Time-Related Patterns Of Schema Evolution

Panos Vassiliadis
Univ. Ioannina
Ioannina, Greece
pvassil@cs.uoi.gr

Alexandros Karakasidis
Univ. Macedonia
Thessaloniki, Greece
a.karakasidis@uom.edu.gr

ABSTRACT

This paper presents a set of patterns on how schema evolution takes place over time in Free Open-Source Systems that are built on top of relational databases. To come up with these timing patterns, we have studied 151 projects of a public dataset of schema histories, with duration more than 12 months. The eight timing patterns of schema evolution are largely grouped in 3 families: (a) the family of limited and focused-in-time change, whose members differ only on when the change happens (early, middle or late life of a project), (b) the family of regular updates, whose members differ in the frequency of change, and (c) the family of middle or late initiation of change, whose members differ in the initiation of schema maintenance as well as the frequency of change.

1 INTRODUCTION

Schema Evolution is the process of alteration of a database schema via the addition, deletion, and update of its constituents and their relationships. In the particular case of relational schema evolution, the elements that can change are the tables, attributes, constraints (including primary and foreign key, non null, data type and value constraints) and views.

The goal of this paper is to investigate the timing nature of schema evolution and extract patterns of how schemata evolve over time.

In sharp contrast with its importance, schema evolution has notoriously been under-studied in the more than 50 years of database research. The literature on case studies, with the exception of a single study [36], appears only after the late '10s, when the existence of public code repositories like GitHub or Gitlab allow the publication of the history of DDL files of Free Open-Source Software. Since then, the literature includes several studies of the characteristics of relational schema evolution with a focus on the volume, change-type breakdown and impact to surrounding queries [36], [10],[24], [50], [31], [8], [37], [47], [46], [11], [44], [12], [4], [42], [33],[45] – however, to the best of our knowledge, none has come up with patterns of how schemata evolve over time (see Section 2).

Patterns. *The major contribution of this paper is that, for the first time in the related literature, a set of time-related patterns on how schema evolution unfolds in time is introduced*. To fill the gap in our scientific body of knowledge we had to overcome several non-straightforward challenges: (i) previous work mostly dealt only with a handful of case studies, making the extraction of patterns impossible, due to the small size of their corpus; (ii) from the methodological point of view, there is no established method in our discipline for extracting patterns from schema histories; (iii) no prior knowledge of how these patterns look like exists. To address the volume challenge, we employ a large corpus of 195 relational, logical-level schema histories ([42], [45]) out of which we extracted 151 schema histories of length higher than 12 months for further study. To address the methodological gap, we have employed methods from empirical software engineering [1], and iteratively grouped the collected schema histories in *patterns of similar time evolution* in a qualitative fashion, later to be quantitatively verified, too. We also verified the *cohesion* and *disjointedness* of the patterns.

A first family of patterns, called Be *Quick of Be Dead*, concerns 2/3 of the dataset corpus, organized in four patterns of very little, focused change. The most rigid of all is the *Flatliner* pattern, which demonstrates no change whatsoever. There are 3 other patterns of minimal change, practically in a single point of change, differing just in the timing of when this happens: *Radical Sign* for early change, *Sigmoid* for middle-life change, and *Late Riser* for late change. The sheer cardinality of this family demonstrates vividly the aversion to change that we observe in the studies of schema evolution at large numbers. A second family of patterns, *Stairway to Heaven*, includes two patterns, *Quantum Steps* and *Regularly Curated*, both with regular steps of schema evolution, the former with few and the latter with more. Finally, the *Scared to Fall Asleep Again* family of patterns includes two patterns that demonstrate late change: *Smoking Funnel* with schema birth in middle-life and regular curation afterwards and *Siesta* with early schema birth, a very long period of inactivity and very late focused changes.

Traits of change. We demonstrate that the anecdotal evidence of "freeze the schema first; then build all the applications on top of it", although certainly majoritarian as a practice, is only partially corroborated, as Fig. 3 suggests. In fact, we observe two important traits. The first trait, *aversion to change*, concerns the 2/3 of the projects and means that curators avoid change as much as they can, resulting in the schema freezing shortly after birth with at most a few changes. The second trait, concerns the 1/3 of the projects, with *observable, regular evolution*, coming in several fashions: rare or dense, yet regular, change (amassing to 25% of the corpus), and, surprisingly, *late change* too, for an 11% of the corpus.

Stats: (a) schemata are typically but not exclusively born early: half projects show schema birth in the first 10% of time; (b) 42% of the projects reach more than 90% of their total activity in the first 25% of the project's life, (c) attainment of the entire change activity is mostly fast: 2 out of 3 projects have zero active months from schema birth to 90% of their total activity.

Characteristics. Change is biased towards expansion and mostly done via table additions and deletions rather than via table refactoring. Patterns do not differ in their project duration. In terms of activity volume, only Smoking Funnel and Regularly Curated projects escape the "start small; show small change next" norm.

Prediction. The point of schema birth, gives a coarse indication of the subsequent evolution: if born in M0 or after the first year, the schema has a strong inclination towards rigidity (75%

and 64% resp.); birth within the first year however, gives a 53% probability, resp.

Contributions. In a nutshell, the contributions of this paper are as follows:

- We present, for the first time in the related literature, a set of time-related patterns, i.e., archetype behaviors, on how schema evolution unfolds in time. Specifically, we introduce 3 families of patterns, including 8 patterns to highlight the traits of change in schemata of FOSS projects. The *Be Quick or Be Dead* family of patterns includes the vast majority of schemata of 2/3 of the studied corpus of projects with aversion to change. At the same time, a sizeable minority of 25% of the corpus comprises the the *Stairway to Heaven* family of patterns with steps of regular change. Third, the the *Scared to Fall Asleep Again* family of patterns refers to a small minority of 11% of the corpus where schema change comes late in the life of a project.
- We validate the introduced pattern-set in terms of common sense, cohesion, disjointedness, completeness and generalization.
- We present the relationship of time-related patterns to other measures of schema evolution, like the total amount of schema change, the point of schema birth and the mixture of change types.

Roadmap. After reviewing related work in Section 2, we present our experimental method in Section 3 and the resulting set of schema evolution patterns in Section 4. In Section 5, we present the validation of the pattern set that we introduce. In Section 6, we discuss the relationship of the introduced patterns to other measures of schema evolution activity. We conclude our deliberations in Section 7.

2 RELATED WORK

Studies of Schema Evolution. Typically, the studies in the area of schema evolution concern the qualitative and quantitative description of schema evolution on the basis of few studied projects (no more than a dozen). Several studies address this question in the field of relational databases [36], [10],[24], [50], [31], [8], [37], [47], [46], [11], [44], [12], [4], [35]. For completeness, we point out the recent work on (i) non-relational databases [21], [3], [34] – see [40] for an overview and (ii) content change [2], [38].

However, none of these studies was able to report on patterns of how evolution happens over time, to a large extent due to the very small corpus of schemata they study. Most studies are interested in the change breakdown per type of change, as well as aggregate information for the schema studied, and apart from the occasional reporting for the schema size over time, there are very few data for time behavior. [36] reports the monthly heartbeat of a single schema with aggregate numbers of changes per schema version (along with time information). [50] gives the number of changes per revision. [10] reports the number of commits per month and the number of changes per commit. [31] gives schema size over %time progress. [8] gives the number of tables, attributes and changes per commit. [37] discusses schema heartbeat characteristics mostly qualitatively. [44] lists the number of tables and foreign keys over time. In [42], the first large-scale study of 195 projects was reported (see a long version in [43]). This corpus lays the foundation allowing us to systematically study patterns of schema behavior at large scale.

It is noteworthy that whereas early studies [36], [10], [31] focused (by selection) on projects with significant change, later large-scale studies [42], [43],[45] restrict this evolutionary profile only for a subset of actively evolving schemata and report *aversion to change* (zero, or just a couple of time-points with schema change) as well as the *gravitation to rigidity* (early change with a long tail of inactivity in the schema line) as the typical pattern of schema evolution for a large majority of the studied schemata.

Evolution is Hard. Schema evolution can have a very large impact to the ecosystem built around the database [23, 39]. To alleviate the difficulty of the evolution, several attempts have been made. [18] and [17] investigate multi-version database management. Attempts to tackle data migration exist too: see [22, 40, 41] and [19]. Research has also investigated how to adapt queries whenever the schema changes [13], [16], [25], [27], [26], [28], [29] – see [5] for an overview.

Schema and Source Co-evolution. In the meanwhile, there are also works on how schema and source code co-evolve, both in the area of studying the joint evolution and in the area of proposing techniques to synchronize schema and applications as they both change [36], [24], [50], [31], [15], [33]. A detailed discussion of studies in source and schema co-evolution is found in [45], which is the closest work to the current paper, discussing the lag between the schema and source-code evolution, without presenting time-related patterns of evolution, however.

Comparison to previous work. To the best of our knowledge, although activity patterns in terms of volume, and statistical patterns of table and foreign keys behavior have been previously studied, *the timing aspect has never been investigated in the past. This is the first time ever that patterns of evolution in terms of time are identified, visually demonstrated and quantitatively characterized*.

3 EXPERIMENTAL METHOD

Our fundamental **research question** is: *Is it possible to detect patterns of schema evolution with respect to when evolution takes place in the lifetime of a schema?*

3.1 Protocol

The basic protocol that we have followed was as follows. First, we have obtained the data set of [42], [45] coming with 195 histories of schemata of Free Open-Source Software Projects based on relational database support. The extracted schema histories include (a) detailed sequences of versions of the schemata and (b) the logical-level changes that took place (tables and attributes added, deleted, maintained) and their quantitative measurement, and, (c) the respective measurements of the history of the source code of their entire encompassing projects, from GitHub. The selection process for the data set of schema histories in [42] used Big-Query to select projects with .sql files from the GitHub Activity and the Library-io datasets; kept only original repositories, with more than 0 stars and more than 1 contributor; cleaned up noise (e.g., projects with the terms 'example, demo, test, migrat' in their path), to end up in 327 repositories, out of which 132 zero-evolution repositories were omitted. For surviving significant projects, their repository was locally cloned and used to extract the schema history. *Thus, the scope of the data set includes as many projects from GitHub as possible that are valid, acknowledged, non-toy, non-demo, and non-test.* For details, we redirect the reader to the respective papers ([42] and [45]). Monthly measurements for both schema and source code evolution accompany the data set. In addition, we visually represented the cumulative progress of schema and source code per project.

