Open-Source Databases: Within, Outside, or Beyond Lehman's Laws of Software Evolution?

Ioannis Skoulis, Panos Vassiliadis, and Apostolos Zarras

Dept. of Computer Science and Engineering
University of Ioannina (Hellas)
{iskoulis,pvassil,zarras}@cs.uoi.gr

Abstract. Lehman's laws of software evolution is a well-established set of observations (matured during the last forty years) on how the typical software systems evolve. However, the applicability of these laws on databases has not been studied so far. To this end, we have performed a thorough, large-scale study on the evolution of databases that are part of larger open source projects, publicly available through open source repositories, and report on the validity of the laws on the grounds of properties like size, growth, and amount of change per version.

Keywords: Schema evolution, software evolution, Lehman's laws.

1 Introduction

Software evolution is the change of a software system over time, typically performed via a remarkably difficult, complicated and time consuming process, software maintenance. In an attempt to understand the mechanics behind the evolution of software and facilitate a smoother, lest disruptive maintenance process, Meir Lehman and his colleagues introduced a set of rules in mid seventies [1], also known as the *Laws on Software Evolution* (Sec. 2). Their findings, that were reviewed and enhanced for nearly 40 years [2], [3], have, since then, given an insight to managers, software developers and researchers, as to *what* evolves in the lifetime of a software system, and *why* it does so. Other studies ([4], [5], [6] to name a few significant ones) have complemented these insights in this field, typically with particular focus to open-source software projects.

In sharp distinction to traditional software systems, database evolution has been hardly studied throughout the entire lifetime of the data management discipline. This deficit in our knowledge is disproportional to the severity of the implications of database evolution, and in particular, of database schema evolution. A change in the schema of a database may immediately drive surrounding applications to crash (in case of deletions or renamings) or be semantically defective or inaccurate (in the case of information addition, or restructuring). Overall, schema evolution threatens the syntactic and semantic validity of the surrounding applications and severely affects both developers and end-users. Given this

M. Jarke et al. (Eds.): CAiSE 2014, LNCS 8484, pp. 379–393, 2014.

importance, it is only amazing to find out that in the past 40 years of database research, only three(!) studies [7], [8] and [9] have attempted a first step towards understanding the mechanics of schema evolution. Those studies, however, focus on the statistical properties of the evolution and do not provide details on the actual events, or the mechanism that governs the evolution of database schemata.

In this paper, *we perform the first large-scale study of schema evolution in the related literature. Specifically, we study the evolution of the logical schema of eight databases, that are parts of publicly available, open-source software projects.* To achieve the above goal, we have collected, cleansed and processed the available versions of the database schemata for the eight case studies. Moreover, we have extracted the changes that have been performed in these versions and, finally, we have come up with the respective datasets that can serve as a foundation for future analysis by the research community (Sec. 3). Concerning the applicability of Lehman's laws to open-source databases, our results show that *the essence of Lehman's laws holds*: evolution is not about uncontrolled growth; on the contrary, there appears to be a stabilization mechanism that employs perfective maintenance to control the otherwise growing trend of increase in information capacity of the database (Sec. 4). Having said that, we also observe that the growth mechanisms and the change patterns are quite different between open source databases and typical software systems.

2 Lehman Laws of Software Evolution in a Nutshell

Meir M. Lehman and his colleagues, have introduced, and subsequently amended, enriched and corrected a set of rules on the behavior of software as it evolves over time [1], [2], [3]. Lehman's laws focus on *E-type systems*, that concern "software solving a problem or addressing an application in the real-world" [2]. The main idea behind the laws of evolution for E-type software systems is that their *evolution is a process that follows the behavior of a feedback-based system.* Being a feedback-based system, the evolution process has to balance (a) *positive feedback*, or else the need to adapt to a changing environment and grow to address the need for more functionality, and, (b) *negative feedback*, or else the need to control, constrain and direct change in ways that prevent the deterioration of the maintainability and manageability of the software. In the sequel we list the definitions of the laws as they are presented in [3], in a more abstract form than previous versions and with the benefit of retrospect, after thirty years of maturity and research findings.

(I) Law of Continuing Change. An E-type system must be continually adapted or else it becomes progressively less satisfactory in use.

(II) Law of Increasing Complexity. As an E-type system is changed its complexity increases and becomes more difficult to evolve unless work is done to maintain or reduce the complexity.

(III) Law of Self Regulation. Global E-type system evolution is feedback regulated.

(IV) Law of Conservation of Organisational Stability. The work rate of an organisation evolving an E-type software system tends to be constant over the operational lifetime of that system or phases of that lifetime.

(V) Law of Conservation of Familiarity. In general, the incremental growth (growth ratio trend) of E-type systems is constrained by the need to maintain familiarity.

(VI) Law of Continuing Growth. The functional capability of E-type systems must be continually enhanced to maintain user satisfaction over system lifetime.

(VII) Law of Declining Quality. Unless rigorously adapted and evolved to take into account changes in the operational environment, the quality of an E-type system will appear to be declining.

