemata (taxa) were extracted, by observing the way that schema evolves.
In this Thesis, we continue this research, and we proceed to the assessment of the taxa that were proposed. Moreover, we observe how the schema evolves over time as well as the measurements that are related to this evolution. The main question we seek to answer is: Assuming a taxon is given, what are the characteristics of the projects that belong to it and vice versa. Moreover, a question that arises is whether

we could find a centroid to each taxon, which would represent its characteristics. Finally, based on the answers that were derived from the study of the previous questions, we group the taxa in larger groups, which we call super taxa, and demonstrate a more clean separation of the evolutionary behavior of their projects. We also study heartbeat, activity, duration for the super taxa and report our findings, along with the identification of centroids for each super taxon.

Εκτεταμενη περιληψη

Γεώργιος Θεόδωρος Καλαμπόκης, Δ.Μ.Σ. στη Μηχανική Δεδομένων και Υπολογιστικών Συστημάτων, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πολυτεχνική Σχολή, Πανεπιστήμιο Ιωαννίνων, Ιούλιος 2021
Τίτλος Διατριβής: Μέθοδος καθορισμού τάξεων σχημάτων βάσεων δεδομένων, με βάση την εξέλιξή τους
Επιβλέπων: Βασιλειάδης Παναγιώτης, Καθηγητής

Η εξέλιξη του λογισμικού αφορά είτε τη διόρθωση τυχόν σφαλμάτων στον αρχικό ή προηγούμενο σχεδιασμό του λογισμικού, είτε την απαίτηση των χρηστών για πρόσθετα χαρακτηριστικά στο λογισμικό. Ωστόσο, προκειμένου το λογισμικό να συνεχίσει να είναι λειτουργικό και βιώσιμο, πρέπει να παρακολουθεί τις νέες απαιτήσεις. Για να επιτευχθεί αυτό, τις περισσότερες φορές, εκτός από τις ενημερώσεις του λογισμικού, πρέπει να προστεθούν νέες πληροφορίες. Η εξέλιξη του σχήματος αναφέρεται στην αλλαγή της εσωτερικής δομής της βάσης δεδομένων, είτε στα πλαίσια αλλαγών στους πίνακες είτε στα χαρακτηριστικά του σχήματος. Ο αντίκτυπος της εξέλιξης αυτής είναι πολύ μεγάλος σε ολόκληρο το λογισμικό, το οποίο είναι χτισμένο γύρω από το σχήμα, πράγμα που καθιστά πολύ σημαντικό να ανακαλύψουμε τον τρόπο με τον οποίο εξελίσσονται τα σχήματα με την πάροδο του χρόνου, καθώς και να εξάγουμε ορισμένα μοτίβα που σχετίζονται με την εξέλιξή τους.
Ιστορικά, λόγω της απουσίας συνόλων δεδομένων, μόνο με την εμφάνιση του λογισμικού ανοικτού κώδικα έγινε εφικτή η διεξαγωγή μελετών για την εξέλιξη των σχημάτων. Πρόσφατα, στη βιβλιογραφία, διεξήχθη η μεγαλύτερη μελέτη που έχει πραγματοποιηθεί ποτέ σχετικά με την εξέλιξη των σχημάτων, με 195 σχήματα που μελετήθηκαν και στην οποία, μελετώντας τον τρόπο με τον οποίο εξελίσσεται το σχήμα αυτών των έργων, εξήχθησαν οικογένειες σχημάτων (taxa).

Στην παρούσα εργασία, συνεχίζουμε την προσπάθεια αυτή και κάνουμε αξιολόγηση αυτών των οικογενειών που προτάθηκαν. Επιπλέον, παρατηρούμε πώς εξελίσσεται το σχήμα με την πάροδο του χρόνου καθώς και ποια είναι τα χαρακτηριστικά που σχετίζονται με αυτή την εξέλιξη. Το κύριο ερώτημα που θέλουμε να απαντήσουμε είναι, αν δοθεί μια οικογένεια, ποια είναι τα χαρακτηριστικά των έργων που ανήκουν σε αυτό, και αντίστροφα. Επιπλέον, ένα ερώτημα που προκύπτει είναι αν θα μπορούσαμε να βρούμε ένα κεντροειδές για κάθε οικογένεια, το οποίο θα αντιπροσωπεύει τα χαρακτηριστικά της εκάστοτε οικογένειας. Τέλος, με βάση τις παρατηρήσεις που προέκυψαν από τη μελέτη των προηγούμενων ερωτημάτων, ομαδοποιούμε τις οικογένειες σε μεγαλύτερες ομάδες, τις οποίες αποκαλούμε "super taxa", και επιδεικνύουμε μια πιο σαφή διάκριση της εξελικτικής συμπεριφοράς των σχημάτων τους. Επίσης, μελετάμε το heartbeat, τη δραστηριότητα και την διάρκεια των super taxa και αναφέρουμε τα ευρήματα μας, σχετικά με την εξακρίβωση κεντροειδών για κάθε taxon.

CHAPTER 1

INTRODUCTION

In the first section of this chapter, we present a brief description of our work and refer to the main directions and the main purpose of our research. In the second section of this chapter, we refer to the structure of this Thesis.

1.1 Goals

Many information systems, to meet market needs, use databases to store, manage, update, and recover data. A relational database consists of tables that are related to each other. A table has fields and records. All the tables of a database including their fields, their records as well as, their constraints, form the schema of the database.

As already mentioned, papers related to schema evolution exist and have done some important steps towards the discovery of behaviors and patterns in schema evolution. Nevertheless, the problem is that all the previous studies have been conducted on a small scale, up to a dozen of projects, and cannot ensure that the conclusions apply in general.

In [12], the author, taking advantage of a big number of open-source projects, conducted the largest study, that has ever been conducted in schema evolution and presented families of schemata based on the stats of the projects. The study aimed

to answer the following research questions: (a) Is schema evolution extensively present? (b) Can we extract families, ("taxa" as in biology) of schemata, with respect to the way they evolve over time? (c) What are the quantitative characteristics of schema evolution and how do they perform for different taxa? The author observed different types of schema evolution, so for this reason he introduced the concept of taxa, which are families of schemata with similar schema evolution. More specifically, the author presented 6 taxa of schema evolution for the domain of Free Open Source Software (FOSS) projects, which were extracted based on the activity and the heartbeat rate of the projects.
In this Thesis, we study in detail, the stats of every project for each one of these taxa, presented in [12], and we attempt to answer the following questions:

i) If given a taxon, can we say specific information about its values as well as its characteristics?"

After observing the stats of the projects and the plots for each taxon, we realized that there are some measurements, which are mostly related to the heartbeat and the activity, that can give us specific information about the schema evolution, during the lifetime of the projects.
Specifically, the characteristics of each taxon can be synopsized as follows:

Frozen taxon

- Zero total activity
- Up to 1 active commits
- Zero schema evolution

ALMOST_FROZEN:

- A few active commits (up to 4 active commits)
- Small total activity
- Most of the projects do not have new tables during their schema evolution

FocusedShot_n_Frozen:

- A few active commits (up to 4 active commits)
- Small to high total activity

- Nearly half of the projects of this taxon have one commit with new tables during their schema evolution

MODERATE:

- Moderate to several number of active commits (up to 23 active commits)
- Small to medium total activity
- Nearly 25% of the projects have several upward steps of evolution

FocusedShot_n_LOW:

- Moderate number of active commits (up to 11 active commits)
- Medium to high total activity
- Nearly 25% of the projects have several upward steps of evolution.

ACTIVE:

- Several number of active commits (up to 64 active commits)
- Very high total activity
- Nearly 50% of the projects have several upward steps of evolution.

ii) Can we give a "centroid" characteristic project of each taxon?

Considering the fact that the taxa presented in [12] were extracted mostly based on the activity and the heartbeat of the projects, in later section we present a project per taxon as a centroid project, including its measurements, and compare it to the general behavior of the taxon. This project represents the characteristics of each taxon with respect to the way its schema evolves and is defined by using measurements related to the heartbeat and the activity of the projects.

iii) *Can we do better than the existing taxonomy?*

After having observed the stats and the plots of all the measurements and especially of those that are related to the heartbeat and the activity of the projects, in Chapter 3, we came up with the idea of merging the similar taxa into larger groups, which we call super taxa, and present them extensively in Chapter 4.

1.2 Thesis Structure

This Thesis consists of 5 sections. Its structure is as follows:
In Section 2, we present the related work and the background of this Thesis.
In Section 3, we discuss the procedure that was followed for the assessment of the taxa, which were presented in [12], and present the results and the conclusions that we end up with.
In Section 4, we examine the possibility of deriving super taxa, by grouping similar taxa into larger groups and discuss the benefits of this procedure.
In Section 5, we discuss the conclusions and the results of this Thesis and also, we refer to potential future work.

CHAPTER 2

RELATED WORK

In the first section of this chapter, we refer to related research that has been done in this specific area, emphasizing how the interest in this area was grown, as well as on the achievements over time. In the second section, we make a quick overview of the paper "Schema Evolution Profiles from the Study of 195 Free Open-Source Software Projects", since this Thesis is an extension of this study.

2.1 Case Studies of Schema Evolution

Sjøberg in his paper [7] conducted a study for measuring modifications to database schemata and their consequences. To accomplish that, the author built a thesaurus tool to monitor the evolution of a large, industrial database application – a health management system (HMS). The thesaurus assists in keeping track of the use of names in the HMS application and reports to user information such as which actions, classes, functions, macros, etc. are defined and where they are used, which fields and relations, this query or update function refers to, etc. It does not perform any changes or conversions itself but instead, it indicates where changes probably have to be done. Finally, the author reports how the schema changed, and concludes that even

a small change to the schema may have major consequences for the rest of the application code.

Specifically, the author's findings can be summarized as follows:

- At the beginning of the development, almost all changes were additions.
- After the system went into production use, there was no diminution in the number of changes, but the additions and deletions were more nearly in balance.
- Every relation has been modified throughout the examination period.
- During the period of examination of HMS, there was a 139% increase in the number of relations and a 274% growth in the number of fields.

Curino, Moon, Tanca, and Zaniolo in their paper [2] present an in-depth analysis of the evolution of the Wikipedia database and its schema, which was short in time, but intense in terms of growth and evolution. In the context of their study, the authors performed a macro and micro classification of the schema changes and then they measured the effect of the changes on applications, by observing the success rate of the query execution among different schema versions.
The main findings of this study are condensed as follows:

- Throughout the analysis, the number of tables increased from 17 to 34 (100% increase) and the number of columns from 100 to 242 (142%).
- Regarding the table/column lifetime as well as the number of revisions per month, each table lasted 103.3 versions (60.4% of the total DB history) and columns lasted 97.17 versions on average (56.8% of the total DB history). The peak of the most revisions per month was spotted in the middle of the timeframe that the database was examined.
- Interestingly, it was noticed that there were two main groups of tables and columns: "short-living" and "long-living".
- Only a small fraction (about 22%) of the queries, designed to run on old schema versions, were still valid throughout the schema evolution.

- The authors, in order to provide a fine-grained analysis of the types of change the schema has been subjected to, exploited Schema Modification Operators (SMOs) as a pure classification instrument. SMO's syntax is similar to that of SQL, DDL, and provides a concise way to describe typical modifications of a database schema and the corresponding data migration. The authors found that the most used SMOs were "ADD COLUMN" and "DROP COLUMN".
- The results of the study showed that MediaWiki has undergone a very intensive schema evolution, as a result of the cooperative, multi-party, open-source development, and administration.

Lin's and Neamtiu's research [4] aimed to identify challenges and solutions associated with the collateral evolution of application programs and databases. The authors define, the situation when the format expected by the data client is different from the format provided by the data server, which may have unexpected behavior, as collateral evolution.
To identify which the most frequent table changes are, *the authors performed a schema evolution study on two real-world widely used applications Mozilla and Monotone*.

- The author's research has revealed that the most frequent table-level modifications are interval changes to the schema of existing tables, followed by table additions, table deletions, and table renaming.
- Regarding the attributes of the tables, most of the changes were additions and deletions.

Afterwards, the authors' purpose was to find out how the application code remains synchronized with the new schema version. To accomplish that, *Mozilla and Monotone were examined, to inspect how these two applications deal with the schema changes*.

- Monotone used a centralized routine to check whether collateral evolution has occurred, and inform the user whether the database is usable, or if migration is required, etc.
- Mozilla had two main approaches to cope with schema changes: The first mechanism (Version-oblivious evolution) simply ignores the collateral evolution problem and assumes that, if a database exists, its schema version matches

the schema version of the application, which may result in unexpected errors. The second approach (Bidirectional schema migration) determines, before accessing the database, both the version X of the application and the version Y of the database schema, and then perform the schema migration, which can be either an upgrade or a downgrade.

Although table and attribute changes are under application program developers' control, the developers face changes they have little control over, like the database file format. The reason why this happens is that DBMS producers often modify the database file format to offer improved performance, reduce storage size, or implement a new standard. *The authors carried out a database format evolution study over the complete lifetime of three major DBMSs (SQLite, MySQL, PostgreSQL).*
Compatibility errors could occur due to different versions of DBMS. MySQL and PostgreSQL face this challenge by backing up the existing data, "dump" the DB contents to a SQL script containing the commands needed to recreate all the database records from scratch, upgrade the DBMS, and run the script to recreate and populate the database at the new format.
Regarding the format changes of each DBMS, they are as follows:

- MySQL over its 14-year existence has had 5 file format changes.
- PostgreSQL over its entire 14-years lifetime has had 21 file format changes.
- SQLite over its 9 years lifetime, has changed the file format 13 times. Though only 3 of those were incompatible file format changes. It seemed to be much more user-friendly. Seamless file format conversion mechanisms used in SQLite should be adopted by other DBMS producers as well.

Wu's and Neamtiu's research [13], focused on schema evolution for embedded databases. The goal of the authors' work was to find a way that would permit safe, dynamic schema updates to embedded databases (EDs).
The main contributions and results of this research were the following:

An approach for extracting ED schemas and detecting schema evolution. The first step towards this goal was to understand how ED schemas evolve. So firstly, the authors created a tool (SCVD) that automates schema extraction and schema evolution

analysis for EDs. Using SCVD, developers can compare old and new applications to find out when and how to correctly migrate an ED from the old schema to the new schema. Regarding the SCVD architecture, it consists of 3 stages:

- In the first stage, the system tracks the release history of an application (source code history extractor).
- In the second stage, it extracts the database schemas embedded in the application (schema extractor). More specifically, it takes as input a list of versions or tags and downloads a corresponding list of source code versions the authors want to analyze.
- In the third stage, it compares the schemas and produces a tally of schema evolution results (schema differencing module). This module is based on mysqldiff, an open-source schema migration assistant.

A study of ED schema evolution of four popular applications over more than 18 cumulative years.

To understand how EDs evolve in practice, the authors used the aforementioned tool (SCVD) to perform a schema evolution study covering a cumulative 18 years of evolution, on four popular open-source programs: Firefox, Monotone, BiblioteQ, and Vienna. The authors' study focused on the table- and attribute-level changes that affect update safety. More specifically, the table changes that were examined were the SMOs "CREATE TABLE" and "DROP TABLE" and the attribute changes were the following: "ADD COLUMN", "DROP COLUMN", "Type Change", "Init Change", "Key Change". The results, after summing up the changes across all the applications, were the following:

- The most frequent operations were ADD COLUMN (32.5%), Type Changes (24.3%), DROP COLUMN (19.3%), DROP TABLE (14.9%), and CREATE TABLE (6.1%). After having observed the previous results, the authors found out that table and column additions/deletions are more important than supporting changes to column types, initializers, and key status.
- The authors also measured the frequency and timing of schema changes for each project. It was observed that schemas tend to change more in the

beginning, and the database structure stabilizes over time because later versions have fewer changes. For that reason, the authors suggested that on-the-fly schema updates are necessary, especially at the beginning of a program's lifetime.

Qiu, Li, and Su in their paper [6] paid attention on how schema-changes impact code and performed a large-scale empirical study on ten popular database applications to gain insight into how schemas and application code co-evolve. In particular, the authors studied the applications' long-time evolution histories from their respective repositories, to understand 1) *if database schemas evolve frequently and significantly*, 2) *how schemas evolve*, and, 3) *how they impact application code*.

Firstly, the authors provided a method that describes their analysis process over the schema and data changes. The method consists of the following steps:

- Locate schema file: The first step extracts the schema files. The authors manually trace the schema files even if their locations or names have been modified.
- Extract DB revisions: The second step identifies DB revisions, which are revisions (commits) that contain modifications to schema files.
- Extract valid DB revisions: Filter DB revisions and keep only those that the authors were interested in.
- Extract atomic changes: After having identified the valid DB revisions for each project, the authors extract all schema changes by manually comparing schema files of contiguous valid DB versions.
- Co-change analysis: Analysis of the real impact caused by these atomic schema changes by mining a project's version control history. The authors use a database application's co-change history to estimate the application code area, affected by a schema change.

After applying the first two steps of the previous method, the authors came up with the following results:

- The ratio of valid over total revisions fell mostly in the 50-90% range.
- Compared to other projects, Joomla had a much lower ratio because DMLs were involved in the same schema file with DDLs, making data-sensitive changes cover a large fraction of invalid revisions.
- The average number of atomic changes per valid revision fell mostly in the 2-7 range.

Finally, the authors considering the previous analysis provided answers and conclusions to the following questions.

"How frequently and extensively do schemas evolve?" The authors, to answer this question, had to do the following steps:

- Firstly, for each stable release/year, the authors calculated the average number of valid DB revisions/atomic schema changes.
- Secondly, it was measured how extensively schemas change, by examining the trend on schema size changes. The authors collected the number of tables/columns in each valid revision to see how much schemas evolve.

To facilitate a more precise evaluation, the authors used two metrics, Growth Rate (GR) and Change Rate (CR):
GR = (#Added Elements - #Deleted Elements) / Initial Elements
CR = (#Added Elements + #Deleted Elements) / Initial Elements

The main results were the following:

- Schemas evolve frequently: On average 65 atomic schema changes occurred per release, and 90 atomic schema changes occurred per year across the ten projects.
- The size of schemas in most projects grew significantly: The GR of tables in 60% of the projects exceeded 100%; the CR of tables in 90% of projects exceeded 100%.

- Columns evolve in a way that is very similar to the one of the tables.

"How do database schemas evolve?" The main results that the authors came up with were the following:

- At the low-level, add table, add column, and change column datatype were the most frequent atomic change types.
- The data also confirmed that referential integrity constraints (such as foreign key and trigger) and procedures (such as stored procedure) are rarely used in practice.
- Addition and change accounted for most of the schema evolution.

"How much application code has co-changed with a schema change?" To answer this question, the authors selected uniformly a random 10% (146) of the valid DB revisions from the total 1,464 valid DB revisions and manually analyzed the co-changed information.

The main results, that the authors ended up with, were the following:

- Their detailed manual study on schema and code co-change history revealed that more than 70% of all valid DB revisions contained effective co-change information, and among these, over 70% have precisions over 80%.
- Schema changes impacted code greatly. For an atomic schema change, developers needed to change about 10~100 LoC on average. For a valid DB revision, which typically contains 25 atomic changes, developers needed to change about 100~1000 LoC.

Skoulis, Vassiliadis, and Zarras [8] during their research performed a large-scale study on the database evolution of large open-source projects and checked the validity of Lehman's laws on properties like size, growth, and amount of change per version. More specifically, the authors isolated databases of eight projects that appeared to be alive. For each dataset, the authors gathered its schema versions, and then used their tool Hecate to get the differences between the two subsequent

committed versions and measures such as the size of the schema, the total number of changes for each transition from a version to the next, or the growth assessed as the difference in the size of the schema between subsequent versions.

The main conclusions that the authors ended up with are the following:

- All projects, with only one exception, had schema changes throughout their lifetime. Database schema evolution happened in discrete time slots and wasn't a continuous process. *The authors concluded that the Law of Continuing Change partially holds*.
- *The evolution of the database schema appeared to obey the behavior of a feedback-based mechanism*.
- Referring to the evolution of size, it was noticed that the schemas follow three fundamental behaviors. In all schemas, exist periods of increase, especially at the beginning of their lifetime or after a large drop in the schema size. Moreover, there were versions with drops in schema size. Those drops were typically sudden and steep and usually took place in short periods of time. Also, in all schemas, there were periods where the size remained stable. In terms of schema growth, change was small. Tables' growth was mostly ranged in small values. Also, the same behavior was noticed for the attributes' growth. *Considering those, the authors concluded that change does not follow the pattern of baseline smooth growth of Lehman*.
- Even though all data sets demonstrated the tendency to grow over time, in all schemas, there were periods of stability where the size of schema did not change, so *the authors concluded that the Law of Continuing Growth holds*.
- *The Law of Conservation of Organizational Stability does not hold*, since there was no constant growth in any of the projects.

Vassiliadis, Zarras, and Skoulis [10] focused on how properties associated with schema evolution such as life duration, or the number of updates of a table are related to observable table properties like the number of attributes or the time of birth of a table. The authors provided answers and conclusions to the following questions:

"Which tables eventually survive, and which ones are deleted?"

According to the authors, the tables can be segregated to the following families:

- *Wide survivors:* Tables with small schema sizes can have various lifetime durations and tables with larger schema sizes live longer.
- *Entry level removals:* Newly born tables tend to be quickly removed with a few or no updates observed.
- *Old timers:* It was noticed that old age tables are rarely removed.

"What is the impact of the lifetime of a table on schema size and vice versa?"

Intending to answer this question, the authors computed the duration for each table in each dataset. To get a normalized measure, the duration of each table was divided by the duration of its database.

Specifically, the author's results and conclusions can be summarized as follows:

- It was observed that long-lived tables tend to appear at the beginning of the database and survive till the end.
- A 26.11 % fraction of tables that appeared at the beginning of the database, survived until the end.
- Nearly half of the tables (approx. 47 %) were small tables with less than 5 attributes.
- The tables with 5 to 10 attributes were approximately one-third of the tables' population and the wide tables with more than 10 attributes were approximately 17 % of the tables.
- The datasets with less evolutionary activity were the ones concentrating outlier values.
- Tables with small schema sizes can have arbitrary durations, whereas tables with larger schema sizes last long.
- Small schema size does not necessarily mean short duration. Nonetheless, tables with 10 or more attributes have high chances of surviving.
- Tables at old age are rarely removed.

"How is table schema size and duration related to its update potential?"

The authors after observing the relation of a table's schema size at its birth with the amount of change the table undergoes, concluded the following two main clusters. The first cluster consists of small size tables with a small amount of change and the second cluster is divided into two subcategories: (i) Medium schema size tables with medium to large amount of changes and (ii) Tables with large schema size with small to medium amount of change.

