Keep Calm & Wait for the Spike! Insights on the Evolution of Amazon Services

Apostolos
Zarras

Panos
Vassiliadis

Ioannis
Dinos

Department of Computer Science &
Engineering
University of Ioannina - Greece
www.cs.uoi.gr

This work was supported from the European Community's FP7/2007-2013 under grant agreement number 257178 (project CHOReOS).

Fundamental question

What are the **patterns** of

web service evolution

from the viewpoint of an

external observer?

Developer concerns

A developer of **client applications of web services** wants to know
- should I use this service?
- will the service evolve and how?
- how will this impact my application?

```
// Create an Amazon SQS queue
CreateQueueRequest createQueueRequest = new CreateQueueRequest("MyQueue");
String myQueueUrl = sqs.createQueue(createQueueRequest).getQueueUrl();
...
//get all the msg's from the queue
ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRequest(myQueueUrl);
List<Message> messages = sqs.receiveMessage(receiveMessageRequest).getMessages();
...
```

Developer concerns

web service evolution

```
// Create an Amazon SQS queue
CreateQueueRequest createQueueRequest = new CreateQueueRequest("MyQueue");
String myQueueUrl = sqs.createQueue(createQueueRequest).getQueueUrl();
...
//get all the msg's from the queue
ReceiveMessageRequest receiveMessageRequest = new ReceiveMessageRequest(myQueueUrl);
List<Message> messages = sqs.receiveMessage(receiveMessageRequest).getMessages();
...
```

data type change?

signature change?

Deprecated operations?
New alternatives?
Renamings?

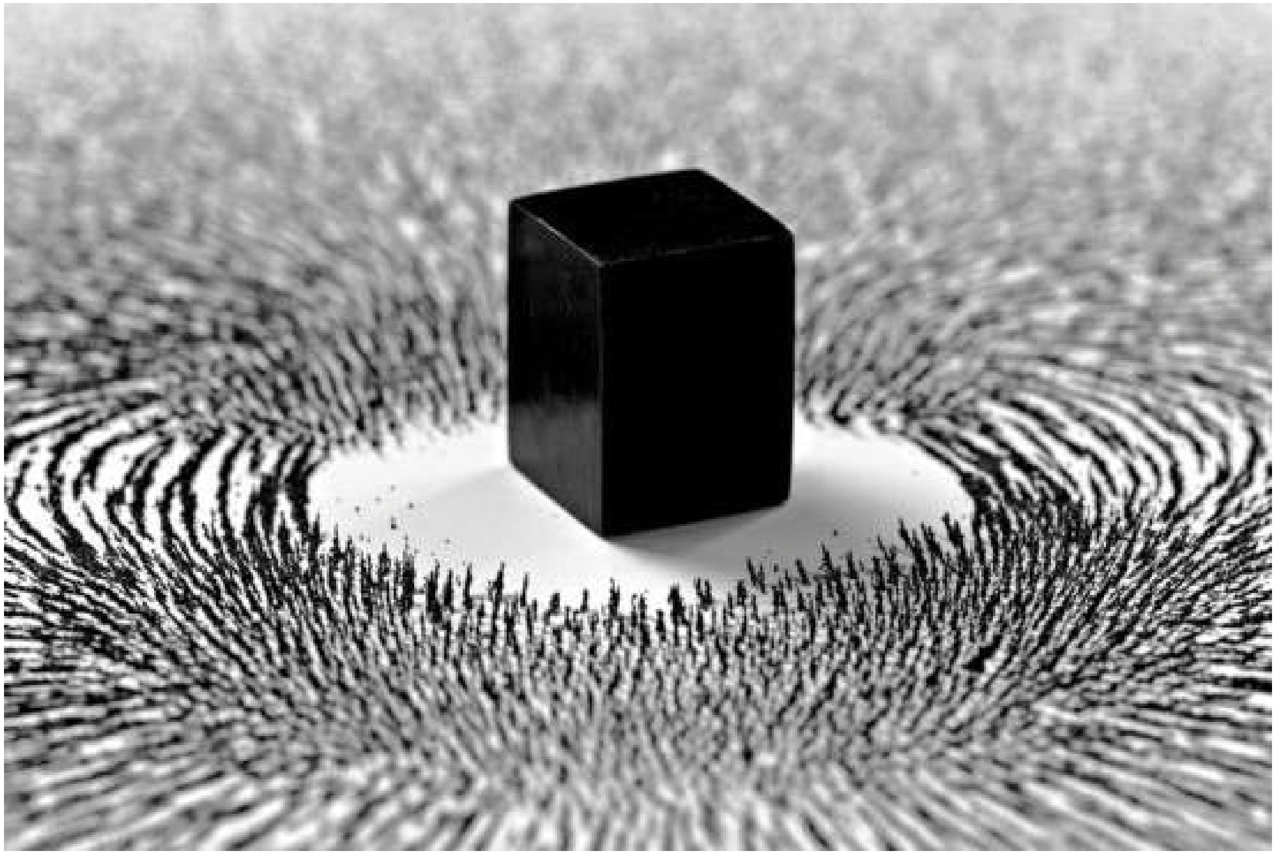
External observer

A developer of client applications of web services is not able to know the internals of the web service provider

© legominifiguresinthewild.blogspot.com

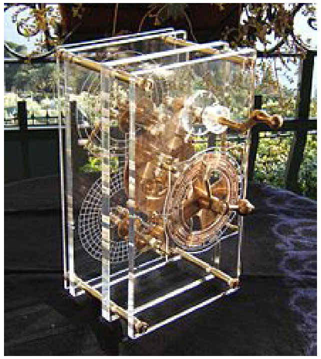
The **external observation of the history of changes** is the only information she has…

Evolution of Dependency Magnets



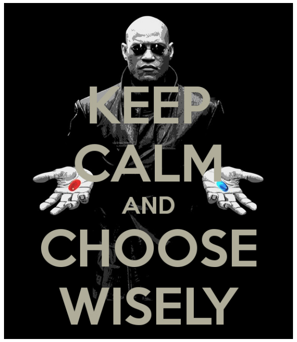
Contribution

We present **patterns** of **web service evolution**
from the viewpoint of an **external observer**



by studying the evolution of **AWS** services based on **Lehman’s laws of software evolution**

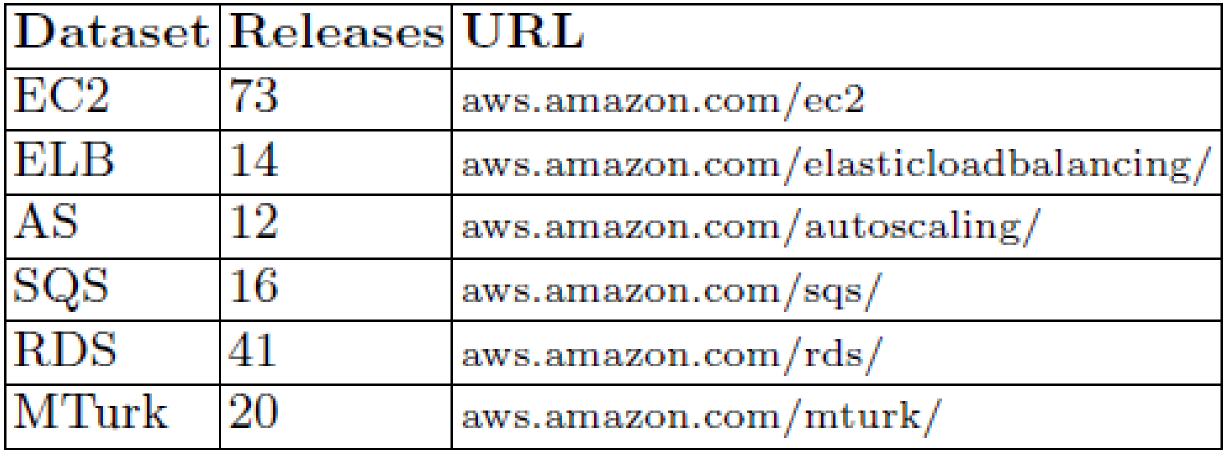
and discuss **recommendations** for assessing the future behavior of a w/s



Roadmap

- Method
- Laws
- Recommendations

Setup & History Extraction



Dataset	Releases	URL
EC2	73	aws.amazon.com/ec2
ELB	14	aws.amazon.com/elasticloadbalancing/
AS	12	aws.amazon.com/autoscaling/
SQS	16	aws.amazon.com/sqs/
RDS	41	aws.amazon.com/rds/
MTurk	20	aws.amazon.com/mturk/

Membrane SOA Model: www.membrane-soa.org/

What did we measure

Service Evolution History:

$$H_s = \{r_1^s, r_2^s, \ldots, r_N^s\}$$

Service Release:

$$r_i^s = (ID, date, Size, Change)$$

Size: *Number of Interfaces, Operations, XML Types*
Change: *Additions, Deletions, and Updates* of Operations between subsequent rel.*

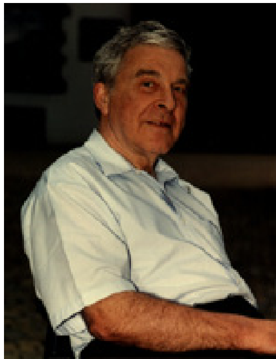
**Operation Updates:* (a) changes in their own structure (e.g., attributes, annotations), or (b) updates in the structure of their constituents (e.g., messages, XML types).

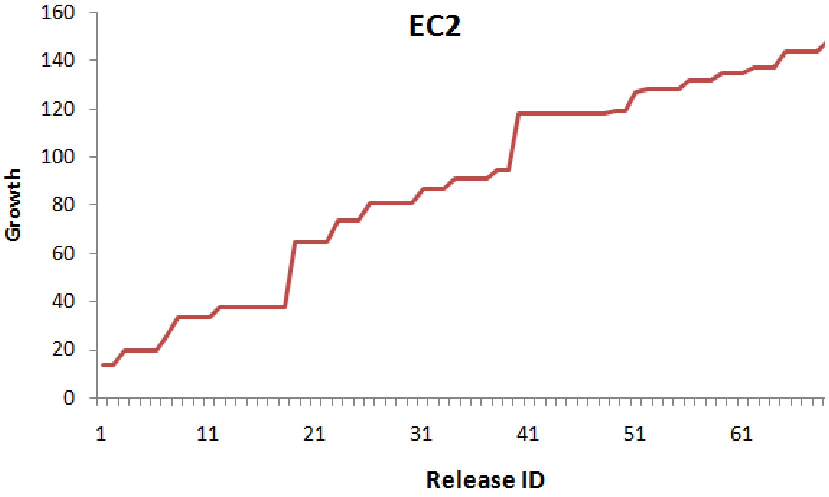
Roadmap

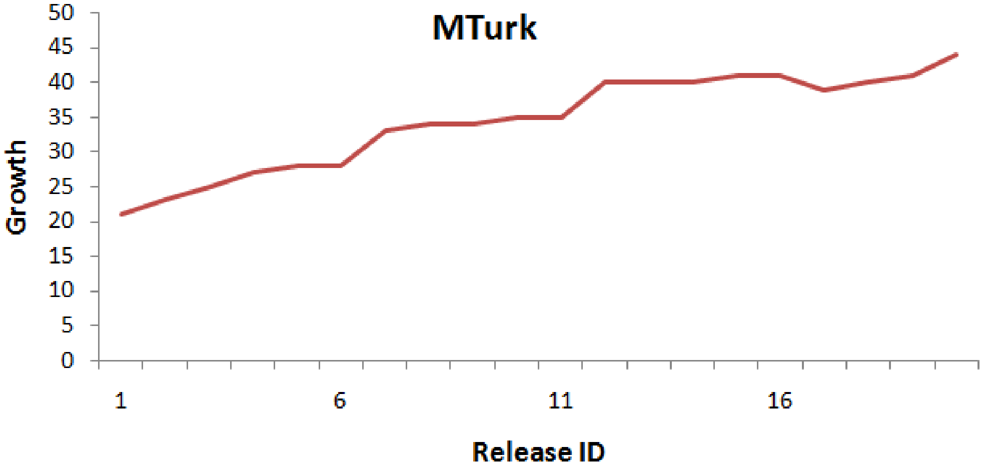
- Method
- Laws
- Recommendations

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).

