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.

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

Ioannis Skoulis, Panos Vassiliadis, Apostolos Zarras

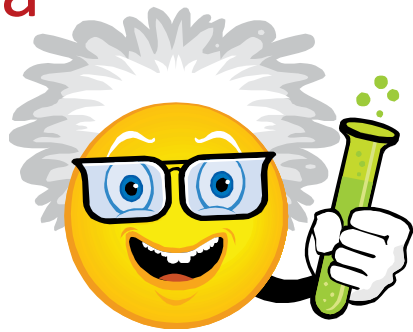
Department of Computer Science and Engineering
University of Ioannina, Hellas

Univ. of Ioannina

WHAT ARE THE LAWS OF DATABASE SCHEMA EVOLUTION?

What are the laws of database (schema) evolution?

- How do databases change?
- In particular, how does the schema of a database evolve over time?



- Long term research goals:
  - Are there any "invariant properties" (e.g., patterns of repeating behavior) on the way database (schemata) change?
  - Is there a theory / model to explain them?

Why care for the laws/patterns of schema evolution?

- Scientific curiosity!
- Practical Impact: DB's are dependency magnets. Applications have to conform to the structure of the db…
  - typically, development waits till the "db backbone" is stable and applications are build on top of it
  - slight changes to the structure of a db can cause several (parts of) different applications to crash, causing the need for emergency repairing

Imagine if we could predict how a schema will evolve over time…

- … we would be able to **"design for evolution"** and **minimize the impact of evolution** to the surrounding applications
  - by applying design patterns
  - by avoiding anti-patterns & complexity increase

  … **in both the db and the code**
- … we would be able to **plan** administration and perfective maintenance tasks and resources, instead of responding to emergencies

Why aren't we there yet?

- Historically, nobody from the research community had access + the right to publish to version histories of database schemata
- Open source tools internally hosting databases have changed this landscape:
  - not only is the code available, but also,
  - public repositories (git, svn, …) keep the entire history of revisions
- We are now presented with the opportunity to study the version histories of such "open source databases"

http://www.cs.uoi.gr/~pvassil/publications/2014_CAiSE/

Our take on the problem

- Collected version histories for the schemata of 8 open-sour projects
  - CMS's: MediaWiki, TYPO3, Coppermine, phpBB, OpenCart
  - Physics: ATLAS Trigger --- Bio: Ensemble, BioSQL

- Preprocessed them to be parsable by our **HECATE schema comparison tool** and exported the transitions between each two subsequent versions and measures for them (size, growth, changes)

- Visualized the transitions in graphs and statistically studied the measures, in a study organized **around the 8 laws of Lehman for software evolution**

… and Lehman's Laws of evolution

SCHEMA EVOLUTION FOR O/S DB'S

Lehman's laws in a nutshell

- An E-Type software system continuously changes over time (I) obeying a complex feedback-based evolution process (VIII) that prohibits the uncontrolled growth of the system (III).
- Positive feedback: due to the need for growth and adaptation to user needs
  - evolution results in an increasing functional capacity of the system (VI),
  - produced by a growth ratio that is slowly declining in the long term (V),
  - with effort typically constant over phases (with the phases disrupted with bursts of effort from time to time (IV)).
- Negative feedback: to regulate the ever-increasing growth and control both the overall quality of the system (VII), with particular emphasis to its internal quality (II).
- **In our context:**
  - E-type system -> **database**, **functional** capacity -> **information** capacity

Lehman's Laws in groups

Three main groups for the Laws:

- Feedback-based System
  - I. Continuing change
  - VIII. Feedback System
  - III. Self Regulation
- Positive feedback
  - VI. Continuing Growth
  - V. Conservation of Familiarity
  - IV. Conservation of Organizational Stability
- Negative feedback
  - II. Increasing Complexity
  - VII. Declining Quality

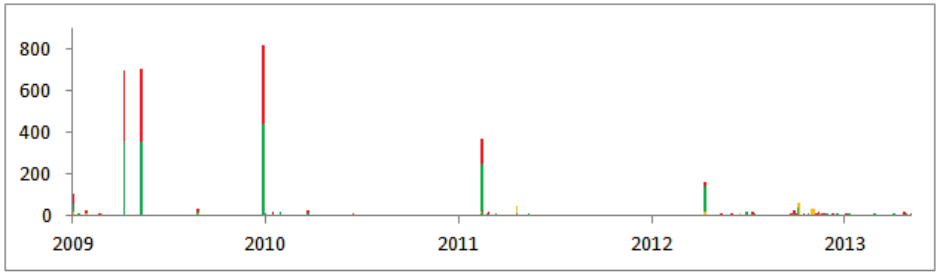
I. Continuing change

"The database schema is <u>continually</u> adapted or else it becomes progressively less satisfactory."

Evaluation: The db must show signs of evolution as time passes

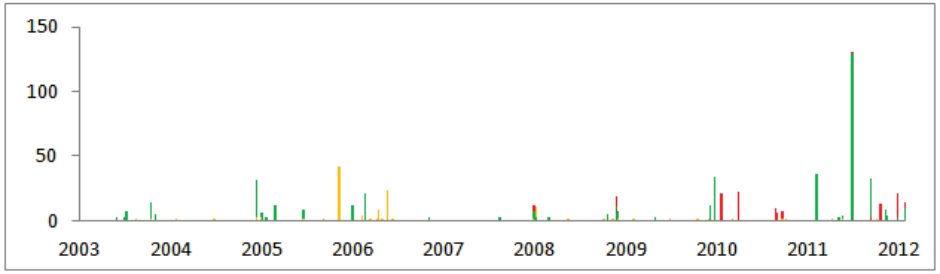
Metrics: heartbeat of changes over time and version

Outcome: Databases do change but not continuously

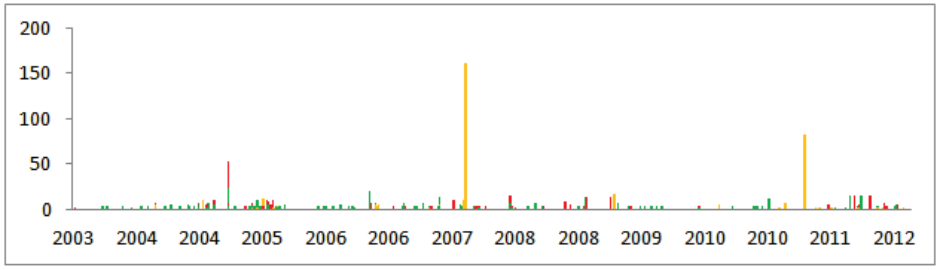


OpenCart

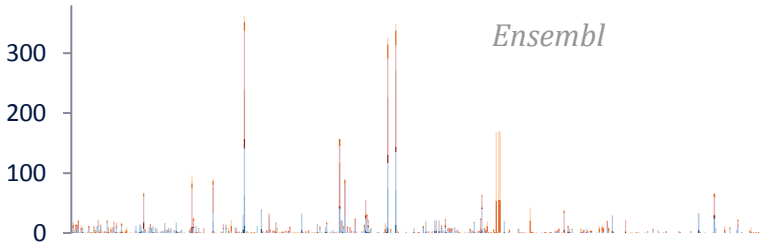
phpBB



TYPO3



MediaWiki



http://www.cs.uoi.gr/~pvassil/publications/201

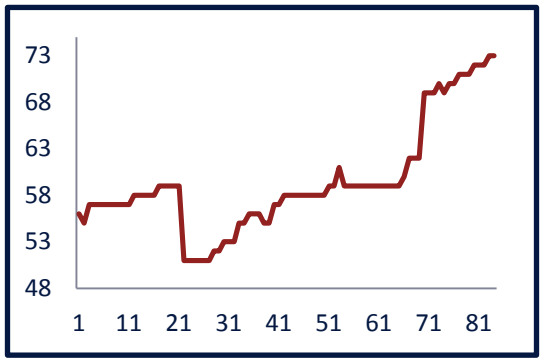
III. Self Regulation

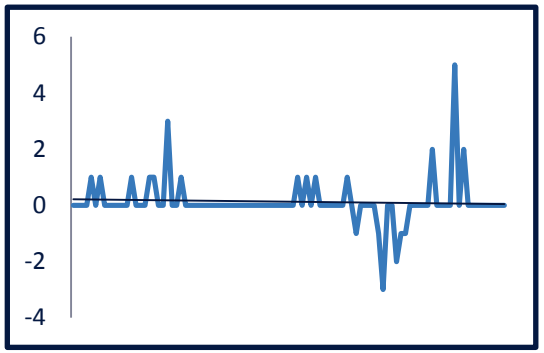
"Database schema evolution is feedback regulated."

Evaluation: i) indication of **patterns in size growth**, ii) existence of **negative feedback** (drop in size and growth locally decreasing), iii) **"ripples" in growth**

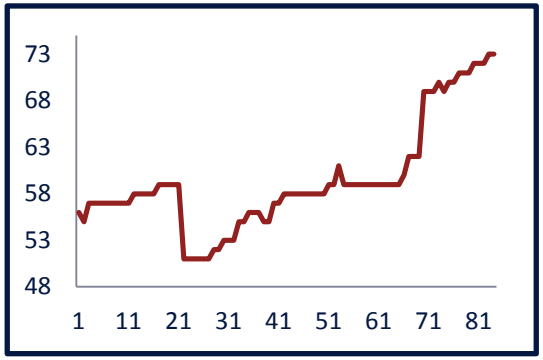
Metrics: size over version, system growth

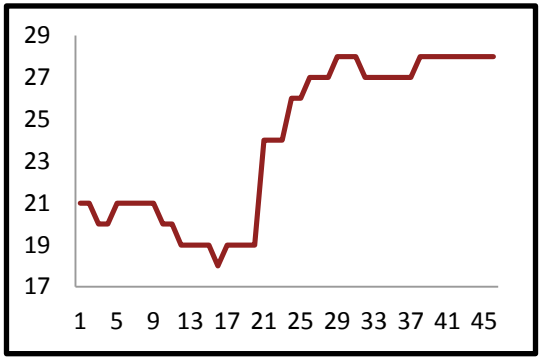
Outcome: feedback is evident, although differently than in traditional software