For the purposes of the research reported here, we have excluded all projects with a life time less or equal to 12 months, in order to be able to observe the events in the life of projects to evolve for a minimum lifespan. Thus, all projects studied here have more than one year's life span. The resulting data set consists of 151 projects.

Second, we manually searched for patterns of the schema line and annotated projects accordingly. This process was *iterative* and based solely on the aforementioned visual representation of the cumulative progress of schema evolution. This bottom-up process took several rounds before it converged into a set of stable patterns, along with a set of exceptions that were intentionally earmarked as such. The choice of manual annotation is typical in research design and intentional: the idea is to have a golden standard of *meaningful*, humanly-verified groups *first*, and *then* to check whether they are also quantitatively meaningful. This allows us to extract the different patterns by careful inspection of *only* the subjects being studied (i.e., to alleviate bias as there are no influences from quantitative properties), and only after a set of meaningful patterns has been extracted, to search for quantitative justifications. To forestall any possible criticism on this tactic, we also refer the reader to [20, 30] for a discussion of manual annotation and [7, 32, 48, 49] as examples from the database literature. Most importantly, we mention [1] for a methodological instruction of Grounded Theory [6], [9], [14], a method suited for situations where patterns have to be extracted from collected data via a manual, iterative generation of patterns via comparing every new data item (here: schema history) to all previous patterns, and adjusting the patterns accordingly.

Third, our initial *qualitative* assessment was also accompanied by *quantitative* evidence that verifies that the essential disjointness of the discovered patterns (see Section 5), along with their cohesion and completeness. A decision tree misclassified only 4 of 151 patterns after the manual classification had taken place (Fig. 5). Our analysis also shows that patterns can be grouped in larger families that are pairwise different, and internally cohesive (with slight visual differences and similar numeric characteristics).

Fourth, once all projects were assigned to the respective patterns, the final data analysis took place. All data, results, charts and auxiliary analyses are available at
https://github.com/DAINTINESS-Group/Schema_Evolution_Datasets.

3.2 Nomenclature and Metrics Used

We use the metrics of [45] and specifically the cumulative fractional activity of *schema and project heartbeats*, which measures the % cumulative change of schema evolution over time (typically, over normalized time progression as a percentage of project lifetime). The unit for measuring schema evolution is the *number of affected attributes* (born with new tables, injected into existing ones, deleted with removed tables, ejected from surviving ones, with their data type changed, or their participation to a primary/foreign key updated). Based on this cumulative measurement of progress, we use the following metrics and landmarks for the life of schemata:

Project Update Period (PUP): the time between the *originating version* (referred to as V_p^0) and the last commit of the project - practically, the life span of the project. The granule of time measurement is the month (and to this end we summed up maintenance activity by month). We also normalize time of PUP; all

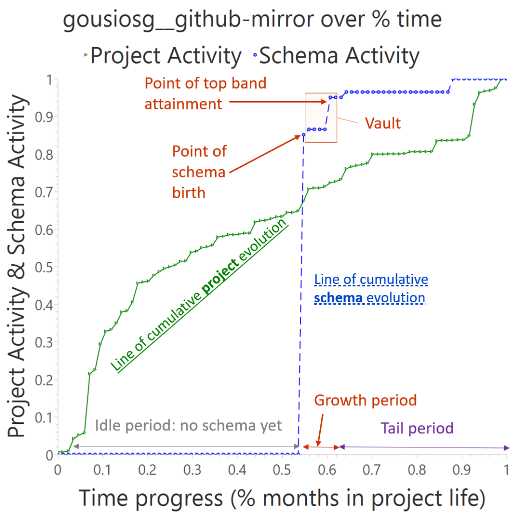


Figure 1: Nomenclature for schema and source code histories [45]. The horizontal axis shows time as a percentage of a project's life. The vertical axis depicts the cumulative progress as a percentage of the total amount of evolution activity, for (a) the schema (dotted, blue line) and (b) the source code (solid, green line). Two points, schema birth and attainment of 90% of total activity, demarcate the growth period. A vault occurs when the transition between schema birth and top-band attainment is less than 10% of the total time.

times used in the charts of this paper, unless otherwise stated, measure time as a percentage of PUP.

Schema birth and volume of activity at birth. Schema birth refers to the point when the schema appears for the first time in the life of a project. Volume measures the percentage over the total schema activity that the birth of the schema carries (quite often the percentage of schema evolution at birth is very high – a clear indicator for the absence of significant change in the schema, esp., if schema birth happens early).

Time of attainment the Top-Band of Schema Activity: the time point of the attainment of the *Top Band* threshold of 90% of the total activity of schema evolution - practically signifies when the schema evolution is close to full attainment.

Intervals between schema birth and reaching the top-band, as well as between the attainment of top-band status and the end of the project's lifetime. Growth is the period between schema birth and top-band. We say that the project has a *vault* if the time span between schema birth and reaching the top-band is lower than 10% of project's life. The interval between top-band attainment and end signifies how long the *tail* of almost-no-change is in the line of the schema activity (frequently, this line is too long).

Activity from schema birth to top-band. We measure active months in the proper interval between the months of schema birth and attainment of top-band (*ActiveGrowthMonths*). To normalize the values, we report them as percentage of (a) the growth period only, and, (b) the entire Project Update Period. The higher the percentage is, the more steps schema evolution took, and the denser the evolution rate is.

3.3 Quantization of the Measures of Schema Evolution

To be able to perform analyses later, it is necessary to quantize the different metrics that have been presented in this section, into ordinal sets of labels. Table 1 presents (a) how the different metrics have been quantized, as well as (b) the number of projects that pertain to each of the produced labels.

The determination of limits was made with the goal of providing labels that (a) are reasonable, and, (b) coarsely divide the metric into labels with similar amounts of projects pertaining to them. As always, the target is not to a-posteriori fit the quantization of the metrics to the patterns. Also, we want to avoid overfitting the labels to the particular data set. The extreme values (zero and one) have particular semantics in most cases, too. For example, discriminating the originating version from the early versions of the schema history is reasonable both statistically, and, semantically. To give an example from Table 1, see the quantization of *Time Point Of Birth*: one third of the schemata were born in V_p^0; another third in the first quarter of the time (i.e., 105 of the 151 projects were born in the first 25% of the project's life); the final third is quantized as middle-life and late. The quantization was made having in mind the goal to produce general classes of behavior rather than overfit the data set (which results in one exception in the Late Riser pattern – to be introduced later– exactly because the labeling was not done with the patterns in mind).

3.4 Statistical Properties of Time-related Measures

3.4.1 Quantization of Metrics. We quantized the important metrics in histograms of 10 buckets, with special care for special values like 0 and 1. All the Shapiro-Wilks normality tests verify the non-normal character of the data with the highest p-value for any of the involved attributes in the order of 10^{-9}. Table 1 demonstrates the breakdown of values of the different measures. The statistical properties of the important time-related measures are as follows:

- *Volume of Birth (%Total Change).* Reading from right to left, we see that out of 151 projects, 39 projects reach *Full* (100%) activity at schema birth, and 83 projects overall reach High or Full activity at schema birth. Overall, *more than half of the projects exceed 75% of total activity at schema birth.*
- *Time point of Birth (%PUP).* A large majority of schemata is born early: 52 schemata (one third of the population) are born in V_p^0. *Two thirds of the projects (105 projects) see schema birth at V_p^0 or before 25% of the PUP. Not shown in the table is that 74 schemata (half the corpus) are born in the first 10% of time.* As the distribution of points of birth follows a power-law shape, the rest of the time points form a long tail.
- *Time point of Entering the Top Band (%PUP). 64 projects (42%) reached the top band immediately at V_p^0 (23 of them) or before 25% of the PUP.* Another 40 projects (26%) came with a late top-attainment at higher than 75% of the PUP. The rest of the projects were spread in the middle half of the PUP. *In other words, projects mostly reach top-band soon, although mid-life and late completion of the evolution do exist too.*
- *Interval Schema Birth-To-TopBand (%PUP) and Interval TopBand-To-End (%PUP).* 88 out of the 151 projects had an interval from schema birth to top-band that was less than 10% of the PUP, with 62 of them in zero time. 115 (75%) of the projects had an interval of less than 35% PUP. *The rise from schema birth to the top is therefore mostly fast.* Again, the distribution of values follows a power-law shape. The inverse situation holds for the TopBand-To-End interval, which typically presents (very) long tails of schema inactivity at the end of the projects' life.
- *Vaults.* In line with the above, 58% of the projects had a single vault in the cumulative progress line, and 42% did not.
- *Active Growth Months. 98 schemata (2/3 of the population) had zero active months in the growth period, from schema birth to the top. Not shown in the table is that another 17 had exactly one month active. Combined, 76% of the population had no more than 1 month of activity from schema birth to top.* If there should be a single metric of evidence for the aversion to change, the super-focused nature of schema evolution, and, schema stabilization, *Active Growth Months* would be the one.