(VIII) Law of Feedback System. E-type evolution processes are multi-level, multi-loop, multi-agent feedback systems.

Before proceeding with our study, we present a first apodosis of the laws, taking into consideration both the wording of the laws, but most importantly their accompanying explanations [3].

An E-Type software system continuously changes over time (I) obeying a complex feedback-based evolution process (VIII). On the one hand, due to the need for growth and adaptation that acts as positive feedback, this process results in an increasing functional capacity of the system (VI), produced by a growth ratio that is slowly declining in the long term (V). The process is typically guided by a pattern of growth that demonstrates its self-regulating nature: growth advances smoothly; still, whenever there are excessive deviations from the typical, baseline rate of growth (either in a single release, or accumulated over time), the evolution process obeys the need for calibrating releases of perfective maintenance (expressed via minor growth and demonstrating negative feedback) to stop the unordered growth of the system's complexity (III). On the other hand, to regulate the ever-increasing growth, there is negative feedback in the system controlling both the overall quality of the system (VII), with particular emphasis to its internal quality (II). The effort consumed for the above process is typically constant over phases, with the phases disrupted with bursts of effort from time to time (IV).

3 Experimental Setup of the Study

Datasets. We have studied eight database schemata from open-source software projects. *ATLAS*[1] is a particle physics experiment at CERN, with the goal of learning about the basic forces that have shaped our universe – famously known for the attempt on the Higgs boson. *BioSQL*[2] is a generic relational model covering sequences, features, sequence and feature annotation, a reference taxonomy, and ontologies from various sources such as GenBank or Swissport. *Ensembl* is

[1] `http://atlas.web.cern.ch/Atlas/Collaboration/`

[2] `http://www.biosql.org/wiki/Main_Page`

a joint scientific project between the European Bioinformatics Institute (EBI)[3] and the Wellcome Trust Sanger Institute (WTSI)[4] which was launched in 1999 in response to the imminent completion of the Human Genome Project. The goal of Ensembl was to automatically annotate the three billion base pairs of sequences of the genome, integrate this annotation with other available biological data and make all this publicly available via the web. *MediaWiki*[5] was first introduced in early 2002 by the Wikimedia Foundation along with Wikipedia, and hosts Wikipedia's content since then. *Coppermine*[6] is a photo gallery web application. *OpenCart*[7] is an open source shopping cart system. *PhpBB*[8] is an Internet forum package. *TYPO3*[9] is a free and open source web content management framework.

Dataset Collection and Processing. A first collection of links to available datasets was made by the authors of [9], [10][10]; for this, these authors deserve honorable credit. We isolated eight databases that appeared to be alive and used (as already mentioned, some of them are actually quite prominent). For each dataset we gathered as many schema versions (DDL files) as we could from their public source code repositories (cvs, svn, git). We have targeted main development branches and trunks to maximize the validity of the gathered resources. *We are interested only on changes of the database part of the project as they are integrated in the trunk of the project.* Hence, we collected all the versions of the database, committed at the trunk or master branch, and ignored all other branches of the project.

We collected the files during June 2013. For all of the projects, we focused on their release for MySQL (except ATLAS Trigger, available only for Oracle). The files were then processed by sequential pairs from our tool, Hecate, that allows the detection of (a) changes at the attribute level, and specifically, attributes inserted, deleted, having a changed data type, or participation in a changed primary key, and (b) changes at the relation level, with relations inserted and deleted, in a fully automated way. Hecate, was then used to give us (a) the differences between two subsequent committed versions, and (b) the measures we needed to conduct this study – for example, the *size of the schema* (in number of tables and attributes), the total number of changes for each transition from a version to the next, which we also call *heartbeat*, or the *growth* assessed as the difference in the size of the schema between subsequent versions. *Hecate, along with all the data sets and our results are available at our group's public repository* https://github.com/DAINTINESS-Group.