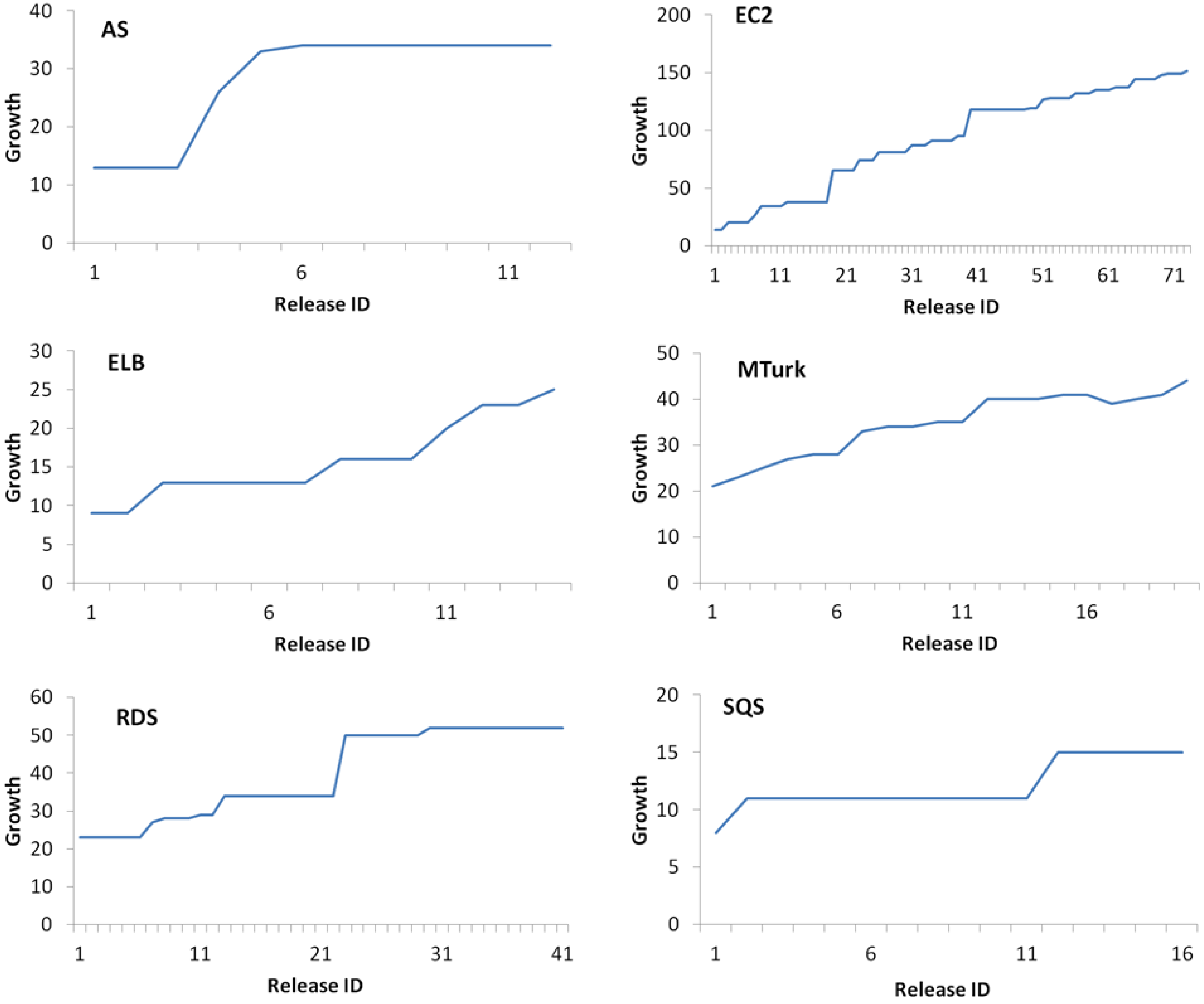




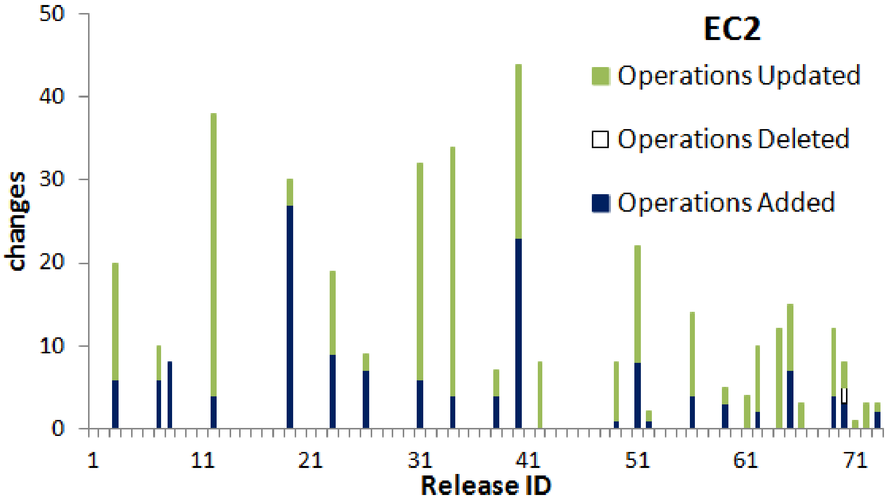


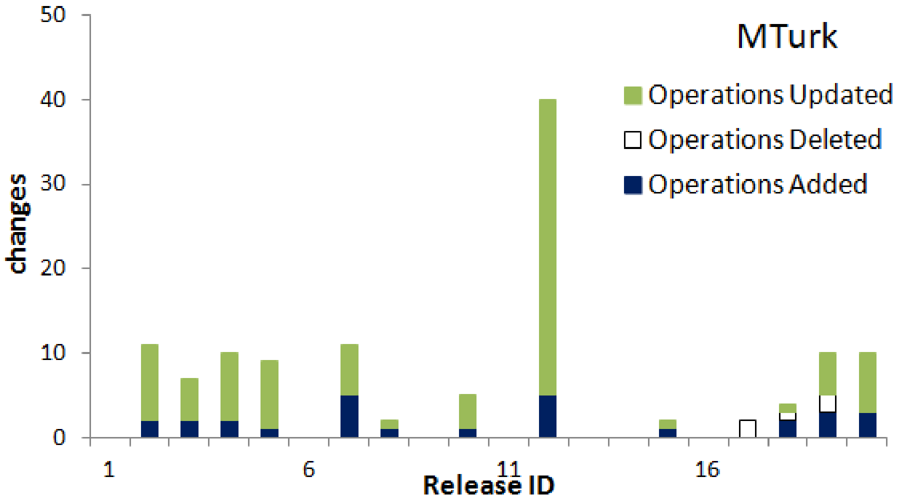
6th Law	**The functional capability of E-type systems must be continually enhanced to maintain user satisfaction over system lifetime**
Criteria	A continuous increasing trend in the growth of the system. We measure the growth of the service as the number of provided operations: $G\left(r_i^s\right) = r_i^s.Size[Opers]$
Validity	**Holds**
Properties	▪ There is an **increasing trend** in the **growth** of the service operations ▪ However, the **increase** is **not continuous**

Growth



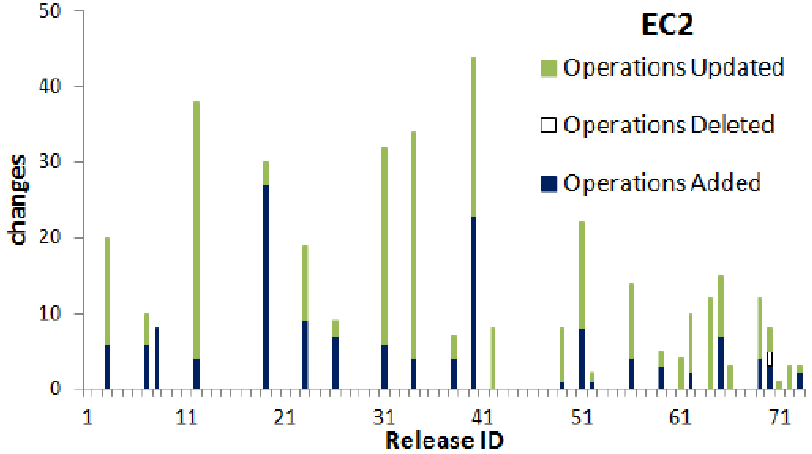
AS
Growth
0
10
20
30
40
1
6
11
Release ID
EC2
Growth
0
50
100
150
200
1
11
21
31
41
51
61
71
Release ID
ELB
Growth
0
5
10
15
20
25
30
1
6
11
Release ID
MTurk
Growth
0
10
20
30
40
50
1
6
11
16
Release ID
RDS
Growth
0
10
20
30
40
50
60
1
11
21
31
41
Release ID
SQS
Growth
0
5
10
15
20
1
6
11
16
Release ID