Additional conclusions that the authors ended up with are as following:

- Considering the table updates, large size tables seem to have fewer possibilities of growth, in contradiction with medium size tables which can carry more information.
- Tables with small lifetimes are subject to small changes, while tables with medium duration undergo small or medium change, and, long-lived tables demonstrate all kinds of change behavior.
- Tables with the highest average transitional update, are born early, live long, and have consequently a large amount of total update.

Panos Vassiliadis and Apostolos V. Zarras in their research paper [11] aimed to discover patterns and behaviors that are tightly related to the survival or the death of the tables of a database. For the purposes of their study, the authors used their tool Hecate, to the schema files of 8 open-source projects, to get measures such as the total number of changes it went through and the change rate. After having examined the graphs, which included measurements, mentioned above, the authors deduce the following facts:

- *"Dead" tables tend to have a short lifetime, which makes sense, because the earlier a table is removed, the smaller the cost of maintaining the surrounding code is.*
- *On contrary to the "dead" tables, "survivor" tables usually have high durations.*
- *Probably the most important finding of the authors was "The electrolysis pattern".* This pattern states that tables, which die, are usually tables with small or medium lifetimes and undergo fewer changes. In contradiction, tables that

survive, usually present medium or big durations and have the tendency, to undergo a greater number of changes, the longer they live.

- *Finally, the authors suggest that the developers try keeping the development of the, related to the tables, code as restrained as possible, preferably encapsulated via views, that will hide the changes from the application code.*

2.2 Overview of the paper "Schema Evolution Profiles from the Study of 195 Free Open Source Software Projects"

The author, in [12], conducted the largest study, that has ever been conducted in schema evolution, and after collecting a big number of open-source projects, he presented families of schemata based on the way they evolve throughout their development lifetime. For the generation of the datasets, the author queried the GitHub Activity Data dataset from Google Cloud BigQuery, for repositories that had .sql files. Then, the author filtered original repositories, with more than 0 stars and more than 1 contributor and finally, after discarding projects with just a single commit, ended up with 195 projects, with at least an extra commit. Once the histories of these 195 projects were collected, The author proceeded to the automatic extraction of changes in the lifetime of projects by using a tool called Hecate. These changes involve table and attribute birth and removal, as well as data type and PK (Primary Key) changes. The last step was the production, of stats and charts for each project, which were used for the extraction of the taxa.
The taxa that emerged at the end of this study [12] are as follows:

- ***0_FROZEN***: Projects with completely frozen schema histories and with zero change at the logical level. (totalActivity = 0)
- ***1_ALMOST_FROZEN***: Projects with histories of very small change, typically with few intra-table attribute modifications. (At most 4 active commits, totalActivity <= 10 updated attributes).

- ***1_FocusedShot_n_FROZEN***: Projects with almost frozen histories but with a single spike of change (not necessarily small) and almost no other change (At most 4 active commits, totalActivity $>$ 10 updated attributes).
- ***2_MODERATE***: Projects with histories of moderate evolution, without spectacular changes, but rather small deltas spread throughout their lives.
- ***3_FocusedShot_n_LOW***: Projects with evolution similar to the moderate one but also with a pair of spikes on their activity.
- ***4_ACTIVE***: Projects with a significant amount of change both as intra-table change and in terms of table generation and eviction.

CHAPTER 3

PROFILING AND EVALUATION OF THE EXISTING TAXONOMY OF SCHEMA EVOLUTION

This chapter critically assesses the taxa presented in [12], through a detailed analysis. To achieve this analysis, some new data and statistics needed to be extracted from the taxa. In the first section of this chapter, an overview of the tool "HeraclitusFire"

as well as the data and statistics, that were extracted and added as information to "HeraclitusFire", are presented. In the second section of this chapter, the main tests that were added to "HeraclitusFire" are reported, to make sure that every information added to the project was valid, as well as to test most of the use cases related to the schema evolution of the input. Afterwards, the third section discusses the correlations of the attributes. More specifically, firstly the metric that was used to compute the correlations of the attributes is presented, then the computed correlations are presented, and at the end of this section, based on this correlation matrix, the most important attributes are extracted. In the fourth section, a profiling of the most important attributes, including their histograms, is provided. In the fifth section, we discuss about patterns and characteristics for each taxon. Then, in the sixth section, we present a centroid-project for each taxon, that represents the characteristics of the respective taxon. Finally, in the last section we proceed to the assessment of the existing taxa, presented in [12], by using some validity metrics from the area of clustering.

3.1 Data and Statistics Extraction from "HeraclitusFire"

Firstly, let's take a look at the tools, that were used in terms of this research. "HeraclitusFire" is a tool that, given the history of a relational database schema, automatically produces visualizations and statistical tests for patterns of schema evolution. As input to "HeraclitusFire" is given the output of "Hecate", which is a tool that, given a folder with the snapshots of the DDL files as input that include the CREATE TABLE statements of the databases, produces transitions from one snapshot ("version") to its next, along with statistics on the types of changes. The data and statistics produced by "HeraclitusFire", were used in our detailed analysis. However, these data were not enough for the goals of this research, because it was important to automatically extract stats like continuous evolution and progressive evolution, so that more attributes and statistics were added. In the following three subsections, the specific measures of evolution that were added to "HeraclitusFire" by category are mentioned, with a description of their meanings. The new information after being calculated is stored in the appropriate CSV file.

3.1.1 Atomic Schema Attributes

Regarding the atomic transition information, having taken into consideration the total activity of each commit of a project, the following attributes, in Table 3.1, that characterize it, were added.

Table 3.1 Atomic Schema Attributes with their descriptions

Attribute	Description
active	A commit is called active if its total activity is greater than 0.
turf	A commit is called turf if it is active, and its total activity is less than 15.
reed	A commit is called reed if its total activity is greater than 15.

These metrics were essential for the computation of the monthly and the summary schema stats, which will be discussed in the next subsections. The attributes turf and reed, explain the evolution in a better way because they distinguish the small from the big evolutions of schemas.

3.1.2 Monthly Schema Attributes

The usage of the month as a time unit was convenient in our case, in terms of the study as well as the visualization of the new data. These monthly measurements provide a continuous regular timeseries, where no value is missing. *In monthly stats, the first month includes not only the commits of that month, but also the initial commit of the project (v0).* As far as the monthly schema additions are concerned, the new attributes for each month are presented in Table 3.2.

Table 3.2 Monthly Schema Attributes with their descriptions

Attribute	Description
reeds	The number of commits that are reeds.

reedRatioAComm	The number of reeds divided by the number of active commits.
reedRatioTComm	The number of reeds divided by the number of total commits.
activityDueToReeds	Sum of the total activities of all commits that are reeds.
turfs	The number of commits that are turfs.
turfRatioAComm	The number of turfs divided by the number of active commits.
turfRatioTComm	The number of turfs divided by the number of total commits.
activityDueToTurfs	Sum of the total activities of all commits that are turfs.
activeCommits	The number of active commits.
activeCommitRatio	The number of active commits divided by the number of total commits.

3.1.3 Summary Schema Attributes

The taxa presented in [12] were produced on the basis of summary information that separates the initial commit (v0) from the rest, which means that the measures of the initial commit (v0) are not being taken into consideration for the calculation of the summary stats. Regarding the summary stats, that involve aggregate measurements for the entire life of a schema, the following attributes, in Table 3.3 were added:

Table 3.3 Summary Schema Attributes with their descriptions

Attribute	Description
reeds	The total number of commits of the project that are reeds.
reedsPostV0	Equal with *reeds* if the first commit is a turf, or (*reeds* – 1) if the first commit is a reed.

reedRatioAComm	The number of *reeds* divided by the number of active commits.
reedRatioTComm	The number of *reeds* divided by the number of total commits.
activityDueToReeds	Sum of the total activities of all commits that are reeds.
activityDueToReedsPostV0	Equal with *activityDueToReeds* if the first commit is a turf, or (*activityDueToReeds – totalActivityV0)* if the first commit is a reed.
turfs	The total number of commits of the project that are turfs.
turfsPostV0	Equal with *turfs* if the first commit is a reed, or (*turfs* – 1) if the first commit is a turf.
turfRatioAComm	The number of *turfs* divided by the number of active commits.
turfRatioTComm	The number of *turfs* divided by the number of total commits.
activityDueToTurf	Sum of the total activities of all commits that are turfs.
activityDueToTurfPostV0	Equal with *activityDueToTurf* if the first commit is a reed, or (*activityDueToTurf – totalActivityV0)* if the first commit is a turf.
activeCommits	The number of active commits.
activeCommitRatePerMonth	The total number of active commits divided by the total number of months the project is alive.
commitRatePerMonth	The total number of commits divided by the total number of months the project is alive.
activeCommitRatio	The total number of active commits divided by the total number of commits.

3.2 Testing of Extracted Data

After the calculation and storage of the data, before their preprocessing, their validity had to be verified. Also, it was very crucial to be confirmed that the changes had not affected the general functionality of the project. To accomplish that, a multitude

of tests for the most important use cases was generated. Moreover, the validity of the extracted data, of five more schemas of projects, was tested, and "truth" files with the expected valid data results were created. The most important use cases for this research, which were tested, are the following:

- *loadData*
- *extractMonthlySchemaStats*
- *extractSchemaLevelInfo*

To confirm that the abovementioned use cases are working properly, their tests on 5 more projects were conducted. Finally, every test for each one of the six projects was fired, and it was realized that everything was working properly and that the calculated data were equal to the expected valid ones.

3.3 Correlations of Attributes

To evaluate the taxa presented in [12], the similarity of the objects of each cluster as well as the degree of separation between the clusters needed to be examined. The attributes were too many to process, so a subset of attributes needed to be selected, by discarding the most correlated attributes. In this section, an overview of the metric that was used, to compute the correlations of the attributes, is provided. Also, the correlations that were computed, as well as the most important attributes that were resulted according to the correlations, are presented.

3.3.1 Kendall Metric

For the calculation of the correlations of the data, the Kendall's Tau Coefficient formula [3] was used which is as follows:

$$\tau = \frac{n_c - n_d}{n(n-1)/2}$$

where:

n_c : number of concordant pairs

n_d : number of discordant pairs

n : number of observations

The reason why Kendall's Tau Coefficient was chosen, is because the sample was small so its complexity O(n^2) was not discouraging. Besides, Kendall's Tau Coefficient manages the outliers effectively because this coefficient is computed based on the rankings of the values, instead of the values themselves.

The steps that need to be done, before using the formula, are as follows:

- Calculate the number of observations.
- Compute the rankings for both of the attributes we want to observe their correlation.
- Then for every pair of ranks (x1, y1) and (x2, y2)
    - If the pair (x1, y1) and (x2, y2) has the property that

      sgn(x2 - x1) = sgn(y2 - y1)

      where $sgn\, x = \begin{Bmatrix} -1 & : & x < 0 \\ 0 & : & x = 0 \\ 1 & : & x > 0 \end{Bmatrix}$

      then, that pair is marked as *concordant pair* (C)

    - Else if the pair (x1, y1) and (x2, y2) has the property that

      sgn(x2 - x1) = - sgn(y2 - y1)

      then that pair is marked as *discordant pair* (D)

- Finally, count the number of concordant pairs (n_c) and the number of discordant pairs (n_d) and apply the formula.

3.3.2 Correlations

After following the aforementioned steps for every pair of our attributes, all the correlations are presented in Figure 3.1.

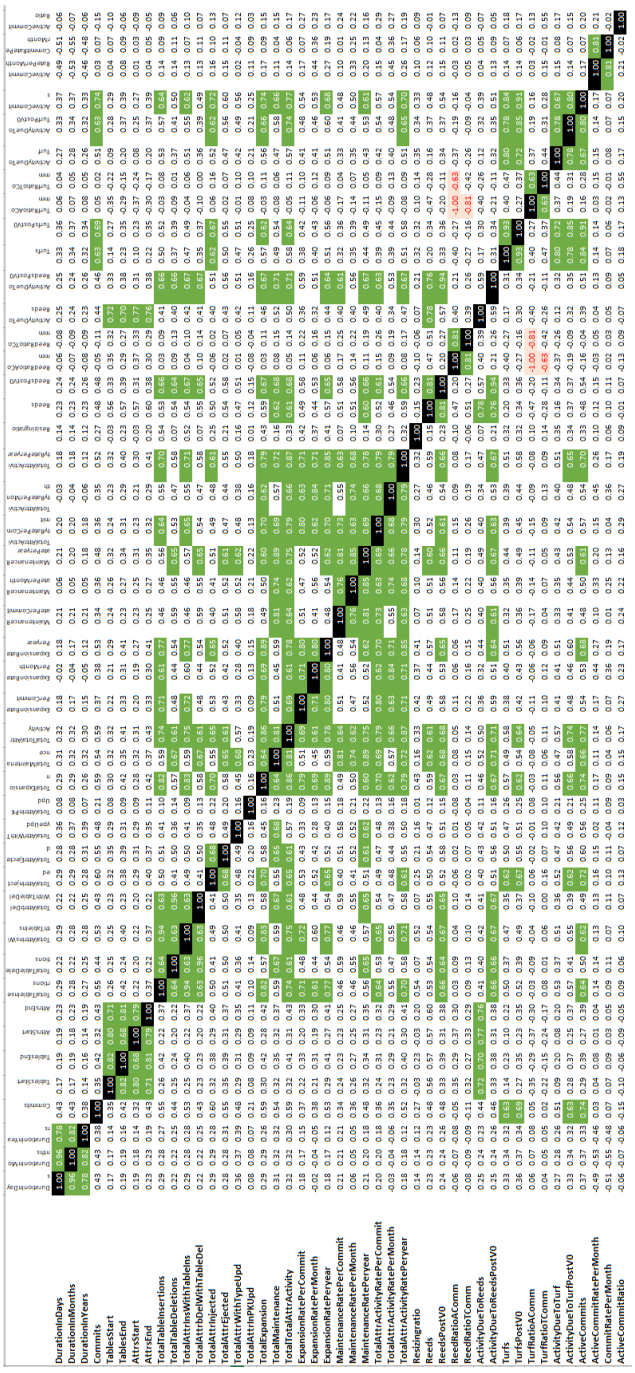


Figure 3.1 Kendall's Tau correlations of data

In Figure 3.1, the symmetric table of correlations for every pair of attributes is presented. Pairs of attributes with correlation values greater equal than 0.6 represent correlated attributes, while other pairs with correlation values less or equal than -0.6 represent anticorrelated attributes.

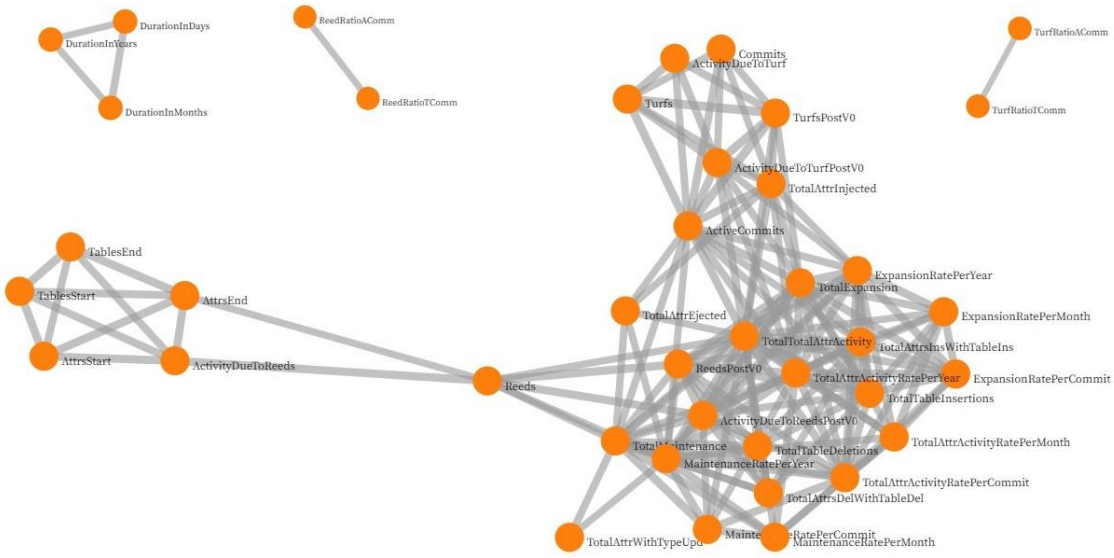


Figure 3.2 Network graph with high correlations between attributes as edges

Figure 3.2 shows all the attributes with their correlations as a graph representation. More specifically, it presents a network, with the attributes as nodes, and the high positive correlations (greater than 0.6) as edges, where the closer the nodes are to each other, the higher the correlation between them is.

3.3.3 Most Important Attributes

Since the number of attributes was very big to process, a procedure was followed to determine the most important attributes. Taking Figures 3.1, 3.2 into consideration, the steps of that procedure are as follows:

First of all, the attribute (*TotalTotalAttrActivity*) was determined from the start to be one of the most important attributes, because of its high significance. The attribute (ResizingRatio), since it was not highly correlated with any other attribute, was marked as significant.
Squares of green values declare a big correlation between the respective pairs of attributes. So, for each such green square, a single attribute was determined to be one of the most important.

- Green Square *(DurationInDays, DurationInMonths, DurationInYears)*
  These attributes were highly correlated. DurationInMonths was determined to be one of the most important attributes.

- The following green squares are associated with specific "activity functions", so since *TotalTotalAttrActivity* was the most general "activity measure", only this was added to the most significant attributes.
  - Green Square *(TablesStart, TablesEnd, AttrsStart, AttrsEnd).*
  - Green Square (*TotalTableInsertions, TotalTableDeletions, TotalAttrInsWithTableIns, TotalAttrbDelWithTableDel).*
  - Green Square (*TotalExpansion, TotalMaintenance, TotalTotalAttrActivity, ExpansionRatePerCommit, ExpansionRatePerMonth, ExpansionRatePerYear, MantainanceRatePerCommit, MantainanceRatePerMonth, MantainanceRatePerYear, TotalActivityRatePerCommit, TotalActivityRatePerMonth, TotalActivityRatePerYear).*

- These attributes (*TotalAttrInjected, TotalAttrEjected, TotalAttrWithTypeUpd, TotalAttrInPKUpd)* did not form a green square but were associated with specific "activity functions" so we decided to keep none of them.

- As far as these attributes (*Reeds, ReedsPostV0, ReedRatioAComm, ReedRatioTComm, ActivityDueToReeds, ActivityDueToReedsPostV0)* are concerned, they are associated with the same metric. The attributes *Reeds, ReedsPostV0*, and *ActivityDueToReedsPostV0* seemed to be highly correlated with most of the attributes, while *ReedRatioAComm, ReedRatioTComm* had close to zero

correlation with the main attribute *TotalTotalAttrActivity*, so in the end, only the *ActivityDueToReeds* attribute was marked as significant.

- These attributes (*Turfs*, *TurfsPostV0*, *TurfRatioAComm*, *TurfRatioTComm*, *ActivityDueToTurf*, *ActivityDueToTurfPostV0*), refer to the same metric. Since *ActivityDueToReeds* was chosen in the previous step, the attributes *ActivityDueToTurf*, *ActivityDueToTurfPostV0* were instantly discarded, and then, based on the correlation matrix, the attributes *Turfs* and *TurfRatioTComm* were marked as significant.
- Finally, concerning these attributes (*ActiveCommits*, *ActiveCommitRatePerMonth*, *CommitRatePerMonth*, *ActiveCommitRatio)*, the attribute *ActiveCommits* was highly correlated with most of the attributes, so it was instantly discarded. The *ActiveCommitRatePerMonth* and *CommitRatePerMonth* attributes seemed to be highly correlated, so we decided to keep only *CommitRatePerMonth*. The attribute *ActiveCommitRatio* was also marked as significant.

After discarding the less important attributes according to the correlation matrix in Figure 3.1, the attributes that finally were marked as important are presented in Figure 3.3.

	DurationInMonths	TotalTotalAttrActivity	Resizingratio	ActivityDueToReeds	Turfs	TurfRatioTComm	CommitRatePerMonth	ActiveCommitRatio
DurationInMonths	1.00	0.32	0.14	0.24	0.34	0.05	-0.55	-0.07
TotalTotalAttrActivity	0.32	1.00	0.33	0.50	0.58	0.11	0.06	0.17
Resizingratio	0.14	0.33	1.00	0.07	0.32	0.14	0.06	0.09
ActivityDueToReeds	0.24	0.50	0.07	1.00	0.17	-0.26	0.05	-0.07
Turfs	0.34	0.58	0.32	0.17	1.00	0.47	0.07	0.18
TurfRatioTComm	0.05	0.11	0.14	-0.26	0.47	1.00	-0.01	0.55
CommitRatePerMonth	-0.55	0.06	0.06	0.05	0.07	-0.01	1.00	-0.02
ActiveCommitRatio	-0.07	0.17	0.09	-0.07	0.18	0.55	-0.02	1.00

Figure 3.3 Most Important attributes

3.4 Data Profiling

In this section, we profile all the attributes of Figure 3.3 that were marked as significant. More specifically, for every attribute, the minimum value, the maximum value, the average, the standard deviation, are computed and a histogram plot is provided. Before performing data profiling, having observed the values of the dataset, two projects ("opencart", "cgrates") of the dataset seemed to have extremely outlier values. Due to that, we have decided not to take these outliers into account to prevent them from affecting the results of the assessment of the existing taxa.
Therefore, *in all our subsequent results we work with only 193 of the 195 projects*.
Figure 3.4 consists of a table with the data profiling metrics of the most important attributes. Figures 3.5 – 3.12 present the histograms of these attributes.