[3] https://www.ebi.ac.uk/

[4] https://www.sanger.ac.uk/

[5] https://www.mediawiki.org/wiki/MediaWiki

[6] http://coppermine-gallery.net/

[7] http://www.opencart.com

[8] https://www.phpbb.com/

[9] http://typo3.org/

[10] http://data.schemaevolution.org

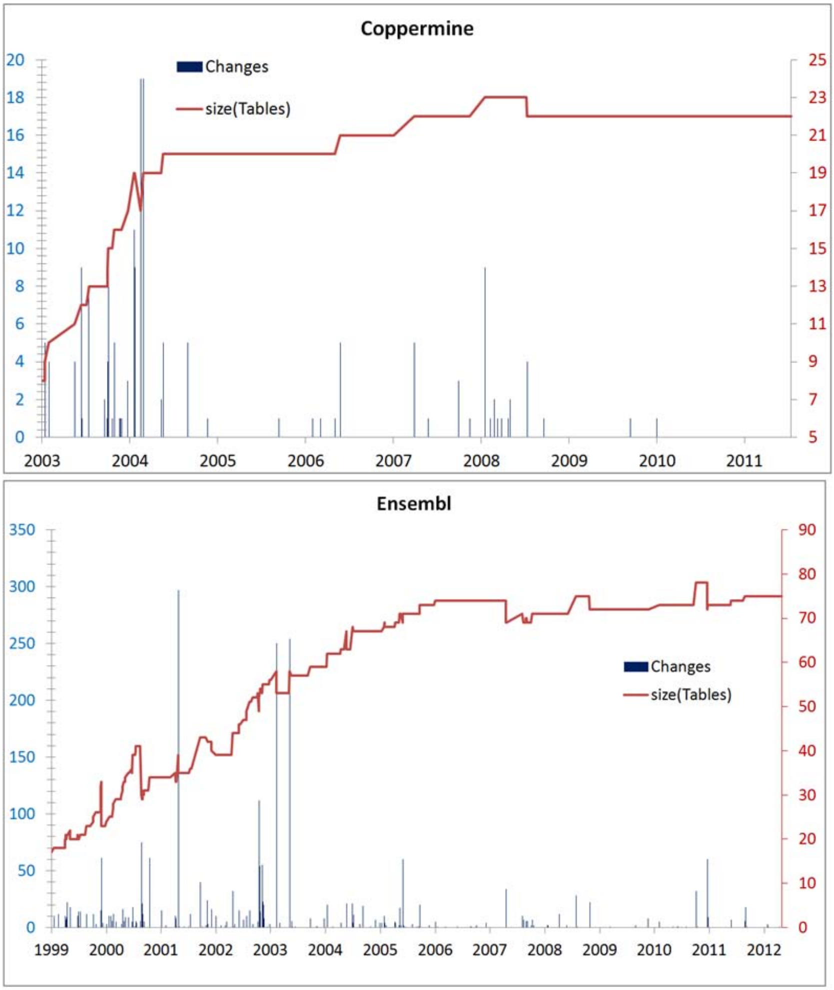

Fig. 1. Combined demonstration of heartbeat (number of changes per version) and schema size (no. of tables) for Coppermine and Ensemble. The left axis signifies the amount of change and the right axis the number of tables.

4 Assessing the Laws for Schema Evolution

The laws of software evolution where developed and reshaped over forty years. Explaining each law in isolation from the others is precarious as it risks losing the deeper essence and inter-dependencies of the laws [3]. To this end, in this section, we organize the laws in three thematic areas of the overall evolution management mechanism that they reveal. The first group of laws discusses the existence of a feedback mechanism that constrains the uncontrolled evolution of software. The second group discusses the properties of the growth of the system,

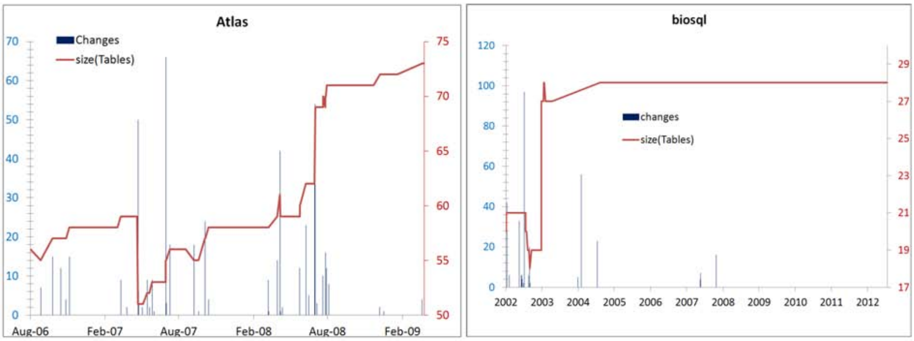

Fig. 2. Combined demonstration of heartbeat (number of changes per version) and schema size (no. of tables) for Atlas and BioSQL. The left axis signifies the amount of change and the right axis the number of tables.

i.e., the part of the evolution mechanism that accounts for positive feedback. The third group of laws discusses the properties of perfective maintenance that constrains the uncontrolled growth, i.e., the part of the evolution mechanism that accounts for negative feedback.

4.1 Is There a Feedback-Based System for Schema Evolution?

Law of Continuing Change (Law I). The main argument of the first law is that the schema continuously changes over time. To validate the hypothesis that the law of continuing change holds, we study the heartbeat of the schema's life (see Fig. 1 and 2 for a combined demonstration of heartbeat and schema size).

With the exception of BioSQL that appeared to be "sleeping" for some years and was later re-activated, in all other cases, we have changes (sometimes moderate, sometimes even excessive) over the entire lifetime of the database schema. An important observation stemming from the visual inspection of our change-over-time data, is that the term *continually* in the law's definition is challenged: *we observe that database schema evolution happens in bursts, in grouped periods of evolutionary activity, and not as a continuous process*! Take into account that the versions with zero changes are versions where either commenting and beautification takes place, or the changes do not refer to the information capacity of the schema (relations attributes and constraints) but rather, they concern the physical level properties (indexes, storage engines, etc) that pertain to performance aspects of the database.