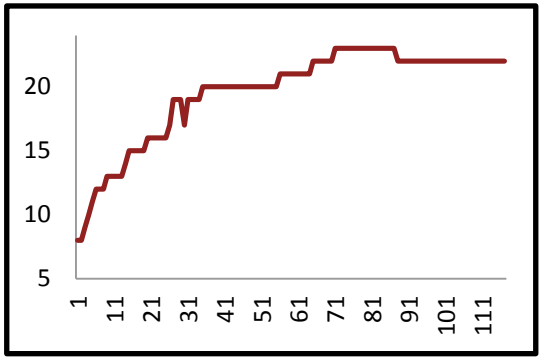


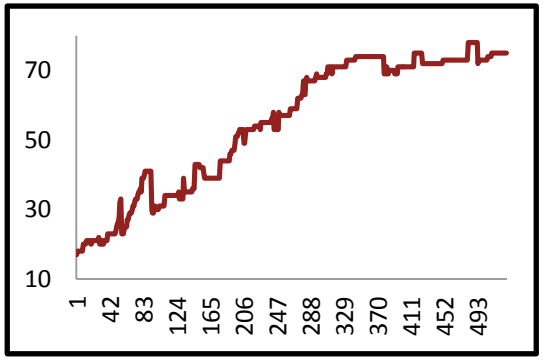


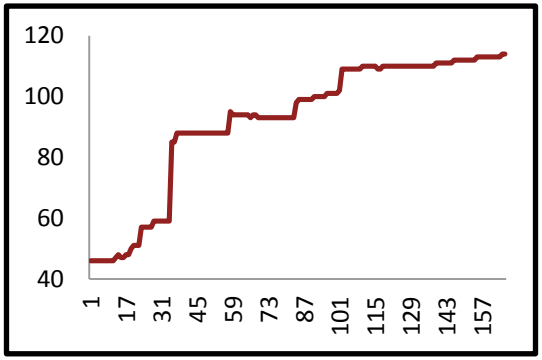
Schema Size (relations)

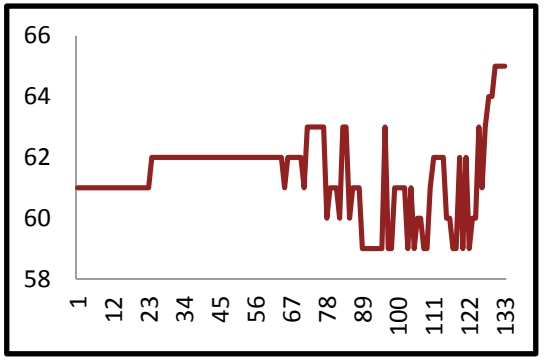


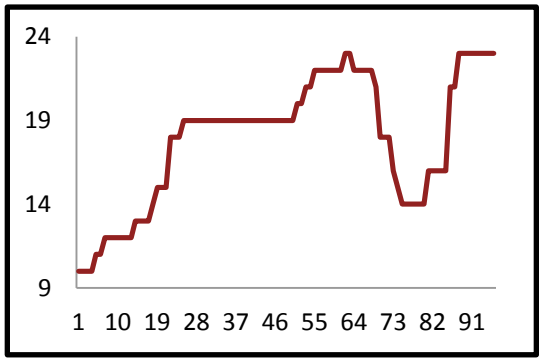


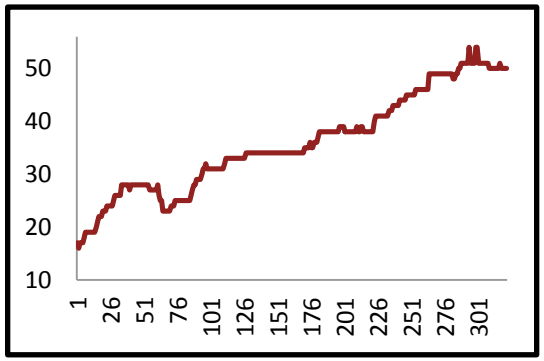












Schema Size

- Apparently, **there is both positive and negative feedback**; still, no Lehman-like recurring patterns

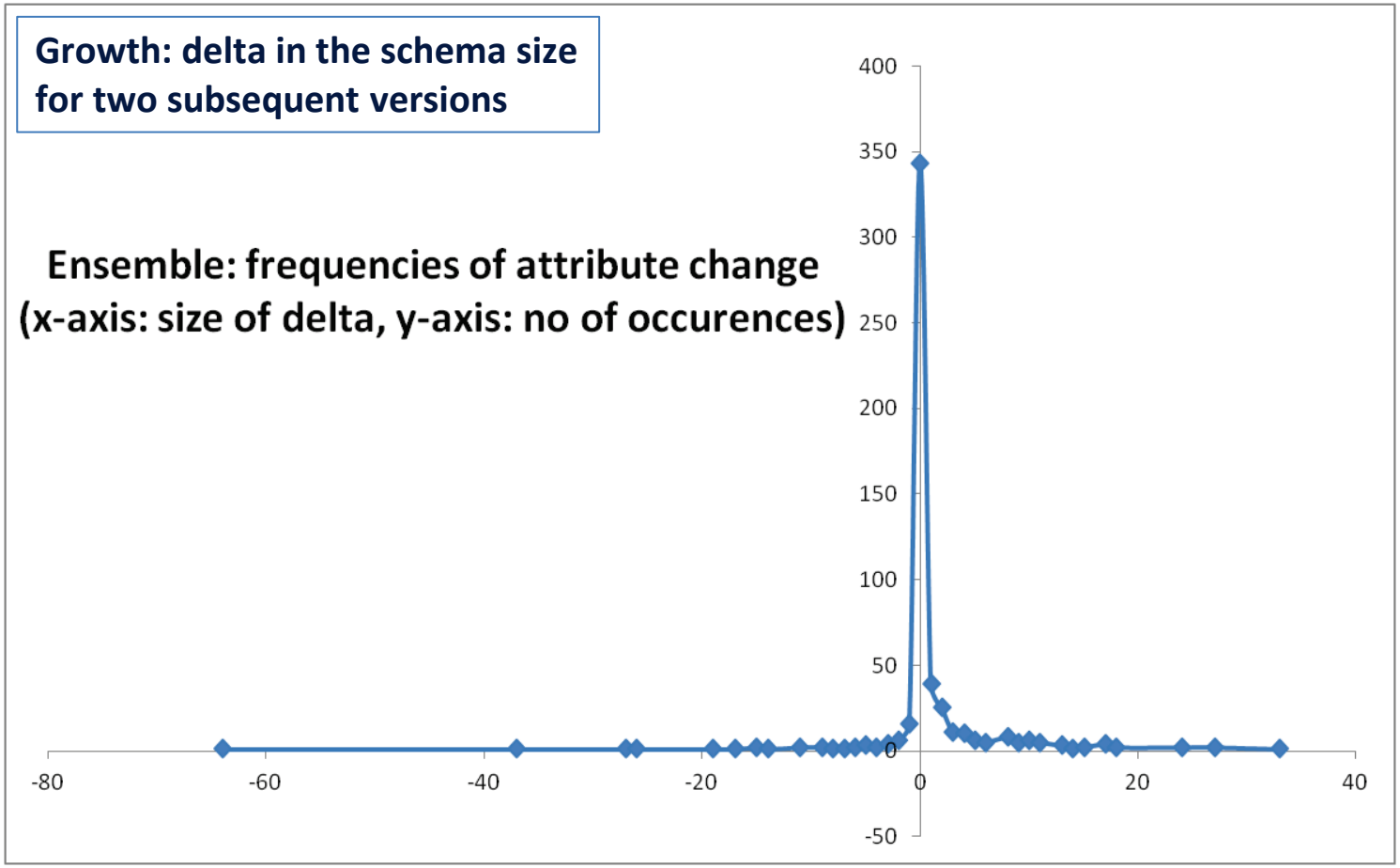
- Overall increase in size
- Periods of **increase**, esp. at beginning and after large drops -> positive feedback
- **Drops**: sudden and steep (in short duration) -> negative feedback
- **Large periods of stability!**
  - Unlike traditional S/W, db's are dependency magnets…

Schema growth is small!

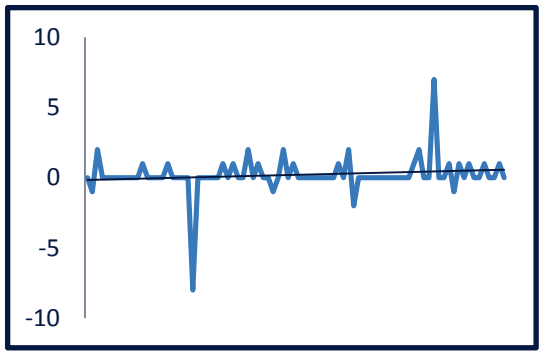
- **Growth is bounded in small values!**
- **Zipfian distribution of growth values around 0**
  - Predominantly: occurrences of zero growth; almost all deltas are bounded between [-2..2] tables
  - [0..2] tables slightly more popular => average value of growth slightly higher than 0
- No periods of continuous change; **small spikes instead**

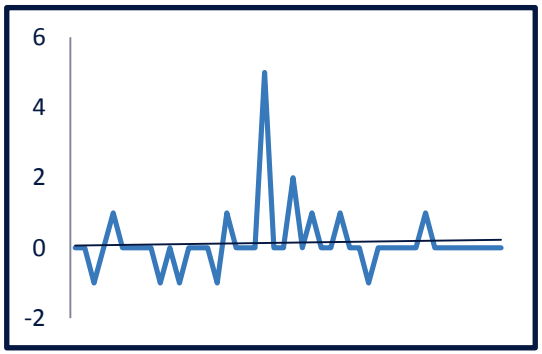
- Due to perfective maintenance, we also have negative values of growth (less than the positive ones).
- Oscillations exist too: positive growth is followed with immediate negative growth or stability

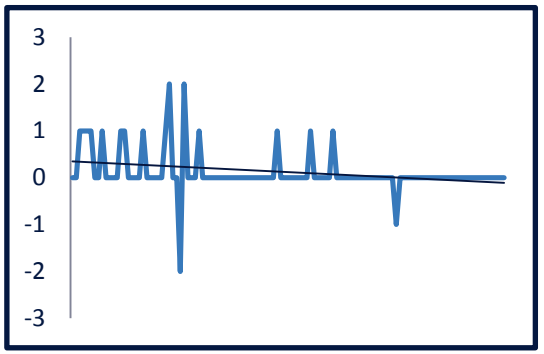
Zipfian model in the distribution of growth frequencies