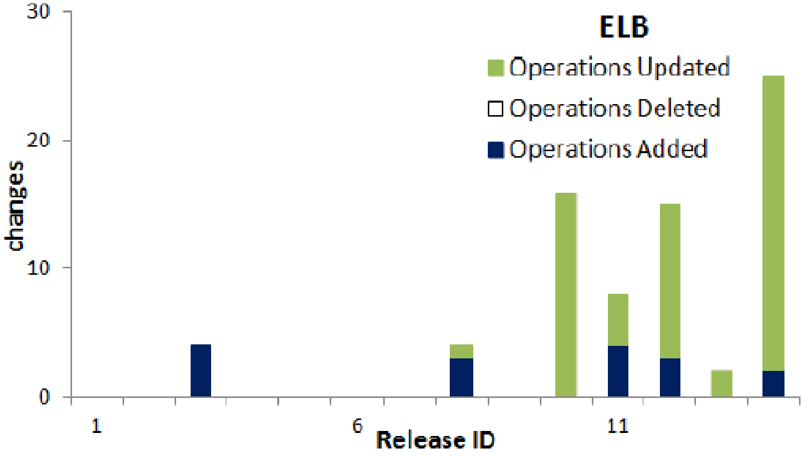


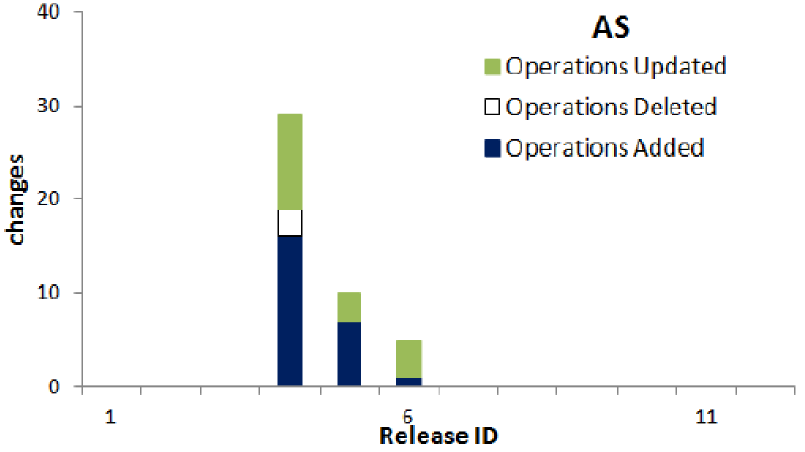


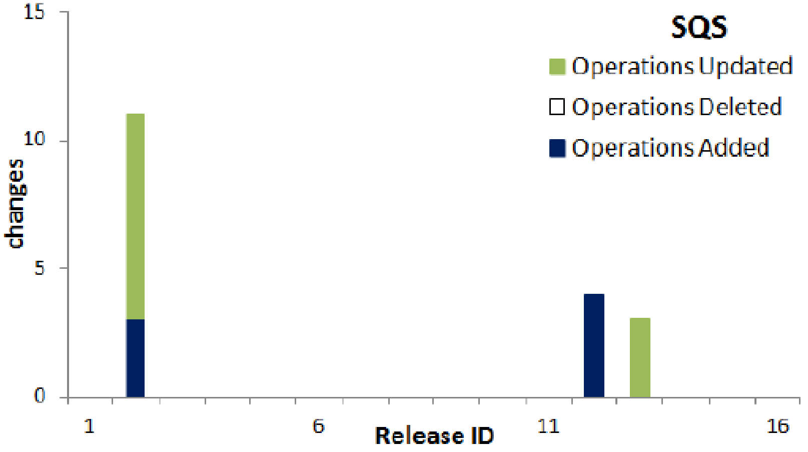
1st Law	**An E-type system must be continually adapted, or else it becomes less satisfactory in use**
Criteria	Heartbeat of changes during the service evolution history
Validity	**Holds**
Properties	▪ Changes are mostly **internal** and involve the structure of the exported operations less frequently ▪ When they do, they involve **mostly updates** and **additions**

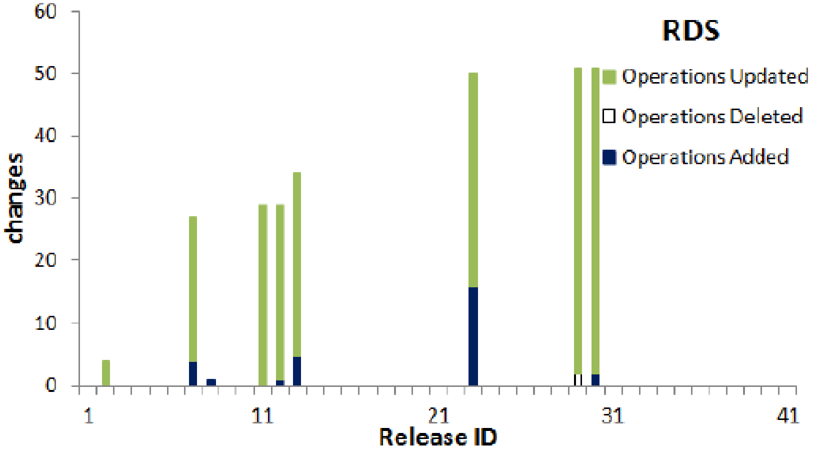
Heartbeat

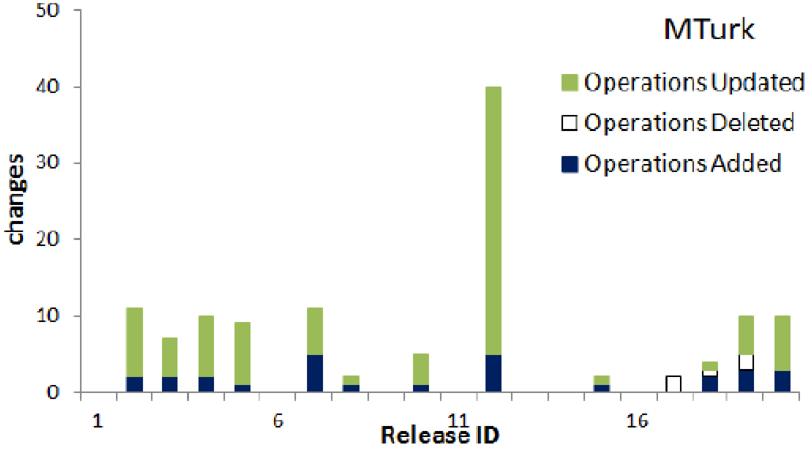

EC2
Operations Updated
Operations Deleted
Operations Added
changes
0
10
20
30
40
50
1
11
21
31
41
51
61
71
Release ID

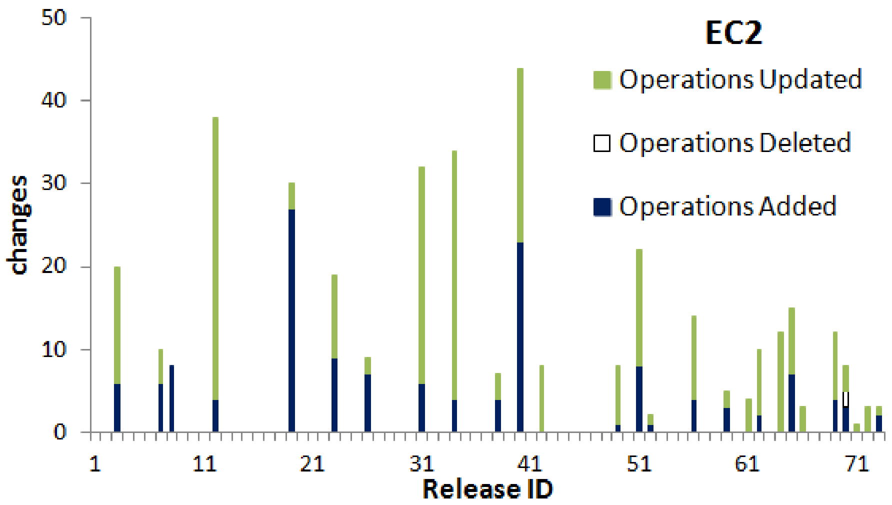

ELB
Operations Updated
Operations Deleted
Operations Added
changes
0
10
20
30
1
6
11
Release ID

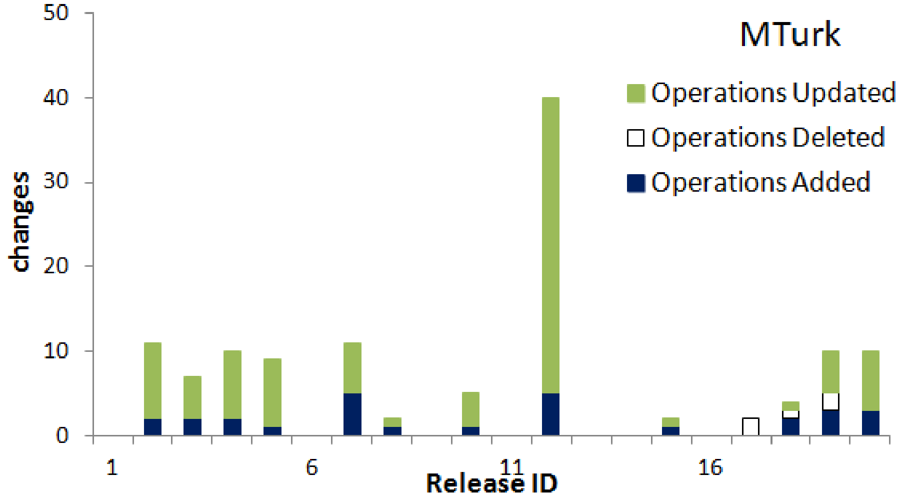

AS
Operations Updated
Operations Deleted
Operations Added
changes
0
10
20
30
40
1
6
11
Release ID


SQS
Operations Updated
Operations Deleted
Operations Added
changes
0
5
10
15
1
6
11
16
Release ID

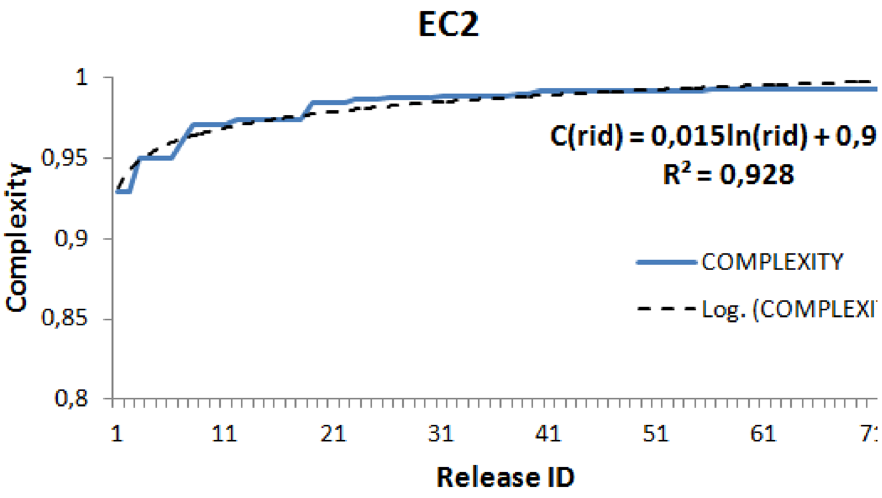

RDS
Operations Updated
Operations Deleted
Operations Added
changes
0
10
20
30
40
50
60
1
11
21
31
41
Release ID

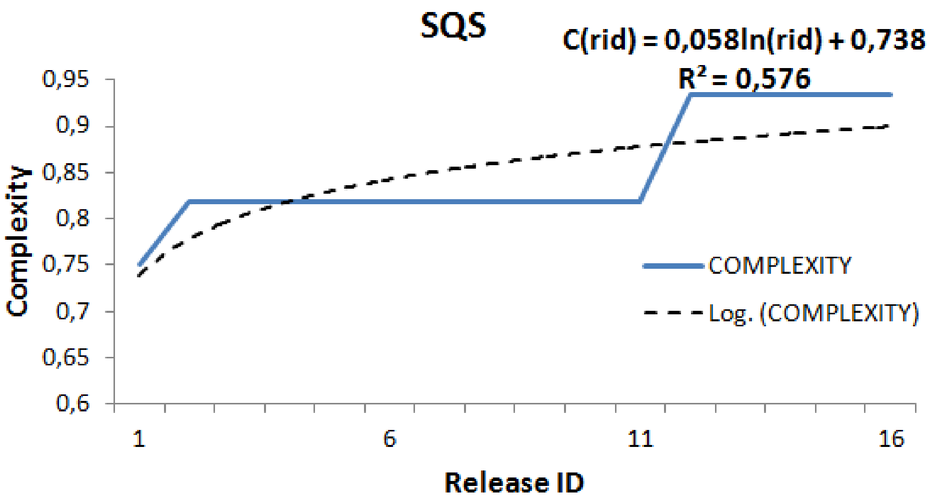

MTurk
Operations Updated
Operations Deleted
Operations Added
changes
0
10
20
30
40
50
1
6
11
16
Release ID