Can we state that this stillness makes the schema "unsatisfactory" (referring back to the wording of the first law by Lehman)? We believe that the answer to the question is negative: since the system hosting the database continues to be in use, user dissatisfaction would actually call for continuous growth of the database, or eventual rejection of the system. This does not happen. On the other hand, our explanation relies on the reference nature of the database in terms of software architecture: if the database evolves, the rest of the code, which is basically using the database (and not vice versa), breaks!

Overall, if we account for the exact wording of the law, we conclude that the law partially holds.

Law of Feedback System (Law VIII). The wording of Law VIII refers to the existence of a self-stabilizing feedback mechanism that governs evolution. Its experimental evaluation typically refers to the possibility of demonstrating adherence to a basic formula of feedback, by estimating the size of the system (here: in terms of number of relations) accurately – i.e., with small error compared to the actual values. The formula typically used [2] is: $\widehat{S}_i = \widehat{S}_{i-1} + \frac{\overline{E}}{\widehat{S}_{i-1}^2}$, where $\widehat{S}$ refers to the estimated system size and $\overline{E}$ is a model parameter approximating effort (actually obtained as the average value of a set of past assessments of E).

Related literature [2] suggests computing $\overline{E}$ as the average value of individual E_i, one per transition. Then, we need to estimate these individual effort approximations. [2] suggests two formulae that we generalize here as follows: $E_i = \frac{s_i - s_\alpha}{\sum_{j=\alpha}^{i-1} \frac{1}{s_j^2}}$, where s_i refers to the actual size of the schema at version i and α refers to the version from which counting starts. Specifically, [2] suggests two values for α, specifically (i) 1 (the first version) and (ii) s_{i-1} (the previous version).

We now move on to discuss what seems to work and what not for the case of schema evolution. We will use the OpenCart data set as a reference example; however, all datasets demonstrate exactly the same behavior.

First, we assessed the formulae of [2]. In this case, we compute the average $\overline{E}$ of the individual E_i over the entire dataset. We employ four different values for α, specifically 1, 5, 10, and n, with n being the entire data set size, and depict the result in Fig. 3, where the actual size is represented by the blue solid line. The results indicate that the approximation modestly succeeds in predicting an overall increasing trend for all four cases, and, in fact, all four approximations targeted towards predicting an increasing tendency that the actual schema does not demonstrate. At the same time, all four approximations fail to capture the individual fluctuations within the schema lifetime.

A better estimation occurred when we realized that back in 1997 people considered that the parameter $\overline{E}$ was constant over the entire lifetime of the project; however, later observations (see [3]) led to the revelation that the project was split in phases. So, for every version i, we compute $\overline{E}$ as an average over the last τ E_j values, with small values for τ (1/5/10).

As we can see in Fig. 3, *the idea of computing the average $\overline{E}$ with a short memory of 5 or 10 versions produced extremely accurate results. This holds for all data sets.* This observation also suggests that, if the phases that [3] mentioned actually exist for the case of database schema, they are really small and a memory of 5-10 versions is enough to produce very accurate results.

Overall, *the evolution of the database schema appears to obey the behavior of a feedback-based mechanism, as the schema size of a certain version of the database can be accurately estimated via a regressive formula that exploits the amount of changes in recent, previous versions.*

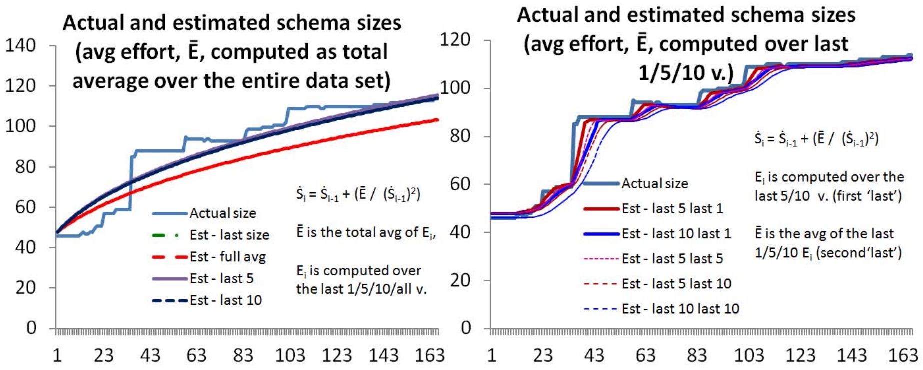

Fig. 3. Actual and estimated schema size via a total (left) and a bounded (right) average of individual E$_i$ for OpenCart; the x-axis shows the version id

Law of Self-Regulation (Law III). Whereas the law simply states that the evolution of software is feedback regulated, its experimental validation in the area of software systems is typically supported by the observation of a recurring pattern of smooth expansion of the system's size(a.k.a. "baseline" growth), that is interrupted with releases of perfective maintenance with size reductions or with releases of growth. Moreover, due to a previous wording of the law (e.g., see [2]) that described change to follow a normal distribution, the experimental assessment included the validation of whether growth demonstrates oscillations around the average value [1,2,3].