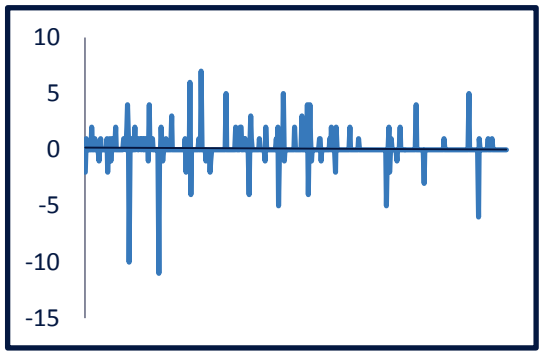


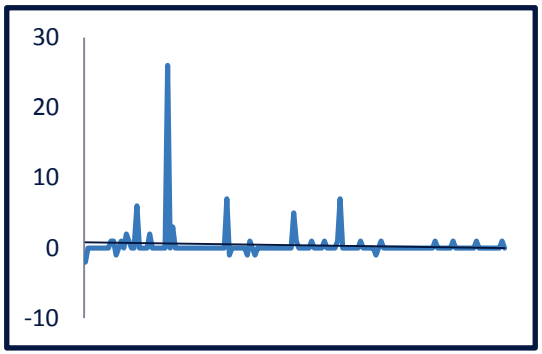
Schema Growth

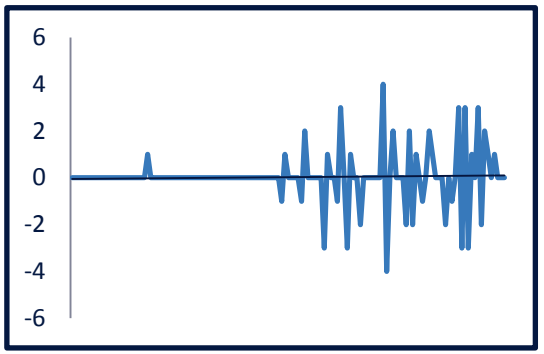


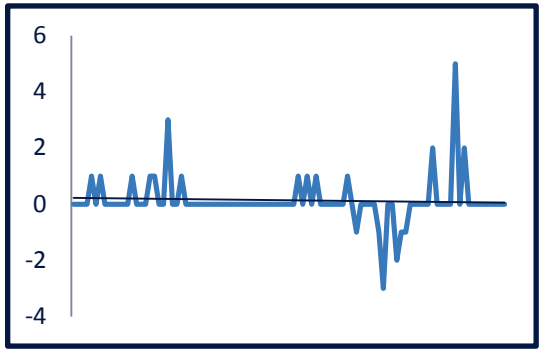


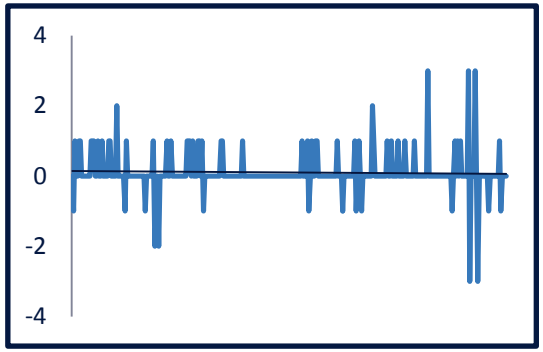












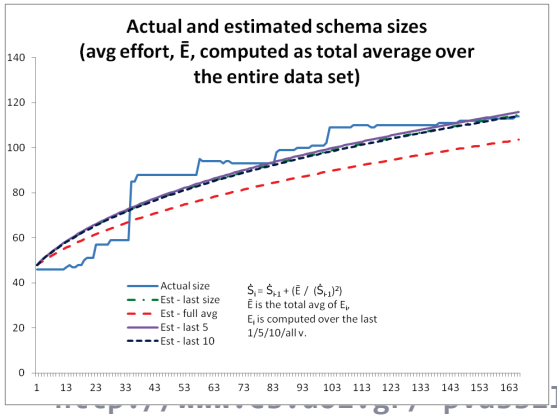
VIII. Feedback System

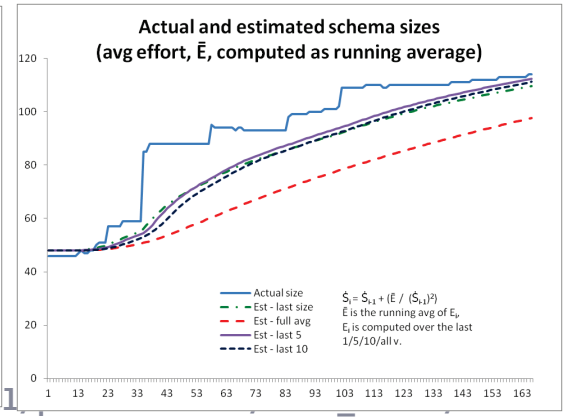
"Database schema evolution processes are multi-level, multi-loop, multi-agent feedback systems."

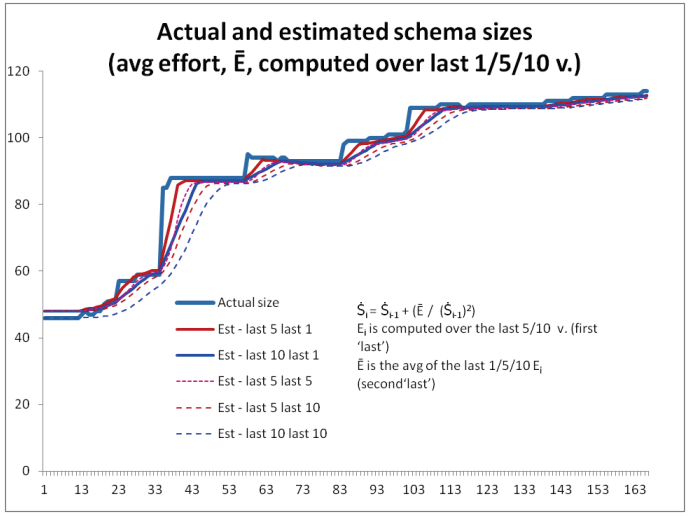
Evaluation: Regression analysis to the estimate size of the database schemata

Metrics: estimated size $\hat{S}_i = \hat{S}_{i-1} + \frac{\overline{E}}{\hat{S}_{i-1}^2}$, effort $E_i = \frac{s_i - s_a}{\sum_{j=a}^{i-1} \frac{1}{s_j^2}}$

Outcome: the regression formula holds, but with short memory!







Laws for Schema Evolution

Three main groups for the Laws:

- Feedback-based System

  - I. Continuing Change
  - VIII. Feedback System
  - III. Self Regulation
- Positive feedback
  - VI. Continuing Growth
  - V. Conservation of Familiarity
  - IV. Conservation of Organizational Stability
- Negative feedback
  - II. Increasing Complexity
  - VII. Declining Quality

VI. Continuing Growth

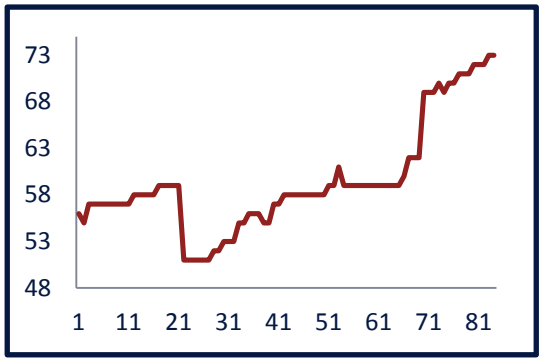
"The informational capacity of databases must be continually enhanced to maintain user satisfaction over system lifetime."

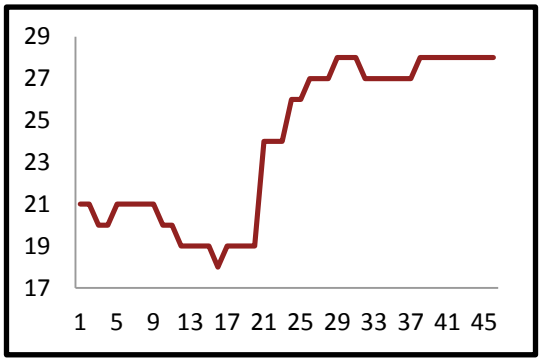
Evaluation: Overall expansion trend for the metrics involved

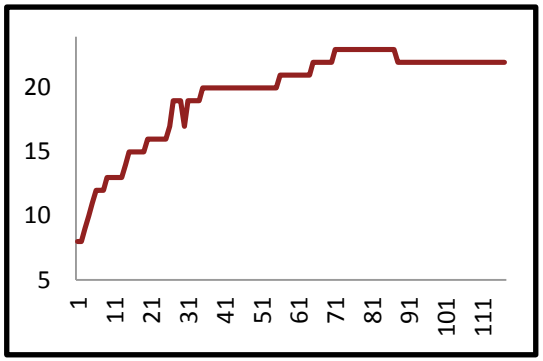
Metrics: number of relations, number of attributes

Outcome: we see phases

- **Stability** (unique for databases compared to traditional SW systems)
- Smooth expansion upwards
- Abrupt change (up or down)







V. Conservation of Familiarity

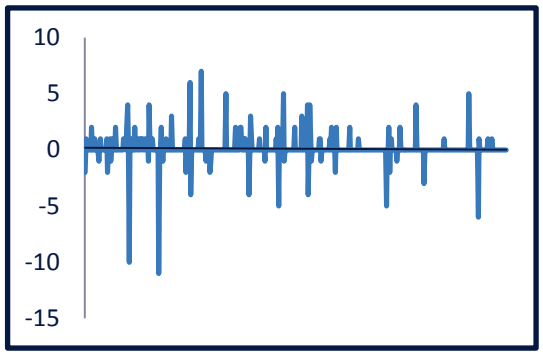
"In general, the incremental growth of database schema is constrained by the need to maintain familiarity."

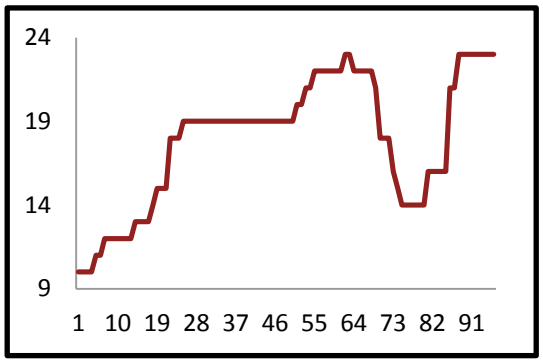
Evaluation: i) **growth is constant or declining with age**, ii) **versions with significant change in size are followed by small growth**