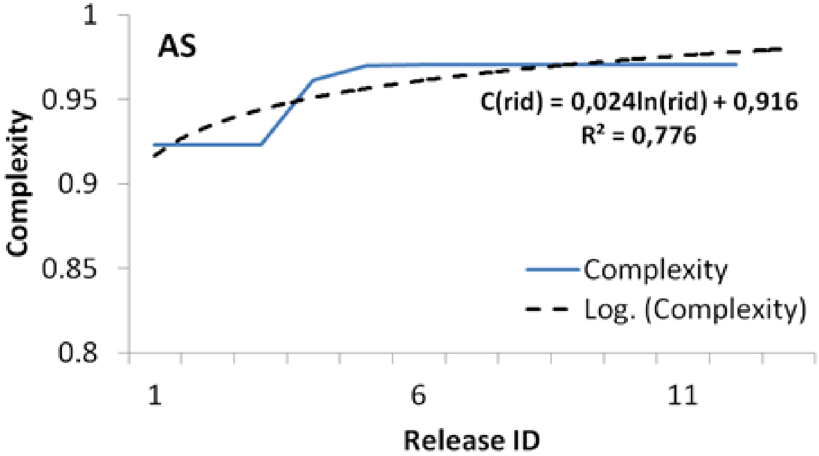
4th Law	**The work rate of an organization evolving an E-type system tends to be constant over the operational lifetime of that system, or phases of that lifetime**
Criteria	Indicators like personnel time dedicated to software evolution is typically unavailable and inaccurate. An approximation suggested by Lehman et al. is number of changes performed per release
Validity	**Inconclusive**
Properties	▪ The amount of **changes** is **not invariant**; also, it is not possible to speak about phases in which the amount of changes remains constant. ▪ On the other hand, it is **not possible to know** precisely the **work done behind the scenes**

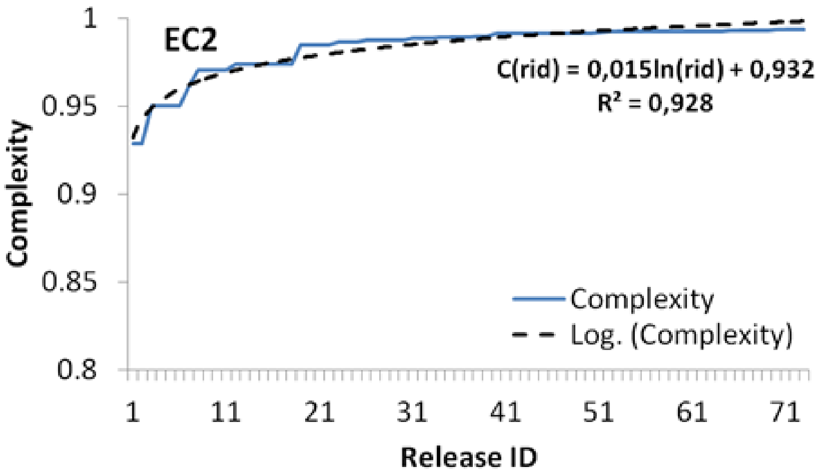


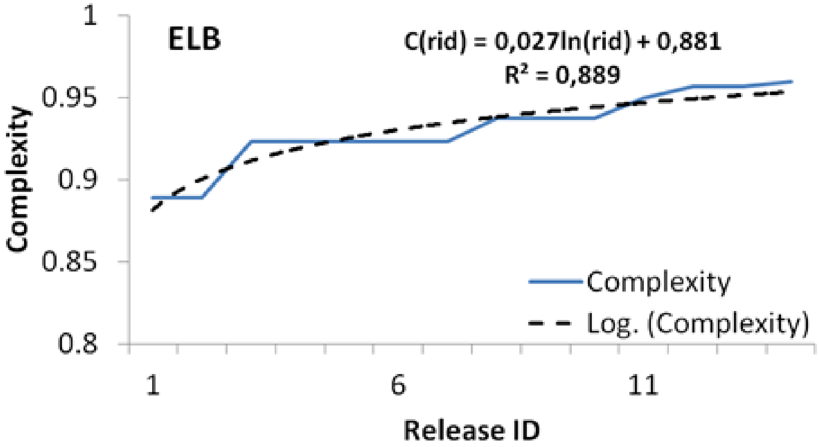


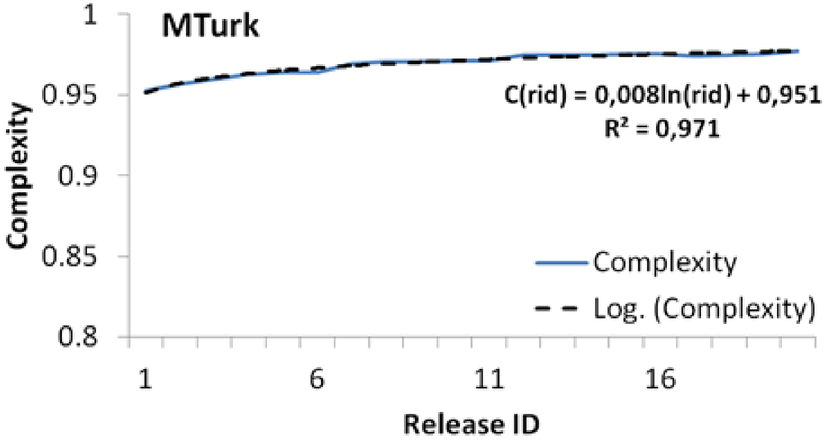
2nd Law	**As an E-type system is changed its complexity increases and becomes more difficult to evolve, unless work is done to maintain or reduce the complexity**
Criteria	Complement of the ratio of the provided interfaces to the operations: $$C\left(r_i^s\right)=1-\frac{r_i^s.Size[Interfaces]}{r_i^s.Size[Opers]}$$
Validity	**Holds**
Properties	▪ Interface complexity, is **high**; it **smoothly increases** over time; usually the increase is **logarithmic**.

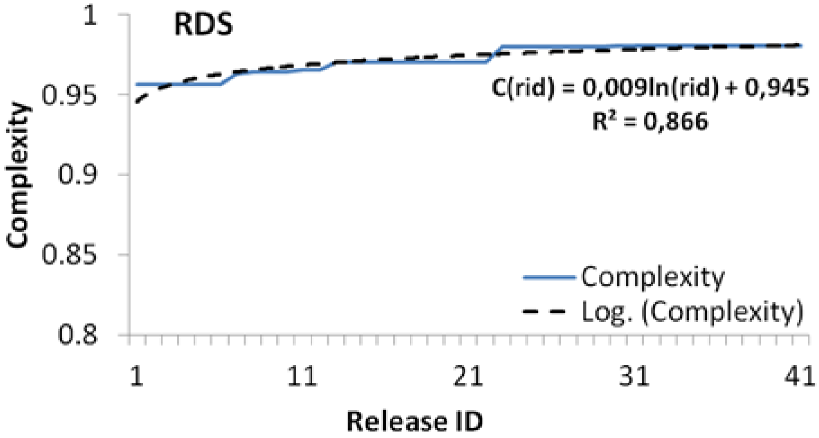
Complexity

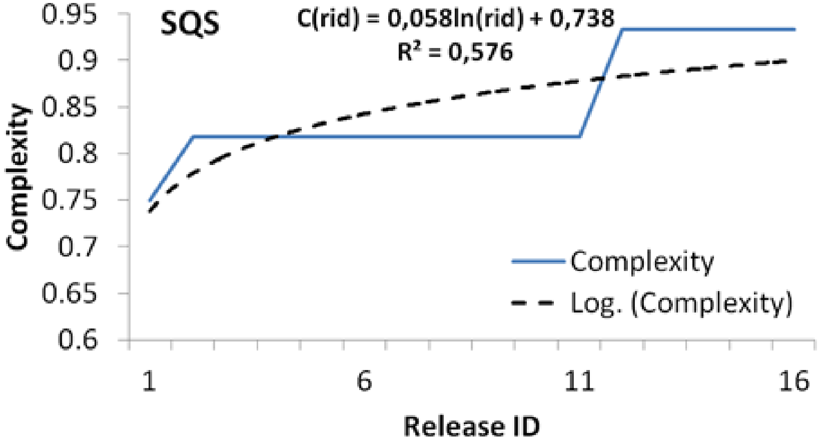

AS
C(rid) = 0,024ln(rid) + 0,916
R² = 0,776
Complexity
1
0.95
0.9
0.85
0.8
1
6
11
Release ID
Complexity
Log. (Complexity)

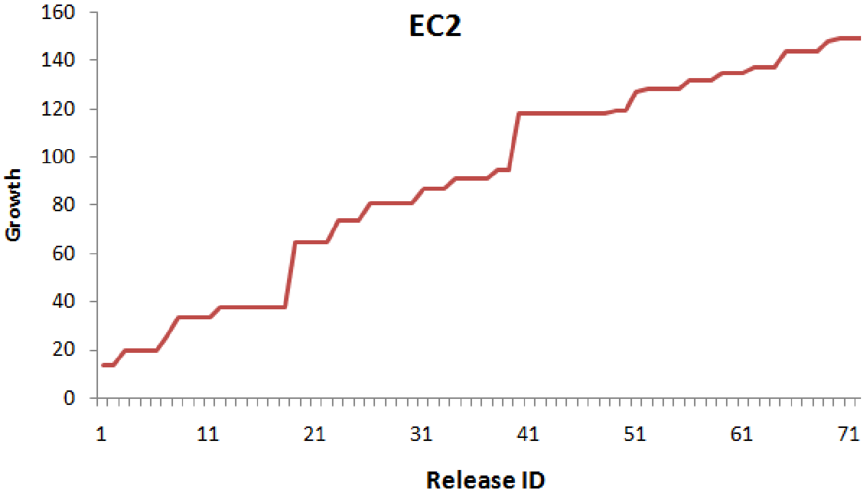

EC2
C(rid) = 0,015ln(rid) + 0,932
R² = 0,928
Complexity
1
0.95
0.9
0.85
0.8
1
11
21
31
41
51
61
71
Release ID
Complexity
Log. (Complexity)

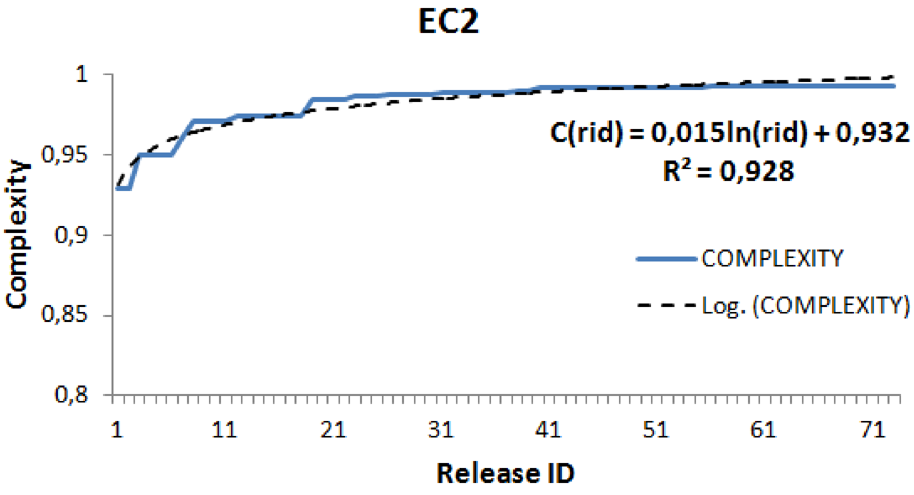

ELB
C(rid) = 0,027ln(rid) + 0,881
R² = 0,889
Complexity
1
0.95
0.9
0.85
0.8
1
6
11
Release ID
Complexity
Log. (Complexity)