3.4.2 Correlations of Metrics. The correlations between the time-related measures are also interesting. Figure 2 demonstrates a "clean view" of the Spearman correlations and anti-correlations. The correlation graph reveals some interesting properties. The *ActiveGrowthMonths* is very tightly related to its normalized versions as percentages of the Project Update Period and the Growth Period. To the extent that *ActiveGrowthMonths* is a simple and intuitive measure we will be using it in our subsequent deliberations as representative of the respective metrics. A more interesting positive correlation occurs when observing the percentage of total schema change that took place at schema birth (*BirthVolume_pctTotal*) and the normalized (as percentage of PUP) Interval

Volume of Birth (%Total Change)	Low <=0.25 (16)	Middle (0.25 ..0.75] (52)		High (0.75..1) (44)	Full 1 (39)
Time Point of Birth (%PUP)	V_p^0 0 (52)	Early (0 .. 0.25] (53)		Middle (0.25..0.75] (33)	Late >75% (13)
Time point of reaching Top Band (%PUP)	V_p^0 0 (23)	Early (0 .. 0.25] (41)		Middle (0.25..0.75] (47)	Late >75% (40)
Interval (%PUP) (birth .. top-band)	Zero 0 (62)	Soon (0 .. 0.1] (26)	Fair (0.1 .. 0.35] (27)	Long (0.35 .. 0.75] (23)	Very Long >75% (13)
Interval (%PUP) (top-band .. end]	Soon <=0.25 (40)	Fair (0 .. 0.25] (48)		Long (0.75..1) (40)	Full 1 (23)
Active months as %growth	Zero 0 (98)	Few (0 .. 0.2] (22)		Fair (0.2 .. 0.75] (22)	High >75% (9)
Active months as %PUP	Zero 0 (98)	Fair (0 .. 0.08] (20)		High (0.08 .. 0.5] (33)	Ultra >50% –

Table 1: Labeling limits of Schema Evolution Metrics (PUP: project life span; Top-band: having reached 90% of total schema evolution). The numbers in parentheses mark the number of projects that pertain to the particular label.

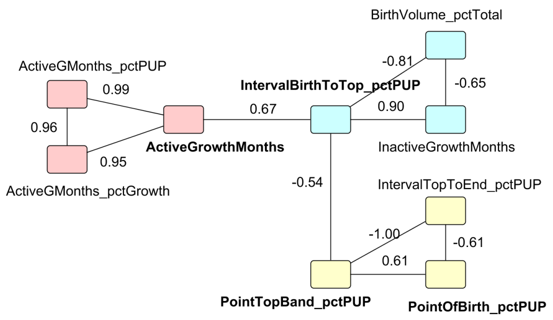


Figure 2: Spearman Correlations of Time-Related Metrics

between Point of Schema Birth and Point of Top Band Attainment (*IntervalBirthToTop_pctPUP*). The higher the change percentage at schema birth, the closer the attainment of the top band. At the same time, it is really intriguing to see that both these measures are strongly related to the months *without* change in the growth period. As we have frequently advocated in the past [42, 45, 46], aversion to change is a constant characteristic that we encounter in our studies. Here, we see it again: the longer it takes to reach the top-band, the more inactive months you include, i.e., people prefer clustered groups of schema changes rather than constant incremental maintenance. Finally, we can see that the Point of Top-Band Attainment and the Interval from the Top-Band to the End of the project are extremely strongly anti-correlated. This is expected, as the later a project reaches the top band, the smaller tail it can have. Also, there is a fairly strong relationship to Point of Birth: a later schema birth time point pushes the top-band attainment later, although the different "character of change" of the projects make them have varying growth intervals that lower their correlation to 0.61.

4 THE PATTERNS OF SCHEMA EVOLUTION

In this section, we identify 8 patterns of change organized in 3 pattern families (see also Figure 4). Specifically:

- The (most voluminous) *Be Quick or Be Dead* family with patterns of focused change around the point of schema birth (*Flatliners*, *Radical Sign*, *Sigmoid*, *Late Risers*)
- The *Stairway To Heaven family* (*Quantum Steps*, *Regular Curation*) for projects with fairly regular rate of change
- The *Scared to Fall Asleep Again* family (*Siesta*, *Smoking Funnel*) with change starting (too) late in the life of the project

All the definitions are based on a small subset of four features; the defining values for each pattern are depicted in bold.

4.1 Flatliners

Flatliners is a category of schemata that are practically frozen and include zero or very few changes to their logical schema, all of which take place in the originating month of schema birth. These projects are born at the originating version of the project and remain unchanged until its end. The respective DDL file receives commits, but these commits serve mostly comments, inclusion of initialization values to reference tables or changes concerning the tuning of the database - in any case all change is completed within the first month, thus generating a flat schema line in the diagram.

Definition 4.1. A **flatliner** schema comes with both a Point of Schema Birth and a Top-Band Attainment Class at V_p^0 .

Birth Class	Top-Band Point Class	Birth-To-Top Interval Class	Active Growth Months
V_p^0	V_p^0	Zero	0

As a consequence, the top band is attained immediately, at the first point of the line, and all subsequent activity measures are zero.

4.2 Radical Sign

The category of Radical Sign projects is the pattern with the largest population of schemata. The shape of the line of the cumulative schema evolution progress is the name-giver of this pattern.

Definition 4.2. A **radical sign** schema comes with a Point of Schema Birth Class V_p^0 or *early* and a Top-Band Attainment Point Class *early*.

Birth Class	Top-Band Point Class	Birth-To-Top Interval Class	Active Growth Months
V_p^0 or **early**	**early**	*	0

Overall, the projects of the radical sign pattern are born early (in the first quarter of the lifespan of the project), and in a very short period after schema birth, they rise to the top-band, and in fact, to 100% of their cumulative schema evolution. The presence of this climbing in a very short period signifies the presence of a sharp vault in the graphical representation of the line of the schema heartbeat. As all this happens early enough, the line of cumulative progress of schema evolution includes a long tail of inactivity.

4.3 Sigmoid

Sigmoid is a category of schemata that are born in the middle of the life of the project and include a very sharp rise to the top band at the point of birth, which is followed by a fairly long tail of frozen inactivity. The line formed is a typical line of a sigmoid function with right angles.

Definition 4.3. A **sigmoid** schema comes with a Point Of Schema Birth Class *middle*, a Top-Band Attainment Point Class *middle*, and a Schema Birth-To-Top Interval Class *zero* or *soon*.

One can think of the sigmoid as the archetypical mother of all the “almost-no-evolution" patterns - i.e., think of the shape of the line of the cumulative schema heartbeat of the other patterns as variations over sigmoid. Sigmoid differs from the radical sign projects in the fact that the schema is born in the middle of the lifetime of the project and very soon (in most cases, immediately) freezes.

Birth Class	Top-Band Point Class	Birth-To-Top Interval Class	Active Growth Months
middle	**middle**	**zero** or **soon**	0-1

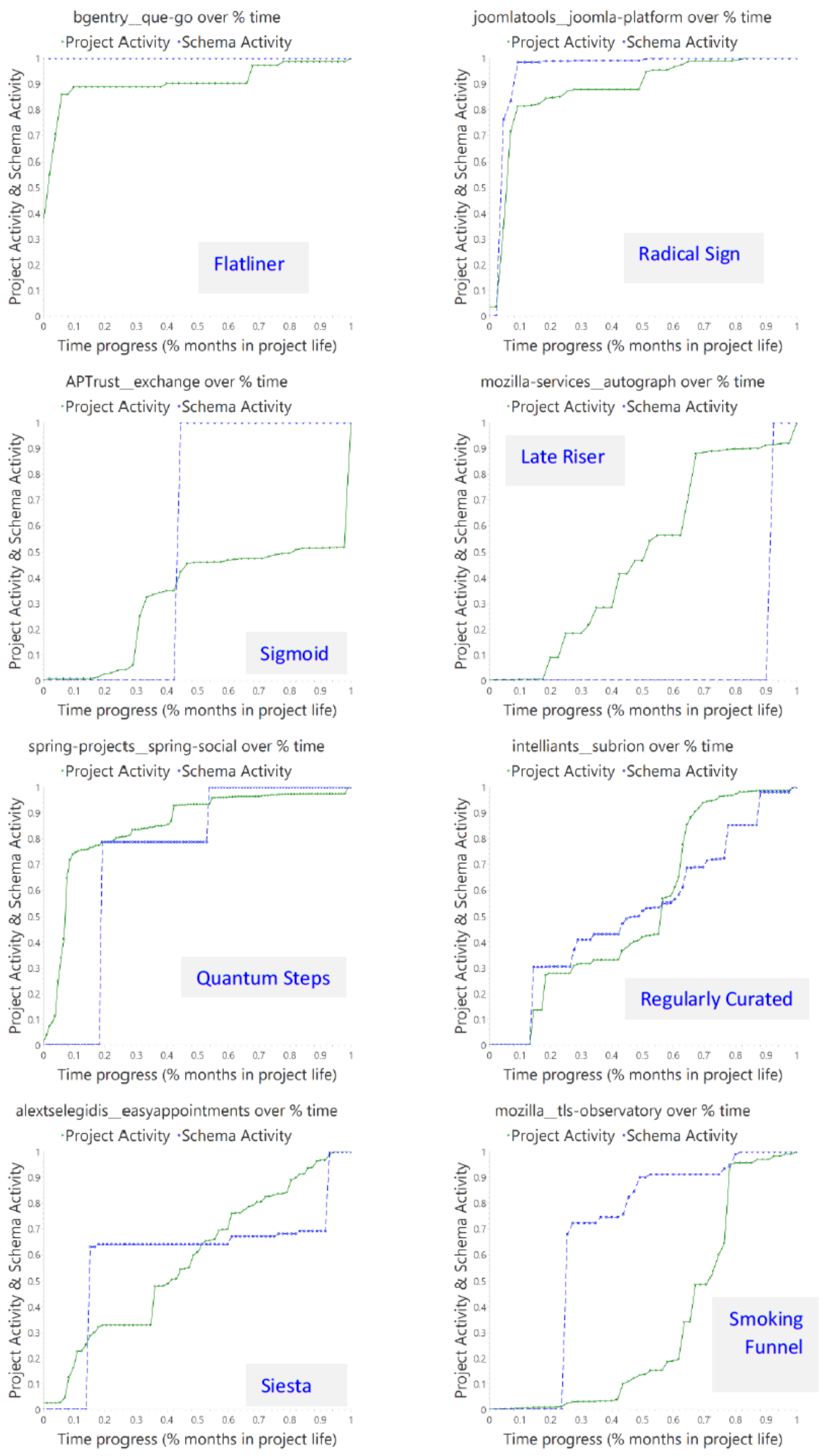


Figure 3: Examples of schema evolution time-related patterns (schema: blue dotted; src: green solid)

4.4 Late Risers

Schemata in the Late Risers pattern are born late and include very little change afterwards. Due to this late birth, there is no early or middle period of the schema lifetime, and thus the life of the schema is pretty much summarized by this sharp vault.