Metrics: schema growth, schema growth rate

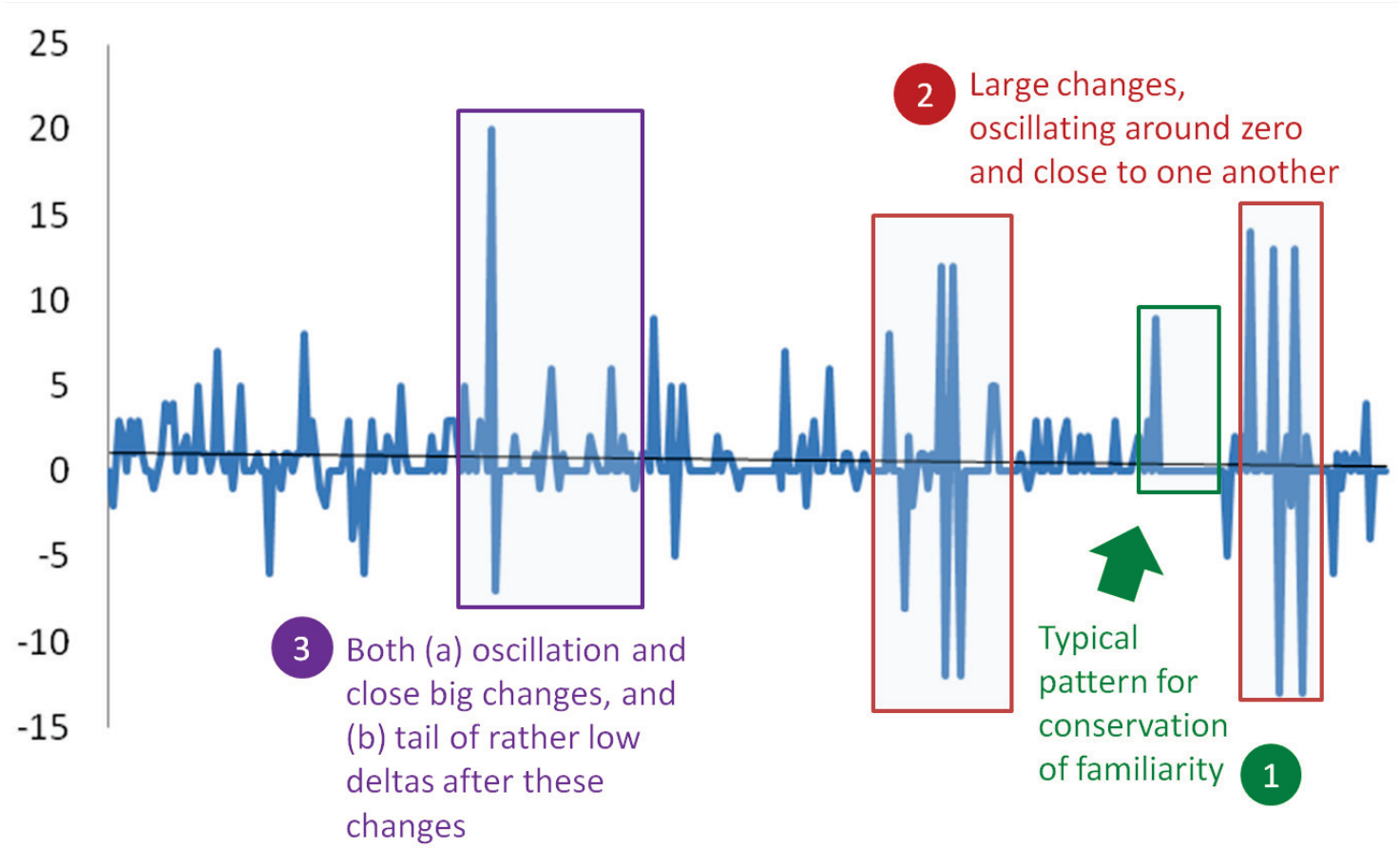
Questions: what happens after large changes?

how does age affect change rates?





V. Conservation of Familiarity



V. Conservation of Familiarity

- We observe a decline in the density of changes with age, but not a decline in growth size
  - Change is frequent in the beginning
  - Large changes and dense periods in any time

- Growth reacts as expected but is it because of the need to maintain familiarity?
  - Other modules are higly dependent on them
  - Effort might be taken to clean and organize a database

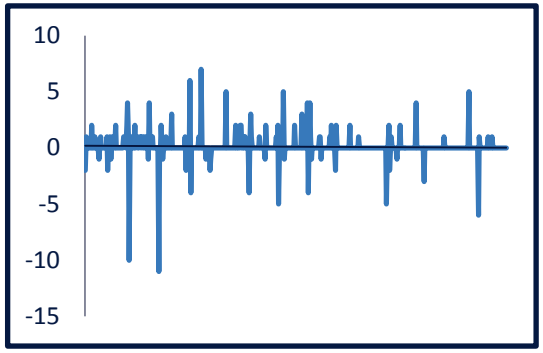
IV. Conservation of Organizational Stability

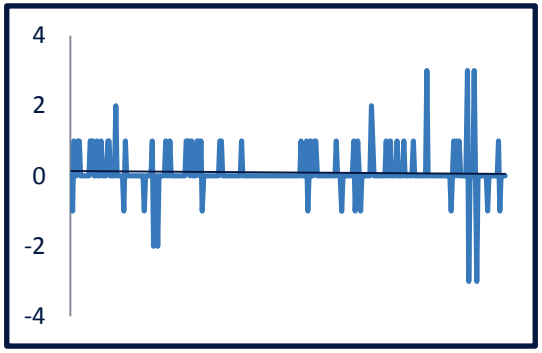
"The work rate of an organization evolving a database schema tends to be constant over the operational lifetime of that schema or phases of that lifetime."

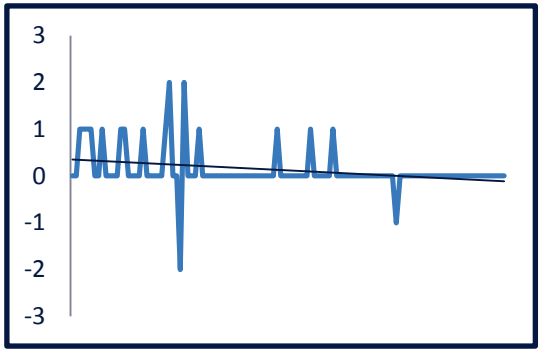
Evaluation: i) detect phases with constant growth, ii) those phases must be connected with abrupt changes

Metrics: schema growth

Outcome: simply not the case (despite the existence of abrupt changes, there is no "constant growth" – instead: stability and spikes)







Laws for Schema Evolution

Three main groups for the Laws:

- Feedback-based System
  -  I. Continuing Change
  - VIII. Feedback System
  - III. Self Regulation
- Positive feedback
  -  VI. Continuing Growth
  - V. Conservation of Familiarity
  - IV. Conservation of Organizational Stability
- Negative feedback
  - II. Increasing Complexity
  - VII. Declining Quality

II. Increasing Complexity

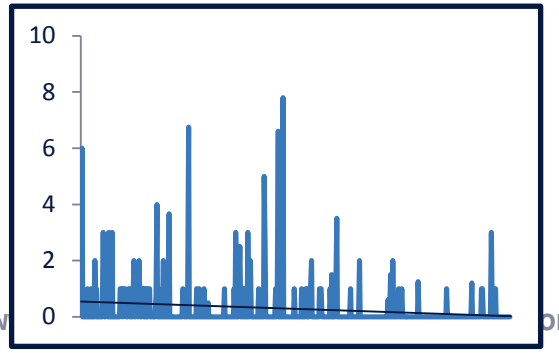
"Efforts to maintain internal quality must be made."

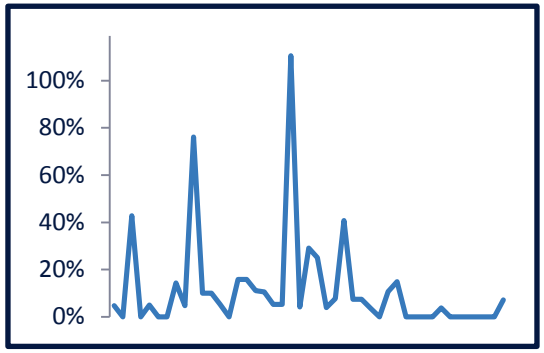
Evaluation: i) We must identify version with perfective maintenance, ii) the VIII law must hold, iii) the approximate complexity must increase

Metrics: $complexity \approx \frac{modules handled}{S_i - S_{i+1}}$ $maintenance\ rate \approx \frac{modules\ handled}{old\ \ size}$

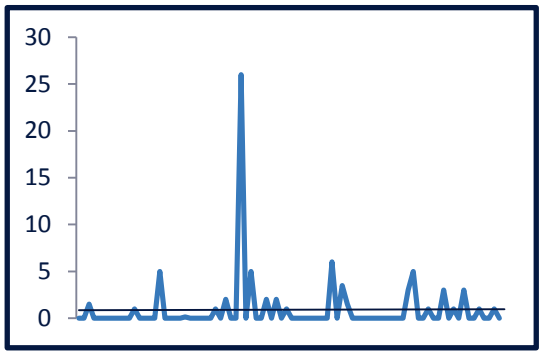
Complexity is an estimate based on the literature.

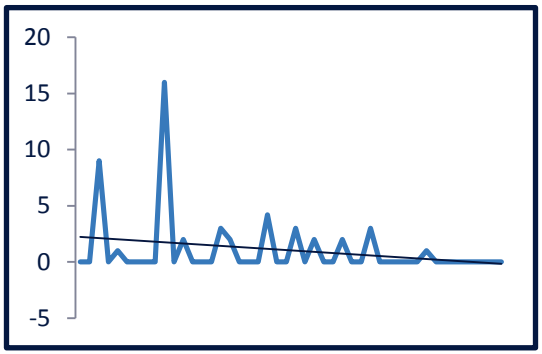
Outcome: unclear -- complexity appears dropping (due to the decline of change density). *Possibly, perfective maintainance has been successful?!*

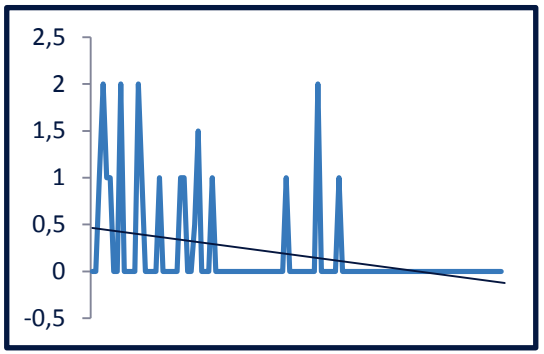


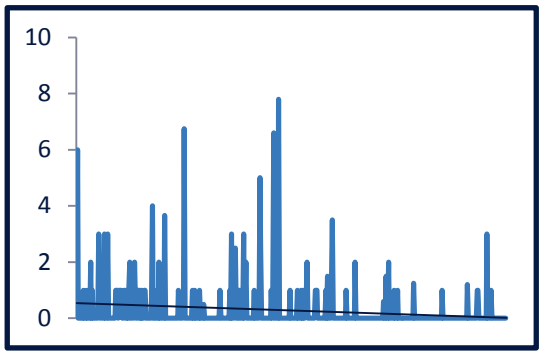


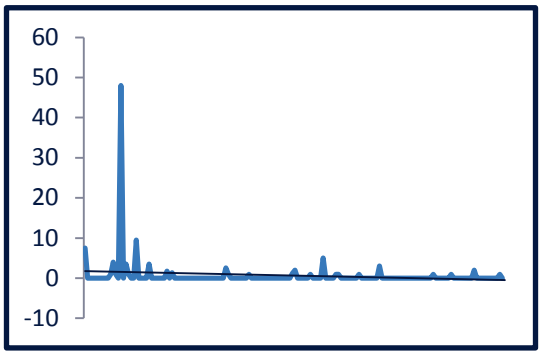
Complexity

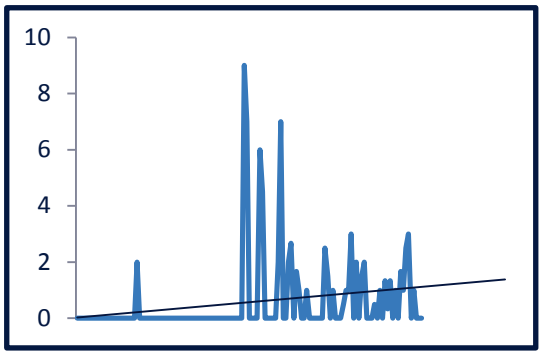

30
25
20
15
10
5
0

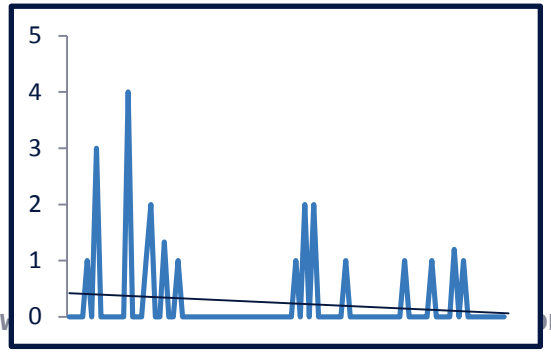

20
15
10
5
0
-5

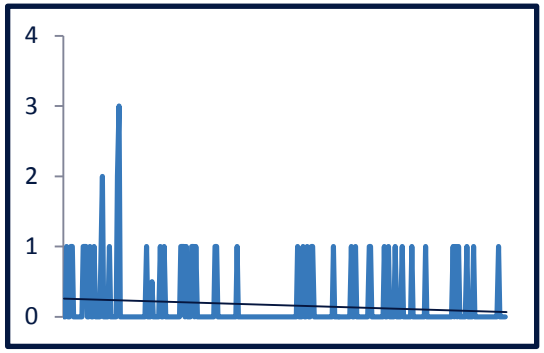

2,5
2
1,5
1
0,5
0
-0,5


10
8
6
4
2
0


60
50
40
30
20
10
0
-10


10
8
6
4
2
0


5
4
3
2
1
0


4
3
2
1
0

VII. Declining Quality

"Unless rigorously adapted and evolved to take into account changes in the operational environment, the quality of a database schema will appear to be declining."