MTurk
C(rid) = 0,008ln(rid) + 0,951
R² = 0,971
Complexity
1
0.95
0.9
0.85
0.8
1
6
11
16
Release ID
Complexity
Log. (Complexity)

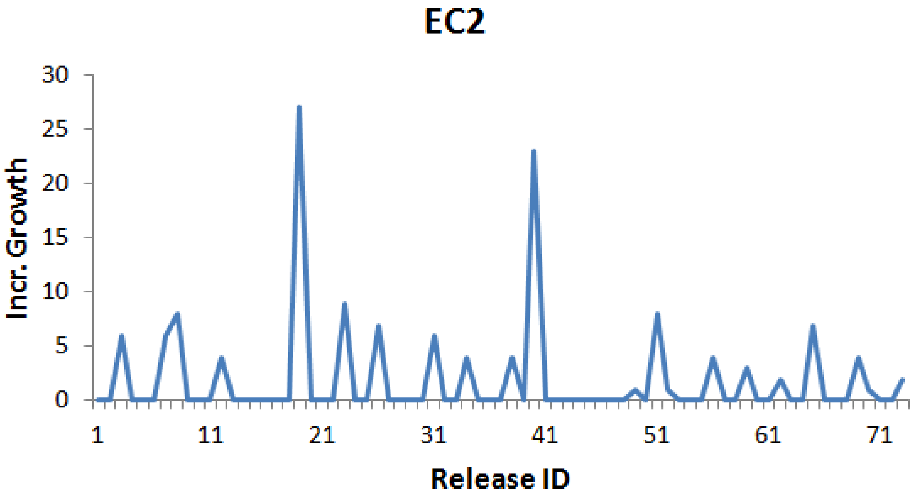

RDS
C(rid) = 0,009ln(rid) + 0,945
R² = 0,866
Complexity
1
0.95
0.9
0.85
0.8
1
11
21
31
41
Release ID
Complexity
Log. (Complexity)

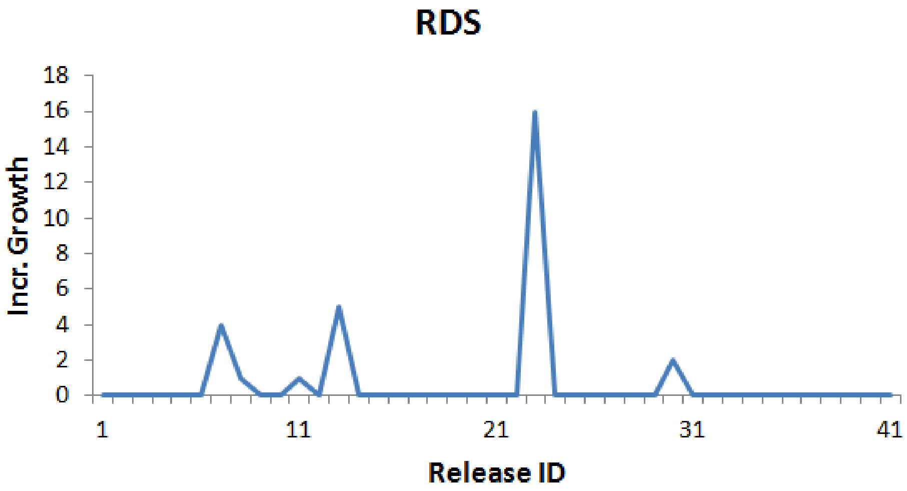

SQS
C(rid) = 0,058ln(rid) + 0,738
R² = 0,576
Complexity
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
1
6
11
16
Release ID
Complexity
Log. (Complexity)





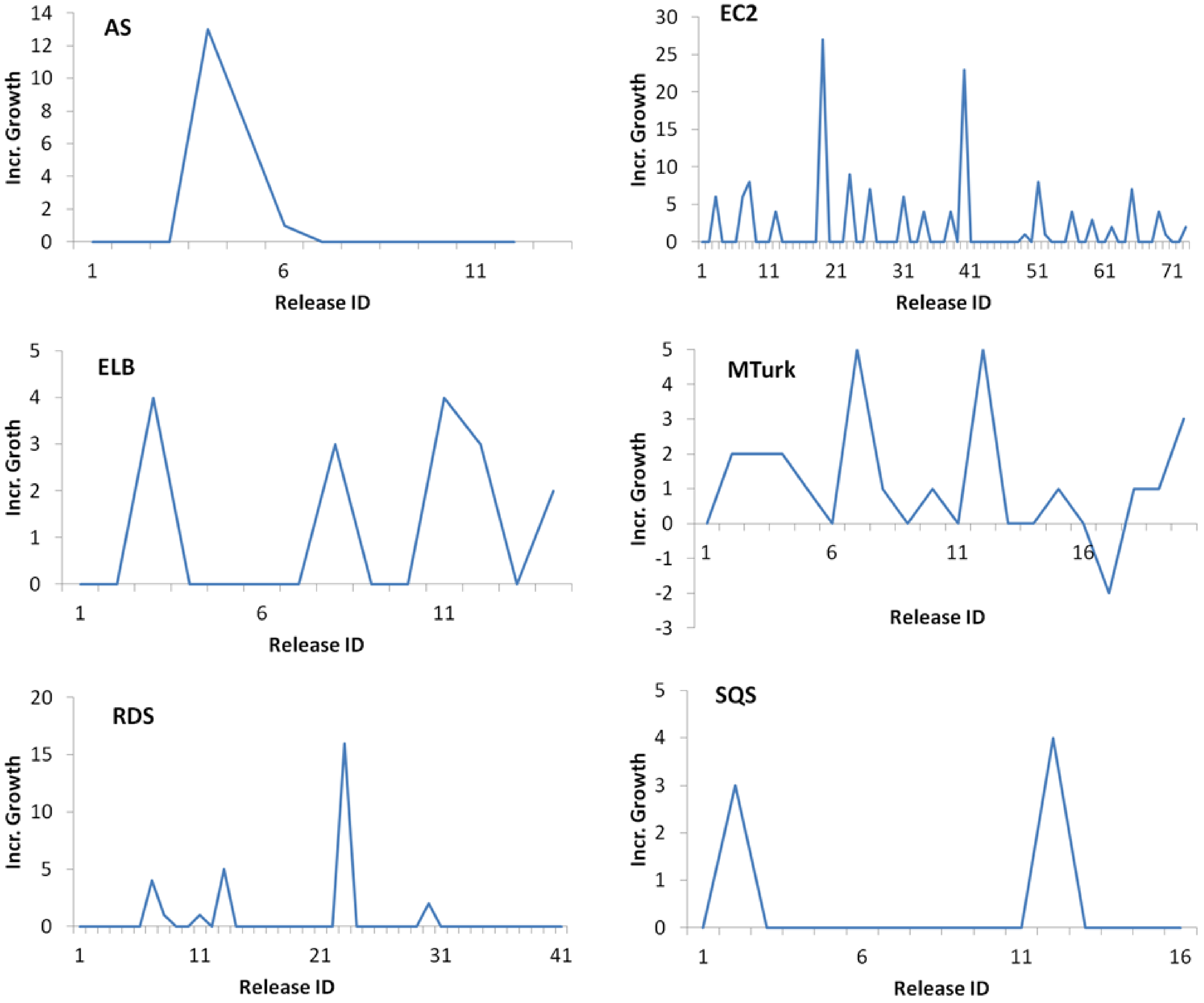
7th Law	**The quality of an E-type system will appear to be declining, unless rigorously maintained and adapted to operational environment changes**
Criteria	▪ The assessment is problematic because the required data are typically not publicly available . ▪ Lehman et al. discuss a more general strategy based on induction: **quality decline**, follows from **functional growth** and **increasing complexity**
Validity	**Inconclusive**
Properties	▪ By following the general strategy suggested by Lehman et al. we have **indications** that the seventh law **holds** for the examined services. ▪ However, there are **no concrete qualitative evaluations**

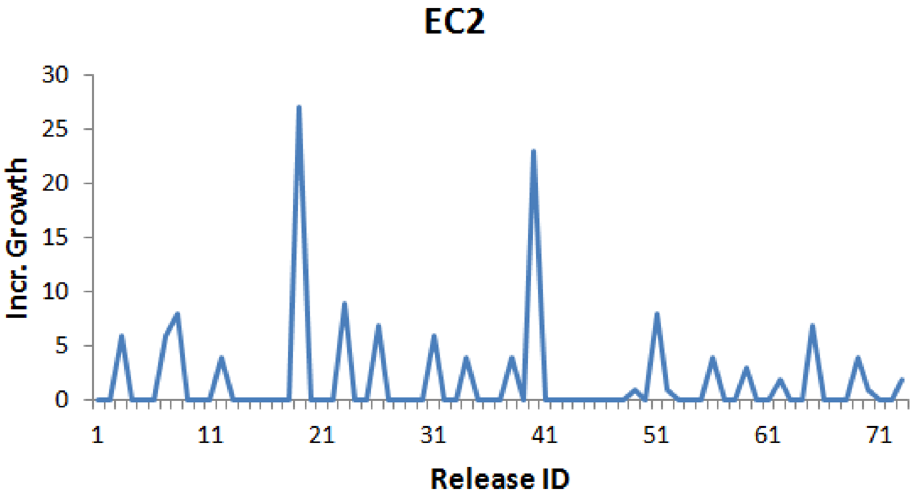


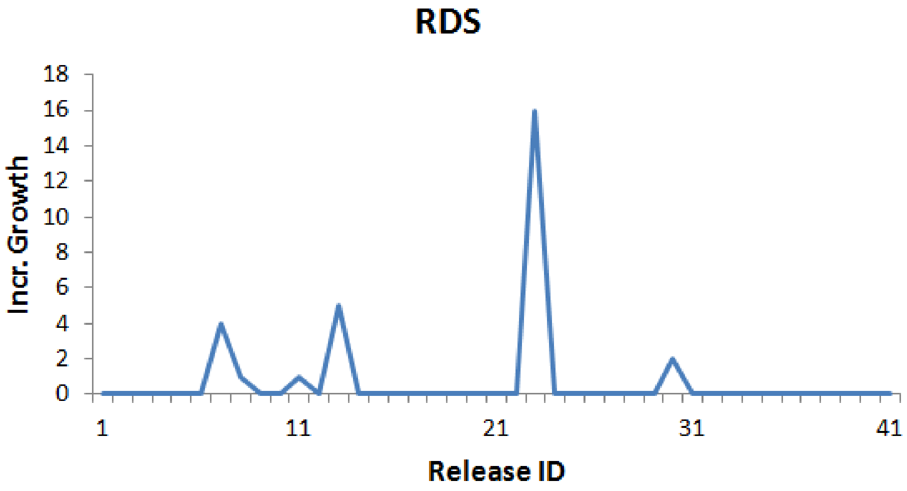


3nd Law	**Global E-type system evolution is feedback regulated**
Criteria	Demonstrated by patterns in the incremental growth: $IG\left(r^s_{i+1}, r^s_i\right) = r^s_{i+1}.Size[Opers] - r^s_i.Size[Opers]$
Validity	**Holds**
Properties	▪ Two **patterns** of incremental growth: **spikes** and **calmness periods**, which together indicate the existence of a stabilization mechanism. ▪ Calmness periods involve **internal improvements** on documentation, bug fixing, security patching and extension of programming facilities.