Attribute	Min	Max	Average	Median	StdDevP	Count	NullCount	Distinct Count
DurationInMonths	1	100	15.61	8.00	19.13	193	0	53
TotalTotalAttrActivity	0	1267	57.34	10.00	137.61	193	0	78
Resizingratio	0.085	6.5	1.34	1.00	0.92	193	0	52
ActivityDueToReeds	0	1665	104.30	27.00	218.42	193	0	92
Turfs	0	57	5.05	2.00	8.88	193	0	26
TurfRatioTComm	0	1	0.47	0.50	0.30	193	0	48
CommitRatePerMonth	0.03	12	1.74	1.00	1.98	193	0	78
ActiveCommitRatio	0.09	1	0.68	0.71	0.25	193	0	47

Figure 3.4 Data Profiling Metrics for 193 projects

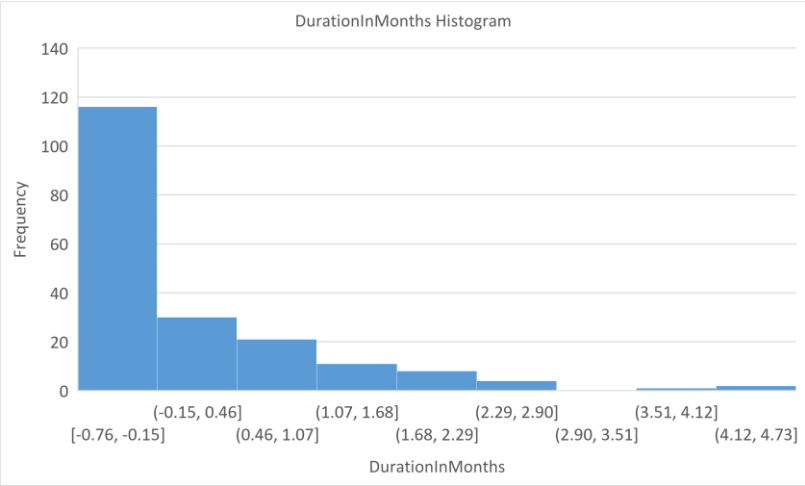


Figure 3.5 Histogram of DurationInMonths

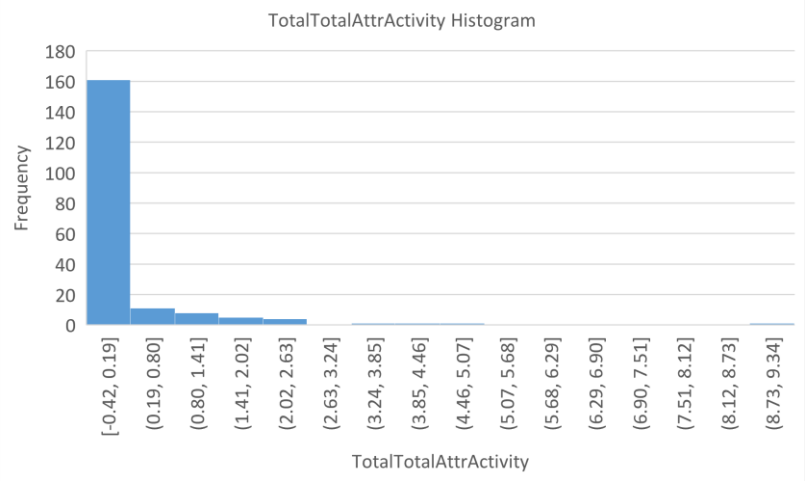


Figure 3.6 Histogram of TotalActivity

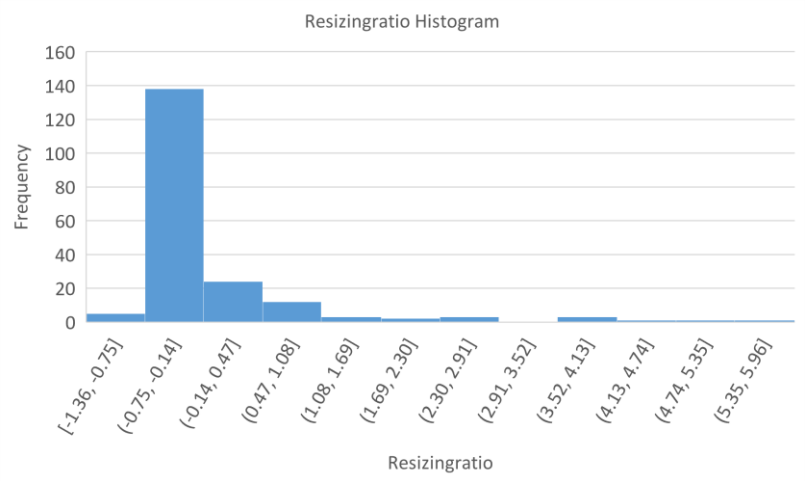


Figure 3.7 Histogram of ResizingRatio

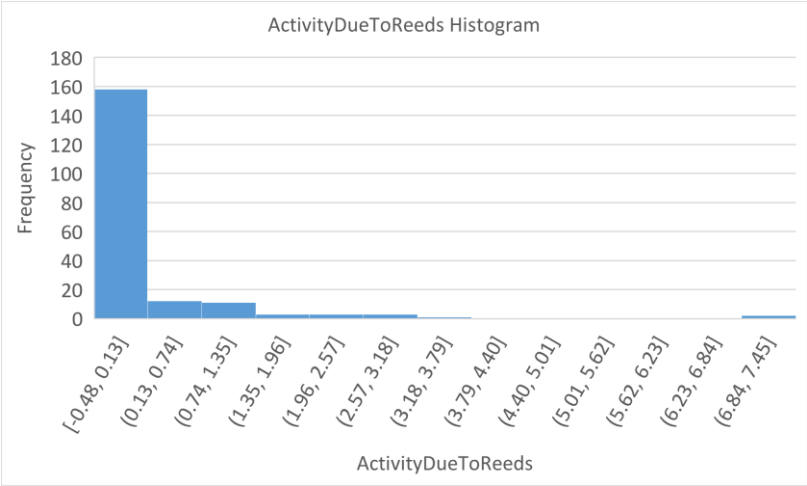


Figure 3.8 Histogram of ActivityDueToReeds

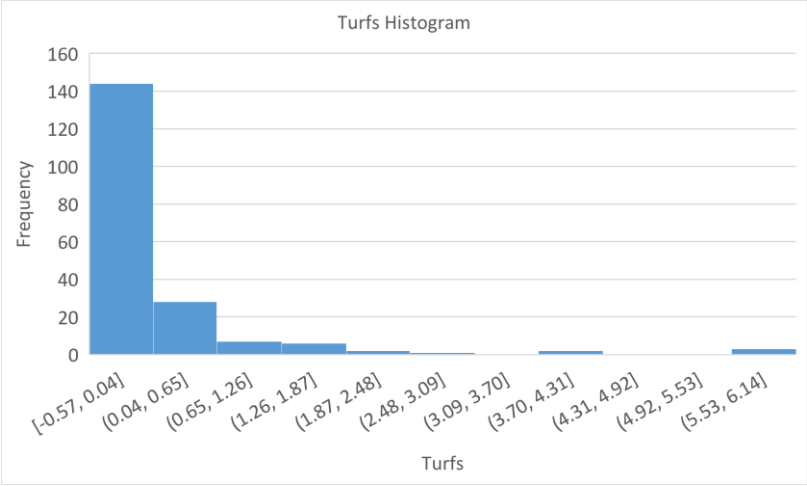


Figure 3.9 Histogram of Turfs

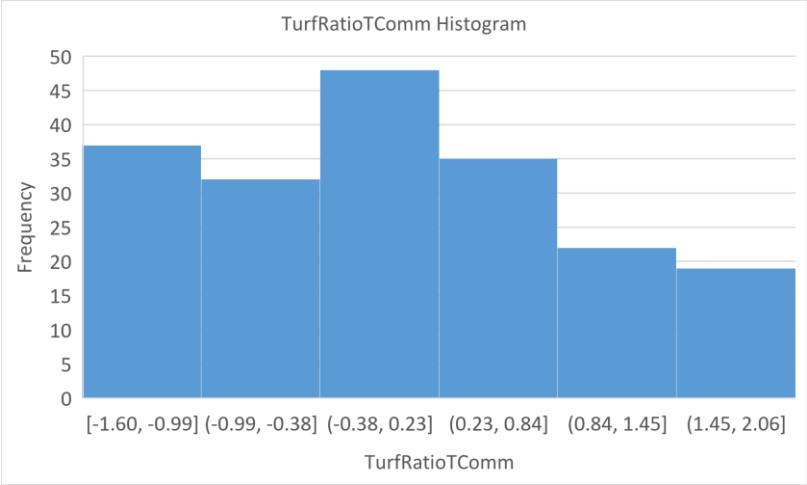


Figure 3.10 Histogram of TurfRatioTComm

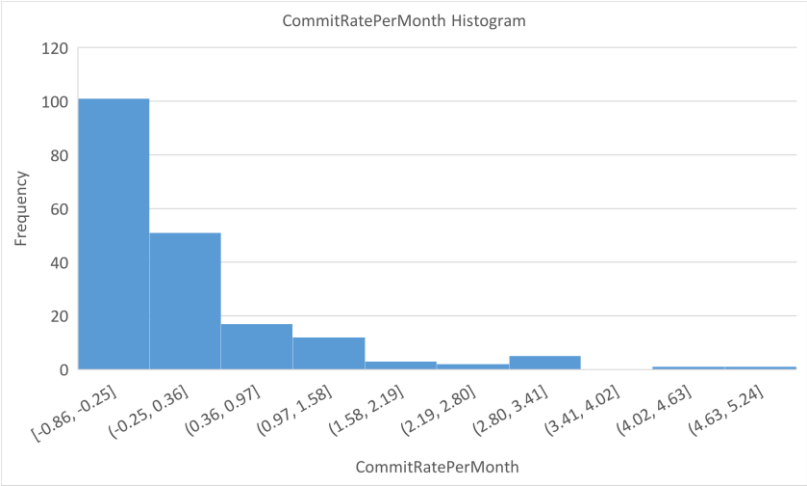


Figure 3.11 Histogram of CommitRatePerMonth

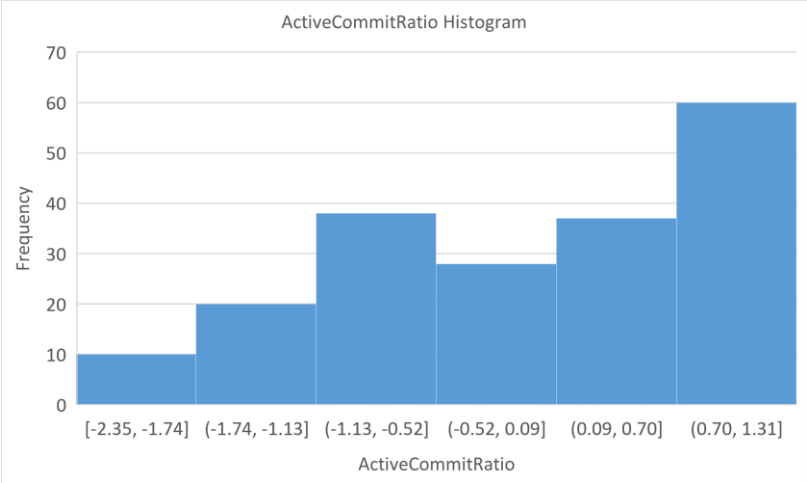


Figure 3.12 Histogram of ActiveCommitRatio

3.5 Behavior and patterns per taxon

In this section, we discuss the behavior and the patterns, that were observed for the studied measurements, and we support any conclusions with extra visual representations.

Regarding the **schema update period**, as shown in Figure 3.13, every taxon seems to have projects with both short and big periods of updates. *However, moving from the frozen to the active class, the schema update period becomes bigger and bigger, the percentages of short periods are decreasing, and the percentages of longer periods are increasing.*

The explanations of the labels, presented in Figure 3.13, are as follows:

- 0_UpTo10Days: Projects with schema update period in the range [0 – 10] days.
- 1_11To180D: Projects with schema update period in the range [11 – 180] days.
- 2_06To12M: Projects with schema update period in the range [181 – 365] days.
- 3_13To36M: Projects with schema update period in the range [366 – 1095] days.
- 4_LONG: Projects with schema update period greater than 1095 days (3 years).

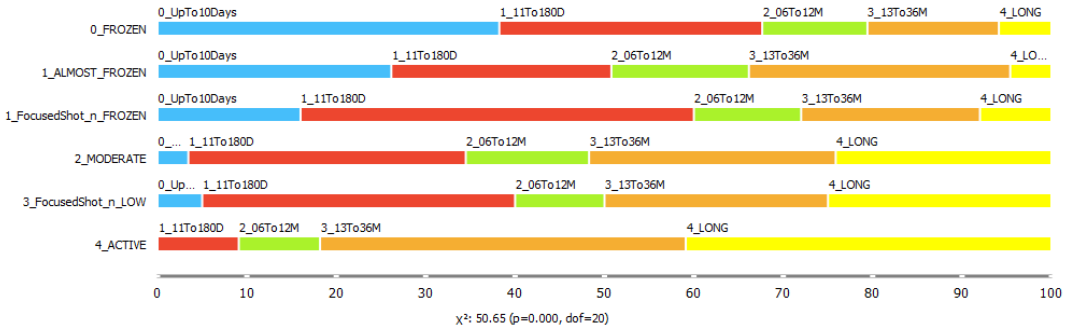


Figure 3.13 Schema Line Update Period Plot

Considering Figure 3.14, which represents the **schema line volume of change** per taxon, it seems like the behavior of each taxon is discrete.

The labels that are presented in Figure 3.14, were produced based on BD where:

$$BD = tableInsertions + tableDeletions$$

More specifically the meanings of the labels in Figure 3.14 are as follows:

- 0_NONE: BD = 0
- 1_SMALL: BD in the range [1 – 2]
- 2_MODERATE: BD in the range [3 - 10]
- 3_HIGH: BD ≥ 11

As in Figure 3.13, so in Figure 3.14, moving from the frozen to the active class, *it seems that the more active a project is, the bigger the percentage of the high volume of changes and the smaller the percentage of the low volume of changes is.*

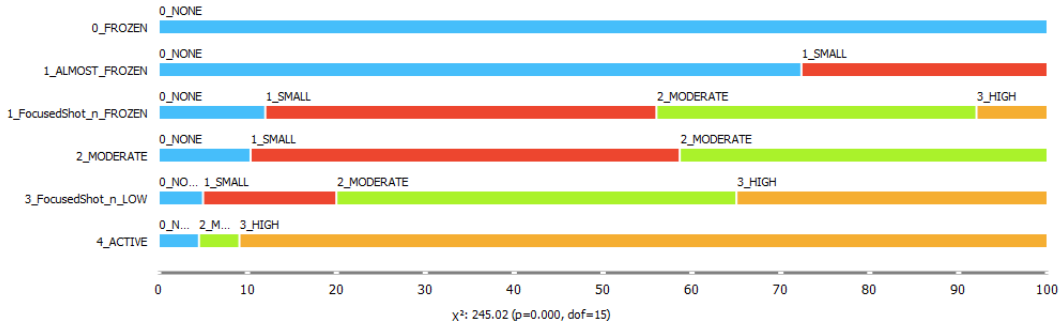


Figure 3.14 Schema Line Volume of Change Plot

Concerning **active commits,** we report their breakdown in the different taxa in Figure 3.15. As mentioned before, a commit is active if its total activity is greater than 0. So, the total number of those commits is the active commits.

Table 3.4 presents the percentages of the active commit classes per taxa where:

- 0_NONE : Active commits = 0
- 1_TOO_FEW : Active commits in the range [1 - 3]
- 2_FEW : Active commits in the range [4 - 10]
- 3_MODERATE : Active commits in the range [11 - 15]
- 4_SEVERAL : Active commits > 15

Table 3.4 Taxa and Active Commits classes

Taxon	0_NONE	1_TOO_FEW	2_FEW	3_MODERATE	4_SEVERAL	Grand Total
0_FROZEN	100%	0%	0%	0%	0%	100%
1_ALMOST_FROZEN	0%	66%	34%	0%	0%	100%
1_FocusedShot_n_Frozen	0%	44%	56%	0%	0%	100%
2_MODERATE	0%	0%	0%	76%	24%	100%
3_FocusedShot_n_LOW	0%	0%	0%	100%	0%	100%
4_ACTIVE	0%	0%	0%	14%	86%	100%
Grand Total	17%	28%	18%	23%	13%	100%

After observing the values of Table 3.4, we conclude that *the active commits separate well the taxa, except for the medium activity classes that are not only separated by the active commits, but also by the amount and concentration of change.*

Taking Figure 3.15 and Table 3.5 into consideration, the taxa are clearly different concerning the active commits, where the range of the active commits, increases, while moving from the frozen to the active class.

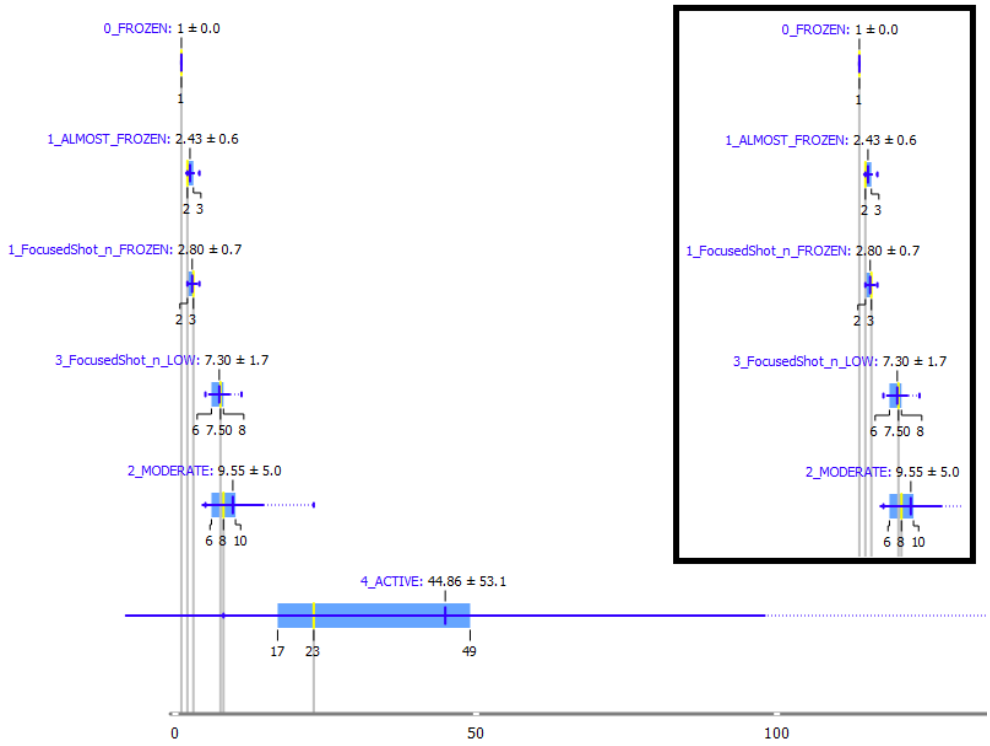


Figure 3.15 BoxPlot of Active Commits

Table 3.5 Active Commits stats per taxon

Taxon	Min	Q1	Median	Mean	Q3	Max	IQR	STDEV
Frozen	1	1	1	1	1	1	0	0
ALMOST_FROZEN	2	2	2	2.43	3	4	1	0.64
FocusedShot_n_FROZEN	2	2	3	2.8	3	4	1	0.76
MODERATE	5	6	8	9.55	10	23	4	5.11
FocusedShot_n_LOW	5	6	7.5	7.3	8	11	2	1.72
ACTIVE	8	16	22	29.25	42.5	64	26.5	18.20

Concerning the **turfs** and the **total activity**, we present their breakdown in the different taxa in Figures 3.16 and 3.17 and the measurements per taxon in Tables 3.5 and 3.6 respectively. As already mentioned, a commit is called turf when it is active and its total activity is less than 15.

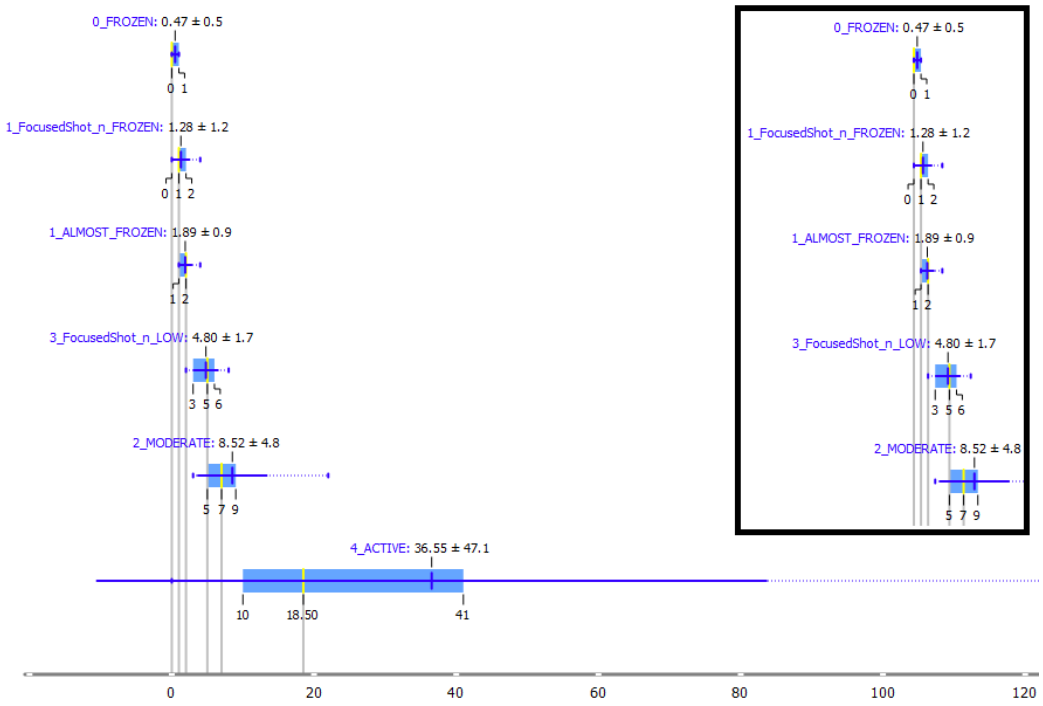


Figure 3.16 BoxPlot of Turfs

Table 3.6 Turfs stats per taxon

Taxon	Min	Q1	Median	Mean	Q3	Max	IQR	STDEV
Frozen	0	0	0	0.47	1	1	1	0.51
ALMOST_FROZEN	1	1	2	1.89	2	4	1	0.89
FocusedShot_n_FROZEN	0	0	1	1.28	2	4	2	1.21
MODERATE	3	5	7	8.52	9	22	4	4.90
FocusedShot_n_LOW	2	3	5	4.8	6	8	3	1.70
ACTIVE	0	9.75	18	23	32.5	57	22.75	17.62

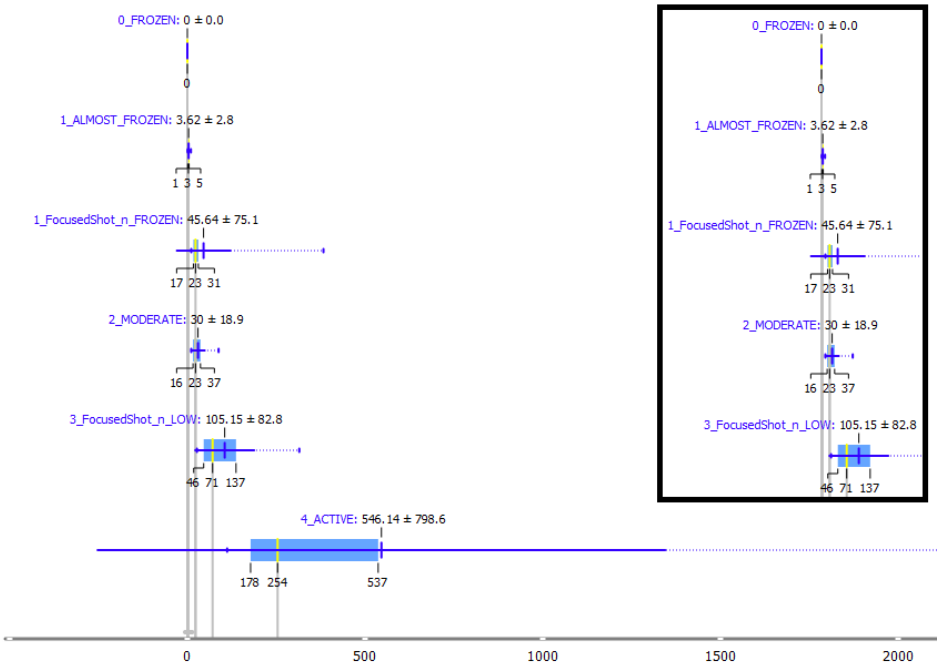


Figure 3.17 BoxPlot of Total Activity

Table 3.7 Total Activity stats per taxon

Taxon	Min	Q1	Median	Mean	Q3	Max	IQR	STDEV
Frozen	0	0	0	0	0	0	0	0
ALMOST_FROZEN	1	1	3	3.62	5	10	4	2.78
FocusedShot_n_FROZEN	11	17	23	45.64	31	383	14	76.62
MODERATE	11	16	23	30	37	88	21	19.19
FocusedShot_n_LOW	27	50.5	71	105.15	131	315	80.5	84.91
ACTIVE	112	177	249	335.9	361.5	1267	184.5	275.68

In Figures 3.16 and 3.17, turfs and total activity are depicted, respectively. Observe that both turfs and total activity are increasing while moving from the frozen towards the active class. So, the families of schemata in these two plots can be easily recognized.