Evaluation: Hold by logical induction, if III, VIII, and II hold

Metrics: not possible to measure external quality

Outcome: We are unsure of the behavior of internal quality so we are even more reluctant towards declaring external quality as improving.

Laws for Schema Evolution

Three main groups for the Laws:

- Feedback-based System
  -  I. Continuing Change
  - VIII. Feedback System
  - III. Self Regulation
- Positive feedback
  -  VI. Continuing Growth
  - V. Conservation of Familiarity
  - IV. Conservation of Organizational Stability
- Negative feedback
  -  II. Increasing Complexity
  - VII. Declining Quality

CONCLUSIONS AND OPEN ISSUES

Can we find any patterns on the way database (schemata) evolve?

- Major scientific goal with tremendous impact on our discipline and economy
- Now possible with the availability of data
- First results here
- Great potential for way too many more to come
- Would be great to pass from observed patterns to laws / theories too

Main results

Schema size (#tables, #attributes) supports the assumption of a feedback mechanism

- Schema size grows over time; not continuously, but with bursts of concentrated effort
- Drops in schema size signifies the existence of perfective maintenance
- Regressive formula for size estimation holds, with a quite short memory

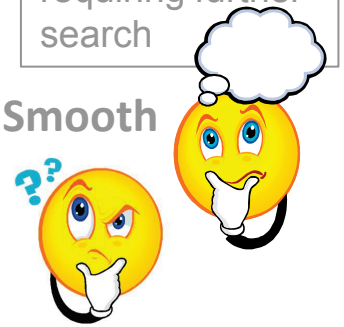
Schema Growth (diff in size between subsequent versions) is small!!

- Growth is small, smaller than in typical software
- The number of changes for each evolution step follows Zipf's law around zero
- Average growth is close (slightly higher) to zero

Patterns of change: no consistently constant behavior

- Changes reduce in density as databases age
- Change follows three patterns: **Stillness**, **Abrupt change** (up or down), **Smooth growth upwards**
- Change frequently follows **spike** patterns
- **Complexity** does **not** increase with age

Grey for results requiring further search



Future Work

- Time related measures and patterns
  - We have occasions were effort is high or low
  - We need better patterns of change over time
- Identifying changes better
  - Capture renames, table splitting, …
- Schema Complexity
  - We lack a representative set of metrics that measure the complexity of a database schema (e.g., via FK's / FD's / semantic rel's / …)

Thank you!

Any questions?

More information

http://www.cs.uoi.gr/~pvassil/publications/2014_CAiSE/

Hecate's Code

https://github.com/DAINTINESS-Group/Hecate

Data sets & results

https://github.com/DAINTINESS-Group/EvolutionDatasets
http://cs.uoi.gr/~iskoulis/results/results_LehmanforDBs.xlsx

Auxiliary slides

Conclusions

- … with a fairly high degree of certainty:
  - Database schemata grow over time; not continuously, but with bursts of concentrated effort
  - Regressive formula for size estimation holds, with a quite short memory
  - Growth is small, smaller than in typical software
  - The number of changes for each evolution step follows Zipf's law
  - Average growth is close to zero
  - Changes reduce in density as databases age

Conclusions

- … requiring further insight:
  - Change follows three patterns:
    - Stillness
    - Abrupt change (up or down)
    - Smooth growth upwards
  - Change frequently follows spike patterns
  - Large changes sequenced one after the other
  - Complexity does not increase with age

Results in detail

- 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.

Results in detail

- 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. (III)
  - 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).

How does our approach stands from the viewpoint of…

PHILOSOPHY OF SCIENCE

“Laws” and other terminological issues

- Pattern: reoccurrence of particular types of events with statistically significant properties in the context of a set of observations
- Law/principle: a statement expressing a fundamental pattern that is omnipresent in a large set of experiments
  - Non falsified yet, based on solid observations
  - Allowing prediction ☺
- Explanation: an inductive statement correlating the explanandum of a law with relevant causes

Disclaimer

- **We do not introduce any laws:**
  - … we just list our observations
  - … we would like to see patterns emerging from an even larger set of data sets
  - … we hope the scientific community can establish when we can switch from patterns to laws
    - Are we safe to assume "Uniformity of Nature" in CS?
    - Pessimistic on the future availability of version histories for the schemata of production db's
- We are very cautious to avoid expressing certainty on any issue related to causality, except for rare & obvious cases; thus **we do not speak of theories / mechanisms / explanations, either!**

Disclaimer: **We are interested in the mechanics of schema evolution, not Lehman's laws**

- Yes, we report on their validity on open-source db schemata
- Yes, we would like to eventually come up with a set of patterns or laws on how schema evolution takes place – even with causalities if possible
- No, we are not interested in Lehman's laws per se
  - For us, they are just a means to understand whether some properties hold for schema evolution
  - We are not assessing their validity at a global setting

Fundamental concerns (1)

Yes, there are differences between traditional SW systems and DB's:

- E-type systems export functionality to their users; on the contrary databases export information capacity, i.e., the ability to store data and answer queries.
  - *Thus, we believe that when it comes to schema evolution, all references to functionality or functional capacity should be restated with a view to information capacity*.
- E-type systems are complete software systems that provide overall solutions to problems in the real world; databases, on the other hand, are typically parts of a larger information system, serving the purpose of accurately answering queries that populate the surrounding information systems with the necessary data.
  - In other words, whereas there is a holistic view of systems in the former case, we have a specific component of a larger ecosystem in the latter.

Fundamental concerns (2)

- **Yet, we say that there is merit in using Lehman's laws as a first tool to study db evolution**
- Databases resemble typical software systems as they have a **specific "data provision" functionality** with a large degree of **independence** and a **stand-alone character**.
  - … as they come with users having requirements from them (in terms of information capacity), developers and administrators that deal with them and the code that surrounds them…

- **DB peculiarity:** ***a database is a fairly independent module of an information system that is more or less insulated from changes to the other modules; at the same time, its evolution can potentially affect every other module.***

Threats to validity

- External validity: we study the evolution of the **logical schema** of **databases in open-source software**.
- We avoid generalizing our findings to databases operating in closed environments or physical properties of db's
- 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 mention).

Threats to validity

- 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.

- 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 …
  - … except for 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.

LEHMAN'S LAWS OF SOFTWARE EVOLUTION

Laws on Software Evolution

- A set of eight rules on the behavior of software as it evolves
- Derived from a study, due to M. Lehman of proprietary software (OS/360)
- Almost 40 years of reviewing and evaluation (first three laws published in 1976)
- Have been recognized for their useful insights as to *what* and *why* evolves in the lifetime of a software system

Laws on Software Evolution

I. Continuing change

"An E-Type system must be continually adapted or else it becomes progressively less satisfactory."

II. 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. Self Regulation

"Global E-type systems evolution is feedback regulated."

IV. Conservation of Organizational Stability

"The work rate of an organization evolving an E-type software system tends to be constant over the operational lifetime of that system or phases of that lifetime."

Laws on Software Evolution

V. Conservation of Familiarity

"In general, the incremental growth of E-type systems is constrained by the need to maintain familiarity."

VI. Continuing Growth

"The functional capacity of E-type systems must be continually enhanced to maintain user satisfaction over system lifetime."

VII. 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. Feedback System

"E-type evolution process are multi-level, multi-loop, multi-agent feedback systems."

Evolution of the laws: 1996 vs 2006

I -- Continuing Change	
(1996) E-type systems must be continually adapted else they become progressively less satisfactory.	(2006) An E-type system must be continually adapted or else it becomes progressively less satisfactory in use.
II -- Increasing Complexity	
(1996) As an E-type system evolves its complexity increases unless work is done to maintain or reduce it.	(2006) 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 -- Self Regulation	
(1996) E-type system evolution process is self regulating with distribution of product and process measures close to normal.	(2006) Global E-type system evolution is feedback regulated.
IV -- Conservation of Organisational Stability (invariant work rate)	
(1996) The average effective global activity rate in an evolving E-type system is invariant over product lifetime.	(2006) 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 -- Conservation of Familiarity	
(1996) As an E-type system evolves all associated with it, developers, sales personnel, users, for example, must maintain mastery of its content and behaviour to achieve satisfactory evolution. Excessive growth diminishes that mastery. Hence the average incremental growth remains invariant as the system evolves.	(2006) In general, the incremental growth (growth ratio trend) of E-type systems is constrained by the need to maintain familiarity.
VI -- Continuing Growth	
(1996) The functional content of E-type systems must be continually increased to maintain user satisfaction over their lifetime.	(2006) The functional capability of E-type systems must be continually enhanced to maintain user satisfaction over system lifetime.
VII -- Declining Quality	
(1996) The quality of E-type systems will appear to be declining unless they are rigorously maintained and adapted to operational environment changes.	(2006) 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 -- Feedback System	
(1996) E-type evolution processes constitute multi-level, multi-loop, multi-agent feedback systems and must be treated as such to achieve significant improvement over any reasonable base.	(2006) E-type evolution processes are multi-level, multi-loop, multi-agent feedback systems.