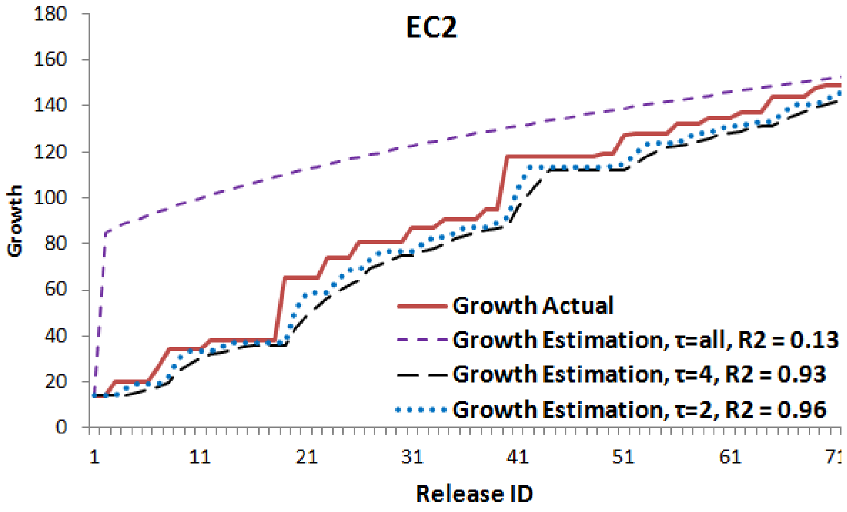
Incr. Growth

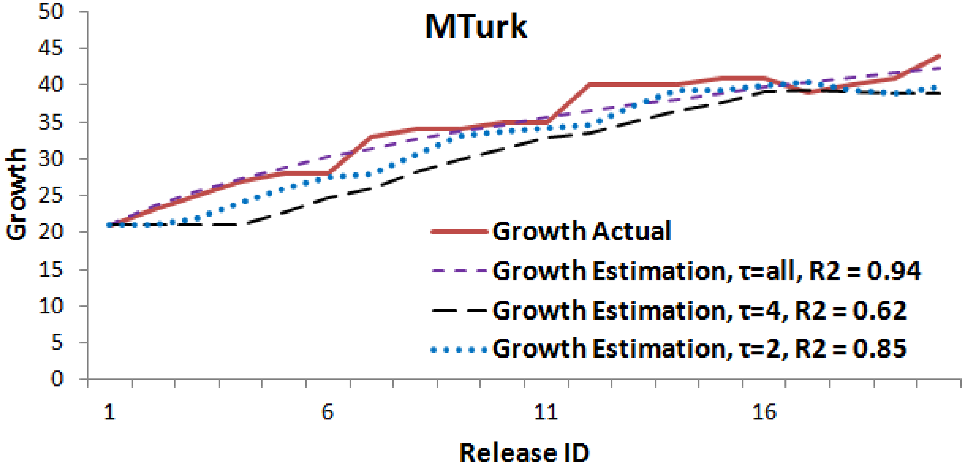

AS
Incr. Growth
0
2
4
6
8
10
12
14
1
6
11
Release ID
EC2
Incr. Growth
0
5
10
15
20
25
30
1
11
21
31
41
51
61
71
Release ID
ELB
Incr. Groth
0
1
2
3
4
5
1
6
11
Release ID
MTurk
Incr. Growth
-3
-2
-1
0
1
2
3
4
5
1
6
11
16
Release ID
RDS
Incr. Growth
0
5
10
15
20
1
11
21
31
41
Release ID
SQS
Incr. Growth
0
1
2
3
4
5
1
6
11
16
Release ID





5th Law	**The incremental growth of E-type systems is constrained by the need to maintain familiarity**
Criteria	1. Releases characterized by high incremental growth, followed by releases with lower incremental growth 2. Declining trend in the incremental growth of the system, due to the increasing complexity of the system
Validity	**Holds**
Properties	▪ There is **no clear declining trend** in the **incremental growth** of the operations ▪ However, releases characterized by **non-zero incremental growth**, tend to be **followed by** releases of **zero incremental growth**.





8th Law	**E-type evolution processes are multi-level multi-loop, multi-agent feedback systems**
Criteria	The actual growth of the system adheres to the inverse square (IS) model $\overline{G}\left(r_i^s\right)=\overline{G}\left(r_{i-1}^s\right)+\frac{\overline{E}}{\overline{G}\left(r_{i-1}^s\right)^2}, \quad where\ \overline{E}=avg_{H_s}\left(E_j\right)$ $E_j=\left(G\left(r_j^s\right)-G\left(r_{j-1}^s\right)\right)*G\left(r_{j-1}^s\right)^2$
Validity	**Holds**
Properties	The **growth** of the examined Web services can be **accurately estimated** via a **feedback-based formula** that exploits changes in previous service releases

Roadmap

- Method
- Laws
- Recommendations

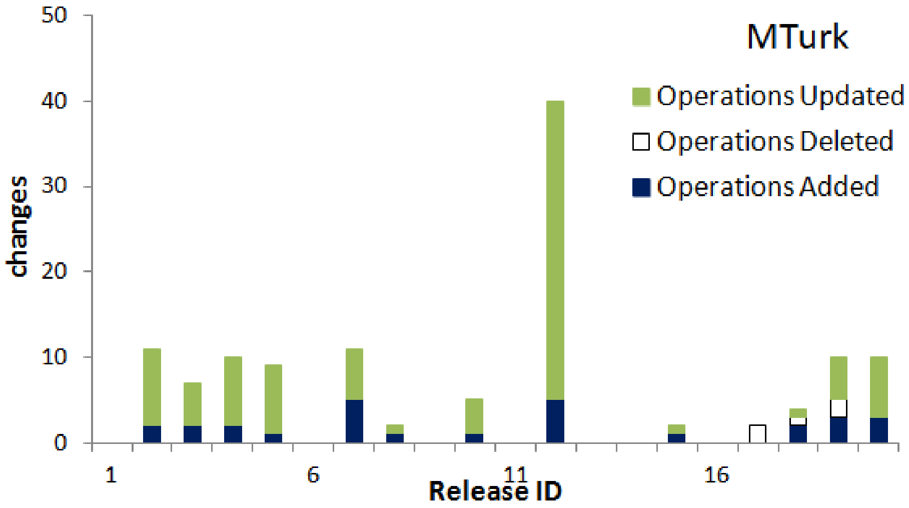
Is this service living a healthy life?

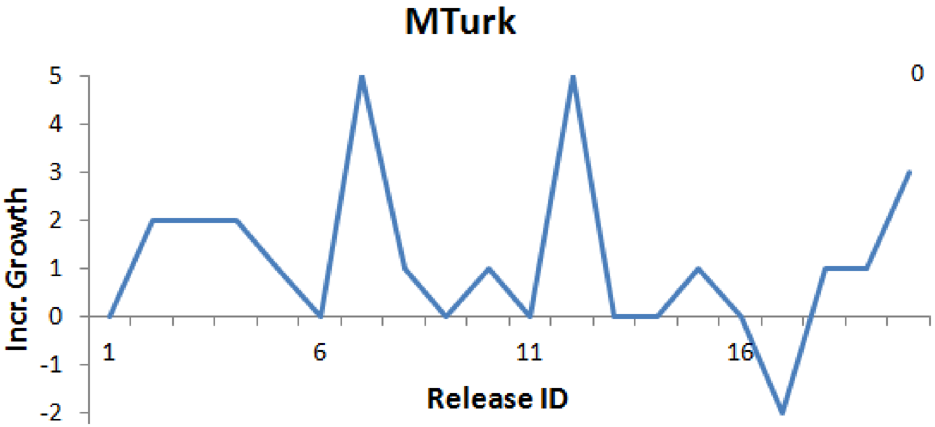
Normal life = calm lives with few excitement

(mostly) periods of calmness + add & update spikes

Checklist:

- change heartbeat
- incremental growth

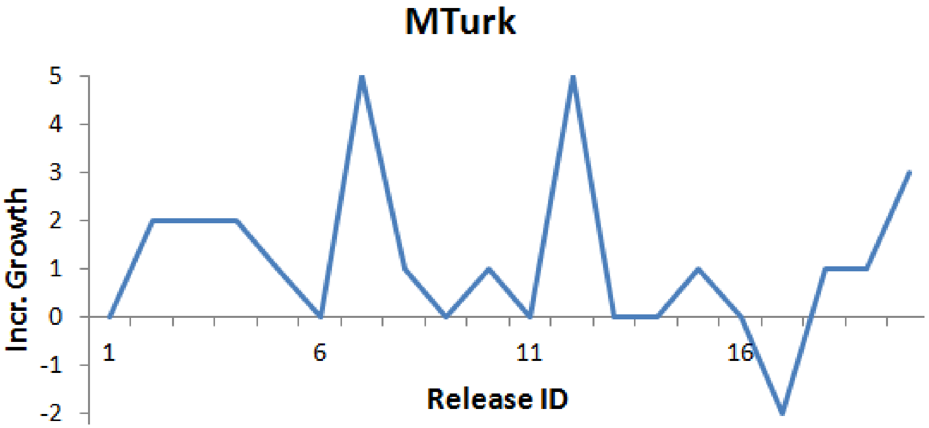


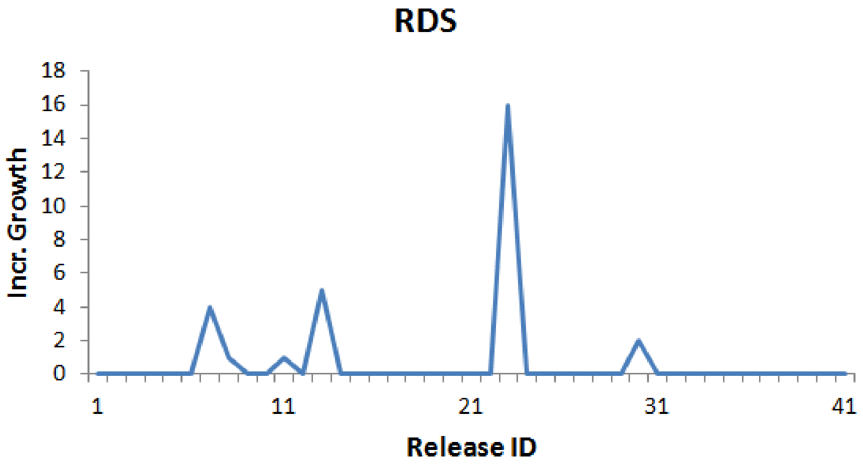


Will I have time to absorb changes?

- Check the incremental growth of the service for a simple pattern:
  - releases with changes of the spec (non-zero incremental growth)

    followed by
  - releases of zero incremental growth
- If yes: there is time to absorb the changes.