Size. The evolution of size can be observed in Fig. 1 and 2. We have to say that we simply do not detect the same behaviour that Lehman did (contrast Fig. 1, 2 to the respective figures of articles [1] and [2]): in sharp contrast to the smooth baseline growth that Lehman has highlighted, the evolution of the size of the studied database schemata provides a landscape with a large variety of *sequences of the following three fundamental behaviors.*

- In all schemata, we can see periods of increase, especially at the beginning of their lifetime or after a large drop in the schema size. This is an indication of positive feedback, i.e., the need to expand the schema to cover the information needs of the users.
- In all schemata, there are versions with drops in schema size. Those drops are typically sudden and steep and usually take place in short periods of time. Sometimes, in fact, these drops are of significantly larger size than the typical change. We can safely say that the existence of these drops in the schema size indicate perfective maintenance and thus, the existence of a negative feedback mechanism in the evolution process.
- In all schemata, there are periods of stability (i.e., size stays still, or –near-still).

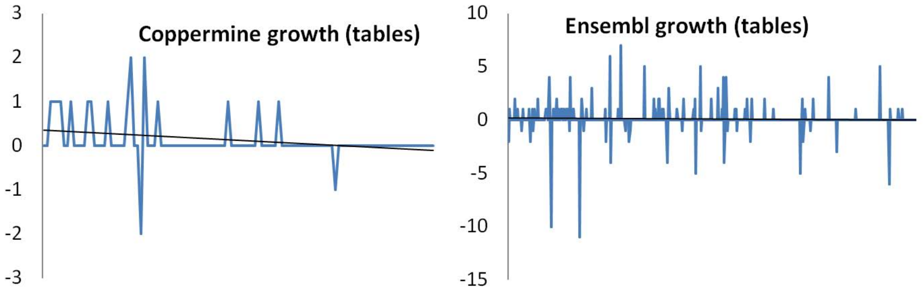

Fig. 4. Growth for Coppermine and Ensembl (over version id, concealed for fig. clarity); the slowly dropping solid line shows the linear interpolation of the values

Growth. Growth (i.e., the difference in the size between two subsequent versions) in all datasets has the following broad characteristics (Fig. 4, 6). In terms of tables, in most cases, growth is small (typically ranging within 0 and 1), and moderately small when it comes to attributes. We *have too many occurrences of zero growth*, typically iterating between small non-zero growth and zero growth. Due to perfective maintenance, we also have negative values of growth (less than the positive ones). We do not have a constant flow of versions where the schema size is continuously changing; rather, we have small spikes between one and zero. Thus, we have to state that the growth comes with *a pattern of spikes*. Due to this characteristic, *the average value is typically very close to zero (on the positive side) in all datasets, both for tables and attributes*. There are *few cases of large change* too; we forward the reader to Law V for a discussion of their characteristics.

We would like to put special emphasis to the observation that *change is small*. In terms of tables, growth is mostly bounded in small values. This is not directly obvious in the charts, because they show the ripples; however, almost all numbers are in the range of [-2..2] – in fact, mostly in the range [0..2]. Few abrupt changes occur. In terms of attributes, the numbers are higher, of course, and depend on the dataset. Typically those values are bounded within [-20,20]. However, the deviations from this range are not many.

In the course of our deliberations, we have observed a pattern common in all datasets: *there is a Zipfian model in the distribution of frequencies*. Observe Fig. 5 that comes with two parts, both depicting how often a growth value appears in the attributes of Ensemble. The x-axis keeps the delta size and the y-axis the number of occurrences of this delta. In the left part we include zeros in the counting (343 occurrences out of 528 data points) and in the right part we exclude them (to show that the power law does not hold only for the most popular value). We observe that there is a small range of deltas, between -2 and 4 that takes up 450 changes out of the 528. This means that, despite the large outliers, change is strongly biased towards small values close to zero.

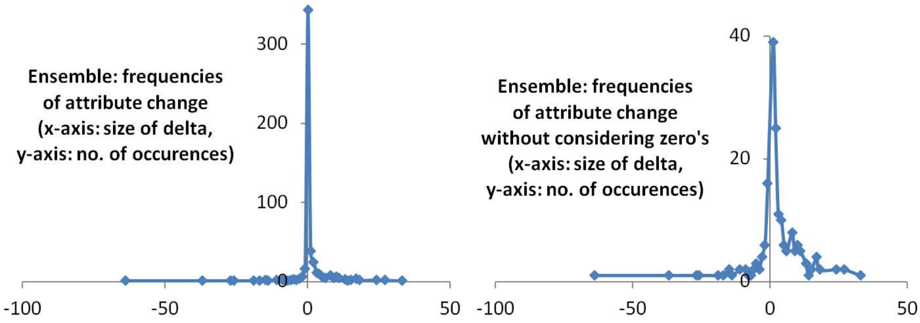

Fig. 5. Frequency of change values for Ensembl attributes

Despite the fact that change does not follow the pattern of baseline smooth growth of Lehman and the fact that change obeys a Zipfian distribution with a peak at zero, we have to say that the presence of feedback in the evolution process is evident; thus the law holds.