/

EXPERIMENTAL SETUP

Datasets

https://github.com/DAINTINESS-Group/EvolutionDatasets

- Content management Systems
  - MediaWiki, TYPO3, Coppermine, phpBB, OpenCart
- Medical Databases
  - Ensemble, BioSQL
- Scientific
  - ATLAS Trigger

Data sets

Dataset	Versions	Lifetime	Tables Start	Tables End	Attributes Start	Attributes End	Commits per Day	% commits with change	Repository URL
ATLAS Trigger	84	2 Y, 7 M, 2 D	56	73	709	858	0,089	82%	http://atdaq-sw.cern.ch/cgi-bin/viewcvs-atlas.cgi/offline/Trigger/TrigConfiguration/TrigDb/share/sql/combined_schema.sql
BioSQL	46	10 Y, 6 M, 19 D	21	28	74	129	0,012	63%	https://github.com/biosql/biosql/blob/master/sql/biosqldb-mysql.sql
Coppermine	117	8 Y, 6 M, 2 D	8	22	87	169	0,038	50%	http://sourceforge.net/p/coppermine/code/8581/tree/trunk/cpg1.5.x/sql/schema.sql
Ensembl	528	13 Y, 3 M, 15 D	17	75	75	486	0,109	60%	http://cvs.sanger.ac.uk/cgi-bin/viewvc.cgi/ensembl/sql/table.sql?root=ensembl&view=log
MediaWiki	322	8 Y, 10 M, 6 D	17	50	100	318	0,100	59%	https://svn.wikimedia.org/viewvc/mediawiki/trunk/phase3/maintenance/tables.sql?view=log
OpenCart	164	4 Y, 4 M, 3 D	46	114	292	731	0,104	47%	https://github.com/opencart/opencart/blob/master/upload/install/opencart.sql
phpBB	133	6 Y, 7 M, 10 D	61	65	611	565	0,055	82%	https://github.com/phpbb/phpbb3/blob/develop/phpBB/install/schemas/mysql_41_schema.sql
TYPO3	97	8 Y, 11 M, 0 D	10	23	122	414	0,030	76%	https://git.typo3.org/Packages/TYPO3.CMS.git/history/TYPO3_6-0:/t3lib/stddb/tables.sql

Hecate: SQL schema diff viewer

- Parses DDL files
- Creates a model for the parsed SQL elements
- Differentiates two version of the same schema
- Reports on the diff performed with a variety of metrics
- Exports the transitions that occurred in XML format

https://github.com/DAINTINESS-Group/Hecate

Experimental Setup

For each dataset:

- We gathered DDL files from public repos
- We collected all commits of the database at the time of the trunk/master branch
- We ignored all other branches
- We ignored commits of other modules of the project
- Focused on MySQL

RELATED WORK

What is the Status in Literature?

- Schema Evolution
  - Three main case studies [Sjob93], [PVSV12], [CMDZ13]
- Software Evolution
  - Laws on Software Evolution [LeRa06]
  - Case studies on proprietary software [BeLe76, Lehm96, Leh+97,LeRP98, RaLe00, LeRa01, LeRa06]
  - Open Source made things easier [GoTu00], [XiSt05], [WeYL08], [XiCN09]

References

[BeLe76] Laszlo A. Belady, M. M. Lehman. A Model of Large Program Development. IBM Systems Journal 15(3), pp: 225-252, 1976.

[CMDZ13] Carlo A. Curino, Hyun J. Moon, Alin Deutsch and Carlo Zaniolo. Automating the database schema evolution process. The VLDB Journal - The International Journal on Very Large Data Bases 22, no. 1 (2013): 73-98.

[CMTZ08] Carlo A. Curino, Hyun J. Moon, Letizia Tanca, Carlo Zaniolo. Schema evolution in Wikipedia: toward a web information system benchmark. In International Conference on Enterprise Information Systems (ICEIS 2008)

[CuMZ08] Carlo A. Curino, Hyun J. Moon, Carlo Zaniolo. Graceful database schema evolution: the PRISM workbench. Proceedings of the VLDB Endowment 1.1 (2008): p. 761-772.

[FePf96] Norman E. Fenton, Shari Lawrence Pfleeger: Software metrics - a practical and rigorous approach. International Thomson 1996, ISBN 978-1-85032-275-7.

[GoTu00] M. W. Godfrey and Q. Tu. Evolution in Open Source Software: A Case Study. In Proceedings of the 16th IEEE International Conference on Software Maintenance (ICSM), pages 131-143, 2000.

[Leh+97] Meir M. Lehman, Juan F. Ramil, Paul Wernick, Dewayne E. Perry, Wladyslaw M. Turski. Metrics and Laws of Software Evolution - The Nineties View. 4th IEEE International Software Metrics Symposium (METRICS 1997), November 5-7, 1997, Albuquerque, NM, USA, pp: 20.

[Lehm96] M. M. Lehman. Laws of Software Evolution Revisited. 5th European Workshop on Software Process Technology (EWSPT '96), Nancy, France, October 9-11, 1996, pp: 108-124.

References

[LeRa01] Meir M. Lehman and J F Ramil. Software Evolution, STRL Annual Distinguished Lecture, De Montfort Univ., Leicester, 20 Dec. 2001. Available at:
http://www.eis.mdx.ac.uk/staffpages/mml/feast2/papers.html
http://www.eis.mdx.ac.uk/staffpages/mml/feast2/papers/pdf/690c.pdf
http://www.eis.mdx.ac.uk/staffpages/mml/feast2/papers/pdf/jfr103c.pdf

[LeRa06] M.M. Lehman and J. C. Fernandez-Ramil. Software Evolution and Feedback: Theory and Practice, chapter Rules and Tools for Software Evolution Planning and Management. Wiley, 2006.

[LeRP98] Meir M. Lehman, Juan F. Ramil, Dewayne E. Perry. On Evidence Supporting the FEAST Hypothesis and the Laws of Software Evolution. 5th IEEE International Software Metrics Symposium (METRICS 1998), March 20-21, 1998, Bethesda, Maryland, USA, pp: 84-88.

[Press00] Roger Pressman. Software Engineering: A Practitioner's Approach: European Adaption. McGraw-Hill, 5th edition, April 2000

[PVSV12] George Papastefanatos, Panos Vassiliadis, Alkis Simitsis, and Yannis Vassiliou. "Metrics for the prediction of evolution impact in ETL ecosystems: A case study." Journal on Data Semantics 1, no. 2 (2012): 75-97.

[RaLe00] Juan F. Ramil, M. M. Lehman. Effort estimation from change records of evolving software (poster). Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000) Limerick Ireland, June 4-11, 2000, pp: 777

References

[Sjøb91] Dag Sjøberg, The Thesaurus – A Tool for Meta Data Management, Technical Report FIDE/91/6, ESPRIT Basic Research Action, Project Number 3070---FIDE, February 1991.

[Sjøb93] Dag Sjøberg. "Quantifying schema evolution." Information and Software Technology 35.1 (1993): 35-44.

[Skou10] Ioannis Skoulis, Hecate: SQL schema diff viewer. Bachelor Thesis, Department of Computer Science, University of Ioannina, 2010

[SEBK] IEEE. Software Engineering Body of Knowledge. IEEE – 2012 SWEBOK Guide V3 – Alpha Version. Available at http://www.computer.org/portal/web/swebok Retrieved at 13 September 2013.

[XiCN09] G. Xie, J. Chen, and I. Neamtiu. 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, pages 51-60, 2009.

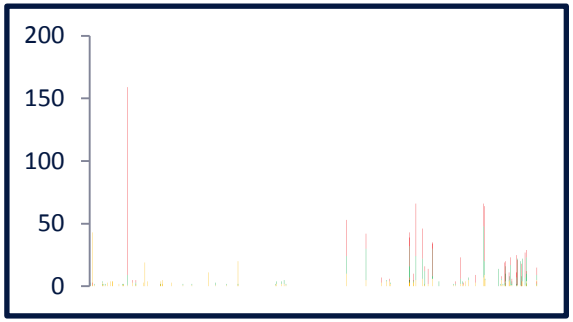
[XiStr05] Z. Xing and E. Stroulia. Analyzing the evolutionary history of the logical design of object-oriented software. IEEE Trans. Software Eng., 31(10):850-868, 2005.

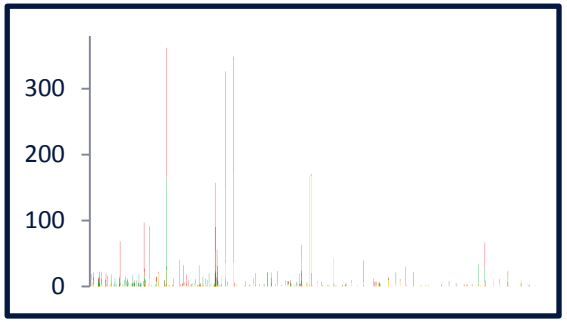
[WeYL08] M. Wermelinger, Y. Yu, and A. Lozano. Design Principles in Architectural Evolution: A Case Study. In 24th International Conference on Software Maintenance (ICSM), pages 396-405, 2008.

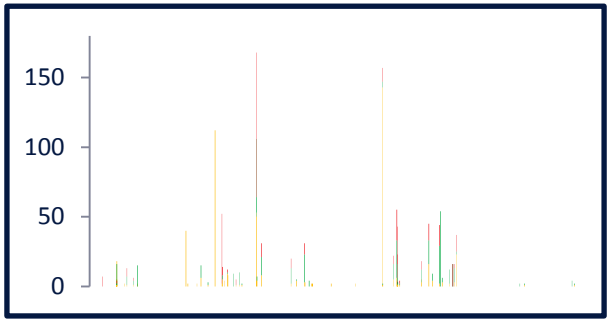
...