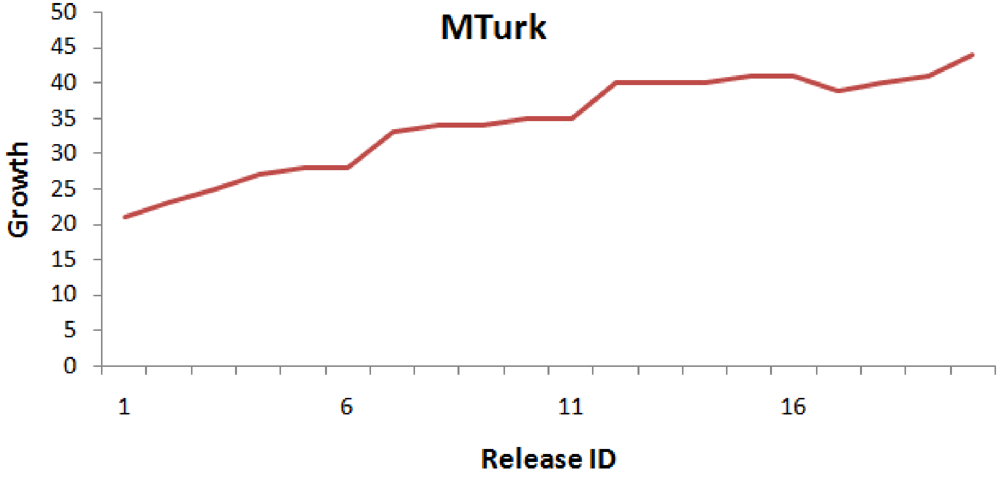


Will I have time to absorb changes …
… & learn about new functionalities?

Checklist:

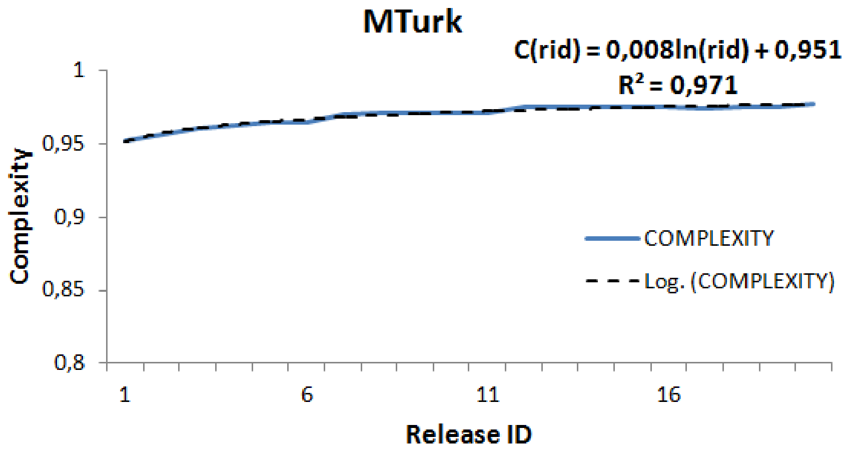
- Is growth increasing with discontinuations?

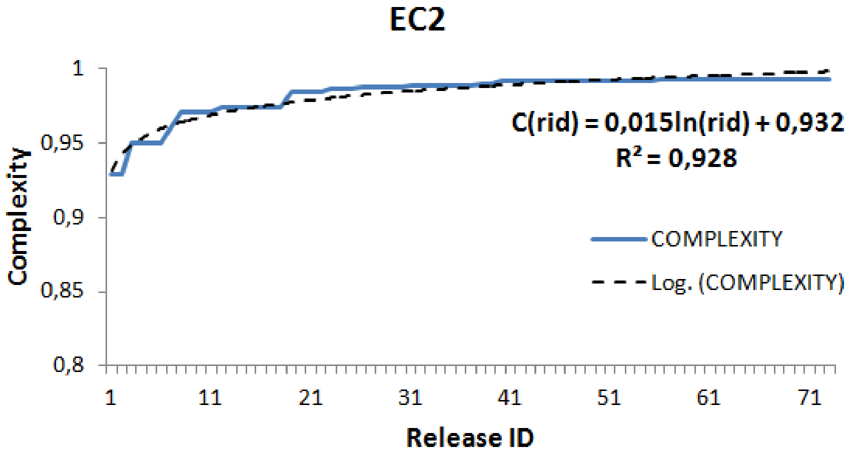
If you observe that the increase is not continuous: you can use the interval for the understanding of the new features.



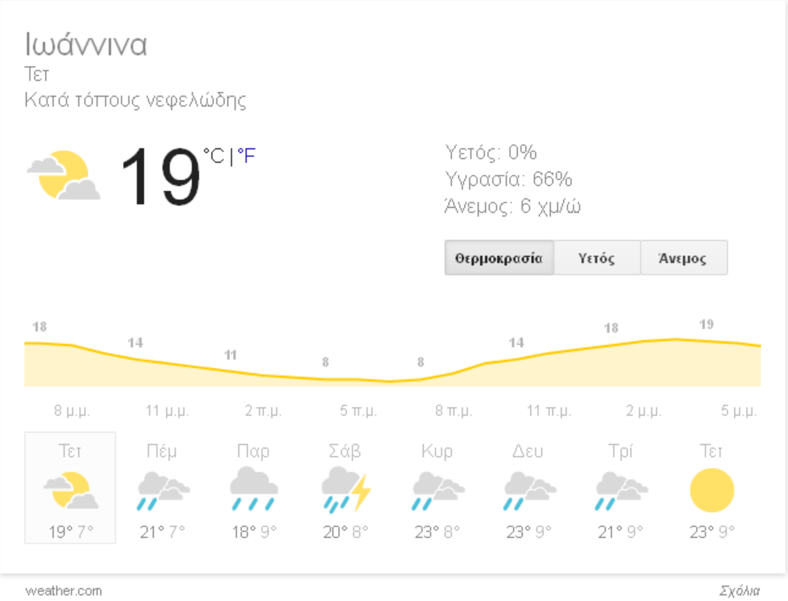
Will the complexity of the service be a problem for service usage?

- Complexity assessment is complex!
  - More than one types of complexity: specification, architectural, structural, etc.
  - **Focus on the one(s) you 're interested in!**
- Complexity is typically high with the tendency to increase
  - … but this can happen in a smooth way …
  - **Neither panic** (refrain from using an otherwise healthy service)**, nor relax**





Can we forecast …



Ιωάννινα
Τετ
Κατά τόπους νεφελώδης

19 °C | °F

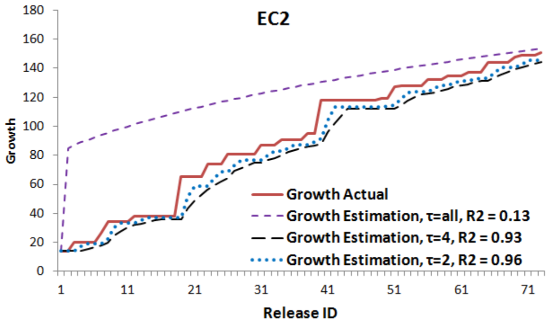
Υετός: 0%
Υγρασία: 66%
Άνεμος: 6 χμ/ω

Τετ	Πέμ	Παρ	Σάβ	Κυρ	Δευ	Τρί	Τετ
19° 7°	21° 7°	18° 9°	20° 8°	23° 8°	23° 9°	21° 9°	23° 9°

weather.com

Σχόλια

- … **the heartbeat of changes?**
  - No!
  - Prepare (accommodate resources) for the worst.
- … **the quality of the service?**
  - No!
  - Again: focus on the quality aspects that you're interested in!
- … **the amount of new functionalities?**
  - Coarsely, yes!



In summary…

- We can monitor web services as **external observers** and assess their evolution patterns
  - Normal life: spikes of increase between calmness
  - Few deletions, although a lot of internal maintenance
  - High complexity, but manageable

Plan for this in advance

There is time to absorb the changes

Can eventually deduce what part of the service is relevant

Keep Calm & Wait for the Spike!

Thank you!

Najlepša hvala!

		Measure	*Confirmed when…*	*Advice to developers of external applications*
***Growth** (positive feedback) is necessary for a healthy life …*				
VI	*Continually grow* [Confirmed]	Growth	Continual enhancement	**Count on the service provider for the provision of new functionalities.** You will have time to understand the expansion.
I	*Continually adapt or die* [Confirmed]	Heartbeat	Continuous adaptation	**Count on the service provider to take maintenance seriously**. Do not be afraid to use an operation: it is likely it will not disappear.
IV	*Constant work rate* [Not Confirmed]	Heartbeat	- Internal data show stable productivity - Stable #changes /release	It is not really possible to forecast the amount of change. Thus, **accommodate resources for the worst case.**
*…but so does **control**! If no maintenance (negative feedback)…*				
II	*Complexity increases* [Confirmed]	Interface Complexity	Complexity increases in the long run (or evidence of rigorous maintenance)	Do not avoid complex services; sooner or later services' complexity increases. **Allocate time and resources for comprehension.**
VII	*Quality declines* [Inconclusive]	Quality measures	Quality declines in the long run (or evidence of rigorous maintenance)	**Allocate time and resources for QoS assessment.**
*There is a **feedback mechanism** after all for a healthy life*				
III	*Feedback regulation* [Confirmed]	Incremental Growth	Observation of ripples as evidence of a stabilization mechanism	**Spikes and calmness periods mean that the services "live" a normal life. Profile & select services that conform to these patterns.**
V	*Familiarity needs constrain incr. growth* [Confirmed]	Incremental Growth	- Spikes followed by flatland - Long term decline of incr. growth	**Spikes and calmness periods** provide ample time to absorb changes. **Profile & select services that conform to these patterns.**
VIII	*Complex feedback mech.* [Confirmed]	IS model	Success of prediction	The expansion of the offered operations can be predicted via an **IS model with short memory**. Plans can be made accordingly.

Q & A

Keep Calm & Wait for the Spike!

My answer to your question is:

YES, I think I can agree with that

Q & A

Keep Calm & Wait for the Spike!

My answer to your question is:

NO, I do not think I can agree

Q & A

Keep Calm & Wait for the Spike!

My answer to your question is:

It’s complicated, let me tell you what I think…

Q & A

Keep Calm & Wait for the Spike!

My answer to your question is:

I don't know about that

…and the only thing I can add is …

```
while( alive ) {
if ( !calm )
{ keep( calm ) };
&&
{ code( ON ) }; }
```

Auxiliary slides

Threats to validity

External Validity

- **DO NOT** over-generalize the results to the overall population of existing Web services

- The findings are **representative** of the **overall population of Amazon services**
- The **assessment approach is general** and can be used to perform further similar studies.
- The **recommendations** for service selection and usage are **general**

Construct Validity

- Membrane SOA, for the **accurate construction** of evolution histories
- Manual inspection of random samples of the collected data

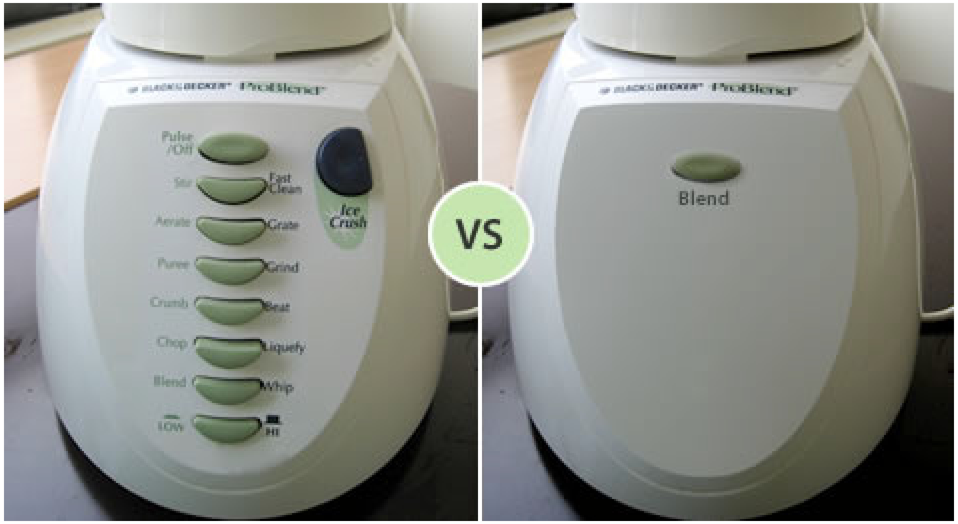
Conclusion Validity

- **Validation** of observed relations and trends with well-known **statistic methods**