Definition 4.4. A **late riser** schema comes with a Point Of Schema Birth Class *late*, a Top-Band Attainment Point Class *late*, and a Schema Birth-To-Top Interval Class *zero* or *soon*.

Birth Class	Top-Band Point Class	Birth-To-Top Interval Class	Active Growth Months
late	**late**	**zero** or **soon**	0

The "Be Quick or Be Dead" family of patterns constitutes a family of very focused change very close to the schema birth point - the only difference of the involved patterns is when schema birth takes place.

4.5 Quantum Steps

The Quantum Steps pattern is characterized by the existence of focused points of schema modification in the period between schema birth and reaching the top band. The schema evolution activity is not dense, but is focused on few (no more than 3) active months in the journey between schema birth and top-band attainment.

Definition 4.5. A **quantum steps** schema comes with two variants, both with less than 3 active growth months:

- a variant with a Point of Schema Birth Class V_p^0 or *early*, and a Top-Band Attainment Point Class *middle*
- a variant with a Point of Schema Birth Class *middle*, a Top-Band Attainment Point Class *late*.

Birth Class	Top-Band Point Class	Birth-To-Top Interval Class	Active Growth Months
V_p^0 or **early**	**middle**	**fair** or **long**	0-3
middle	**late**	**fair** or **long**	0-3

The quantum steps pattern is probably the most heterogeneous and inclusive pattern of all. In the major variant of the pattern, the projects are mostly early born and come with all possible volumes at schema birth, the line reaches the top band mostly in medium time intervals from schema birth in few focused points of schema evolution, and once the top band is reached, a fairly long tail follows. In the second variant of the pattern, the schema is born in the middle of the life of the project, and, slowly arrives to the top band late in the lifetime of the project (thus with a short tail afterwards). Again, the change is small and rare.

4.6 Regular Curation

The Regularly Curated pattern includes more active projects with respect to the spread and density of change over time.

Definition 4.6. A **regularly curated** schema comes with two variants, both with more than 3 active growth months:

- a variant with a Point Of Schema Birth Class V_p^0 or *early*, and a Top-Band Attainment Point Class *middle* or *late*
- a variant with a Point Of Schema Birth Class *middle*, a Top-Band Attainment Point Class *late*.

Birth Class	Top-Band Point Class	Birth-To-Top Interval Class	Active Growth Months
V_p^0 or **early**	**middle** or **late**	**long** or **very long**	> 3
middle	**late**	**fair** or **long**	>3

Regularly curated projects have lives with frequent change of the schema. Much like the quantum steps pattern, they come with two basic variants, which share the same time characteristics with the quantum steps pattern, but just differ in the volume of change. Again, in the first variant, the regularly curated projects are born early with a fairly small percentage of the overall change at schema birth. Then, progressively, over their lives, the schema is consistently maintained and evolved. The top-band is typically reached (very) long after schema birth, towards the last quarter of the project's life with a fair number of steps and a typically short tail. In a second variant, the schema is born in the middle of the life of the project, and all the involved durations are shrunk accordingly.

The "Stairway to Heaven" family of patterns: both patterns, involve a fairly regular pattern of change, with change steps distributed across time. Although different in the change rate, both patterns refer to projects that do not reach the top band in a single shot, as with the previous family of patterns, and progressively climb to the top-band over a long period of time.

4.7 Siesta

The Siesta pattern includes projects whose schema is born early (even at the originating version of the project) and, typically, at a significant amount of its cumulative change. After birth, however, the schema is frozen for a long period of time, although quite later, towards the end of its life, the schema receives changes again.

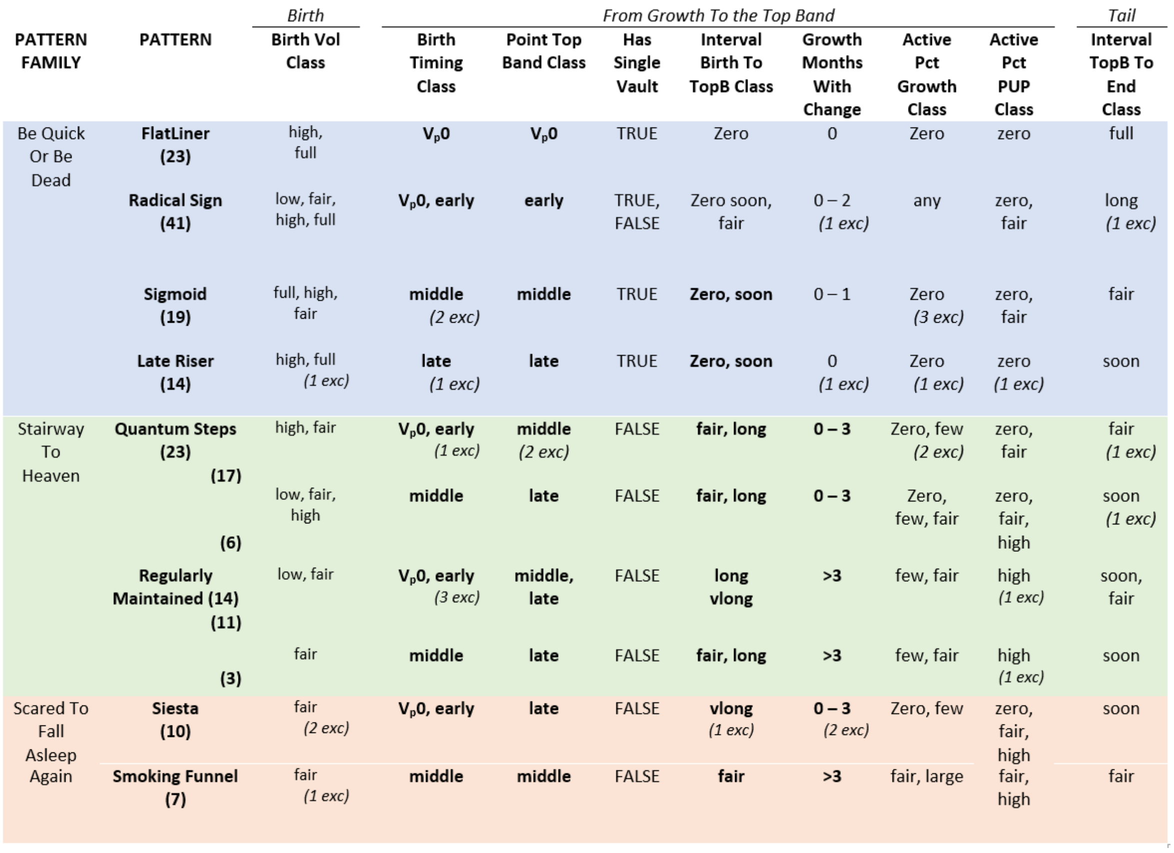
Definition 4.7. A **siesta** schema comes with a Point of Schema Birth Class V_p^0 or *early*, a Top-Band Attainment Point Class *late*, and a Schema Birth-To-Top Interval Class *very long*.

Birth Class	Top-Band Point Class	Birth-To-Top Interval Class	Active Growth Months
V_p^0 or **early**	**late**	**very long**	0-3

Typically, Siesta projects are born early, and although they remain in idleness for long periods, they eventually demonstrate change in the schema later in their life. As such, they come without a vault in their line and take more than 75% of the project's lifetime to reach the top band of cumulative schema activity.

4.8 Smoking Funnel

The Smoking Funnel pattern takes its name from the shape of the cumulative schema evolution line. The schemata of this pattern



		Birth	*From Growth To the Top Band*							*Tail*
PATTERN FAMILY	**PATTERN**	**Birth Vol Class**	**Birth Timing Class**	**Point Top Band Class**	**Has Single Vault**	**Interval Birth To TopB Class**	**Growth Months With Change**	**Active Pct Growth Class**	**Active Pct PUP Class**	**Interval TopB To End Class**
Be Quick Or Be Dead	**FlatLiner (23)**	high, full	**V_p0**	**V_p0**	TRUE	Zero	0	Zero	zero	full
	Radical Sign (41)	low, fair, high, full	**V_p0, early**	**early**	TRUE, FALSE	Zero soon, fair	0 – 2 *(1 exc)*	any	zero, fair	long *(1 exc)*
	Sigmoid (19)	full, high, fair	**middle** *(2 exc)*	**middle**	TRUE	**Zero, soon**	0 – 1	Zero *(3 exc)*	zero, fair	fair
	Late Riser (14)	high, full *(1 exc)*	**late** *(1 exc)*	**late**	TRUE	**Zero, soon**	0 *(1 exc)*	Zero *(1 exc)*	zero *(1 exc)*	soon
Stairway To Heaven	**Quantum Steps (23)** **(17)**	high, fair	**V_p0, early** *(1 exc)*	**middle** *(2 exc)*	FALSE	**fair, long**	**0 – 3**	Zero, few *(2 exc)*	zero, fair	fair *(1 exc)*
	(6)	low, fair, high	**middle**	**late**	FALSE	**fair, long**	**0 – 3**	Zero, few, fair	zero, fair, high	soon *(1 exc)*
	Regularly Maintained (14) **(11)**	low, fair	**V_p0, early** *(3 exc)*	**middle, late**	FALSE	**long vlong**	**>3**	few, fair	high *(1 exc)*	soon, fair
	(3)	fair	**middle**	**late**	FALSE	**fair, long**	**>3**	few, fair	high *(1 exc)*	soon
Scared To Fall Asleep Again	**Siesta (10)**	fair *(2 exc)*	**V_p0, early**	**late**	FALSE	**vlong** *(1 exc)*	**0 – 3** *(2 exc)*	Zero, few	zero, fair, high	soon
	Smoking Funnel (7)	fair *(1 exc)*	**middle**	**middle**	FALSE	**fair**	**>3**	fair, large	fair, high	fair

Figure 4: An overview of the different characteristics of the time-related patterns of schema evolution

(a) are born in the middle of the life of the project, (b) jump to a medium level of total schema activity when born, and, (c) continue to produce a fairly dense rate of schema evolution after their birth.