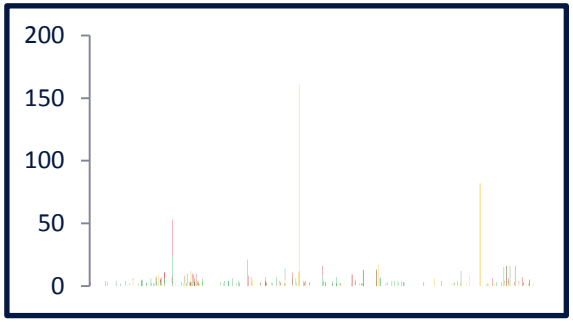
SEVERAL CHARTS

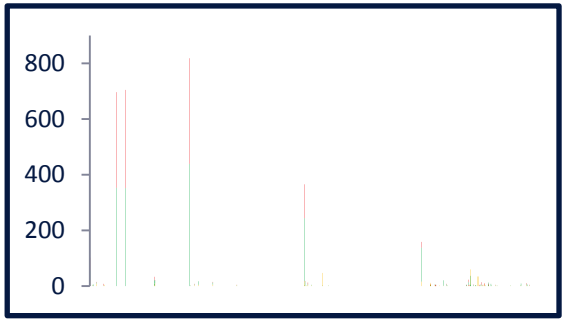
Change over time

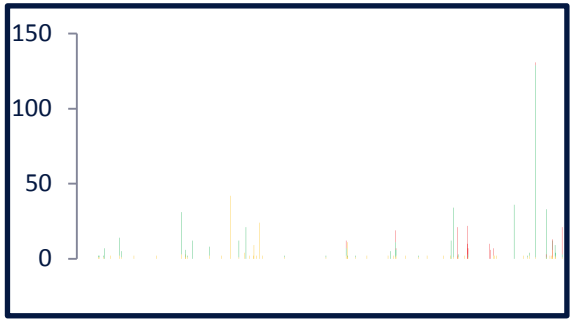


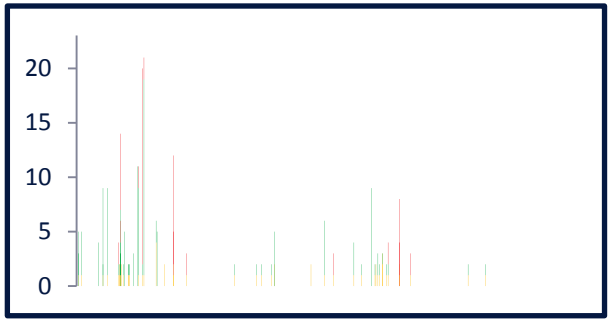




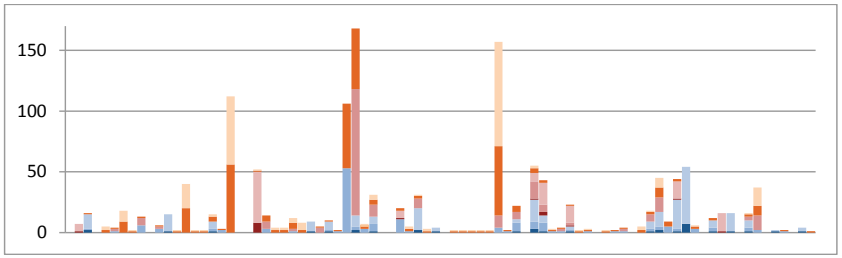




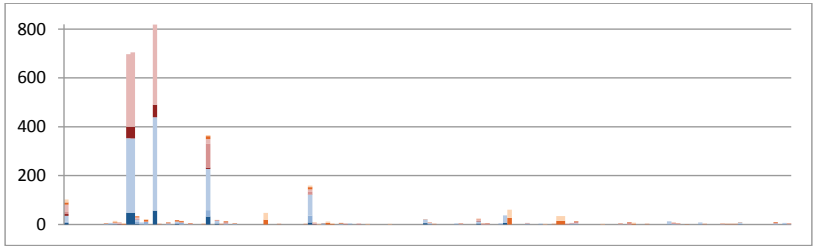




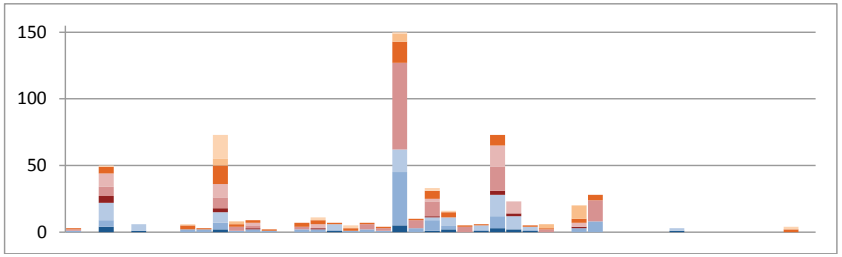
Change over version



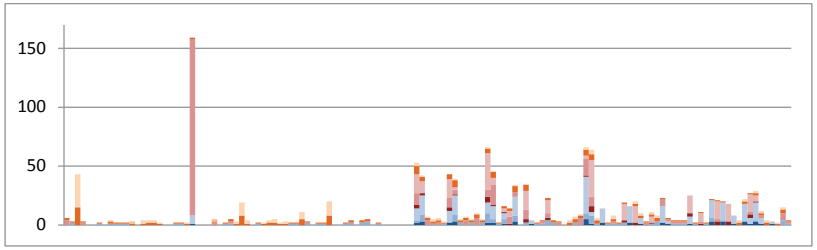
ATLAS Trigger



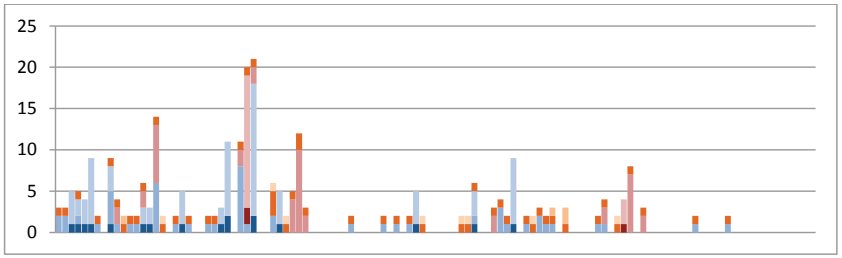
OpenCart



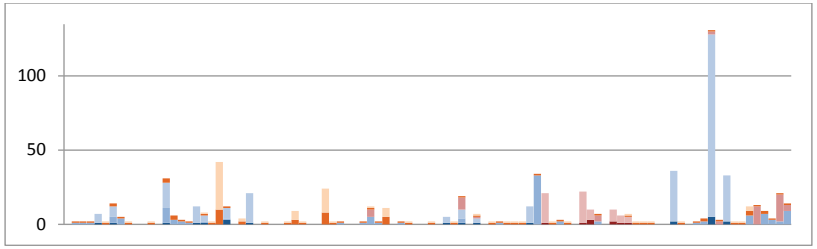
BioSQL



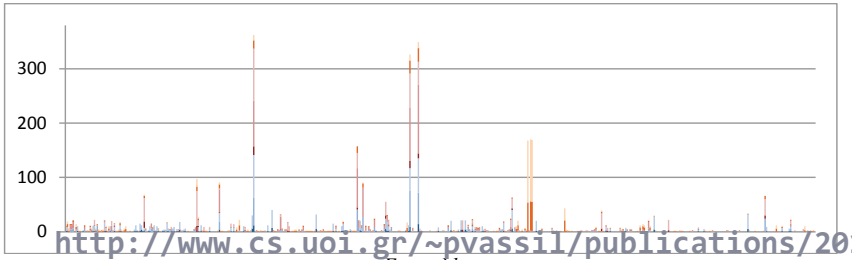
phpBB



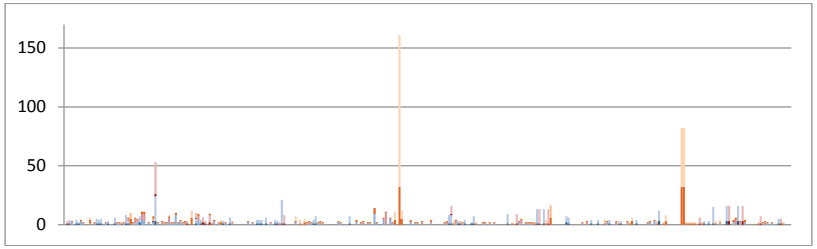
Coppermine



TYPO3

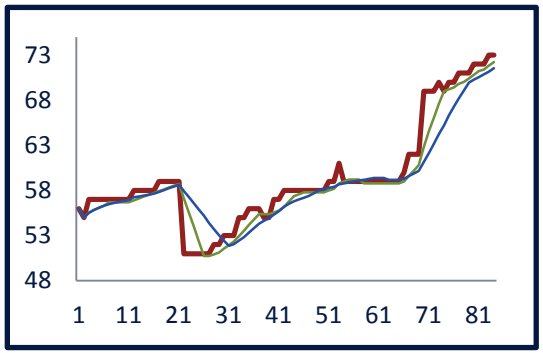


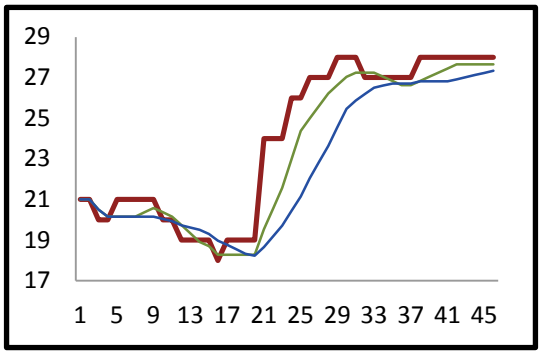
Ensembl

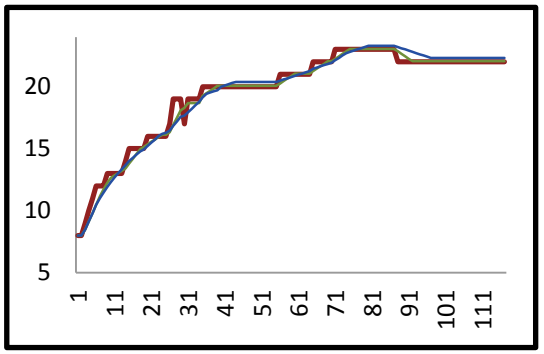


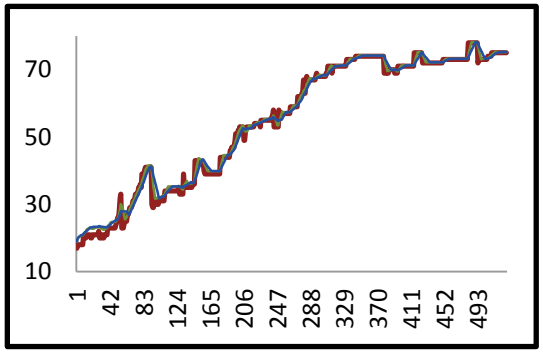
MediaWiki

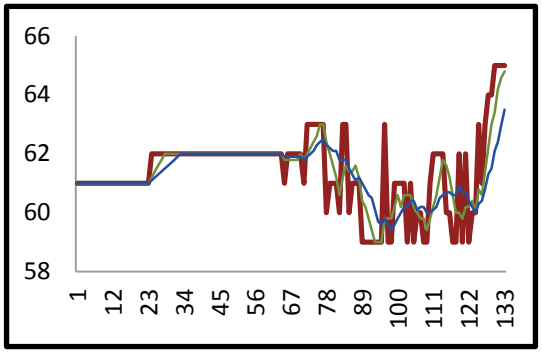
Estimated Size

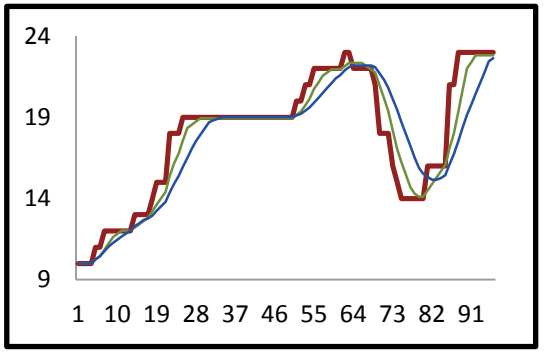