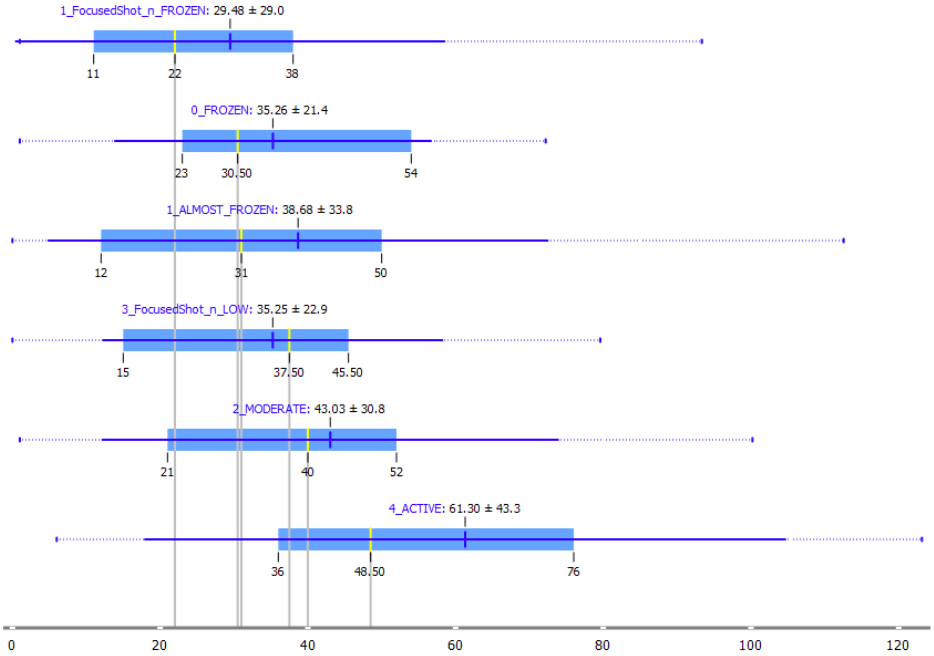


Figure 3.18 BoxPlot of Project Update Period (months)

Table 3.8 Project Update Period stats per taxon

	Taxon	MIN	MAX	AVERAGE	MEDIAN	STDEV
Project Update Period (Months)	0_FROZEN	1	80	35.26	30.5	21.69
	1_ALMOST_FROZEN	0	155	38.68	31	34.06
	1_FocusedShot_n_FROZEN	1	116	29.48	22	29.61
	2_MODERATE	1	126	43.03	40	31.39
	3_FocusedShot_n_LOW	0	94	35.25	37.5	23.54
	4_ACTIVE	6	198	61.3	48.5	44.46

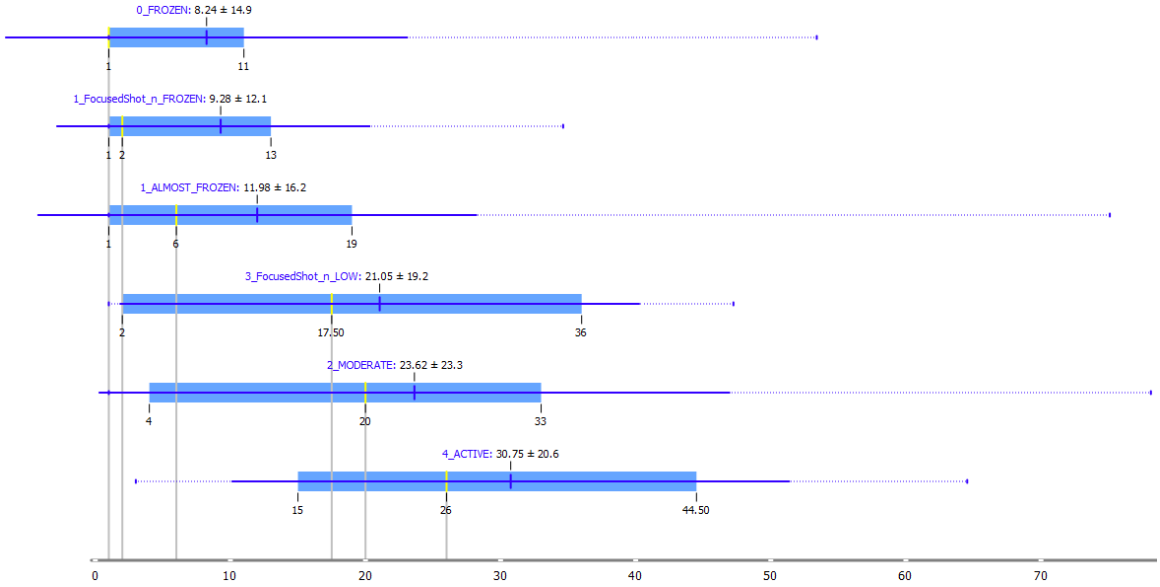


Figure 3.19 BoxPlot of Schema Update Period (months)

Table 3.9 Schema Update Period stats per taxon

	Taxon	MIN	MAX	AVERAGE	MEDIAN	STDEV
Schema Update Period (Months)	0_FROZEN	1	69	8.24	1	15.07
	1_ALMOST_FROZEN	1	99	11.98	6	16.34
	1_FocusedShot_n_FROZEN	1	46	9.28	2	12.36
	2_MODERATE	1	100	23.62	20	23.73
	3_FocusedShot_n_LOW	1	57	21.05	17.5	19.7
	4_ACTIVE	3	84	30.75	26	21.15

In Figures 3.18 and 3.19, the plots of the project update period and the schema update period are presented, respectively. Observe Figure 3.18 and Table 3.8: The update periods of the projects of all families, except for the active class, seem to be big and very similar. More specifically, considering also the values presented in Table 3.9 as well as Figure 3.19, observe that *none of the families of the projects is inactive and that they simply just have different schema evolution profiles.*

To get a deeper insight of the Schema Update Period (SUP), Figure 3.20 presents the breakdown per SUP class and taxon. Active taxa are biased towards longer Schema Update Periods, Moderate and FS-Low are mostly biased towards medium

SUP ranges, whereas "frozen land" is mostly oriented to small SUP periods (with the prominent exception of some Almost Frozen projects, that have almost frozen SUPs between 1 and 3 years.

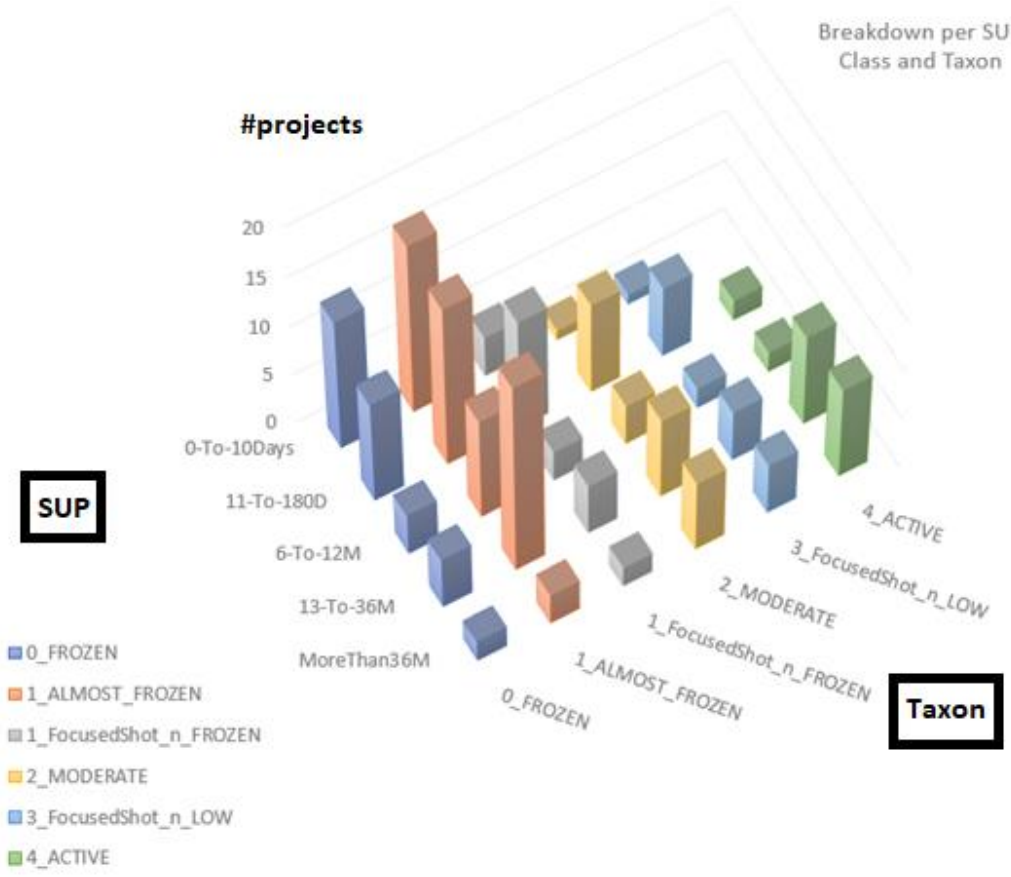


Figure 3.20 3D Column-Plot of Schema Update Period: The left axis demonstrates the Schema Update Period, organized in labeled intervals, the right axis demonstrates the taxon and the height of each bar demonstrates the frequency, i.e. the number of projects of a specific taxon with a specific SUP interval.

In Figure 3.21 a short overview of the patterns of the evolution of the schema line, i.e., the number of tables over time, per taxon is presented.
We classify the patterns of schema line evolution as follows:

- 0_FLAT : Zero schema evolution
- Single Rise : A single step of evolution, upwards
- Multi-Step Rise : Several upward steps, like a staircase

- Drop : Drop of the number of tables over time.
- Turbulent & DoUndo : Mix of Up's and Down's and DoUndo commits

According to Figure 3.21, observe that as the activity increases, more steps of evolution are observed and the pattern of zero evolution decreases. *However, 52% of the total of all projects, belongs to the pattern of zero evolution.* In Figure 3.22 we present a line plot with the patterns per taxon. As in Figure 3.21 so in Figure 3.22, observe how the evolution is increasing while moving from the frozen to the active class.

More specifically:

- In the 0_FROZEN class all projects have zero schema evolution, as expected.
- For the 1_ALMOST_FROZEN class, 75% of 0_FLAT means that most of the projects in this class, during their evolution, did not have new tables.
- In the 1_FocusedShot_n_FROZEN class, 52% of Single Rise means that nearly half of the projects of this class had one commit with new tables.
- In the 2_MODERATE and 3_FocusedShot_n_LOW classes, nearly 25% of the projects seem to have several upward steps of evolution.
- Finally, in the 4_ACTIVE class, 50% of the projects seem to have several upward steps of evolution, on the contrary to only 9% of projects with zero schema evolution.

0_FLAT	Single Rise	Multi Step Rise	Drop	Turbulent & DoUndo	Grand Total	Taxon
100%					100%	0_FROZEN
75%	18%		5%	2%	100%	1_ALMOST_FROZEN
36%	52%		4%	8%	100%	1_FocusedShot_n_FROZEN
10%	41%	24%	7%	17%	100%	2_MODERATE
20%	30%	25%		25%	100%	3_FocusedShot_n_LOW
9%	9%	50%	14%	18%	100%	4_ACTIVE
52%	23%	12%	5%	9%	100%	Grand Total

Figure 3.21 Patterns of schema evolution per taxon

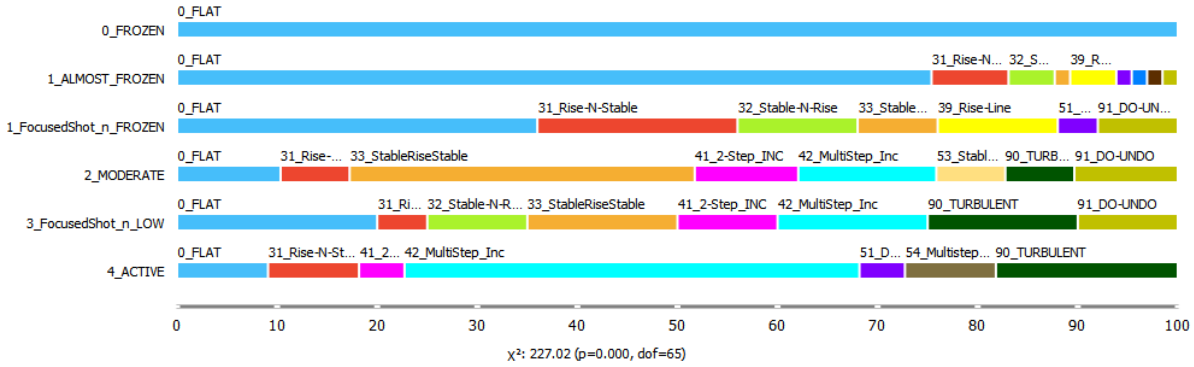


Figure 3.22 Patterns Line Plot

Finally, in Figure 3.23, a decision tree, for the taxa on the basis of attributes TotalActivity, ActiveCommits, Reeds, ActiveCommitRatio, is presented.

The steps for the generation of the decision tree presented in Figure 3.23 are as follows:

- We have calculated the Z-scores from the actual values of the attributes by following the procedure explained in Method A.
- Then, we used the produced Z-scores to the Orange tool, which produced the decision tree of Figure 3.23.

To get a deeper insight into the decision tree, a translation of the Z-scores of the attributes is reported in Table 3.10. In the decision tree, a taxon is marked as well-configured, if all parts of the taxon are on the same leaf of the decision tree, which means that the attributes separate the taxa well and the taxa are distinct. For instance, the 0_FROZEN, 1_ALMOST_FROZEN, and 1_FocusedShot_n_FROZEN classes seem to be well-configured. On the other hand, the 2_MODERATE and 3_FocusedShot_n_LOW classes seem like they have been misclassified and their merge into a bigger class can be examined. Moreover, it is also noticed that, in this configuration of classes, some projects belong in 3_FocusedShot_n_LOW and their total activity is so big that they fit with the 4_ACTIVE class.

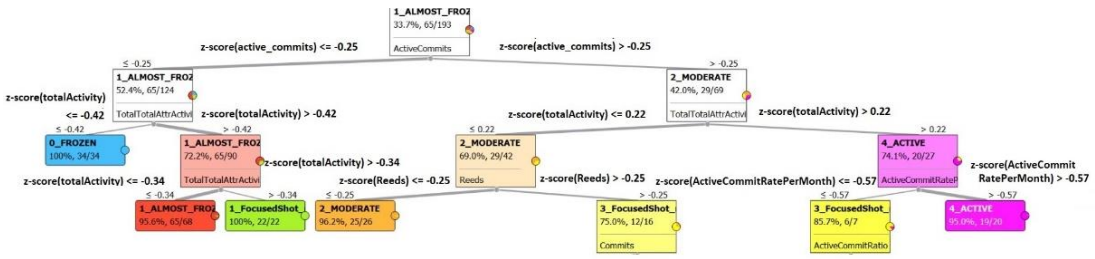


Figure 3.23 Decision Tree of the taxa as reported by the Orange tool

Table 3.10 Translations of Z-scores to the actual values

Attribute	Z-score	Actual Value
ActiveCommits	-0.25	4
TotalTotalAttrActivity	-0.42	0
	-0.34	10
	0.22	88
Reeds	-0.25	1
ActiveCommitRatePerMonth	-0.57	0.36

3.6 Centroids and characteristics per taxon

In this section, we discuss if we could define a project of each taxon as a centroid-project, that would represent the characteristics and the behavior of the respective taxon. The taxa presented in [12], were extracted mostly based on the activity and the heartbeat of the projects. For this reason, to define the centroid project per taxon, we decided to take into account only activity and heartbeat measures (totalActivity, Reeds, Turfs, ActiveCommits).

Before we proceed to the calculation of the centroid-project, firstly we calculate the Z-scores for all the measurements of the dataset, which allow us to compare scores that are from different samples (in our case different taxa that may have different means and standard deviations).

The steps of the procedure, for the calculation of the centroid-project within each taxon are as follows:

- For each taxon, we calculate the average value for all the measurements of interest.
- For each project we calculate the distances between every such measurement and its average.
- We sum up all the distances, that we calculated on the previous step, per project.
- Finally, we find the smallest of the sums of the previous step. The smallest sum represents the sum of the distances of the project, whose measures are closer to the average measures of the respective taxon.

After following the aforementioned procedure, we ended up with the following results, as presented in Table 3.11, where for each taxon the actual values as well as the Z-scores, in parenthesis, are reported.

Table 3.11 Centroid-Project per taxon

Taxon	**Centroid-Project**	Total Activity	Reeds	Turfs	Active Commits
Frozen	*damnpoet__yiicart*	0 (-0.42)	1 (-0.25)	0 (-0.57)	1 (-0.55)
ALMOST_FROZEN	*RubyMoney__money-rails*	3 (-0.39)	1 (-0.25)	2 (-0.34)	3 (-0.35)
FocusedShot_n_FROZEN	*accgit__acl*	31 (-0.19)	2 (0.22)	1 (-0.46)	3 (-0.35)
MODERATE	*mapbox__osm-comments-parser*	34 (-0.17)	1 (-0.25)	9 (0.44)	10 (0.33)
FocusedShot_n_LOW	*anchorcms__anchor-cms*	125 (0.49)	2 (0.22)	6 (0.11)	8 (0.14)

ACTIVE	*pods-frame-work__pods*	352 (2.14)	9 (3.54)	21 (1.80)	30 (2.29)

The characteristics of each taxon compared to the values of the centroid-project are reported below:

Frozen Taxon Characteristics (Centroid-Project: *damnpoet__yiicart)*

- *Zero total activity*
  - *totalActivity(damnpoet__yiicart) = 0*
- *Just one active commit (Either a reed or a turf)*
  - *activeCommits(damnpoet__yiicart) = 1*

Almost Frozen Taxon Characteristics (Centroid-Project: *RubyMoney__money-rails)*

- At most 3 active commits (Mix of turfs and reeds)
  - *activeCommits(RubyMoney__money-rails) = 3*
- Small total activity (less than 10 updated attributes)
  - totalActivity(*RubyMoney__money-rails) = 3*

FocusedShot_n_FROZEN Taxon Characteristics (Centroid-Project: *accgit__acl)*

- At most 3 active commits (Mix of turfs and reeds)
  - *activeCommits(accgit__acl) = 3*
- Medium to high total activity (More than 10 updated attributes and less than 383 updated attributes)
  - totalActivity(*accgit__acl) = 31*

Moderate Taxon Characteristics (Centroid-Project: *mapbox__osm-comments-parser)*

- More than 4 active commits (Mix of turfs and reeds)
  - *activeCommits(mapbox__osm-comments-parser) = 10*
- Medium total activity (More than 10 and less than 88 updated attributes)
  - totalActivity(*mapbox__osm-comments-parser)* = 34

FocusedShot_n_LOW Taxon Characteristics

(Centroid-Project: *anchorcms__anchor-cms)*

- More than 4 active commits (Mix of turfs and reeds)
  - *activeCommits(anchorcms__anchor-cms)* = 8
- High total activity (more than 27 and less than 315 updated attributes)
  - totalActivity(*anchorcms__anchor-cms) = 125*

Active Taxon Characteristics (Centroid-Project: *pods-framework__pods)*

- Very high total activity (more than 111 and less than 1268 updated attributes)
  - totalActivity(*pods-framework__pods)* = 352
- More than 7 active commits (Mix of turfs and reeds)
  - *activeCommits(pods-framework__pods) = 30*

Considering the values of the centroid-projects and especially the total activity and the active commits, the taxa are distinguished from each other. However, observe that the active commits of the centroid-projects reveal the possibility of deriving larger groups, which will be discussed further in later sections.

3.7 Assessment of existing taxa and validity metrics

In this section, we assess the taxa presented in [12] in terms of their validity, with the calculation of some validity metrics from the area of clustering. Afterwards, we present some plots to observe visually and discuss the quality of the clusters.

3.7.1 Cohesion and Separation metrics

The validity metrics, that were used, are as follows:

- *Cohesion*: This metric [9] measures how closely related are the objects within the same cluster. Intuitively, cohesion means that members of a group are similar. A lower within-cluster variation is an indicator of good compactness, which means that the lower the cohesion is, the better the clustering is.

Cohesion (SSE) can be computed by the following formula:

$$\text{SSE} = \sum_{i} \sum_{x \in C_i} (x - m_i)^2$$

where:

x ∈ C_i : Every element in C_i cluster.

m_i : Centroid of i cluster.