Definition 4.8. A **smoking funnel** schema comes with a Point of Schema Birth Class *middle*, a Top-Band Attainment Point Class *middle*, and a Schema Birth-To-Top Interval Class *fair*.

Birth Class	Top-Band Point Class	Birth-To-Top Interval Class	Active Growth Months
middle	**middle**	**fair**	> 3

The Smoking Funnel schemata are born in the middle of the project's lifetime with a medium amount of the total schema evolution at birth. In contrast to the first family of patterns of super-focused change, where the top-band is reached immediately, Smoking Funnel comes without vaults, but just a fair amount of the total change at schema birth. The interval between schema birth and top band is typically fair and the tail contains change. In other words, once born, the schemata of Smoking Funnel projects are regularly evolved.

The "Scared to Fall Asleep Again" family of patterns: the aforementioned two patterns, although very different in their characteristics, resemble in that they include projects where the change is not focused in a single point, and happens towards the end of the lifetime of the project.

Pattern	#prjs	Exceptions	Overlaps
Flatliner	23	–	–
Radical Sign	41	–	–
Sigmoid	19	2	–
Late Riser	14	1	–
Quantum Steps	23	2	–
Regularly Curated	14	–	–
Smoking Funnel	7	–	–
Siesta	10	3	–

Table 2: Exceptions and Overlaps of the Definitions of Schema Evolution time-related patterns

5 VALIDATION OF THE PATTERNS

In this section, we demonstrate the justification of the introduced patterns via the answering of the following validation questions:

- *VQ1: Are these patterns genuine and reasonable? How can we guarantee that the separation is not artificial and a-posteriori fitted to the numbers?*
- *VQ2: Can we claim that the classification of projects into different patterns is producing patterns that are (a) internally cohesive and (b) pairwise disjoint?*
- *VQ3: How generalizable, i.e., how representative of the general behavior of projects, are the results?*
- *VQ4: Is the taxonomy produced complete? How possible is it that other behaviors do exist too?*

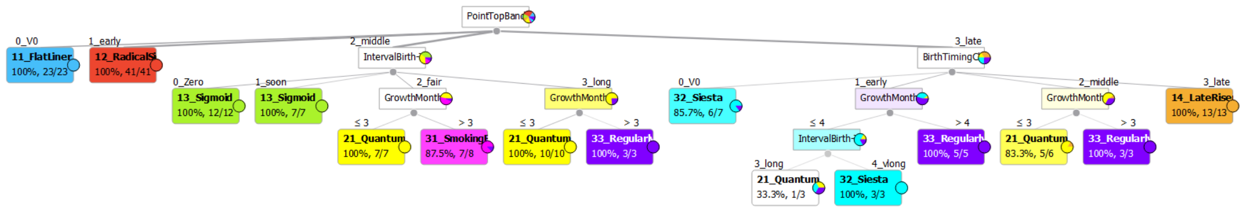


Figure 5: Schema evolution time-related patterns classified via a decision tree

5.1 Are these Patterns Genuine and Reasonable?

The main justification of the presented patterns lies in the fact that they were produced as a result of primarily manual analysis, and a-posteriori quantified justification. So, they were primarily judged as reasonable, meaningful patterns, by inspecting the charts of schema evolution of the collected projects, and only afterwards, validated by inspecting their measurable properties.

The protocol started with a manual classification of the available projects, via the visual inspection of the schema evolution cumulative progress – i.e., solely by observing the projects' charts like the ones appearing in Figure 3. Once the first grouping of charts had be completed, the question raised was: "is this reasonable and meaningful?" The following qualitative characterizations were made:

- The "*be quick or be dead*" family of few sharp changes: there were obviously too many projects with too little activity done in single "shot". The only internal separation of the patterns of the family was with respect to when the schema was born and, almost immediately, practically frozen.
- The "*stairway to heaven*" family of regular changes: another non-trivial fraction of the projects demonstrated progressive evolution characteristics at the beginning and middle life of the schema, and possibly at the late phases too – with variations, however, at the rate of change time-points.
- The "*scared to fall asleep again*" family: another small part of the corpus demonstrated signs of late change (patterns differ at the existence of midlife change hiatus).

Although not completely expected, compared to what previous research results would report, the results of the manual grouping are meaningful and reasonable. Despite the fact that previous results had mostly focused to the *volume* of activity without taking the timing so much into consideration, what we do know from the related literature (see Section 2, mainly [31], [37], [44]) is that quite frequently (although not exclusively) schema evolution is significantly reduced, or frozen, after some time. Therefore, the family of few sharp changes was quite expected. The family of moderate, regular change was not a surprise either with respect to the state of the art knowledge, which ascertained the existence of projects with moderate and high volume of change ([36], [10],[24], [50], [31]). The presence of late-oriented change was a surprise, however, given the state of the art and anecdotal evidence: although we knew there are projects that produce schema change at later periods of their life, the time-related patterns of change of this family were never discussed in the literature or otherwise.

As a side note, we must say that the aversion to change is not omnipresent in the literature. A major reason is that studies with just a handful of schemata ([36], [10],[24], [50], [31], [37]) had to pick schemata that do demonstrate some change, otherwise there is not much to report. On the other hand, studies like [42], [43] that work with large numbers of schemata, or studies like [34] working with necessity sampling of projects, clearly show the aversion to change. Interestingly, the study of nosql schemata in [34] reveals time behaviors that are visually quite close to the patterns reported here (hinting to a potential universality of the patterns that remains to be verified).

It is also obligatory to report that the original process produced some line chart patterns that were subsequently re-arranged into more reasonable families. Specifically:

- The Radical Sign pattern is the union of two sub-groups of the original process. Practically, we merged a small subgroup of 6 schemata born at the originating version of the project (with slightly different shapes), with the subgroup of schemata being born early, but not on V_p^0. The merge is reasonable in the sense that the differences are minor.
- The Quantum Steps and the Regularly Curated patterns are agglomerations of smaller line patterns that merged into two cohesive groups, which are also pairwise disjoint in terms of the volume of time-points of change. The formal condition on the volume of time-points of change, which distinguishes the two patterns, came later, and as a result, reclassified 7 projects. We believe it is obligatory to report this fact as part of the experimental protocol; however, we also believe that the rearrangement is a-posteriori justified.

Moreover, we can also observe several desirable properties in our pattern set. First, few exceptions do exist (see the following subsection). The result of the pattern extraction process was neither perfectly fit to the data, nor with colossal deviations. Second, the patterns are disjoint and cover significantly (not fully) the space of possible behaviors (see the subsection on disjointedness and completeness). Moreover, each pattern came with a reasonable size, neither overly dominating the rest, or being insignificant.

Overall, based on all the above, the most fundamental guard against statistics-oriented patterns came from the process itself: all the statistics-related tasks, like the quantization of the time-related attributes, the pairwise comparison of patterns or the inspection of other attributes were performed only once the bottom-up manual inspection and classification had been completed.

The pattern definitions came also after the manual classification was performed, and had a very small effect, just for the Regularly Curated family. Wherever a project seemed more related to a pattern despite the violation of the definition, the project remained in the pattern to which it was originally assigned. This is why exceptions also do exist, as we will demonstrate in the sequel. So, basically, the subsequent statistical analysis is testing the hypothesis: "did the manual grouping produce meaningful groups?"

5.2 Pattern Cohesion

Pattern cohesion refers to the internal homogeneity of the projects that pertain to each pattern. We encourage the reader to inspect the detailed material at the accompanying web site of the paper for their visual homogeneity. Table 2 presents the exceptions to the patterns with respect to their definition for our 151 projects. The 7 we found exceptions are too few and not particularly significant to question the cohesion of the patterns:

- Two projects classified as Sigmoid violate the "middle-born" part of the definition by being born early.
- A Late Riser project reaches the top band in middle life, violating the requirement of late attainment of top-band.
- Siesta has 2 projects exceeding the 0-3 months growth activity in the end, and another project that reaches growth just 'long' after schema birth (and not 'very long').
- A Quantum Step project reaches top late rather than middle.

We also quantized each project's time series to a vector of 20 measurements, one per interval of 5% of time (i.e., at 0%, 5%, . . . of time), and computed the centroid for each pattern. We measured the Mean Distance to Centroid for each pattern. The MDC ranges from 0.06 to 1.25, which is reasonable for vectors of 20 measurements in $[0 \dots 1]$ (see the accompanying web site of the paper).

5.3 Disjointedness

Disjointedness requires that the different patterns are essentially different with one another. It is straightforward to verify that our classification scheme attains disjointedness, as the formal definitions cover disjoint areas in the space produced by the Cartesian Product of values for the defining attributes. However, apart from formally, the pattern set is also hiding an inherent, essential disjointedness. The essential disjointedness is depicted in Figure 6 reporting the placement of patterns in the active domain space formed by the Cartesian Product of the domains of the involved class-based metrics. We report only the active domain that contains only the combinations which are populated by projects, and, for each point of this multidimensional space, we report the number of projects that pertain to it. So, for each part of the active domain, we show how many projects "live" there, and to which pattern they belong. With the exception of (a) a couple of Siesta projects being born early that is overlapping with Regularly Curated projects of similar definition, and, (b) the Quantum Steps and Regularly Curated patterns that span a large area of the domain space of values (understandably, as they are produced by the union of similar sub-families) which they would share had they not being discriminated by change rate, the rest of the patterns are focused in a specific area of the domain space and disjoint from the others. Quantum Steps and Regularly Curated are still disjoint; we just point out that they are the only patterns practically separated by change rate in their growth period.