4.2 Properties of Growth for Schema Evolution

Law of Continuing Growth (Law VI). The sixth law of continuing growth requires us to verify whether the information capacity of the system (schema size) continuously grows. In all occasions, the schema size increases in the long run (Fig. 1, 2). We frequently observe some shrinking events in the timeline of schema growth in all data sets. However, *all data sets demonstrate the tendency to grow over time*. However, we also have differences from the traditional software systems that the law studies: as with Law I, the term ”continually” is questionable. As already mentioned (refer to Law III and Fig. 1, 2), change comes with frequent (and sometimes long) periods of *stability*, where the size of the schema does not change (or changes very little).

Therefore we can conclude that *the law holds, albeit modified to accommodate the particularities of database schemata.*

Law of Conservation of Familiarity (Law V). A first question, of central interest for the fifth law's intuition is: "What happens after excessive changes? Do we observe small ripples of change, showing the absorbing of the change's impact in terms of corrective maintenance and developer acquaintance with the new version of the schema?" An accompanying question, typically encountered in the literature, is: "What is the effect of age over the growth and the growth ratio of the schema?" Is it slowly declining, constant or oblivious to age? Again, we would like to remind the reader on the properties of growth, discussed in Law III of self-regulation: the changes are small, come with spike patterns between zero and non-zero deltas and the average value of growth is very close to zero.

Concerning the ripples after large changes, we can detect several patterns. Observe Fig. 6, depicting attribute growth for the MediaWiki dataset. Due to

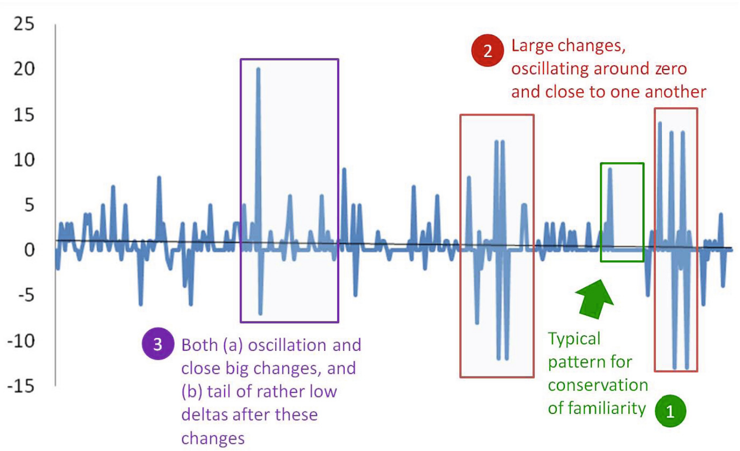

Fig. 6. Different patterns of change in attribute growth of Mediawiki (over version-id, concealed for fig. clarity)

the fact that this involves the growth of attributes, the phenomena are amplified compared to the case of tables. Reading from right to left, we can see that there are indeed cases where a large spike is followed by small or no changes (case 1). However, within the small pool of large changes that exist overall, it is quite frequent to see sequences of large oscillations one after the other, and quite frequently being performed around zero too (case 2). In some occurrences, we see both (case 3).

Concerning the effect of age, we do not see a diminishing trend in the values of growth; however, *age results to a reduction in the density of changes and the frequency of non-zero values in the spikes. This explains the drop of the average value in almost all the studied data sets* (Fig. 4): the linear interpolation drops; however, this is not due to the decrease of the height of the spikes, but due to the decrease of their density.

The heartbeat of the systems tells a similar story: typically, change is quite more frequent in the beginning, despite the fact that existence of large changes and dense periods of activities can occur in any period of the lifetime. Fig. 1 and 2 clearly demonstrate this by combining schema size and activity. This trend is typical for almost all of the studied databases (with phpBB being the only exception, demonstrating increased activity in its latest versions with the schema size oscillating between 60 and 63 tables).

Concerning the validity of the law, we *believe that the law is possible but not confirmed*. The law states that the growth is constrained by the need to maintain familiarity. However, the peculiarity of databases, compared to typical software systems, is that there are other good reasons to constrain growth: (a) a high degree of dependence of other modules from the database, and, (b) an intense effort to make the database clean and organized. Therefore, conservation of familiarity, although important cannot solely justify the limited growth. The extent of the contribution of each reason is unclear.

Law of Conservation of Organizational Stability (Law IV). To validate the hypothesis that the law of conservation of organizational stability holds, we need to establish that the project's lifetime is divided in phases, each of which (a) demonstrates a constant growth, and, (b) is connected to the next phase with an abrupt change. Moreover, abrupt changes should occur from time to time and not all the time (resulting in extremely short phases).