More specifically, $(x - m_i)^2$ is the distance of each element of a cluster from the centroid of the cluster. The lower the distance is, the lower the variation in the cluster is, which means that the more similar the elements of the cluster to one another are. The sum of the distances of all the elements of cluster i is the cohesion of cluster i and the sum of the cohesions of all the clusters is the total cohesion.

- *Separation*: This metric [9] measures how distinct or well-separated a cluster is from other clusters. The bigger the separation is the better the clustering is, because it means that the clusters have fewer common characteristics.

  Separation (BSS) can be computed by the following formula:

$$BSS = \sum_{i} |C_i| \, (m - m_i)^2$$

  where:

  C_i : Number of elements of i cluster.

  m : Mean value of the whole dataset.

  m_i : Mean value of i cluster.

More precisely, $(m - m_i)^2$ is the distance of the mean of a cluster i from the mean value of the whole dataset. The number of elements of i cluster (C_i) is used as a

weight in the aforementioned formula so that the larger the cluster is, the more significant its impact on the separation is. The multiplication of (C_i) with the $(m - m_i)^2$ distance is called the separation of cluster i, and the sum of the separations of all the clusters is the total separation. The bigger the separations of the clusters are, the more distinct the clusters are.

The abovementioned validity metrics were computed for many combinations of attributes, where the most worth mentioning groups of attributes are the following:

- *Group A*: TotalTotalAttrActivity, ResizingRatio, DurationInMonths.
- *Group B*: TotalTotalAttrActivity, Turfs, ActivityDueToReeds.
- *Group C*: TotalTotalAttrActivity, Turfs.
- *Group D*: TotalTotalAttrActivity, Turfs, ActiveCommits.

For the computation of the cohesion and separation metrics presented in Figure 3.24, the Z-scores of the attributes were used instead of the actual values. For each group, the cohesion and separation metrics per taxon are presented, as well as their total values. *This procedure aims to find out, which attributes seem to determine the taxa the best way, in terms of their in-cluster variation and their separation.*

Group	Class	Size	SSE	BSS
A	0_FROZEN	34	20.49	15.51
	1_ALMOST_FROZEN	65	51.25	17.97
	1_FocusedShot_n_FROZEN	25	47.43	3.91
	2_MODERATE	29	105.47	12.91
	3_FocusedShot_n_LOW	20	62.42	5.52
	4_ACTIVE	20	138.89	97.22
	Total (All Classes)	**193**	**425.95**	**153.05**
B	0_FROZEN	34	73.80	14.93
	1_ALMOST_FROZEN	65	6.73	25.37
	1_FocusedShot_n_FROZEN	25	18.81	5.03
	2_MODERATE	29	11.50	7.22
	3_FocusedShot_n_LOW	20	12.30	2.98
	4_ACTIVE	20	197.82	202.50
	Total (All Classes)	**193**	**320.96**	**258.04**
C	0_FROZEN	34	0.11	14.92
	1_ALMOST_FROZEN	65	0.66	18.10
	1_FocusedShot_n_FROZEN	25	7.89	4.67
	2_MODERATE	29	9.05	5.57
	3_FocusedShot_n_LOW	20	7.93	2.43
	4_ACTIVE	20	151.03	163.63
	Total (All Classes)	**193**	**176.67**	**209.33**
D	0_FROZEN	34	0.11	25.04
	1_ALMOST_FROZEN	65	0.91	28.79
	1_FocusedShot_n_FROZEN	25	8.02	8.09
	2_MODERATE	29	16.03	8.02
	3_FocusedShot_n_LOW	20	8.47	2.53
	4_ACTIVE	20	211.18	261.82
	Total (All Classes)	**193**	**244.72**	**334.28**

Figure 3.24 Cohesion and Separation Metrics

After observing Figure 3.24, which presents the cohesion and separation metrics for the groups of attributes, the conclusions about the taxa presented in [12] are as follows:

- The values of cohesion and separation in Groups A and B show that neither the resizing of the projects caused by the commits, nor the duration of the commits, nor the activity due to big commits, seem to have been taken into consideration for the extraction of the taxa. The values of cohesion and separation of Groups A and B are worse than Groups C and D, which makes sense

since the classification of the taxa was made according to the total activity and the active commits of the projects.

- The SSE in the "4_ACTIVE" class seems to remain high in all groups. This variation can be explained because this class consists either of projects that have a big amount of small active commits or of projects that have fewer active commits but bigger ones.
- Regarding the size of the 1_ALMOST_FROZEN class, the value of separation in all groups is big compared to the other classes, which is logical, due to the fact that separation, as already mentioned, is proportional to the size of the cluster.
- In all groups, the values of the metric separation are not that high, but this seems not to be a problem, since the taxa, as shown in Figure 3.25, seem to be distinct.

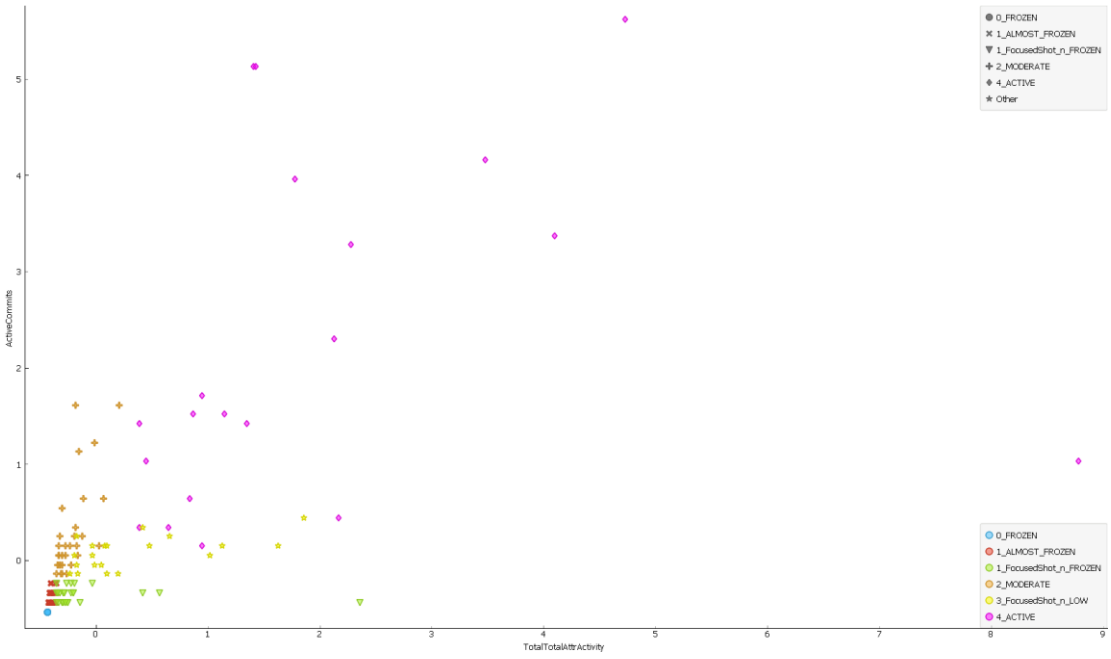


Figure 3.25 ScatterPlot: TotalTotalActivity - ActiveCommits

3.7.2 Silhouette Coefficient

To get a deeper insight into the quality of the clustering of projects to taxa, the "Orange" tool, which is an open-source data visualization and analysis tool, was used to plot the silhouette scores of each project of each class.

- *Silhouette coefficient*: The silhouette value [1] is a measure of how similar an object is to its cluster (cohesion) compared to other clusters (separation). The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. If many points have a low or negative value, then the clustering configuration may have too many or too few clusters.

  Silhouette score can be computed by the following formula:

$$Silhouette\ Score = \frac{b - a}{\max(a, b)}$$

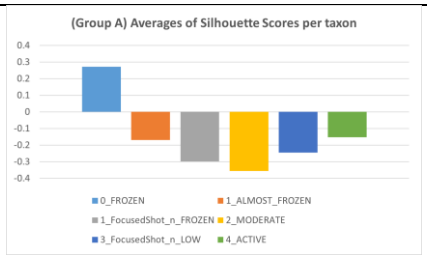
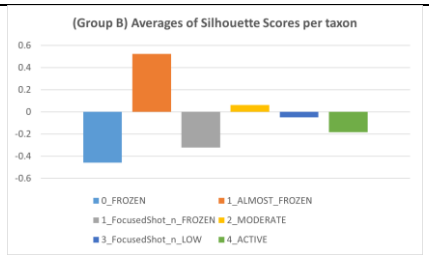
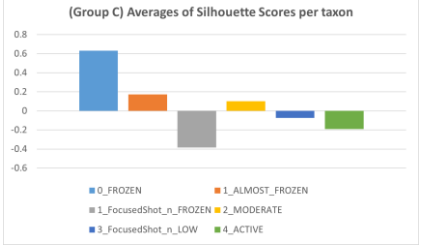
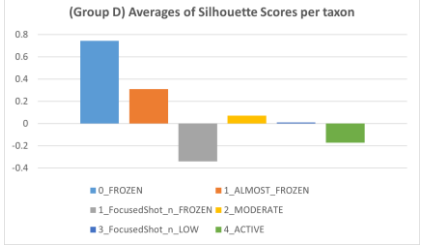
  where:

  a : The average distance between each point within a cluster.

  b : The average distance between all clusters.

The averages of the silhouette scores per taxon of the aforementioned groups are presented in Table 3.12. Also, the visualization of all the silhouette scores for the aforementioned groups is presented in Table 3.13.

Table 3.12 Averages of Silhouette scores per taxon

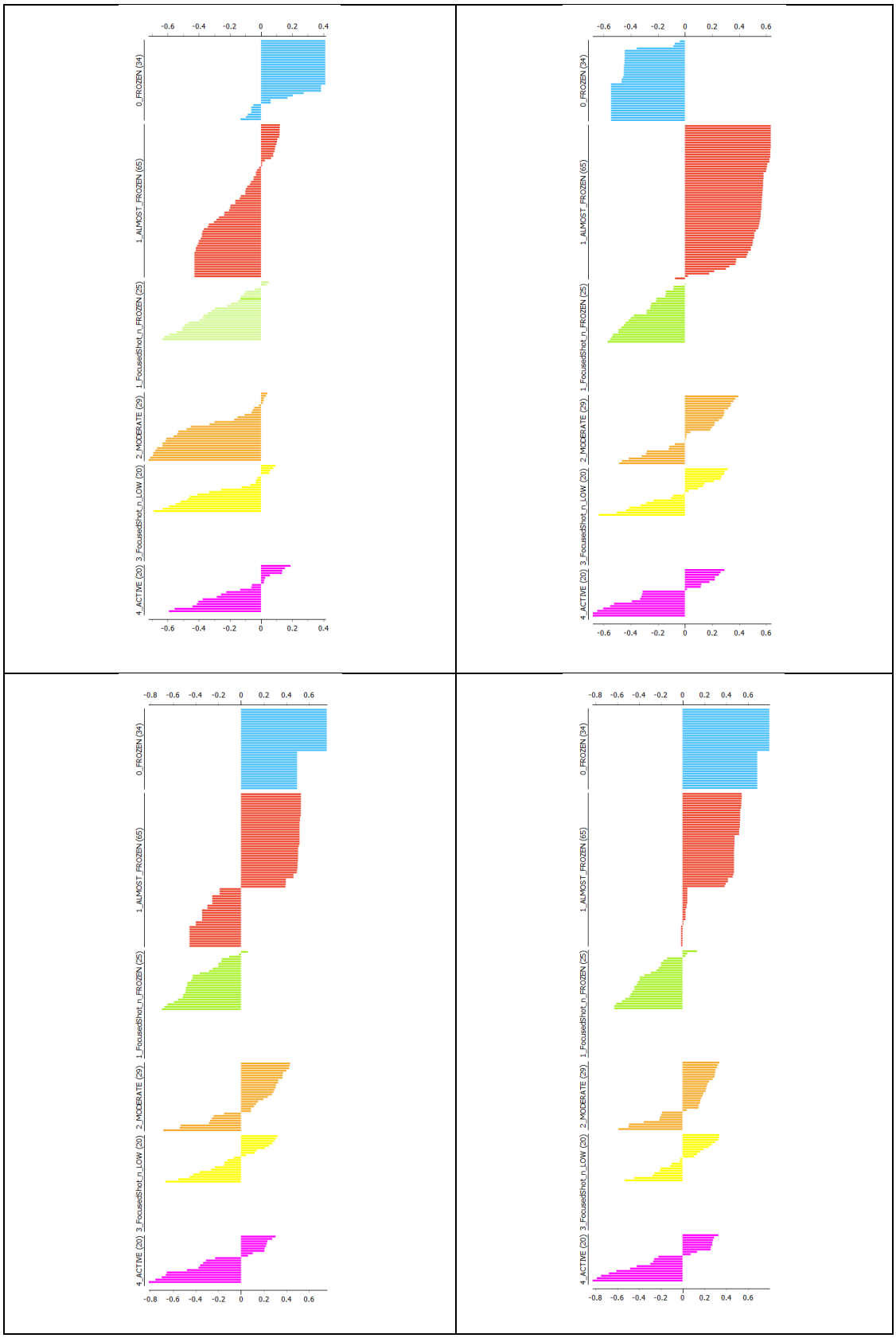


Table 3.13 Silhouette Plots (Upper Left: Group A, Upper Right: Group B, Bottom Left: Group C, Bottom Right: Group D)

As shown in Figure 3.25 the clusters are distinct, but their distance is not that big, which is the reason why the silhouette values in the silhouette plots, presented in Tables 3.12 and 3.13, are not that good, because the silhouette coefficient is based on the average distance between all clusters (b). However, things are getting better in groups B, C, and D, which makes sense, considering the fact that the taxa presented in [12] were extracted based on the active commits, the total activity, and the reeds of the projects as shown in the decision tree in Figure 3.26. Having said all this, the question, that naturally follows is, whether we could find a better solution to clustering these projects.

As a summary of the plots, the metrics, the patterns and the decision tree that we explored in this chapter, we noticed that the taxa 0_FROZEN, 1_ALMOST_FROZEN, and 1_FocusedShot_n_FROZEN have similar behavior in terms of their schema growth with zero to minimum schema evolution. Moreover, the taxa 2_MODERATE and 3_FocusedShot_n_LOW also are similar to each other with medium schema evolution. Finally, the taxon 4_ACTIVE is different from all the other taxa and includes projects with very active commits and many schema changes. These similarities in schema evolution between some taxa, that we observed, are the reason why we came up with the idea that merging these similar taxa into larger groups (super taxa) would probably make sense and give us some good results.

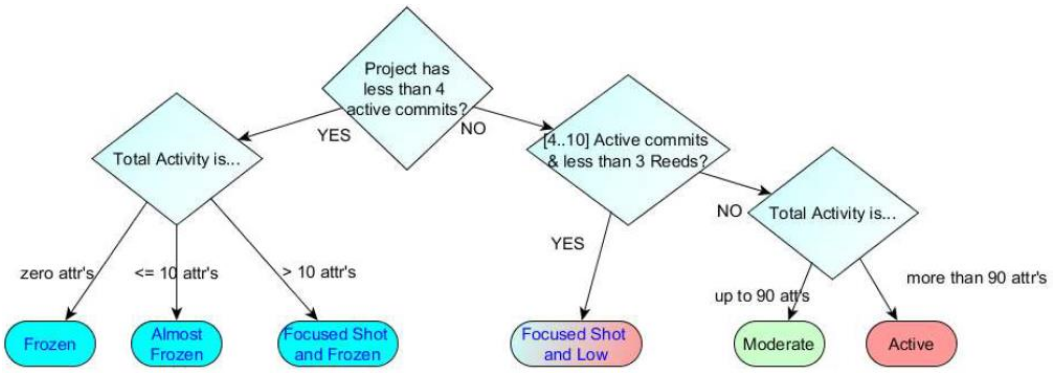


Figure 3.26 Decision Tree presented in [12]

CHAPTER 4

EXAMINATION OF THE RELATIONSHIP BETWEEN SUPER TAXA AND SCHEMA MEASUREMENTS

In this chapter, in order to address the research opportunity detected in Chapter 3, we introduce the notion of super taxa, which are generalizations of the taxa presented in [12] and were generated by the merge of similar taxa into larger groups. Afterwards, we discuss how these super taxa are related to the heartbeat, the activity, the table-level activity measurements, as well as the durations. At the end of this chapter, we summarize our findings.

4.1 The possibility of deriving super taxa

As already mentioned in the previous chapter, the data and the plots provide a hint that it is possible that there is room for abstracting into **super taxa** by merging similar taxa into larger groups. The idea behind the generation of super taxa remains the same as in the original taxa in [12], meaning that they are also based on the evolutionary activity of the projects. In this chapter, we merge taxa with similar behavior into bigger ones, as shown in Table 4.1, to make them more distinct and see what the outcome is. With that being said, the super taxa that were produced, along with the rationale of their derivation, are as follows:

Table 4.1 Table of Super Taxa with their characteristics

Super-taxon	Taxa	Super-Taxon Characteristics
Cold	• 0_FROZEN • 1_ALMOST_FROZEN • 1_FocusedShot_n_FROZEN	• Low total activity in average. • At most 4 active commits.
Mild	• 2_MODERATE • 3_FocusedShot_n_LOW	• More than 4 active commits. • Majority of projects has less equal than 3 reeds.
Hot	• 4_ACTIVE	• More than 4 active commits. • Majority of projects has more than 3 reeds.

- ***0_COLD***

  This super taxon was generated by the merge of 0_FROZEN, 1_ALMOST_FROZEN, and 1_FocusedShot_n_FROZEN classes. The individual

classes were determined to be merged, because of their small, in average, total activity, as well as their small number of active commits. Regarding the child taxa, the 0_FROZEN taxon consisted of projects with zero total activity, the 1_ALMOST_FROZEN taxon consisted of projects with small total activity and a small number of active commits and the 1_FocusedShot_n_FROZEN taxon consisted of projects with a small number of active commits but higher total activity, which was caused by a couple of focused-shot big commits. Nevertheless, these taxa are very similar to each other and their common line is their "cold" behavior (zero to a few active commits), and this is the reason we decided to group them in the same super taxon.

- ***1_MILD***

  This super taxon was produced by the combination of 2_MODERATE and 3_FocusedShot_n_LOW original classes. These two taxa were determined to be merged due to their medium to high activity. As far as the child taxa are concerned, the 2_MODERATE taxon consists of projects with medium total activity and a medium number of active commits, and the 3_FocusedShot_n_LOW consists of projects with medium to high total activity and a medium number of active commits. However, the medium activity of the projects in these taxa led us to group them in the 1_MILD super taxon.

- ***2_HOT***

  This super taxon 2_HOT is the same as the original 4_ACTIVE taxon, which differs a lot from the other taxa and consists of projects with extremely high activity during their lifecycle. For this reason, we decided to let this taxon the same as the original 4_ACTIVE taxon.

Figure 4.1 presents how the original taxa presented in [12] are merged and produce the super taxa.

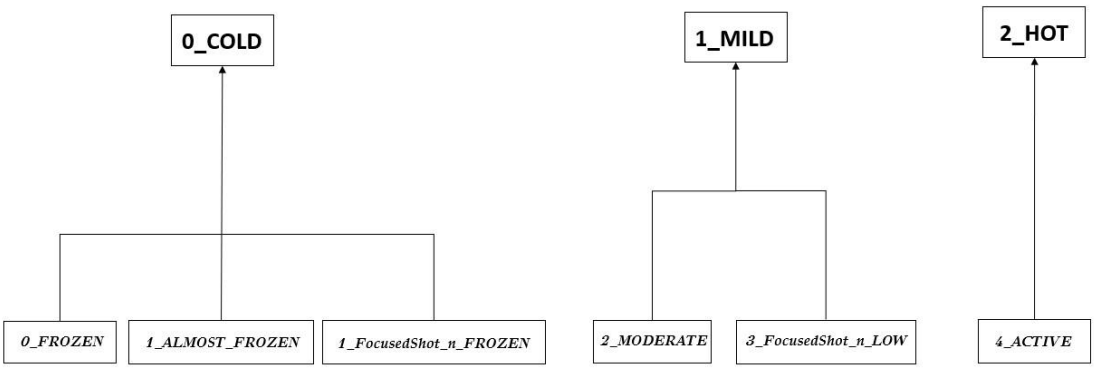


Figure 4.1 Super Taxa

4.2 Super taxa and Heartbeat

In this section, we refer to the relation between the super taxa and all the attributes and metrics that are related to the heartbeat.

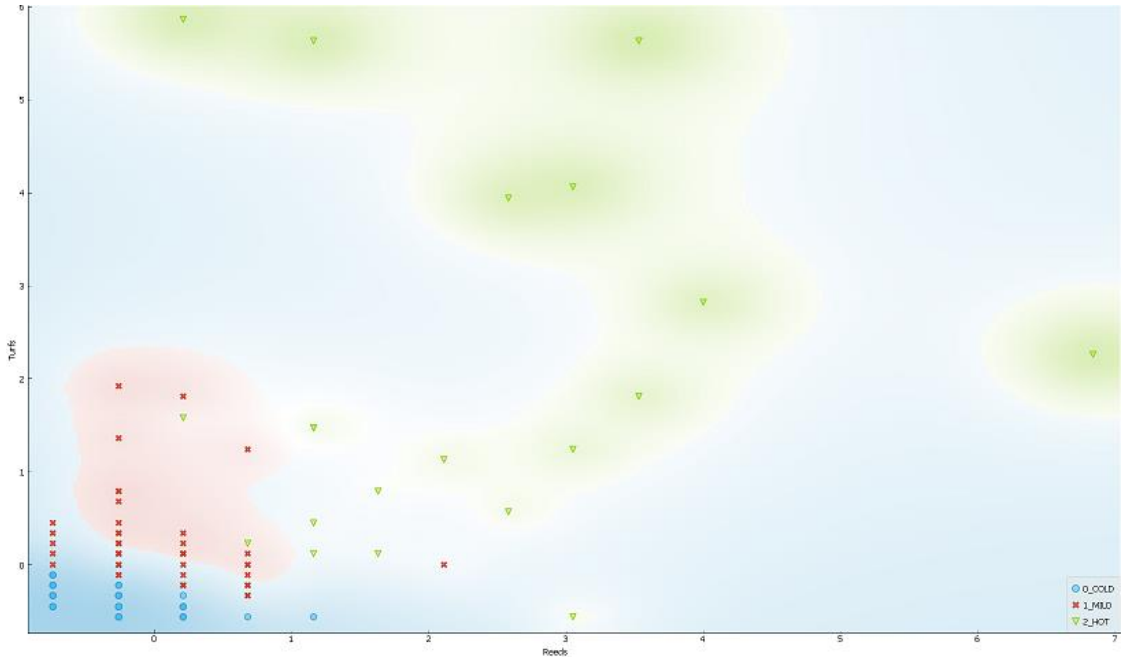


Figure 4.2 Scatter Plot of Reeds – Turfs

Reeds - Turfs

As far as the reeds and turfs are concerned, their relationship with the super taxa is presented in Figure 4.2.