As a side-effect of the validation of pattern disjointedness, we have extracted a simple decision tree from the labeled values of their properties, *after the manual annotation had been completed* (Figure 5). The simple classification tree shows that the patterns can be fairly well (although not 100%) separated automatically, with only 4 out of 151 projects that would have been erroneously classified under this classification scheme.

5.4 Threats to Validity, Generalization and Completeness

The main factors that affect the validity of our findings are factors concerning the validity of the data set per se. As our data set is a targeted subset of [42], several threats to validity are common. We do refer the interested reader to [42] for extensive comments on the validity and constraint ourselves to a concise treatment of the most important facts, for the autonomy of the paper.

Scope. The scope of the paper concerns the evolution in time of the logical level of relational schemata in meaningful Free Open-Source Software projects, hosted in GitHub. We are not covering proprietary schemata outside the FoSS domain. We do not cover conceptual or physical schemata. We are also restricted in relational schemata and not XML, JSON, or another format.

Generalization and External Validity. The external validity of a study concerns the extent to which its findings are generalizable to the general population. Here, we work with the corpus of projects coming from [42] and [45], and, as mentioned in Section 3, we have established that it is representative enough of the *entire* population described in the aforementioned scope.

A second threat to the external validity of our findings has to do with the process of this paper. We believe that our findings are representative of how schemata evolve in time, based on the following observations. First, we have based our findings on a substantial corpus of 151 significant projects with more than 12 months duration (which means ample time for the patterns to appear). Second, the patterns produced are cohesive, disjoint and not overfitted to the data set used.

Experimental validity. We have checked our change detection and chart-generating software via tests and code reviews; hence, we are fairly confident on its correctness, and thus, trust its output. We have also rechecked our manual allocation of projects to patterns, as this was an iterative process.

5.5 Completeness

We cannot exclude the possibility that other patterns of change do exist, although the overall situation hints that this is rather unexpected. First, a large part of the space of possible value combinations is already covered. Second, although there do exist value combinations missing from Figure 6, several of them are unattainable – e.g., a project with late schema birth, is obligatorily restricted to have a late top-band attainment and a short tail.

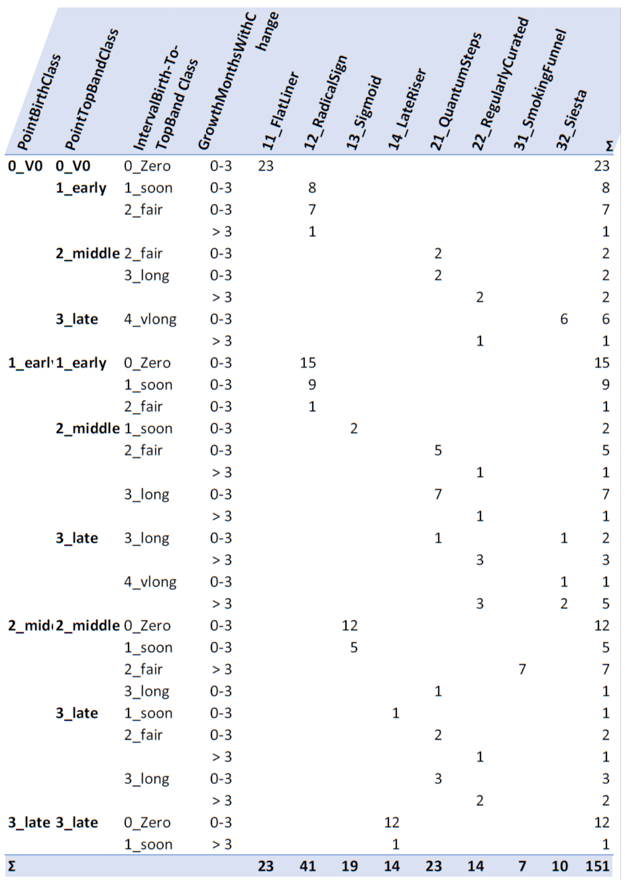
6 RELATIONSHIP TO OTHER MEASURES OF EVOLUTION

In this section, we discuss how different metrics of activity are related to the time-related patterns that we introduce.

6.1 Relationship to Activity-Related Evolution Measures

Interestingly, the time-related patterns are quite orthogonal to most of the fundamental measures of schema evolution activity. Most of the patterns, with the notable exception of *Smoking Funnel* and *Regularly Curated*, demonstrate quite similar behavior.

Concerning *Total Schema Activity*, i.e., the total amount of schema change that eventually took place in the life of the project after schema birth, it is rather indifferent to the patterns. Because



PointBirthClass	PointTopBandClass	IntervalBirth-To-TopBand Class	GrowthMonthsWithChange	11_FlatLiner	12_RadicalSign	13_Sigmoid	14_LateRiser	21_QuantumSteps	22_RegularlyCurated	31_SmokingFunnel	32_Siesta	Σ
0_V0	0_V0	0_Zero	0-3	23								23
	1_early	1_soon	0-3		8							8
		2_fair	0-3		7							7
			> 3		1							1
	2_middle	2_fair	0-3					2				2
		3_long	0-3					2				2
			> 3						2			2
	3_late	4_vlong	0-3								6	6
			> 3						1			1
1_earl	1_early	0_Zero	0-3		15							15
		1_soon	0-3		9							9
		2_fair	0-3		1							1
	2_middle	1_soon	0-3			2						2
		2_fair	0-3					5				5
			> 3						1			1
		3_long	0-3					7				7
			> 3						1			1
	3_late	3_long	0-3					1			1	2
			> 3						3			3
		4_vlong	0-3								1	1
			> 3						3		2	5
2_mid	2_middle	0_Zero	0-3			12						12
		1_soon	0-3			5						5
		2_fair	> 3							7		7
		3_long	0-3					1				1
	3_late	1_soon	0-3				1					1
		2_fair	0-3					2				2
			> 3						1			1
		3_long	0-3					3				3
			> 3						2			2
3_late	3_late	0_Zero	0-3				12					12
		1_soon	> 3				1					1
Σ				23	41	19	14	23	14	7	10	151

Figure 6: Coverage of the space of possible values by the time-related patterns of schema evolution

of the typically low values that schema evolution reaches, the values of change are clustered along small numbers. The only exceptions are Smoking Funnel and Regular Curation, who escape the tight overlap of the other patterns. There is a progressive shift from (a) the *be quick or be dead* patterns (Radical Sign with a median at 13 attributes and the rest less than 3), to (b) Siesta and Quantum steps with somewhat higher medians (17 and 22 attributes changed), and, (c) to the Smoking Funnel and Regular Curation which come with orders-of-magnitude higher values (medians of 189 and 250 changed attributes, respectively). The subdivision of *Total Schema Activity* to *Total Schema Expansion* and *Total Schema Maintenance* follows the same behavior.

Why do we see this behavior? It is important to highlight that the behavior towards schema evolution is not obligatorily in sync with the behavior towards source code evolution. For example, even in really frozen patterns, like, e.g., the populous Radical Sign, we can see vast ranges of PUP values. Similarly, for the schema size at birth. Concerning the differentiation of Smoking Funnel and Regularly Curated projects (the two smallest patterns in terms of population), it is important to highlight that these two include the most active projects in terms of total amount of change.

Thus, *Smoking Funnel and Regularly Curated projects start bigger and contain higher amounts of schema evolution activity. One could argue that this combination of attributes quantitatively discriminates these two groups of projects from the others.*

6.2 Relationship to the Point of Schema Birth

Predicting how a schema will evolve given some original evidence is an extremely difficult problem. To the best of our knowledge, there is no attempt to the problem so far for two reasons. First, to a very large extent, the evolution depends heavily on the particularities of the development and curation team, along with the idiosyncrasies of the project itself. Thus, predictions would require a very detailed charting of both project and anthropocentric characteristics. Second, one needs a really large corpus of schema histories to come up with some broad approximation from a statistical point of view.

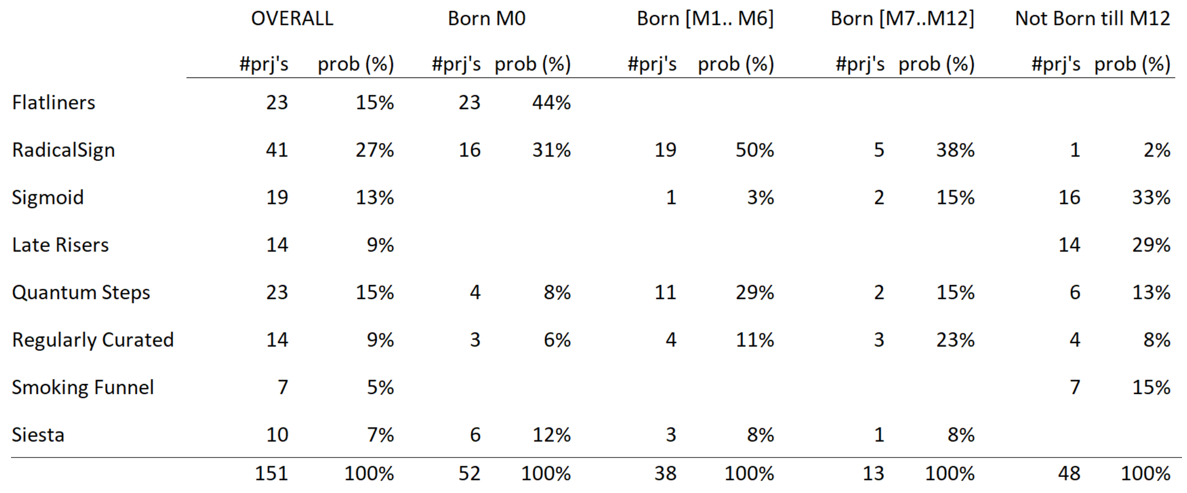
We make such a preliminary attempt in this section, to give a broad overview of how the point of schema birth and the future behavior of a project in terms of its pattern of schema evolution relate. Figure 7 depicts (a) the probabilities overall, and, (b) depending on the month of schema birth (as an absolute value). The rationale behind Figure 7 is: "Assume a curator, or an external assessor, who extracts (e.g., via git log and out tool set) the history of changes of a software project and its relational database. Can the curator make an educated guess on the future of how the schema will evolve?" Based on the statistics from our 151 projects, we argue that the point of schema birth gives some interesting estimations.