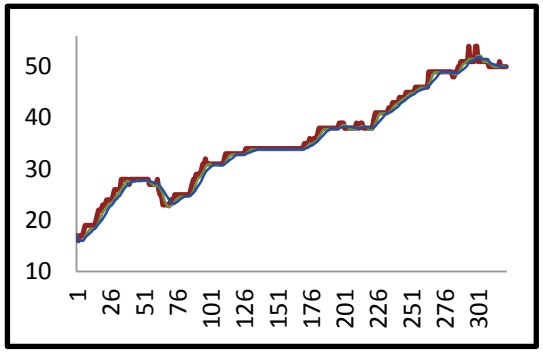


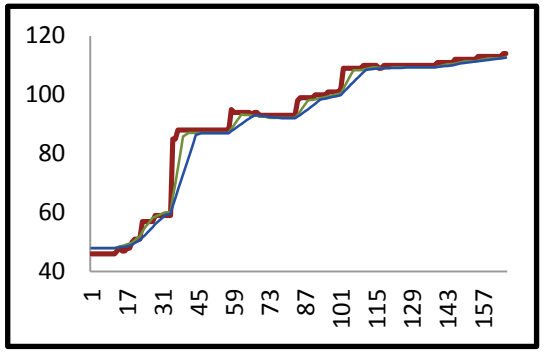






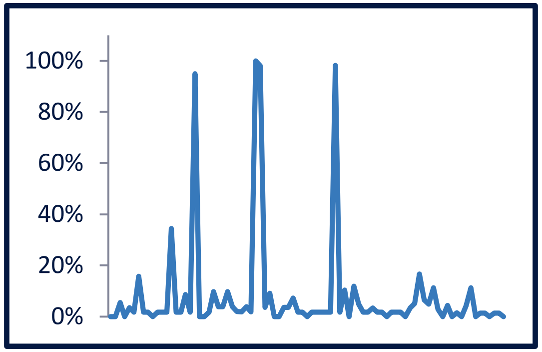


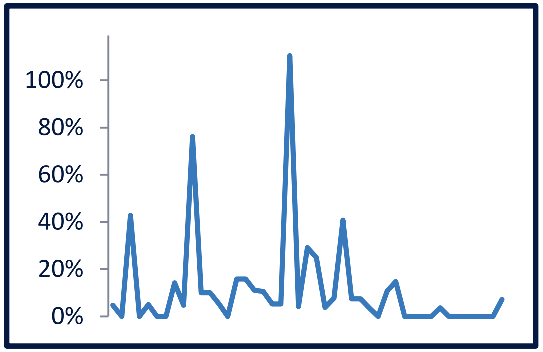


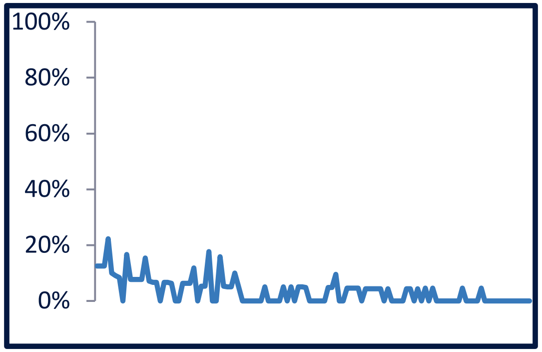


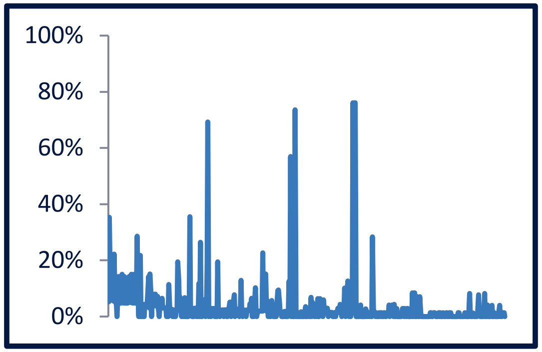
Actual size Est - last 5 last 1 Est - last 10 last 1

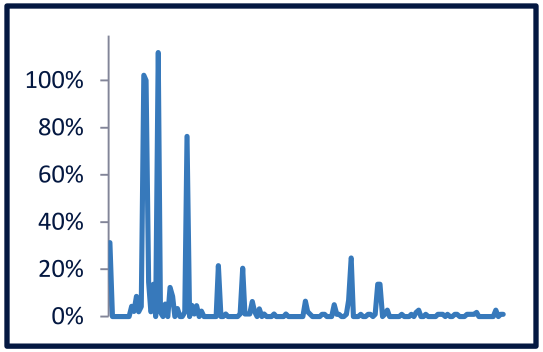
Maintenance Rate

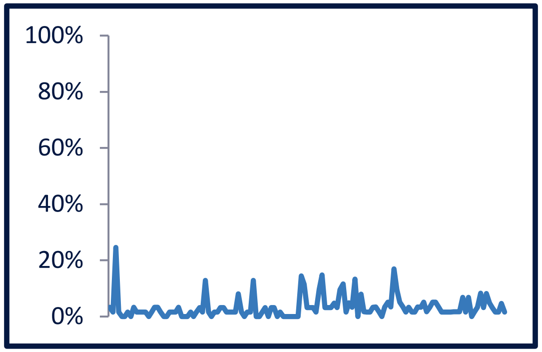


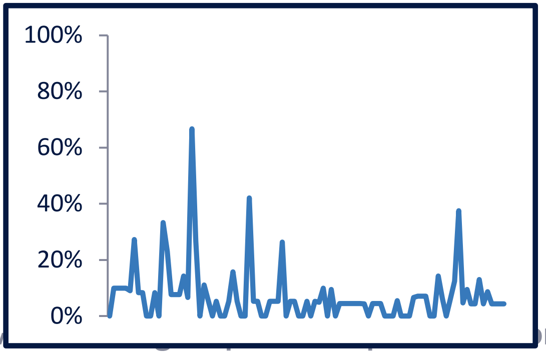


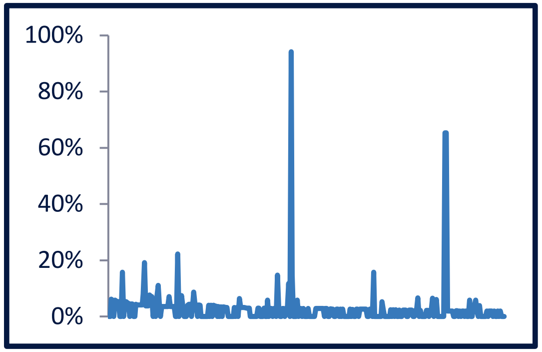




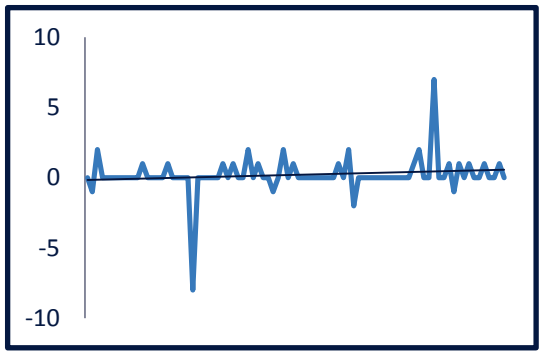


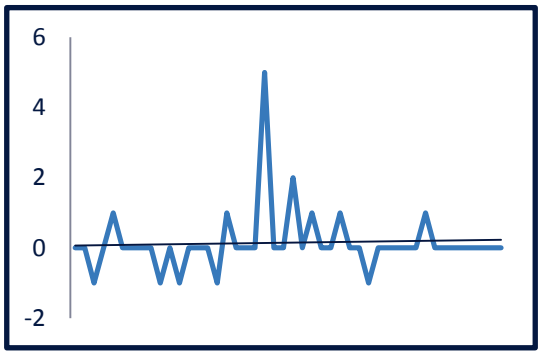


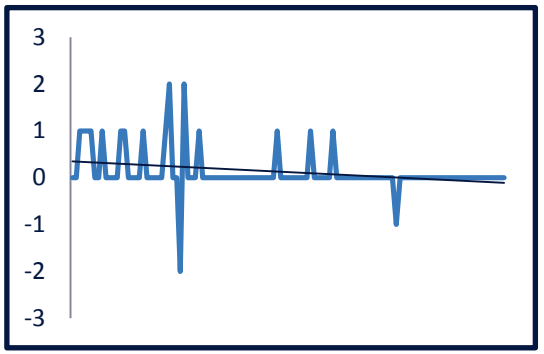


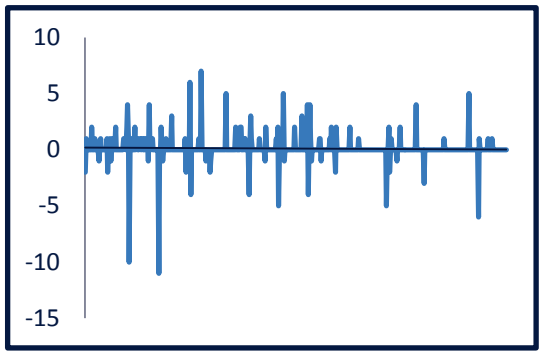


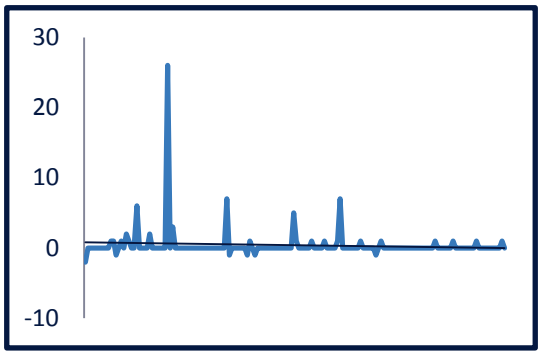
Schema Growth

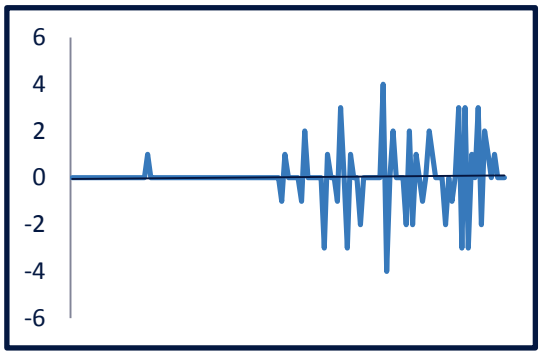


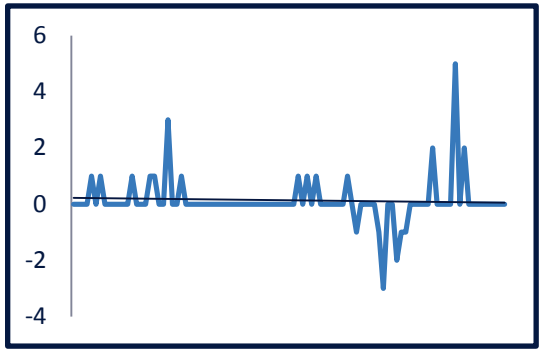


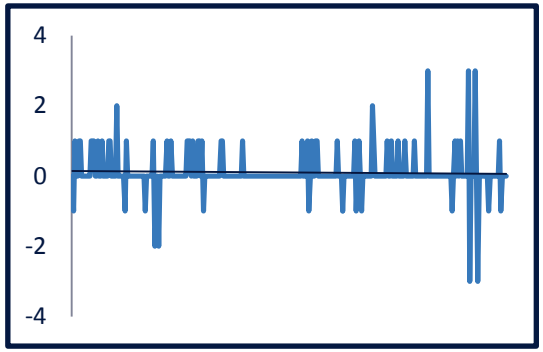












Schema Size (relations)