Observe Figure 4.2: ***Regarding reeds and turfs, the super taxa seem to be well configured with a few exceptions in 1_MILD and 2_HOT classes.***

The number of reeds, in Figure 4.3 seems to distinguish well the 2_HOT class from the two other classes, but not the 0_COLD class from the 1_MILD class. On the other hand, regarding the number of turfs, in Figure 4.4, the taxa seem to be distinct from each other, which makes sense, since the classification is based on the active commits, so, moving from the 0_COLD to the 2_HOT class, the number of active commits is obviously increasing.

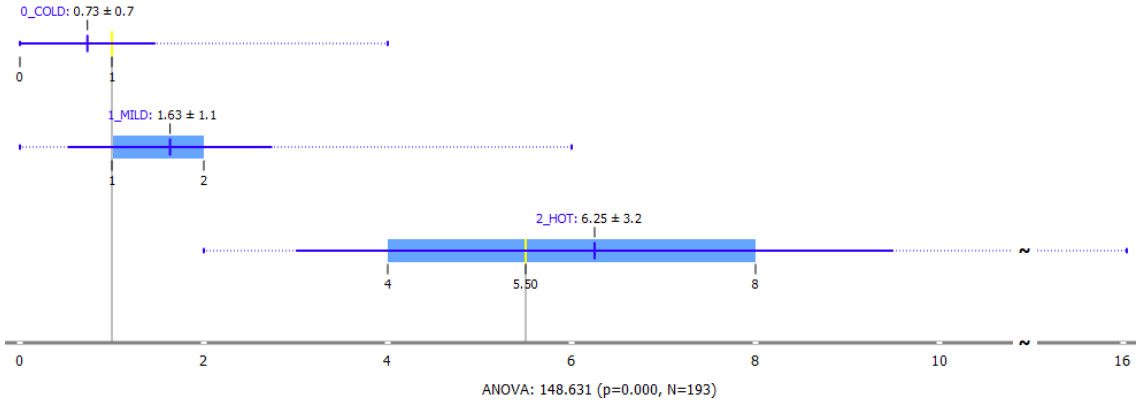


Figure 4.3 BoxPlot of Reeds

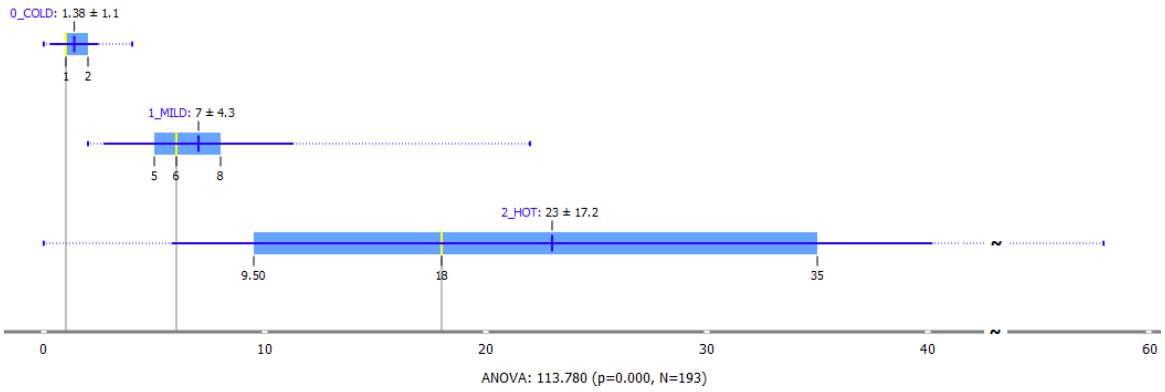


Figure 4.4 BoxPlot of Turfs

Active Commits

The number of active commits, as shown in Figure 4.5, increases while moving from the 0_COLD class to the 2_HOT class, as expected. Especially, ***0_COLD seems to be totally distinct from the other classes, in terms of the number of active commits, which makes sense, because 0_COLD class has zero to few active commits, by definition.***

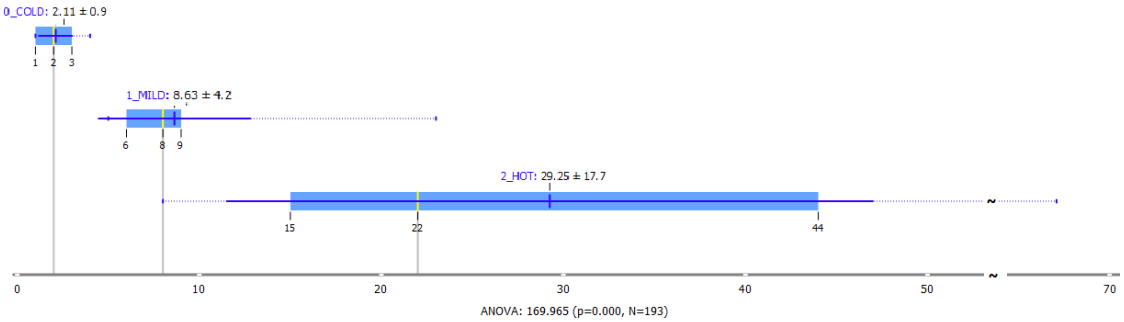


Figure 4.5 BoxPlot of Active Commits

Reed – Turfs (Ratios)

In Figures 4.6 and 4.7 we present the ratio of turfs to active commits and the ratio of reeds to active commits. In Figure 4.6, we surprisingly observe that the 0_COLD has the biggest ratio of reeds to active commits. This can be explained by the small number of active commits that the 0_COLD class has, which makes its reeds ratio seem high. Regarding the ratio of turfs to active commits, as shown in Figure 4.7, the 1_MILD class has the highest ratio of turfs to active commits.

Similar behavior with the aforementioned ratios we observe at the ratio of turfs to the total commits and the ratio of reeds to the total commits in Figures 4.8 and 4.9.

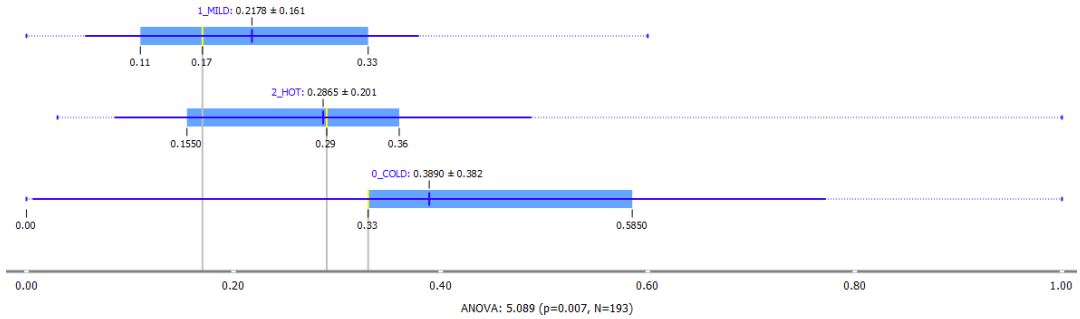


Figure 4.6 BoxPlot of ReedRatioAComm

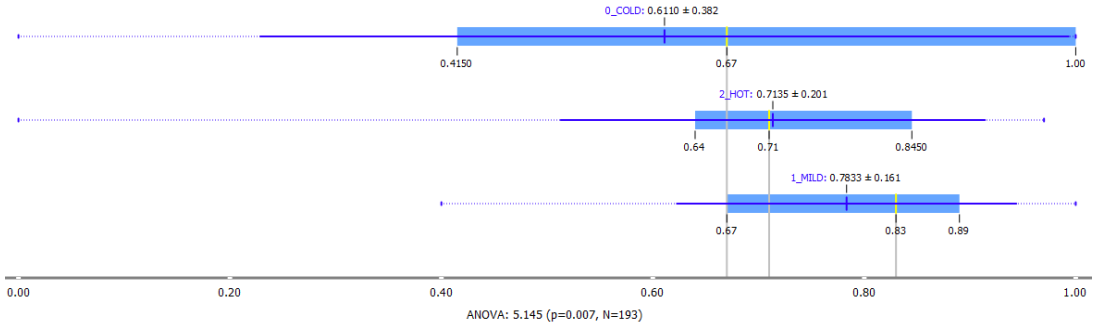


Figure 4.7 BoxPlot of TurfRatioAComm

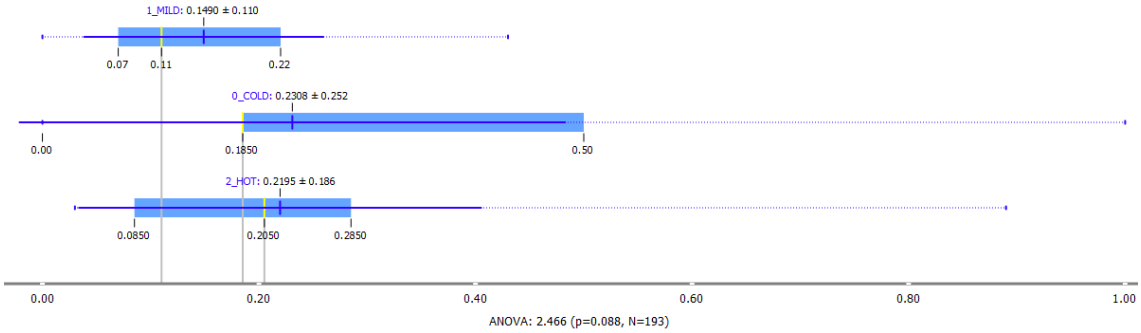


Figure 4.8 BoxPlot of ReedRatioTComm

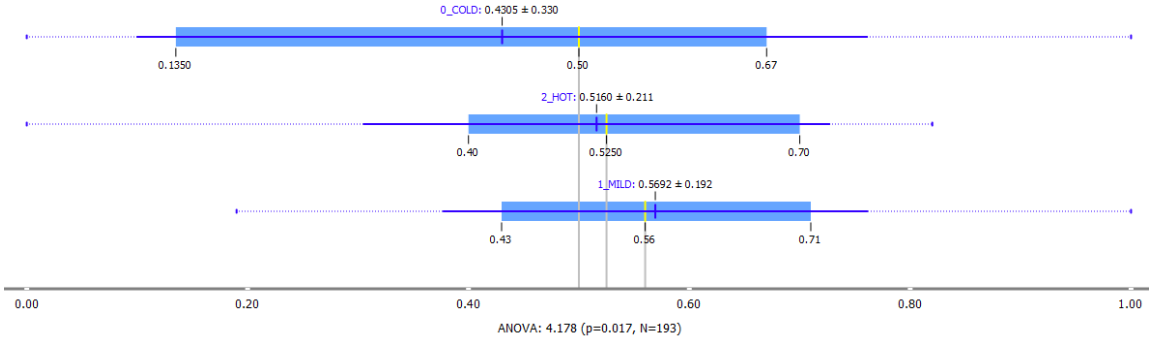


Figure 4.9 BoxPlot of TurfRatioTComm

Active Commits (Ratio)

In Figure 4.10 we can see the relationship between the percentage of active commits to total commits and the super taxa. As we move from the cold to the hot class the ratio of active commits increases, as expected.

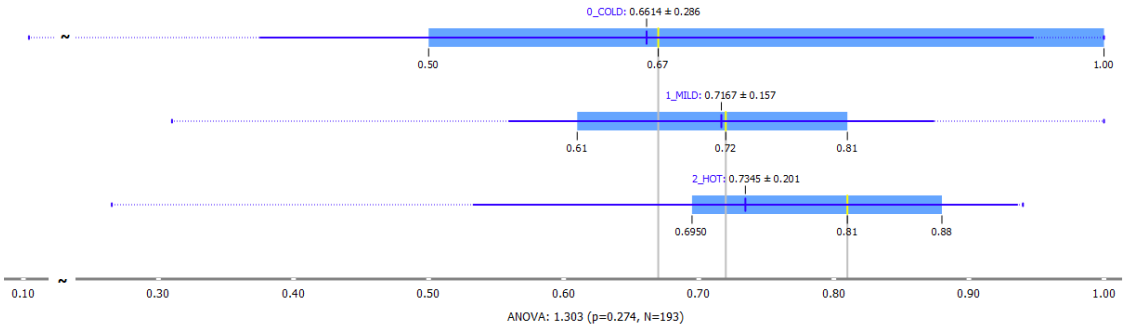


Figure 4.10 BoxPlot of ActiveCommitRatio

Summary of Heartbeat

In Table 4.2 we report the averages of some measures related to the heartbeat of the projects. *The more active a taxon is, the bigger the number of the active commits, reeds and turfs it has. On the other hand all ratios seem to be close for the different taxa.*

Table 4.2 Averages of some measures related to heartbeat per taxon

Measure	0_COLD	1_MILD	2_HOT
Reeds	1.00	2.00	6.00
Turfs	1.00	7.00	23.00
Active Commits	2.00	9.00	29.00
ActiveCommitRatio	0.66	0.72	0.73
ReedRatioAComm	0.39	0.22	0.29
TurfRatioAComm	0.61	0.78	0.71
ReedRatioTComm	0.23	0.15	0.22
TurfRatioTComm	0.43	0.57	0.52

Finally, to get a better intuition about how the heartbeat is related to the super taxa, we present in Figure 4.11 a decision tree, which was produced, given all the attributes that are related to the Heartbeat as input, by the Orange tool.

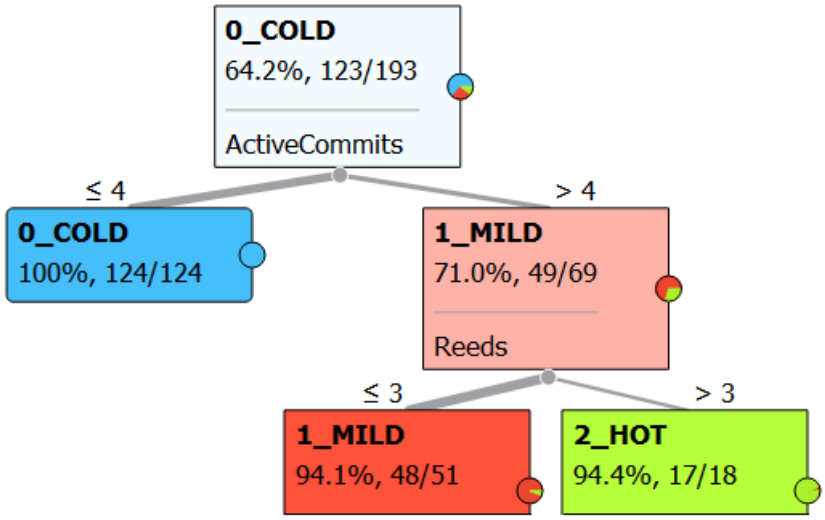


Figure 4.11 Decision Tree – Heartbeat

We can clearly conclude that *the super taxa are highly related to the heartbeat of the projects*. More specifically, projects with less or equal than 4 active commits automatically belong to the 0_COLD super taxon. Additionally, projects with more than 4 active commits, belong to 1_MILD super taxon, if they have less or equal than 3 reeds, whereas if they have more than 3 reeds they belong to the 2_HOT super

taxon. Exceptions at the aforementioned rules are 4 misclassified projects out of 193, meaning 2% (Three projects that were supposed to be in 1_MILD super taxon, are misclassified to the 2_HOT super taxon, and one project that was supposed to be in the 2_HOT super taxon, is misclassified to the 1_MILD super taxon). Taking into account the percentages of the well-classified projects per taxon (100% for 0_COLD, 94.1% for 1_MILD, and 94.4% for 2_HOT), there is no doubt, that the heartbeat is a crucial factor for the discrimination of the taxa.

4.3 Super taxa and Activity

In this section, we discuss the relation between the super taxa and all the attributes and metrics, that are related to the activity of the projects.
Regarding the total activity, as shown in Figure 4.12, the super taxa are well separated from each other. *The concentration of the values at each taxon shows that, moving from the 0_COLD to the 2_HOT class the total activity increases.* However, there are a few outliers at the 0_COLD class with high total activity. These outliers in the taxonomy of [12] belonged to the 1_FocusedShot_n_FROZEN taxon, which contained projects with a small number of active commits (less than 4) and total activity greater than 10.

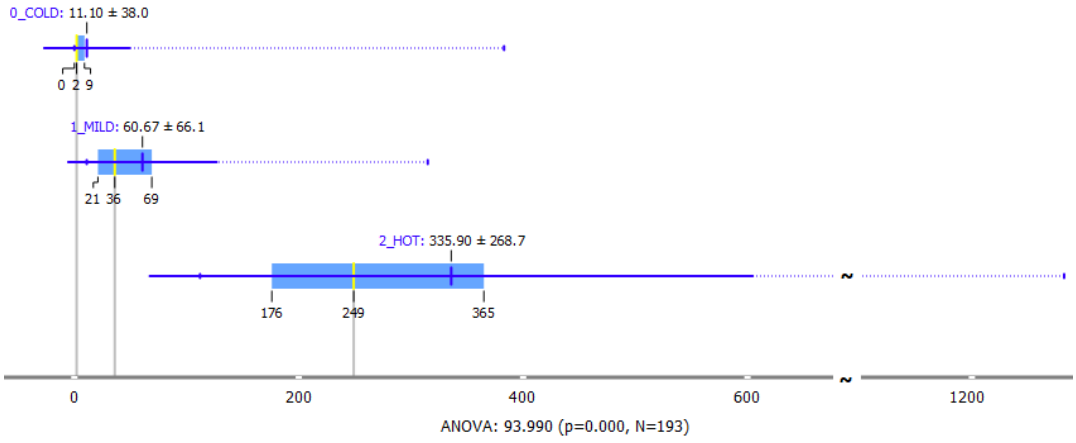


Figure 4.12 BoxPlot of Total Activity

The relationship of total maintenance to the super taxa, as shown in Figure 4.13, is similar to the relationship of the taxa to total activity. Observe *that the more active the class is, the more maintenance it goes through*, which makes sense.

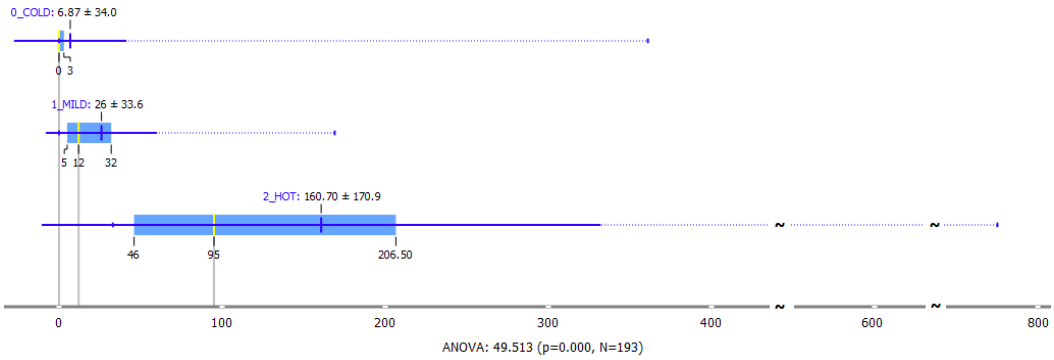


Figure 4.13 BoxPlot of TotalMaintenance

In Figure 4.14 the plot presents the relation between the total expansion and the super taxa. Observe that the taxa are a lot more distinct from each other than with respect to the other activity measures, which indicates that *total expansion is a good taxon discriminator*.

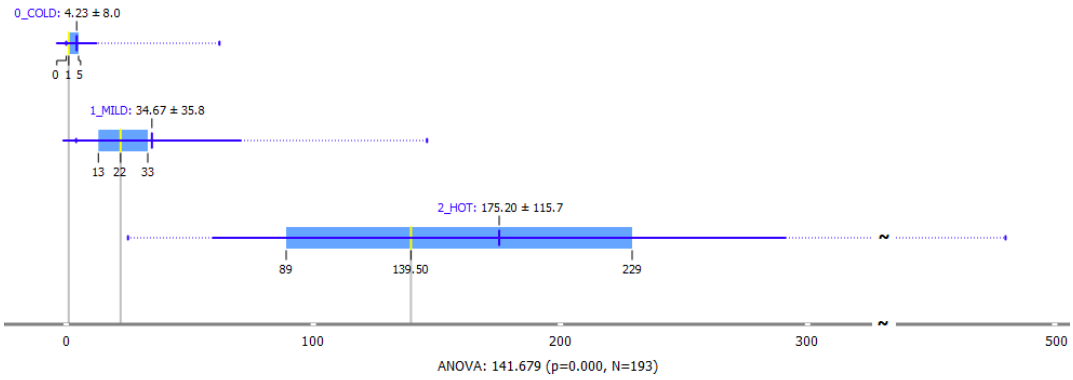


Figure 4.14 BoxPlot of TotalExpansion

Figures 4.15 and 4.16 show how the total attributes that are injected and ejected, are related to the super taxa. Observe that these figures, especially the total attributes injected which relates well to expansion as a discriminator, separate well the taxa, as well as the fact that, *moving from the 0_COLD to the 2_HOT class, the number of injections-ejections are increasing*.