- Assuming that the schema is born during the first month of the project life, the probability that the schema will be completely frozen is 75% (!), as a flatliner or a radical sign. A 14% probability remains for steady regular curation via the Quantum Steps or Regular Curation patterns.
- Assuming the schema is born in the next six months, there is a 53% chance that the schema will follow a sharp, focused evolution, and a 40% chance it will follow a regularly curated life.
- The next six months demonstrate a drop in the number of projects born (13), albeit with pretty much the same distribution of probabilities for the families. The only significant change is that the probability of a more active regular curation rises to 23%.
- As the point of schema birth exceeds the first year, there is a 64% chance of a sharp focused change, 21% for a regular maintenance, and a 15% for a smoking funnel behavior.

As a side observation, Figure 7 also gives some results to the question: "when are schemata born, in the life of a project?". The data indicate (i) a probability of 34% (52/151) for a schema to be born in M0, (ii) a 60% probability (90/151) for the schema to be born in the first 6 months of the project (including M0), (iii) *a 68% (103/151) probability for the schema to be born in the first year of the project's life, and, (iv) quite surprisingly, a 31% probability of a project having its schema being born after the first year of the project's life.*

6.3 Relationship to mixture of data type change

We have studied how patterns relate to the internal breakdown of change, in terms of expansion (attribute birth with new tables, or injection to existing ones) and maintenance (attribute deletion, data type or PK change). With the exception of few Radical Sign projects that are oriented towards maintenance, the "Be Quick or Be Dead" family involves small change, frequently being zero, and an inclination towards expansion. Due to the small volume of change, the patterns of the family are frequently monothematic in their internal breakdown of change types. In contrast, the rest of



	OVERALL		Born M0		Born [M1.. M6]		Born [M7..M12]		Not Born till M12	
	#prj's	prob (%)	#prj's	prob (%)	#prj's	prob (%)	#prj's	prob (%)	#prj's	prob (%)
Flatliners	23	15%	23	44%						
RadicalSign	41	27%	16	31%	19	50%	5	38%	1	2%
Sigmoid	19	13%			1	3%	2	15%	16	33%
Late Risers	14	9%							14	29%
Quantum Steps	23	15%	4	8%	11	29%	2	15%	6	13%
Regularly Curated	14	9%	3	6%	4	11%	3	23%	4	8%
Smoking Funnel	7	5%							7	15%
Siesta	10	7%	6	12%	3	8%	1	8%		
	151	100%	52	100%	38	100%	13	100%	48	100%

Figure 7: Probability of a schema following a certain time-related pattern, given its point of birth

the time-related patterns come with higher volumes of changes, which is also related to a variety of change types. Both expansion and maintenance are performed with the granule of change being mostly the entire table (being inserted or deleted), rather than the internal restructuring of existing tables. A detailed study is included in the accompanying material of the paper.

7 CONCLUSIONS AND FUTURE WORK

The core contribution of this paper is the provision of a series of patterns and pattern families on how developers and data curators regulate schema evolution over time. We work with a large corpus of 151 schema histories, and introduce 3 families of 8 patterns of schema change: the *Be Quick or Be Dead* family of sharp, focused change, the *Stairway to Heaven* family of steps of schema change, and the *Scared to Fall Asleep Again* family with schema change late in the project's life.

A major result, mainly involving the Be Quick or Be Dead family is that *the "freeze-and-build" aversion to change is indeed present at large and concerns 2/3 of the corpus*. The result is significant, because, now that we know that schemata evolve over time mostly rarely and sharply, we can act in several ways.

- Research-wise, the research community can instigate research efforts to facilitate a more active schema life for databases. A major implication of the aversion-to-change finding is that we need to reflect on our schema design and software development premises. A clear problem here is the loose coupling between software applications and the underlying schema structure of their database. Schema evolution, apart from the operational costs of data migration, breaks the mapping to the surrounding code, thus incurring significant costs. Therefore, as researchers, we should fundamentally rethink how schemata and applications are coupled in ways that allow smooth adaptation of both to change.
- Moreover, software development teams can organize how they chart schema-to-source mappings, in order to lessen the effects of schema change to the code, and be able to facilitate schema change better than what they do now.

At the same time, *the rest of the projects evolve differently in time, with regular (rare or dense) change; and, surprisingly, late change too*. Thus, we prove that we cannot rely solely on the premise of aversion to change. Section 6.2 gives *insights on how likely it is to encounter change or rigidity in the life of a schema.*

- Project managers can exploit the statistics of Figure 7 upfront, to reserve time for handling change and its impact. Hence, projects managers can prepare in advance on how they plan / expect / drive schema change.

Finally, this paper concerns a type of result that is (sadly) atypical for our data engineering community: the enrichment of our knowledge –in a principled manner– on how the artifacts that we invent (here: relational databases) are actually used in practice.

- *Part of our contribution is the topic itself, along with the nomenclature, measures and methodological principles used.*
- A clear implication, thus, is that the paper opens a road for future research – e.g., schema evolution of non-traditional data (e.g., NoSQL), or in non-FOSS projects.

Concerning possible roads for the future, we can envision several possibilities.

- First, as already mentioned, Nosql schemata are a clear case where this method can be applied, esp., since there are obvious similarities with the few results that currently exist [34]. This becomes more interesting if we consider the anecdotal claim that nosql schemata are more “alive" in terms of evolutionary activity. The entire issue remains open to investigation.
- Another unsolved research problem is the provision of solid foundations for the prediction of future behavior on the basis of a meaningful model. This is a fairly important result to be pursued, as it can allow the deduction of laws on how schemata evolve.
- An even harder road to follow is to obtain and study schema histories from proprietary schemata. For the last 50 years of the database discipline, this has been practically impossible, limiting our attempts to the study of FOSS schemata, as soon as public repositories allowed us to do so, in the last decade.
- Finally, developing educational material and practical exercises for our students, in order to train them on the practical aspects of the topic is another important road for the future.

REFERENCES

[1] Steve Adolph, Wendy Hall, and Philippe Kruchten. 2011. Using grounded theory to study the experience of software development. *Empir. Softw. Eng.* 16, 4 (2011), 487–513. https://doi.org/10.1007/S10664-010-9152-6

[2] Tobias Bleifuß, Leon Bornemann, Dmitri V. Kalashnikov, Felix Naumann, and Divesh Srivastava. 2021. The Secret Life of Wikipedia Tables. In *Proceedings of the 2nd Workshop on Search, Exploration, and Analysis in Heterogeneous Datastores (SEA-Data 2021) co-located with 47th International Conference on Very Large Data Bases (VLDB 2021), Copenhagen, Denmark, August 20, 2021 (CEUR Workshop Proceedings, Vol. 2929)*. CEUR-WS.org, 20–26.

[3] Angela Bonifati, Peter Furniss, Alastair Green, Russ Harmer, Eugenia Oshurko, and Hannes Voigt. 2019. Schema Validation and Evolution for Graph Databases. In *38th International Conference on Conceptual Modeling, ER 2019, Salvador, Brazil, November 4-7, 2019, Proceedings*. Springer, 448–456.

[4] Dimitri Braininger, Wolfgang Mauerer, and Stefanie Scherzinger. 2020. Replicability and Reproducibility of a Schema Evolution Study in Embedded Databases. In *ER 2020 Workshops CMAI, CMLS, CMOMM4FAIR, CoMoNoS, EmpER (Lecture Notes in Computer Science, Vol. 12584)*. Springer, Vienna, Austria, 210–219.

[5] Loredana Caruccio, Giuseppe Polese, and Genoveffa Tortora. 2016. Synchronization of Queries and Views Upon Schema Evolutions: A Survey. *ACM Trans. Database Syst.* 41, 2 (2016), 9:1–9:41.

[6] Kathy Charmaz. 2006. *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis.* Sage, Thousand Oaks.

[7] Christina Christodoulakis, Eric B. Munson, Moshe Gabel, Angela Demke Brown, and Renée J. Miller. 2020. Pytheas: Pattern-based Table Discovery in CSV Files. *Proc. VLDB Endow.* 13, 11 (2020), 2075–2089.

[8] Anthony Cleve, Maxime Gobert, Loup Meurice, Jerome Maes, and Jens H. Weber. 2015. Understanding database schema evolution: A case study. *Sci. Comput. Program.* 97 (2015), 113–121.

[9] Juliet M. Corbin and Anselm Strauss. 1990. *Basics of qualitative research: Grounded theory procedures and techniques.* Sage, Thousand Oaks, CA, US.

[10] Carlo Curino, Hyun Jin Moon, Letizia Tanca, and Carlo Zaniolo. 2008. Schema Evolution in Wikipedia - Toward a Web Information System Benchmark. In *Proceedings of the Tenth International Conference on Enterprise Information Systems, ICEIS 2008, June 12-16, 2008*. Volume DISI, Barcelona, Spain, 323–332.

[11] Julien Delplanque, Anne Etien, Nicolas Anquetil, and Olivier Auverlot. 2018. Relational Database Schema Evolution: An Industrial Case Study. In *2018 IEEE International Conference on Software Maintenance and Evolution, ICSME 2018, September 23-29, 2018*. IEEE Computer Society, Madrid, Spain, 635–644.

[12] Konstantinos Dimolikas, Apostolos V. Zarras, and Panos Vassiliadis. 2020. A Study on the Effect of a Table's Involvement in Foreign Keys to its Schema Evolution. In *39th International Conference on Conceptual Modeling, ER 2020, November 3-6, 2020 (Lecture Notes in Computer Science, Vol. 12400)*. Springer, Vienna, Austria, 456–470.

[13] Spyridon K. Gardikiotis and Nicos Malevris. 2009. A two-folded impact analysis of schema changes on database applications. *Int. J. Autom. Comput.* 6, 2 (2009), 109–123.