If we focus on the essence of the law, we can safely say that it does not hold. The heartbeats of Fig. 1 and 2 and the arbitrary sequencing of spikes and stability (Fig. 4, 6) make it impossible to speak about constant growth, even in phases. The open-source nature of our cases plays a role to that too.

4.3 Perfective Maintenance for Schema Evolution

Law of Increasing Complexity (Law II). The law states that complexity increases with age, unless effort is taken to prevent this – nevertheless, the law does not prescribe a clear assessment method for its validity. The rationale behind verifying the law dictates the observation of (a) an increasing trend in complexity of a software system, battled by (b) a perfective maintenance activity that attempts to reduce it and demonstrated by drops in the system size and rate of expansion. As there is no precise definition and measurement of complexity in the law, different metrics have been employed (coupling, cyclomatic complexity, etc. – see [6] for a review). Unfortunately, as most of these metrics are non-applicable to the case of databases, we take a definition already found in Lehman [1]: *complexity is defined as the number of modules handled (in our case tables added or modified) over the absolute value of growth per transition.* This formula approximates how much effort has been invested in expanding the system over the actual difference achieved (large values demonstrate too much effort for too small change).

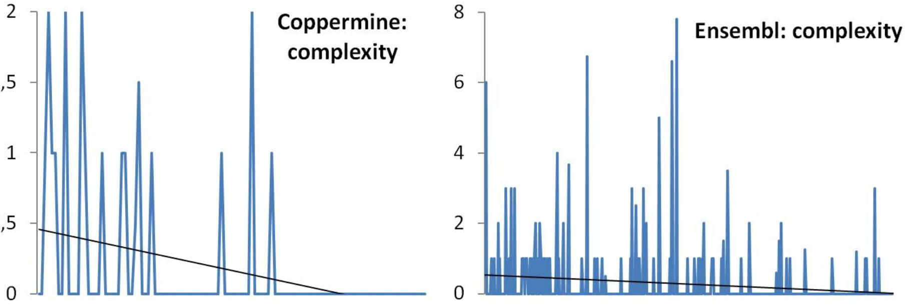

Fig. 7. Complexity for Coppermine and Ensembl (over version-id, concealed for clarity)

Related literature typically speaks for increasing complexity [1], [2], [3], [6], although there have been counterarguments for the case of open source software [5]. In our case, *in all the datasets but Biosql, complexity, as defined in the previous paragraph, does not increase* (Fig. 7). The phenomenon must be coupled with the drop in change density (Law V) and although we cannot provide undisputable explanation, we offer the synergy of two causes: (a) the increasing dependence of the surrounding code to the database that makes developers more cautious to perform schema changes as they incur higher maintenance costs, and, (b) the success of the perfective maintenance, which results in a clean schema, requiring less corrective maintenance in the future.

Although we cannot confirm or disprove the law based on undisputed objective measurements, we have indications that the second law partially holds, albeit with completely different connotations than the ones reported by Lehman for typical software systems: in the case of database schemata, complexity, when measured as the fraction of expansion effort over actual growth, drops.

Law of Declining Quality (Law VII). The seventh law postulates that quality declines with age unless the system is rigorously adapted to its external environment. Lehman and Fernandez-Ramil [3] avoid both (a) a definition of quality "the definition, measurement, modelling and monitoring of software quality-related characteristics are very dependent on application, organisation, product and process characteristics and goals", and, (b) giving any other support to the law than a logical proof: as the system expands over time, its complexity rises and thus the addressing of user requirements and removal of defects becomes more and more difficult, unless work is done to confront the phenomenon ("the decline in software quality with age, appears to relate to a growth in complexity that must be associated with ageing").

We have already demonstrated that the rationale behind complexity increase is not supported by our observations. At the same time, we cannot assess schema quality with undisputed means. Therefore, we cannot confirm or disprove the law based on undisputed objective measurements.

5 Discussion

In this section, we summarize fundamental *observations* and *patterns* that have been detected in our study. We intentionally avoid the term *law*, as we do not have unshakeable evidence for their explanation. Apart from the *empirical grounding*, due a very large amount of datasets that obey the same patterns (which we believe we have fairly attained), we would require an undisputed *rationalized grounding*, that can be obtained via a clear explanation of the underlying mechanism that guides them, also established on measured, undisputed data.