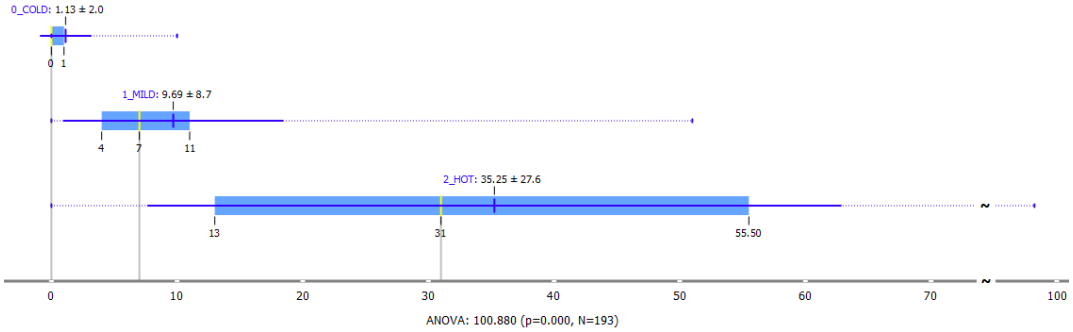


Figure 4.15 BoxPlot of TotalAttrInjected

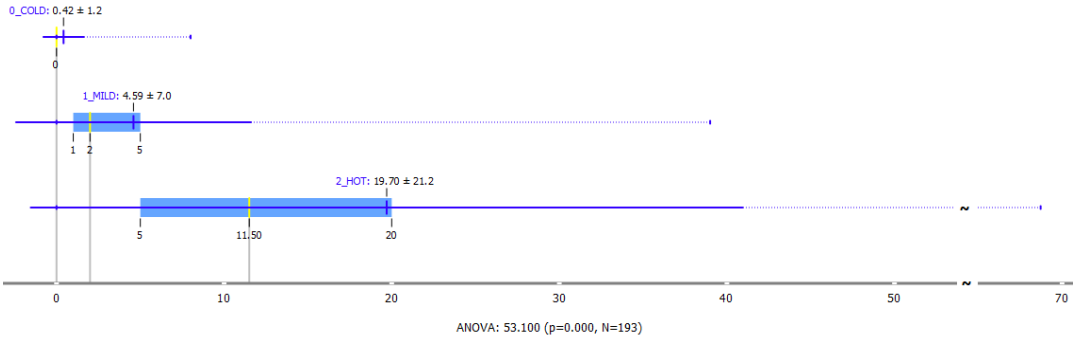


Figure 4.16 BoxPlot of TotalAttrEjected

In Figures 4.17, 4.18, and 4.19 we present the plots of the rates per commit of total activity, total maintenance, and total expansion, respectively. *The rates mentioned above, for every commit, look very similar to each other. At the same time, they all show an increase as we move from the 0_COLD class to the 2_HOT class.*

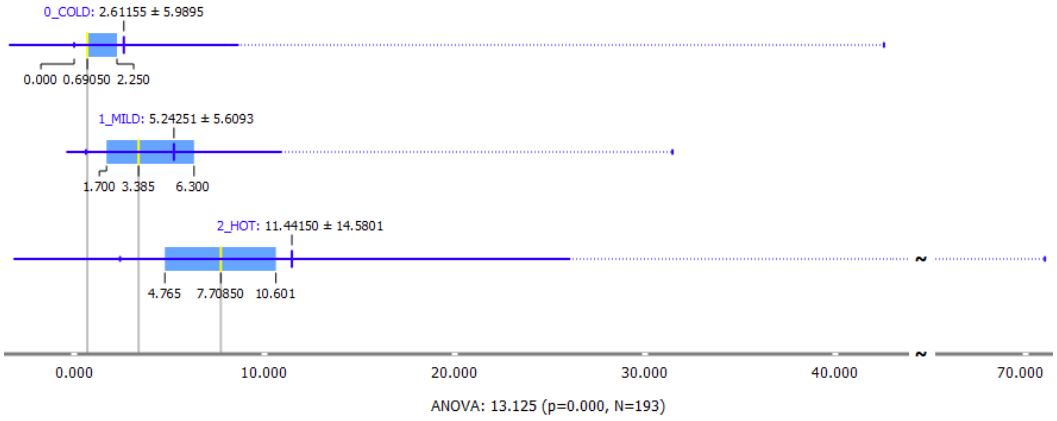


Figure 4.17 BoxPlot of TotalActivityRatePerCommit

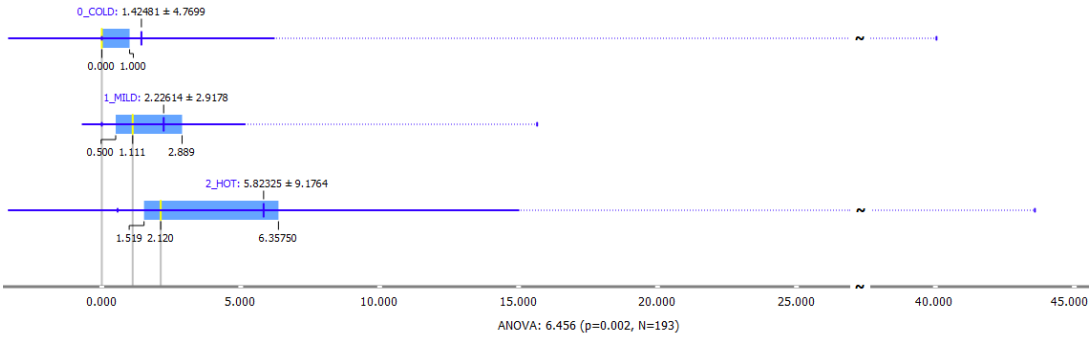


Figure 4.18 BoxPlot of MaintenanceRatePerCommit

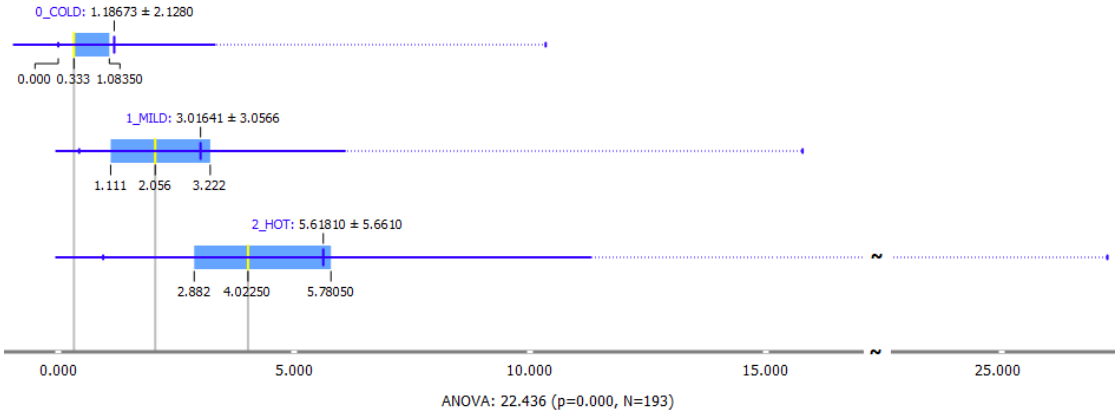


Figure 4.19 BoxPlot of ExpansionRatePerCommit

In Figures 4.20, 4.21, and 4.22, observe the corresponding plots for the same ratios per year. All these representations suggest that the taxa are clearly different concerning these activity metrics, and as the taxon becomes more active, all the related measures increase: the mean, the median, the IQR, and the range of these metrics.

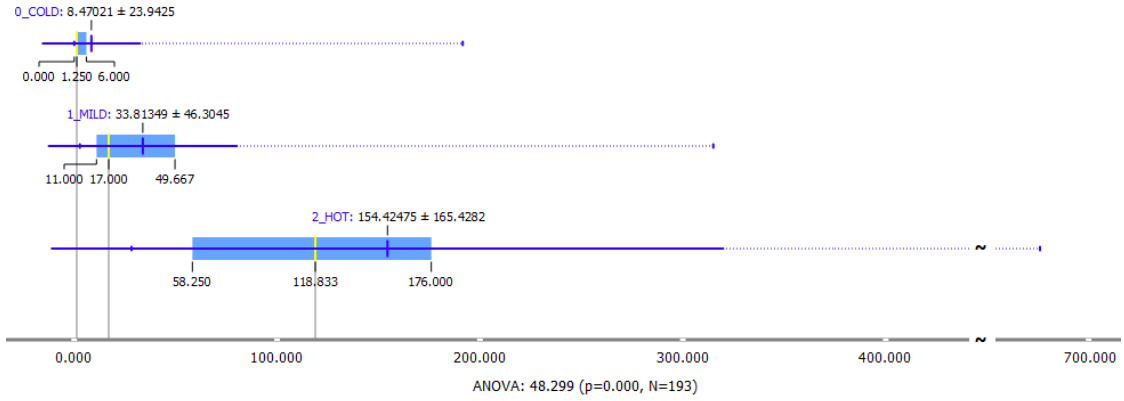


Figure 4.20 BoxPlot of TotalActivityRatePerYear

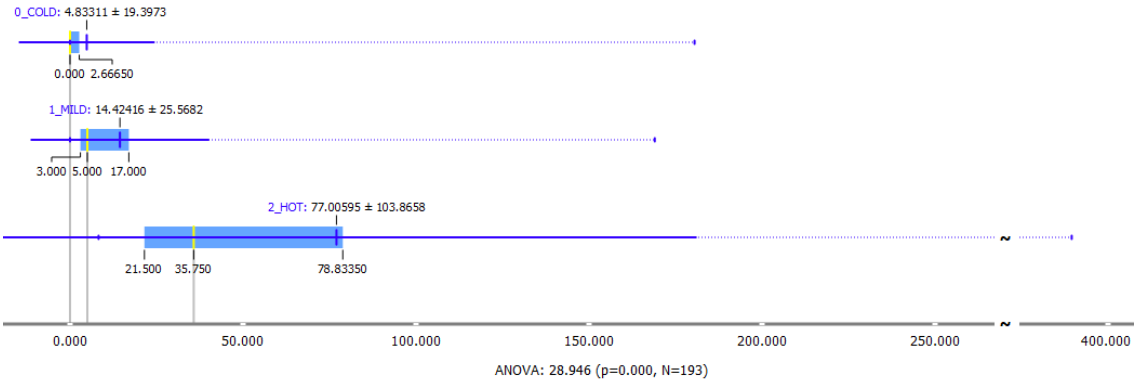


Figure 4.21 BoxPlot of MaintenanceRatePerYear

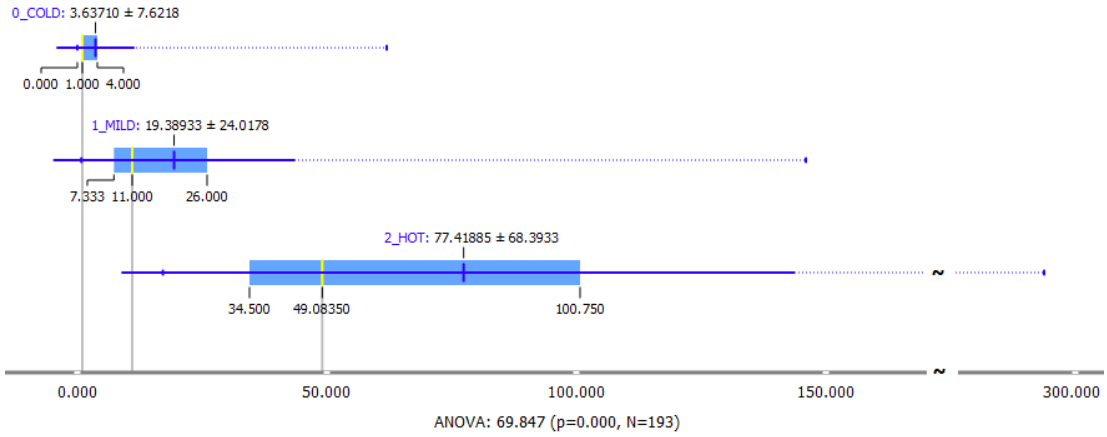


Figure 4.22 BoxPlot of ExpansionRatePerYear

It is a very important fact that there is a consistency to the plots that were presented, according to the discrimination of the taxa, which means that all the examined attributes, related to the activity, agree with the taxonomy.

In Table 4.3 we report some measures related to the activity of the projects. *Considering the activity of the projects, the measures show that the taxa are distinct among each other.*

Table 4.3 Averages of some activity-measures per taxon

Measure	0_COLD	1_MILD	2_HOT
TotalAttrInjected	1.00	10.00	35.00
TotalAttrEjected	0.00	5.00	20.00
TotalExpansion	4.00	35.00	175.00
TotalMaintenance	7.00	26.00	161.00
TotalTotalAttrActivity	11.00	61.00	336.00
TotalAttrActivityRatePerCommit	2.61	5.24	11.44
TotalAttrActivityRatePeryear	8.00	34.00	154.00

Finally, to get a deeper understanding of how the activity is related to the super taxa, let's take a look at the decision tree, which was generated by the Orange tool, presented in Figure 4.23. At first glance, observe that this decision tree is not as simple and straightforward as the one we observed in Figure 4.11, which was produced from the heartbeat metrics. This is because there are more misclassified projects in this decision tree. *More specifically, there are 20 misclassified projects out of 193, meaning 10% (Most of the misclassified, meaning 7%, are projects that belong to the 1_MILD class and have been misclassified either to the 0_COLD class or to the 2_HOT class).* Ideally, we would like the 0_COLD leaves to be at the left, the 1_MILD at the center and the 2_HOT at the right, side of the tree.

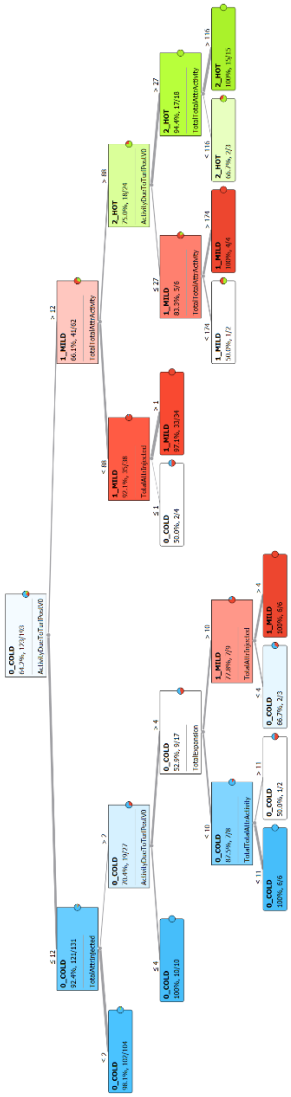


Figure 4.23 Decision Tree – Activity

4.4 Super taxa and Table-Level Activity Measurements

In this section, we will discuss the relevance between the super taxa and the table-level activity.

Figures 4.24 and 4.25 present, how the taxa are separated, based on the metrics Tables@Start and Tables@End.

The concentration of the values in taxa shows that the taxa are distinct. However, it is interesting to highlight that there are a few projects in the 0_COLD class with the biggest number of tables. This means, that the number of tables of the projects in these cases, is neither related to the activeness of the projects nor to the super taxa.

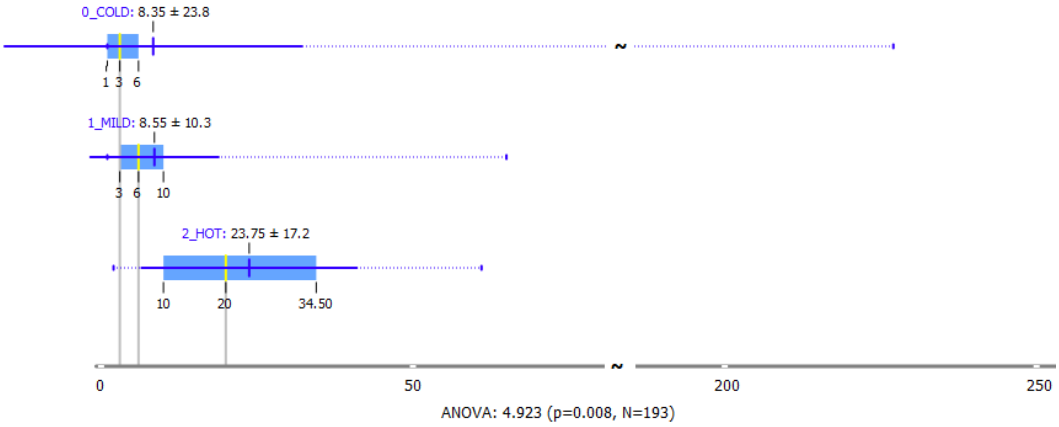


Figure 4.24 BoxPlot of Tables@Start

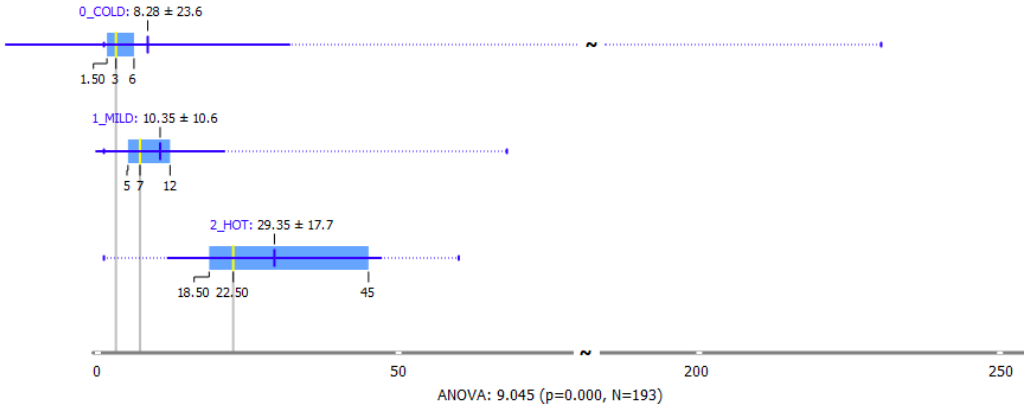


Figure 4.25 BoxPlot of Tables@End

Observe Figures 4.26 and 4.27: *The more active the taxon is, the bigger the number of the tables that are inserted is, which verifies the assumption that, the activity of a taxon is related to the number of the table operations.*

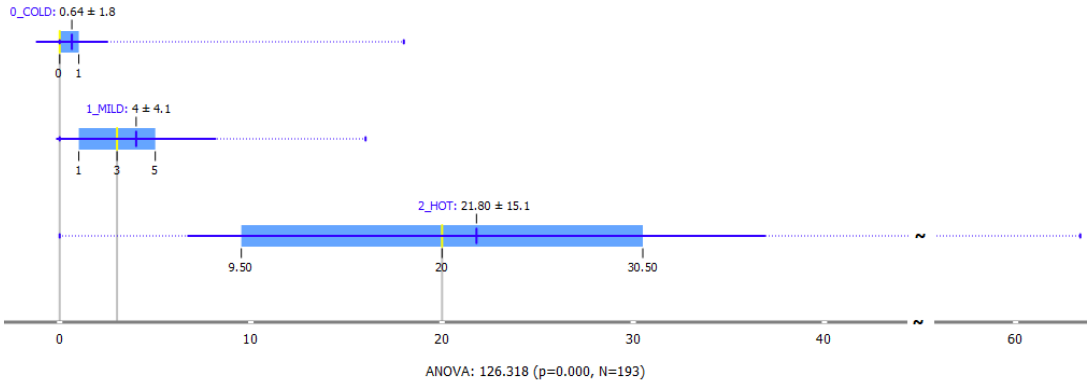


Figure 4.26 BoxPlot of TotalTableInsertions

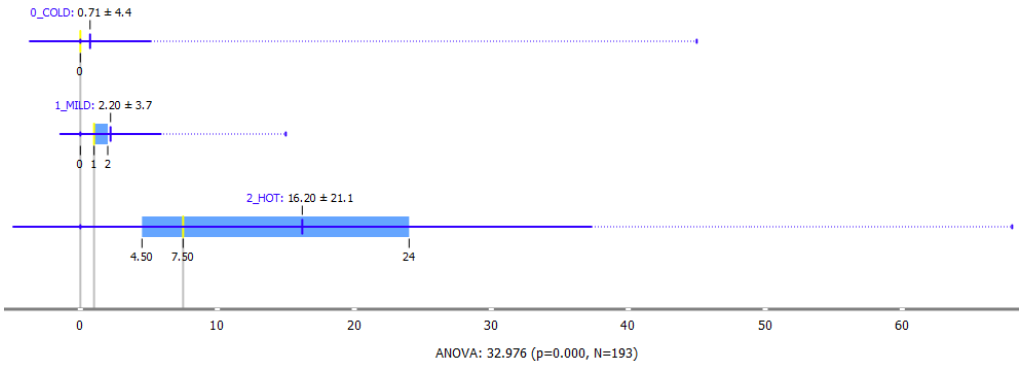


Figure 4.27 BoxPlot of TotalTableDeletions

In Figure 4.28 the relationship between the resizing ratio of the projects and the super taxa is presented. *It seems that the resizing ratio neither discretizes the taxa nor is related to the evolution of the projects.*

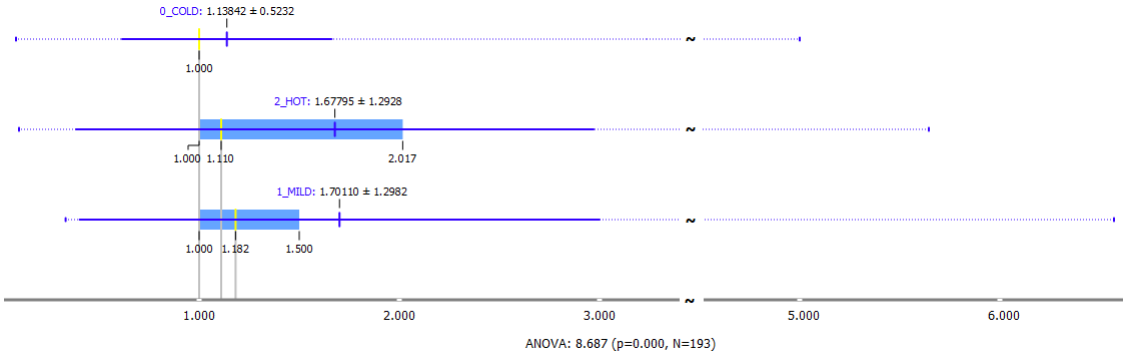


Figure 4.28 BoxPlot of ResizingRatio

In Figure 4.29 we present the schema line volume of change plot.
As already mentioned, the labels that are presented in Figure 4.29, were produced based on BD where:

$$BD = tableInsertions + tableDeletions$$

The meanings of the labels in Figure 4.29 are as follows:

- 0_NONE: BD = 0
- 1_SMALL: BD in the range [1 – 2]
- 2_MODERATE: BD in the range [3 - 10]
- 3_HIGH: BD ≥ 11

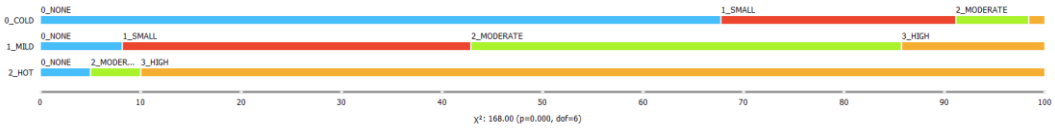


Figure 4.29 Schema Line Volume of Change Plot - Super Taxa

It is even more clear now with the super taxa that, *the more active a taxon is, the bigger the percentage of the high volume of changes and the smaller the percentage of the low volume of changes, is.*

In Table 4.4, some table-level measures are reported per taxon. *As the number of the insertions and deletions increases, so does the activeness of the project. The change of the size of the projects, as already mentioned does not play any role both in the evolution of schema and the discretization of the taxa.*

Table 4.4 Averages of some table-level measures per taxon

Taxon	0_COLD	1_MILD	2_HOT
#Tables@Start	8.00	9.00	24.00
#Tables@End	8.00	10.00	29.00
TotalTableInsertions	1.00	4.00	22.00
TotalTableDeletions	1.00	2.00	16.00
Resizingratio	1.14	1.70	1.68
BD	1.00	6.00	38.00

In Figure 4.30, we present the decision tree, which was generated by the Orange tool, based on the attributes TotalTableInsertions, TotalTableDeletions, Tables@Start, Tables@End. *Again, observe that the taxa are distinct, with a couple of misclassified*

exceptions. More precisely, there are 44 misclassified projects out of 193, meaning 22% (Most of the misclassified, meaning 18%, are projects that belong to the 1_MILD class and have been misclassified either to the 0_COLD class or to the 2_HOT class). Moreover, the number of tables that a project has at the beginning does not seem to be an indicator of its further activity. Nevertheless, the number of tables that are inserted is proportional to the activeness of the projects.