[14] Barney Glaser and Anselm Strauss. 1967. *The Discovery of Grounded Theory: Strategies for Qualitative Research.* Aldine, Chicago Illinois.

[15] Mathieu Goeminne, Alexandre Decan, and Tom Mens. 2014. Co-evolving code-related and database-related changes in a data-intensive software system. In *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering, CSMR-WCRE 2014, February 3-6, 2014*. IEEE Computer Society, Antwerp, Belgium, 353–357.

[16] Michael Hartung, James F. Terwilliger, and Erhard Rahm. 2011. Recent Advances in Schema and Ontology Evolution. In *Schema Matching and Mapping*, Zohra Bellahsene, Angela Bonifati, and Erhard Rahm (Eds.). Springer, 149–190.

[17] Kai Herrmann, Hannes Voigt, Torben Bach Pedersen, and Wolfgang Lehner. 2018. Multi-schema-version data management: data independence in the twenty-first century. *VLDB J.* 27, 4 (2018), 547–571. https://doi.org/10.1007/s00778-018-0508-7

[18] Kai Herrmann, Hannes Voigt, Jonas Rausch, Andreas Behrend, and Wolfgang Lehner. 2018. Robust and simple database evolution. *Inf. Syst. Frontiers* 20, 1 (2018), 45–61.

[19] Abdulrahman Kaitoua, Tilmann Rabl, Asterios Katsifodimos, and Volker Markl. 2019. Muses: Distributed Data Migration System for Polystores. In *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*. 1602–1605.

[20] Barbara A. Kitchenham, Lech Madeyski, David Budgen, Jacky Keung, Pearl Brereton, Stuart M. Charters, Shirley Gibbs, and Amnart Pohthong. 2017. Robust Statistical Methods for Empirical Software Engineering. *Empir. Softw. Eng.* 22, 2 (2017), 579–630.

[21] Meike Klettke, Hannes Awolin, Uta Störl, Daniel Müller, and Stefanie Scherzinger. 2017. Uncovering the evolution history of data lakes. In *IEEE International Conference on Big Data, BigData 2017, December 11-14, 2017*. IEEE Computer Society, Boston,A, USA,, 2462–2471.

[22] Meike Klettke, Uta Störl, Manuel Shenavai, and Stefanie Scherzinger. 2016. NoSQL schema evolution and big data migration at scale. In *2016 IEEE International Conference on Big Data (IEEE BigData 2016), Washington DC, USA, December 5-8, 2016*. 2764–2774.

[23] Thomas A. Limoncelli. 2019. SQL is no excuse to avoid DevOps. *Commun. ACM* 62, 1 (2019), 46–49. https://doi.org/10.1145/3287299

[24] Dien-Yen Lin and Iulian Neamtiu. 2009. Collateral evolution of applications and databases. In *Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops, August 24-28, 2009*. ACM, Amsterdam, Netherlands, 31–40.

[25] Petros Manousis, Panos Vassiliadis, and George Papastefanatos. 2013. Automating the Adaptation of Evolving Data-Intensive Ecosystems. In *Proceedings of 32th International Conference on Conceptual Modeling (ER 2013), Hong-Kong, China, November 11-13, 2013*. 182–196.

[26] Petros Manousis, Panos Vassiliadis, and George Papastefanatos. 2015. Impact Analysis and Policy-Conforming Rewriting of Evolving Data-Intensive Ecosystems. *Journal of Data Semantics* 4, 4 (2015), 231–267. https://doi.org/10.1007/S13740-015-0050-3

[27] Petros Manousis, Panos Vassiliadis, Apostolos V. Zarras, and George Papastefanatos. 2015. Schema Evolution for Databases and Data Warehouses. In *5th European Summer School on Business Intelligence, eBISS 2015, July 5-10, 2015, Tutorial Lectures (Lecture Notes in Business Information Processing, Vol. 253)*. Springer, Barcelona, Spain, 1–31.

[28] Andy Maule, Wolfgang Emmerich, and David S. Rosenblum. 2008. Impact analysis of database schema changes. In *30th International Conference on Software Engineering (ICSE 2008), May 10-18, 2008*, Wilhelm Schäfer, Matthew B. Dwyer, and Volker Gruhn (Eds.). ACM, Leipzig, Germany, 451–460.

[29] George Papastefanatos, Panos Vassiliadis, Alkis Simitsis, and Yannis Vassiliou. 2010. HECATAEUS: Regulating schema evolution. In *ICDE*. IEEE, Long Beach, California, USA, 1181–1184.

[30] Amandalynne Paullada, Inioluwa Deborah Raji, Emily M. Bender, Emily Denton, and Alex Hanna. 2021. Data and its (dis)contents: A survey of dataset development and use in machine learning research. *Patterns* 2, 11 (2021), 100336.

[31] Dong Qiu, Bixin Li, and Zhendong Su. 2013. An empirical analysis of the co-evolution of schema and code in database applications. In *Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE'13, August 18-26, 2013*. ACM, Saint Petersburg, Russian Federation, 125–135.

[32] Mohammad Raza and Sumit Gulwani. 2020. Web Data Extraction using Hybrid Program Synthesis: A Combination of Top-down and Bottom-up Inference. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, June 14-19, 2020*. ACM, online conference, 1967–1978.

[33] Stefanie Scherzinger, Wolfgang Mauerer, and Haridimos Kondylakis. 2021. DeBinelle: Semantic Patches for Coupled Database-Application Evolution. In *37th IEEE International Conference on Data Engineering, ICDE 2021*. IEEE, Chania, Greece, 2697–2700.

[34] Stefanie Scherzinger and Sebastian Sidortschuck. 2020. An Empirical Study on the Design and Evolution of NoSQL Database Schemas. In *39th International Conference on Conceptual Modeling, ER 2020, November 3-6, 2020, Proceedings (Lecture Notes in Computer Science, Vol. 12400)*. Springer, Vienna, Austria, 441–455.

[35] Robert E. Schuler, Jitin Singla, Brinda Vallat, Kate L. White, Helen M. Berman, and Carl Kesselman. 2023. Database Evolution, by Scientists, for Scientists: A Case Study. In *19th IEEE International Conference on e-Science, e-Science 2023, Limassol, Cyprus, October 9-13, 2023*. 1–10.

[36] D. Sjøberg. 1993. Quantifying schema evolution. *Information and Software Technology* 35, 1 (1993), 35–44.

[37] Ioannis Skoulis, Panos Vassiliadis, and Apostolos V. Zarras. 2015. Growing up with stability: How open-source relational databases evolve. *Information Systems* 53 (2015), 363–385.

[38] Divesh Srivastava, Tobias Bleifuß, Leon Bornemann, Dmitri V. Kalashnikov, and Felix Naumann. 2022. Exploring and Analyzing Change: The Janus Project. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, Mohammad Al Hasan and Li Xiong (Eds.). ACM, 3.

[39] Michael Stonebraker, Raul Castro Fernandez, Dong Deng, and Michael L. Brodie. 2017. Database Decay and What To Do About It. *Commun. ACM* 60, 1 (2017), 11. https://doi.org/10.1145/3014349

[40] Uta Störl, Meike Klettke, and Stefanie Scherzinger. 2020. NoSQL Schema Evolution and Data Migration: State-of-the-Art and Opportunities. In *23rd International Conference on Extending Database Technology, EDBT 2020, March 30 - April 02, 2020*. OpenProceedings.org, Copenhagen, Denmark, 655–658.

[41] Uta Störl, Alexander Tekleab, Meike Klettke, and Stefanie Scherzinger. 2018. In for a Surprise When Migrating NoSQL Data. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*. IEEE Computer Society, 1662.

[42] Panos Vassiliadis. 2021. Profiles of Schema Evolution in Free Open Source Software Projects. In *37th IEEE International Conference on Data Engineering, ICDE 2021,April 19-22, 2021*. IEEE, Chania, Greece, 1–12.

[43] Panos Vassiliadis and George Kalampokis. 2022. Taxa and super taxa of schema evolution and their relationship to activity, heartbeat and duration. *Inf. Syst.* 110 (2022), 102109. https://doi.org/10.1016/j.is.2022.102109

[44] Panos Vassiliadis, Michail-Romanos Kolozoff, Maria Zerva, and Apostolos V. Zarras. 2019. Schema evolution and foreign keys: a study on usage, heartbeat of change and relationship of foreign keys to table activity. *Computing* 101, 10 (2019), 1431–1456.

[45] Panos Vassiliadis, Fation Shehaj, George Kalampokis, and Apostolos V. Zarras. 2023. Joint Source and Schema Evolution: Insights from a Study of 195 FOSS Projects. In *26th International Conference on Extending Database Technology, EDBT 2023, March 28-31, 2023*. OpenProceedings.org, Ioannina, Greece, 27–39.

[46] Panos Vassiliadis and Apostolos V. Zarras. 2017. Schema Evolution Survival Guide for Tables: Avoid Rigid Childhood and You're En Route to a Quiet Life. *Journal of Data Semantics* 6, 4 (2017), 221–241.

[47] Panos Vassiliadis, Apostolos V. Zarras, and Ioannis Skoulis. 2017. Gravitating to rigidity: Patterns of schema evolution - and its absence - in the lives of tables. *Information Systems* 63 (2017), 24–46.

[48] Enzo Veltri, Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. 2023. Data Ambiguity Profiling for the Generation of Training Examples. In *39th IEEE International Conference on Data Engineering, ICDE 2023, April 3-7, 2023*. IEEE, Anaheim, CA, USA, 450–463.

[49] Gerardo Vitagliano, Mazhar Hameed, Lan Jiang, Lucas Reisener, Eugene Wu, and Felix Naumann. 2023. Pollock: A Data Loading Benchmark. *Proc. VLDB Endow.* 16, 8 (2023), 1870–1882.

[50] Shengfeng Wu and Iulian Neamtiu. 2011. Schema evolution analysis for embedded databases. In *Workshops Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011*. IEEE Computer Society, Hannover, Germany, 151–156.