Feedback-Based Behavior for Schema Evolution. *As an overall trend, the information capacity of the database schema is enhanced – i.e., the size grows in the long term (VI). The existence of perfective maintenance is evident in almost all datasets with the existence of relation and attributes removals, as well* *as observable drops in growth and size of the schema (sometimes large ones). In fact, growth frequently oscillates between positive and negative values (III). The schema size of a certain version of the database can be accurately estimated via a regressive formula that exploits the amount of changes in recent, previous versions (VIII).* Based on the above, we can state that the essence of Lehman's laws applies to open-source databases too: *Schema evolution demonstrates the behavior of a feedback-regulated system, as it obeys the antagonism between the need for expanding its information capacity to address user needs and the need to control the unordered expansion, with perfective maintenance.*

Observations Concerning the Heartbeat of Change. *The database is not continuously adapted, but rather, alterations occur from time to time, both in terms of versions and in terms of time (I). Change does not follow patterns of constant behaviour (IV). Age results in a reduction of the density of changes to the database schema in most cases (V).*

Schema Growth Is Small (Observations): *Growth is typically small in the evolution of database schemata, compared to traditional software systems (III). The distribution of occurrences of the amount of schema change follows a Zipfian distribution, with a predominant amount of zero growth in all data sets. Plainly put, there is a very large amount of versions with zero growth, both in the case of attributes and in the case of tables. The rest of the frequently occurring values are close to zero, too. The average value of growth is typically close to zero (although positive) (III) and drops with time, mainly due to the drop in change density (V).*

Threats to Validity. We start with a *fundamental inquiry*: are databases E-type systems, so that this research is meaningful in the first place? Despite a fundamental difference (as databases involve information and not functional capacity), databases, closely resemble E-type systems as they address the real problem of query answering, come with their own user community (developers, DBA's), and act as fairly independent modules in information systems. Concerning the *external validity* of our study, its context concerns *the study of the evolution of the logical schema of databases in open-source software.* We avoid generalizing our findings to databases operating in closed environments and we stress that our study has focused only on the logical structure of databases, avoiding physical properties (let alone instance-level observations). Overall, we believe we have provided a safe, representative experiment with a significant number of schemata, having different purposes in the real world and time span (from rather few (40) to numerous (500+) versions). Our findings are generally consistent (with few exceptions that we mentioned). Concerning *internal validity* and cause-effect relationships, we avoid directly relating age with phenomena like the dropping density of changes or the size growth; on the contrary, we attribute the phenomena to a confounding variable, perfective maintenance actions, which we anticipate to be causing the observed behavior. When it comes to *construct validity*, all the measures we have employed are accurate, consistent with the metrics used in the related literature and appropriate for assessing the law to which they are employed. The only exceptions to this statement are Laws II and VII dealing with the complexity and the quality of the schemata. Both terms are very general and the related database literature does not really provide adequate metrics other than size-related (which we deem too simple for our purpose); our own measurement of complexity requires deeper investigation. Therefore, the undisputed assessment of these laws remains open.

Future Work. The extension of the study to more datasets, possibly non-relational too, and the study of databases in closed environments for large periods of time, are possible roads for future research. Concerning the current findings of our study, the detailed understanding of the feedback mechanism, especially when it comes to ageing and complexity (Law II) as well as patterns of growth (Laws III and V), or patterns in the heartbeat of the evolution, are open issues worth investigating.

Acknowledgment. This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Thales. Investing in knowledge society through the European Social Fund.

References

1. Belady, L.A., Lehman, M.M.: A model of large program development. IBM Systems Journal 15(3), 225–252 (1976)
2. Lehman, M.M., Ramil, J.F., Wernick, P., Perry, D.E., Turski, W.M.: Metrics and laws of software evolution - the nineties view. In: 4th IEEE International Software Metrics Symposium (METRICS 1997), p. 20 (1997)
3. Lehman, M.M., Fernandez-Ramil, J.C.: Rules and Tools for Software Evolution Planning and Management. In: Software Evolution and Feedback: Theory and Practice. John Wiley and Sons Ltd. (2006) ISBN-13: 978-0-470-87180-5
4. Xing, Z., Stroulia, E.: Analyzing the evolutionary history of the logical design of object-oriented software. IEEE Trans. Software Eng. 31(10), 850–868 (2005)
5. Fernández-Ramil, J., Lozano, A., Wermelinger, M., Capiluppi, A.: Empirical studies of open source evolution. In: Software Evolution, pp. 263–288 (2008)
6. Xie, G., Chen, J., Neamtiu, I.: Towards a better understanding of software evolution: An empirical study on open source software. In: 25th IEEE International Conference on Software Maintenance (ICSM 2009), Edmonton, Alberta, Canada, pp. 51–60 (2009)
7. Sjøberg, D.: Quantifying schema evolution. Information and Software Technology 35(1), 35–44 (1993)
8. Papastefanatos, G., Vassiliadis, P., Simitsis, A., Vassiliou, Y.: Metrics for the prediction of evolution impact in etl ecosystems: A case study. J. Data Semantics 1(2), 75–97 (2012)
9. Curino, C., Moon, H.J., Tanca, L., Zaniolo, C.: Schema evolution in wikipedia: toward a web information system benchmark. In: Proceedings of ICEIS 2008. Citeseer (2008)
10. Curino, C.A., Moon, H.J., Zaniolo, C.: Graceful database schema evolution: the prism workbench. Proceedings of the VLDB Endowment 1, 761–772 (2008)