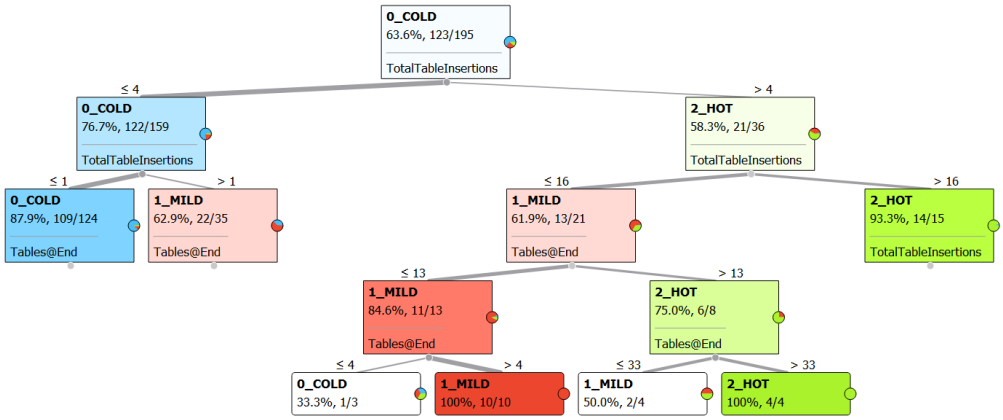


Figure 4.30 Decision Tree - Table Level Activity

4.5 Super taxa and Durations

In this section, we study how the lifetime duration of the projects is related to the super taxa. In Figure 4.31, we present the plot of the schema duration of the projects, in months, per taxon and in Table 4.5 we present the stats of the schema update period per class.

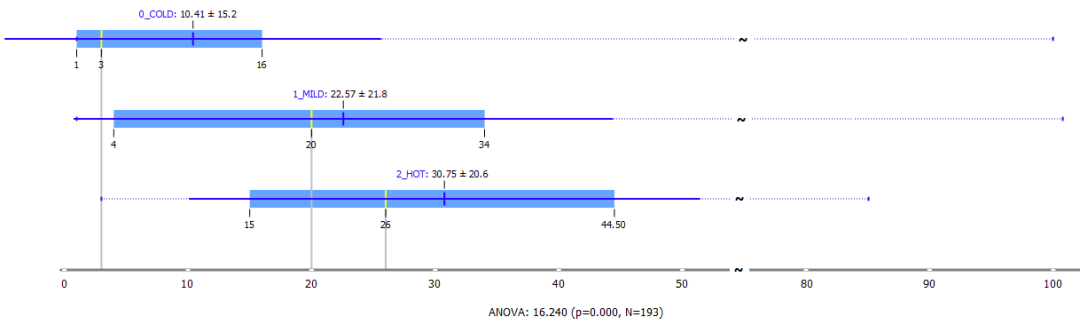


Figure 4.31 BoxPlot of Schema Update Period (months)

The concentration of the values is similar for all the taxons. The 1_MILD class overlaps both the 0_COLD and the 2_HOT class, which shows that the activity is due to the nature of the project. Because of that, there are 1_MILD projects with schema update period less than the average of the 0_COLD's ones and bigger than the average of the 2_HOT's ones. These conclusions can also be confirmed after observing the values of the Table 4.5 and the Figure 4.32, where the explanations of the labels are as follows:

- 0_UpTo10Days: Projects with schema update period in the range [0 – 10] days.
- 1_11To180D: Projects with schema update period in the range [11 – 180] days.
- 2_06To12M: Projects with schema update period in the range [181 – 365] days.
- 3_13To36M: Projects with schema update period in the range [366 – 1095] days.
- 4_LONG: Projects with schema update period greater than 1095 days (3 years).

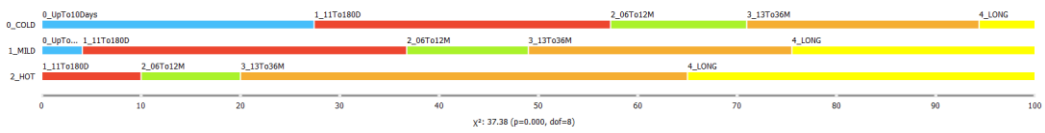


Figure 4.32 BoxPlot of Schema Update Period Class

Table 4.5 Schema Update Period stats per class

Schema Update Period (Months)	CLASS	MIN	MAX	AVERAGE	MEDIAN	STDEV
	0_COLD	1	99	10.41	3	15.25
	1_MILD	1	100	22.57	20	21.99
	2_HOT	3	84	30.75	26	21.15

Observe Figure 4.33 and Table 4.6: *It seems that the project update period is not necessarily proportional to the activity of the projects. All classes seem to have similar project update periods.*

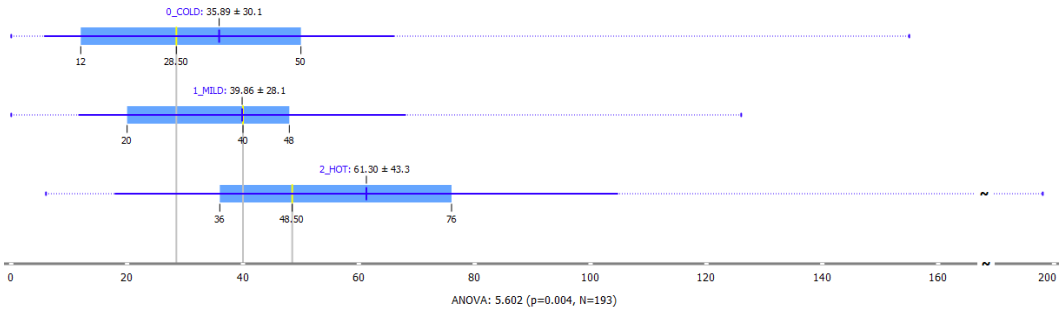


Figure 4.33 BoxPlot of Project Update Period (Months)

Table 4.6 Project Update Period stats per class

Project Update Period (Months)	CLASS	MIN	MAX	AVERAGE	MEDIAN	STDEV
	0_COLD	0	155	35.89	28.5	30.22
	1_MILD	0	126	39.86	40	28.44
	2_HOT	6	198	61.30	48.5	44.46

With that being said about the duration metrics, we conclude that the activeness of a taxon is more related to the schema update periods, rather than the project update periods. However, none of them can be considered as an absolute taxon indicator.

4.6 Centroids and characteristics per super taxon

In this section we define a centroid for each super taxon and then we discuss the characteristics of super taxa.

For the computation of the centroids, we followed the same procedure as in chapter 3. In Table 4.7 we present the centroid-project of each taxon including its actual measures as well as its Z-scores, in parenthesis.

Table 4.7 Centroid-project per super taxon

Taxon	**Centroid-Project**	Total Activity	Reeds	Turfs	Active Commits
Cold	yiier__forum	7 (-0.37)	1 (-0.25)	1 (-0.46)	2 (-0.45)
Mild	jasdel__harvester	55 (-0.02)	2 (0.22)	6 (0.11)	8 (0.14)
Hot	pods-framework__pods	352 (2.14)	9 (3.54)	21 (1.80)	30 (2.29)

The characteristics of each taxon compared to the values of the centroid-project are reported below:

Cold Taxon Characteristics (Centroid-Project: *yiier__forum)*

- *Zero to high total activity with range from 0 to 382*
  - totalActivity(yiier__forum) = 7
- *At most 3 active commits (Mix of turfs and reeds)*
  - activeCommits(yiier__forum) = 2

Mild Taxon Characteristics (Centroid-Project: *jasdel__harvester)*

- More than 4 active commits (Mix of turfs and reeds)
  - *activeCommits(jasdel__harvester) = 8*
- Medium to high total activity
  - totalActivity(*jasdel__harvester) = 55*

Hot Taxon Characteristics (Centroid-Project: *pods-framework__pods)*

- Very high total activity (more than 111 and less than 1268 updated attributes)
  - totalActivity(*pods-framework__pods)* = 352
- More than 7 active commits (Mix of turfs and reeds)
  - *activeCommits(pods-framework__pods) = 30*

Considering the values of the centroid-projects the taxa are clearly distinguished from each other. Moving from the cold to the hot taxon, both the total activity and the number of active commits noteworthy increase.

4.7 Summary of findings

In this chapter, after the observations we made in chapter 3 about the similarities of some taxa, we decided to merge these similar taxa into larger groups and observe what results we get. Our conclusions and findings are summarized as follows:

- Regarding ***Schema Heartbeat***:

  Reeds, Turfs, and Active Commits distinguish well the super taxa. This conclusion was extracted by the observation of the box plots, where the values for each taxon are distinct as well as from the scatter plot of reeds and turfs, where each taxon forms distinct color regions. Moreover, we observed in the decision tree that the taxa can be defined easily only by using the attributes Active Commits and Reeds.

- Regarding ***Activity***:

As already mentioned, the hotter the projects are, the bigger the general activity of the projects is. As we observed in the box plots, all activity measures and especially totalActivity, totalExpansion, and totalAttrsInjected are good taxon discriminators and confirm the assumption that the activity of the projects increases as we move from the cold to the hot super taxon. Additionally, in the decision tree, the areas of each super taxon are distinct; specifically, the cold land is on the left side of the tree, the mild land is on the center of the tree and the hot land is on the right side of the tree.

- Regarding ***Table-Level Activity Measurements***:

  We observed that the resizing ratio is not an indicator of how active a taxon is. The tables@Start and tables@End measures are bigger when moving to the hot class, but still, there are some exceptions and cannot discriminate the taxa alone. The measures totalTableInsertions and totalTableDeletions are bigger when we refer to the hot taxon projects than to the cold super taxon and discriminate the taxa well. We also observed in the schema line plot, that the more active a taxon is, the bigger the percentage of the high volume of changes and the smaller the percentage of the low volume of changes, is.

- Regarding ***Duration***:

  We observed that both the project duration and the schema duration, cannot discriminate the taxa, alone. We highlighted that the values of schema update period for each taxon are very close to each other. This indicates that the activity is due to the nature of the project and is not necessarily proportional to the schema update period of it. Additionally, the project update period is not necessarily proportional to the activity of the projects, as we observed that all classes had similar project update periods.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

In this thesis, we studied in great detail the schema evolution of 193 projects, as well as we assessed the taxonomy of schema evolution which was produced in [12]. Firstly, we presented HeraclitusFire, the tool which we used in our analysis, as well as some new data and statistics that were added as information because they were necessary for our further research. Afterwards, since the metrics that were extracted were too many to handle, we observed the correlations of them and ended up with the most important attributes, by discarding the most correlated ones. Then we introduced the taxonomy presented in [12] and proceeded to the evaluation of this taxonomy with the assistance of a few validity clustering metrics. Finally, after observing similar behaviors among some taxa, in chapter 4 we merged them to see how this new taxonomy is related to the heartbeat, the activity, the table-level measures, and the duration of the projects.

Concerning the taxa of [12] we can report that:

- *Regarding the schema update period, every taxon seems to have projects with both short and big periods of updates. However, moving from the frozen to the active taxon, the schema update period becomes bigger and bigger, the percentages of short periods are decreasing, and the percentages of longer periods are increasing.*
- *The more active a taxon is, the bigger the percentage of the high volume of changes in the schema line and the smaller the percentage of the low volume of changes is.*

- *The active commits separate well the taxa, except for the medium activity classes that are not only separated by the active commits, but also by the amount and concentration of change.*
- *Turfs and total activity are good taxon discriminators and proportional to the activeness of the taxon.*
- *The update periods of the projects of all families, except for the active class, seem to be big and very similar.*
- *Considering the values of the centroid-projects and especially the total activity and the active commits, the taxa are distinguished from one another.*
- *The validity metrics were better for groups consisting of attributes related to the heartbeat and the activity of the projects.*

Concerning the possibility of defining super taxa on the basis of the taxa of [12] we can report that:

- *Reeds, Turfs, and Active Commits distinguish the super taxa well.*
- *The super taxa can be discriminated easily on grounds of a decision tree which is defined only by using the attributes Active Commits and Reeds.*
- *The more active a super taxon is, the bigger the number of the active commits, reeds and turfs it has. On the other hand, all ratios seem to be close for the different super taxa.*
- *All the activity measures and especially totalActivity, totalExpansion, and totalAttrsInjected are good super taxon discriminators and confirm the assumption that the activity of the projects increases as we move from the cold to the hot super taxon.*
- *In the decision tree that was produced by activity measurements, the areas of each super taxon are distinct.*
- *The resizing ratio is not an indicator of how active a taxon is.*

- *Regarding the schema update period, the activity is due to the nature of the projects.*
- *Project update period is not related to the activity.*

In the future, it is possible to explore, which percentage of the projects per taxon abides by patterns like progressive reduction of activity, intense spikes of activity, commits of massive maintenance, commits of zero maintenance, etc, and see if the taxa can predict the patterns. Also, it is possible to define for each pattern, the property values that each project needs to fulfill. Then, one can check if intersections of these properties can be used as discriminators to create taxa, and observe which taxa are created after applying them.

REFERENCES

[1] S. Aranganayagi, K.Thangavel [ArTh07]. Clustering Categorical Data using Silhouette Coefficient as a Relocating Measure. International Conference on Computational Intelligence and Multimedia Applications, 2007.

[2] Carlo A. Curino, Hyun J. Moon, Letizia Tanca, Carlo Zaniolo [CMTZ08]. Schema Evolution in Wikipedia - Toward a Web Information System Benchmark. ICEIS '08, 2008.

[3] M. G. Kendall [Kend38]. A New Measure of Rank Correlation. Biometrika, Vol. 30, No. ½, pp. 81-93, June 1938.

[4] Dien-Yen Lin, Iulian Neamtiu [LiNe09]. Collateral Evolution of Applications and Databases. IWPSE-Evol'09, 2009.

[5] Jessica Lin, Eamonn Keogh, Li Wei, Stefano Lonardi [LKWL07]. Experiencing SAX: a novel symbolic representation of time series. Springer Science+Business Media, LLC '07, 2007.

[6] Dong Qiu, Bixin Li, Zhendong Su [QLSu13]. An Empirical Analysis of the Co-evolution of Schema and Code in Database Applications. ESEC/FSE '13, 2013.

[7] Dag Sjøberg [Sjøb93]. Quantifying Schema Evolution. Information and Software Technology, 35(1), pp. 35-44, January 1993.

[8] Ioannis Skoulis, Panos Vassiliadis, and Apostolos Zarras [SVZa14]. "Open-Source Databases: Within, Outside, or Beyond Lehman's Laws of Software Evolution?". CAiSE '14, 2014.

[9] Tan, P. N., Steinbach, M., & Kumar, V [TSKu06]. Cluster analysis: Basic concepts and algorithms. In P. N. Tan, M. Steinbach, & V. Kumar (Eds.), Introduction to data mining. Boston, MA: Pearson Addison Wesley, 2006.

[10] Panos Vassiliadis, Apostolos V. Zarras, Ioannis Skoulis [VaZS15]. How is Life for a Table in an Evolving Relational Schema? Birth, Death, and Everything in Between. International Conference on Conceptual Modeling, 2015.

[11] P. Vassiliadis, A. Zarras [VaZa17]. Survival in schema evolution: putting the lives of survivor and dead tables in counterpoint. 29th International Conference on Advanced Information Systems Engineering (CAiSE), 2017.

[12] P. Vassiliadis [Vass21]. Schema Evolution Profiles from the Study of 195 Free Open Source Software Projects. 37th IEEE International Conference on Data Engineering (ICDE 21), Crete, Greece, 19-22 April 2021.

[13] Shengfeng Wu, Iulian Neamtiu [WuNe11]. Schema Evolution Analysis for Embedded Databases. ICDE Workshops 2011, 2011.

APPENDIX A

NORMALIZATION METHODS

A.1 Method A: Z-Scoring

A Z-score is a numerical measurement that describes a value's relationship to the mean of a group of values. If a Z-score is 0, it indicates that the data point's score is identical to the mean score. Z-scores may be positive or negative, with a positive value indicating the score is above the mean and a negative score indicating it is below the mean.

Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1. Standardization assumes that dataset values fit a Gaussian distribution (bell curve) with a well-behaved mean and standard deviation. Even if this expectation is not met, time-series data can still be standardized, but may not get reliable results. To standardize the data, the mean and standard deviation of the data series are required. They can be calculated by the following math formulas:

- $\mu = \frac{sum(x_i)}{count(x_i)}$

where μ is the Mean of data series, $sum(x_i)$ is the Sum of all values and $count(x_i)$ is the Count of all values.

- $\sigma = \sqrt{\frac{\sum(x_i-\mu)^2}{count(x_i)}}$

where σ is the Standard Deviation of data series.

Finally, the new standardized value, for each value of the data series, can be computed, with the following math formula:

$$z = \frac{y - \mu}{\sigma}$$

where z is the new standardized value (also known as Z-score).

A.2 Method B: SAX (Symbolic Aggregate approXimation)

According to [5] paper, even though many symbolic representations of time series have been introduced over the past decades, they all suffer from two fatal flaws. Firstly, the dimensionality of the symbolic representation remains the same with the original data, and as a result, virtually all data mining algorithms scale poorly with dimensionality. Secondly, although distance measures can be defined on the symbolic approaches, these distance measures have little correlation with distance measures defined on the original time series.
SAX is a transformation method to convert a numeric sequence (time series) to a symbolic representation. More specifically, it is about the transformation of a time series to a sequence of symbols with a predefined length and alphabet. The produced string's length is less than the original series' length. Firstly, data is being transformed into the Piecewise Aggregate Approximation (PAA) representation, and then the PAA representation is symbolized into a discrete string. Sax representation is very robust to wandering baseline and outliers and can be computed incrementally.

Unlike other representations Sax representation of time series allows:

- Lower bounding of Euclidean distance
- Lower bounding of the DTW distance
- Dimensionality Reduction
- Numerosity Reduction

In general, the technique consists of the *following steps*:

- Compute the Piecewise Aggregate Approximation (PAA) of the time series.
- Compute the lookup table for the given alphabet, assuming that the values of the time series are normally distributed.
- Transform the generated PAA of the time series to a symbolic sequence by using the lookup table.

To the following subsections, an extending analysis of the steps of SAX representation takes place.

A.2.1 Normalization of Data

Before the transformation of the data series into the PAA representation, normalization of data is required.
Normalization is a rescaling of the data from the original range so that all values are within the range of 0 and 1. To normalize the data, the maximum and the minimum value that exists in the data series, are necessary. Afterwards, each value of the time series can be normalized by using the following math formula:

$$y = \frac{x - min}{\max - min}$$

where x is each value of the time series and y is the new normalized value.

A.2.2 PAA Representation

After the normalization of the data, the computation of the Piecewise Aggregate Approximation (PAA) of the data series, is ready to take place. PAA (Piecewise Aggregate Approximation) corresponds to downsampling of the original time series,

and, in each segment (segments have fixed size), the mean value is retained. The basic idea behind the algorithm is to reduce the dimensionality of the input time series by splitting them into equal-sized segments which are computed by averaging the values in these segments. Before proceeding to the computation of the PAA, the number of equal-sized segments must be defined. One of the most important advantages of PAA is that provides *dimensionality reduction*.

Assuming a time series Y=Y1, Y2,..., Yn of length (n) to be split or reduced into a series X=X1, X2,..., Xm where m≤nm≤n, the overall equation describing the elements in the reduced series can be summed up by the formula:

$$\bar{X}_i = \frac{m}{n} * \sum_{j=n/N(i-1)+1}^{(n/M)\cdot i} x_j$$

The above equation provides the mean of the elements in the equi-sized frame which makes up the vector of the reduced dimensional series. Nonetheless, there are immediate special cases:

- m=n: The reduced series is an exact copy of the original sequence.
- m=1: The reduced series is the mean of the original sequence.

The second case is a special case where the result is a piecewise constant approximation. Unlike the normal case, where the original input vector is split into frames and the mean of the values in the frame is computed, in this case, the reduced series is the mean of the original sequence.

A.2.3 Discretization

The final stage of SAX is discretization. In SAX, discretization of time series is the process where the numeric PAA representation is transformed into a symbolic representation. It is desirable to have a discretization technique that will produce symbols with equiprobability. The steps for the discretization are as following:

- Firstly, the alphabet that is going to be used for the generation of the symbolic representation must be declared.

- Next, given that the normalized time series have a highly Gaussian distribution, the "breakpoints", that will produce a predefined number of equal-sized areas under the Gaussian curve, must be determined.

  Definition 1 Breakpoints: Breakpoints are a sorted list of numbers $B = \beta 1, ..., \beta a-1$ such that the area under a $N(0, 1)$ Gaussian curve from βi to $\beta i+1 = 1/a$ ($\beta 0$ and βa are defined as $-\infty$ and ∞, respectively).

  Using these breakpoints, a lookup table with symbols is being created.

- Finally, for the transformation of the PAA representation into the symbolic representation, taking into consideration the lookup table, a symbol to each value of the PAA representation is assigned.

Publications of the Author

G. Angelopoulos, G. T. Kalampokis and M. Dasygenis, "An Internet of Things humanoid robot teleoperated by an open source Android application," 2017 Panhellenic Conference on Electronics and Telecommunications (PACET), 2017, pp. 1-4, doi: 10.1109/PACET.2017.8259978.

Available at: https://ieeexplore.ieee.org/document/8259978

Short Biographical Sketch

Georgios Theodoros Kalampokis is an M.Sc. graduate student at the Department of Computer Science and Engineering (CSE) of the University of Ioannina, Greece. He received his M.Eng. degree from the Department of Informatics and Telecommunications Engineering (now Department of Electrical & Computer Engineering) of the University of Western Macedonia in 2017. His research interests include data analysis, pattern extraction, data visualization, Internet of Things (IoT), etc.