What did we measure

Interface Complexity:

$$C\left(r_i^s\right) = 1 - \frac{r_i^s.Size[Interfaces]}{r_i^s.Size[Opers]}$$



What did we measure

Incremental Growth:

$$IG\left(r_{i+1}^{s}, r_{i}^{s}\right) = r_{i+1}^{s}.Size[Opers] - r_{i}^{s}.Size[Opers]$$

Growth:

$$G\left(r_{i}^{s}\right) = r_{i}^{s}.Size[Opers]$$

What did we measure

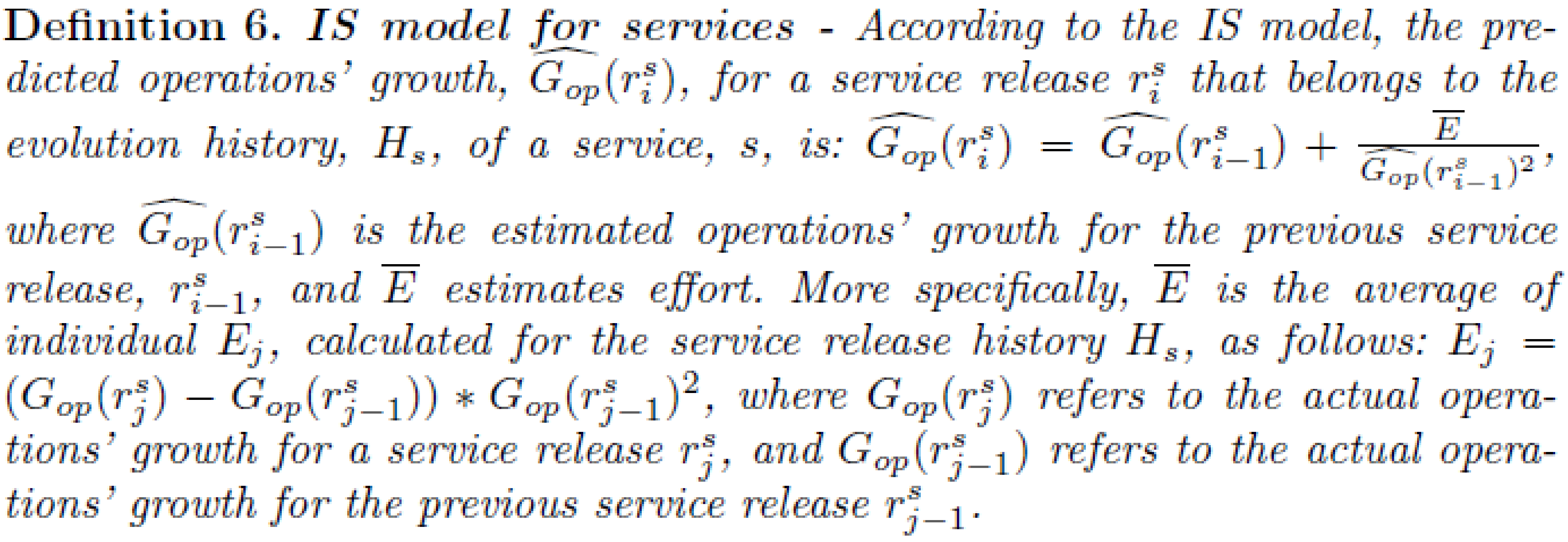
Inverse Square Model:

$$\overline{G}\left(r_i^s\right) = \overline{G}\left(r_{i-1}^s\right) + \frac{\overline{E}}{\overline{G}\left(r_{i-1}^s\right)^2}$$

where

$$\overline{E} = avg_{H_s}\left(E_j\right),$$

$$E_j = \left(G\left(r_j^s\right) - G\left(r_{j-1}^s\right)\right) * G\left(r_{j-1}^s\right)^2$$



Definition 6. *IS model for services - According to the IS model, the predicted operations' growth, $\widehat{G_{op}}(r_i^s)$, for a service release r_i^s that belongs to the evolution history, H_s, of a service, s, is: $\widehat{G_{op}}(r_i^s) = \widehat{G_{op}}(r_{i-1}^s) + \frac{\overline{E}}{\widehat{G_{op}}(r_{i-1}^s)^2}$, where $\widehat{G_{op}}(r_{i-1}^s)$ is the estimated operations' growth for the previous service release, r_{i-1}^s, and $\overline{E}$ estimates effort. More specifically, $\overline{E}$ is the average of individual E_j, calculated for the service release history H_s, as follows: $E_j = (G_{op}(r_j^s) - G_{op}(r_{j-1}^s)) * G_{op}(r_{j-1}^s)^2$, where $G_{op}(r_j^s)$ refers to the actual operations' growth for a service release r_j^s, and $G_{op}(r_{j-1}^s)$ refers to the actual operations' growth for the previous service release r_{j-1}^s.*

$$\overline{G}\left(r_i^s\right)=\overline{G}\left(r_{i-1}^s\right)+\frac{\overline{E}}{\overline{G}\left(r_{i-1}^s\right)^2}$$

where

$$\overline{E}=avg_{H_s}\left(E_j\right),$$

$$E_j=\left(G\left(r_j^s\right)-G\left(r_{j-1}^s\right)\right)*G\left(r_{j-1}^s\right)^2$$

		Measure	*Confirmed when…*	*Advice*
I	*Continually adapt or die* [Confirmed]	Heartbeat	Continuous adaptation	**Count on the service provider to take maintenance seriously**. Do not be afraid to use an operation: it is likely it will not disappear.
II	*Complexity increases unless battled* [Confirmed]	Interface Complexity	Complexity increases in the long run (or evidence of rigorous maintenance)	Do not avoid complex services; sooner or later services' complexity increases. **Allocate time and resources for comprehension.**
III	*Feedback regulation* [Confirmed]	Incremental Growth	Observation of ripples as evidence of a stabilization mechanism	Spikes and calmness periods mean that the services "live" a normal life. **Profile & select services that conform to these patterns.**
IV	*Constant work rate* [Not Confirmed]	Heartbeat	- Internal data show stable productivity - Stable #changes per release (approx.)	It is not really possible to forecast the amount of change. Thus, **accommodate resources for the worst case.**
V	*Familiarity needs constrain incr. growth* [Confirmed]	Incremental Growth	- Spikes followed by flatland - Long term decline of incr. growth	**Spikes and calmness periods** provide ample time to absorb changes. **Profile & select services that conform to these patterns.**
VI	*Continually grow* [Confirmed]	Growth	Continual enhancement	**Count on the service provider for the provision of new functionalities.** You will have time to understand the expansion.
VII	*Quality declines* [Inconclusive]	Quality measures	Quality declines in the long run (or evidence of rigorous maintenance)	**Allocate time and resources for QoS assessment.**
VIII	*Complex feedback mech.* [Confirmed]	IS model	Success of prediction	The expansion of the offered operations can be predicted via an **IS model with short memory**. Plans can be made accordingly.

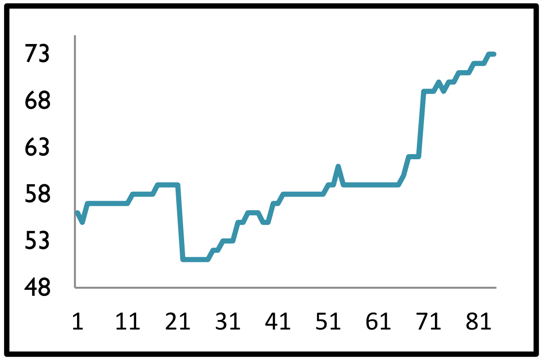
Focaefs+ @ ICWS11

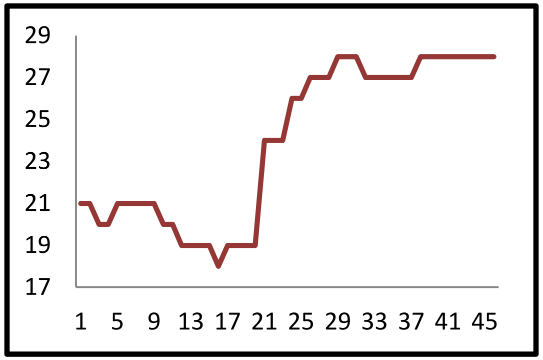
- Present a tool, VTracker for detecting changes
- Analyze AWS EC2, FedEx Rate, FedEx Movement Inf. Serv., Paypal, Bing Search
- Findings include
  - Domination of operation additions in some WS
  - Domination of operation updates in some others + some specific versions of the former group
  - Absence of operation deletions
- Taxonomy & discussion of data type changes

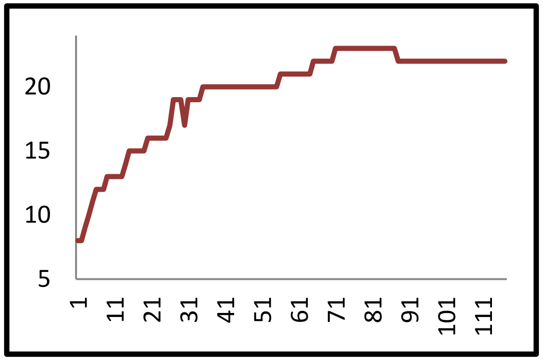
Romano & Pinzger @ ICWS12

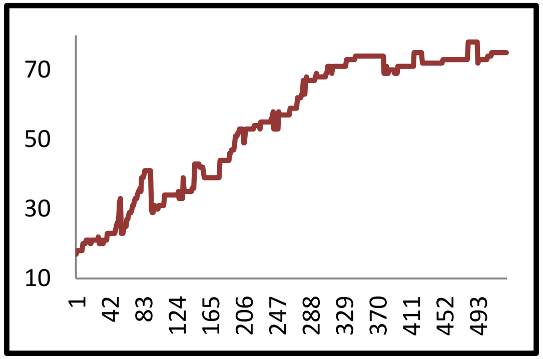
- Present a tool, WSDLDiff for detecting changes (more fine-grained than VTracker)
- Analyze AWS EC2, FedEx Rate, FedEx Ship, FedEx Package Movement Inf. Serv.
- Findings include:
  - Domination of operation additions @ AWS
  - Domination of operation updates @ Fedex
  - Absence of operation deletions
  - Too many data type updates in 3 out of 4 services
  - Each service comes with its own change profile, where some type of changes dominates (element additions for EC2, enumeration additions for Fedex.*)

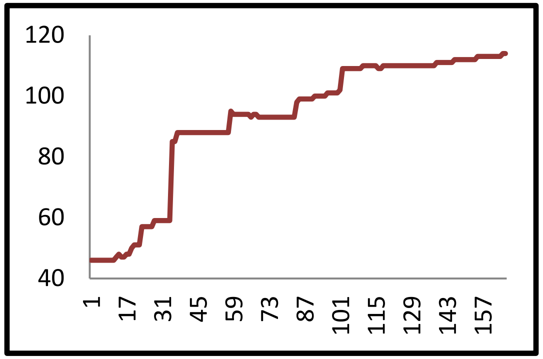
Skoulis+ @ CAiSE’14: DB Schema evolution

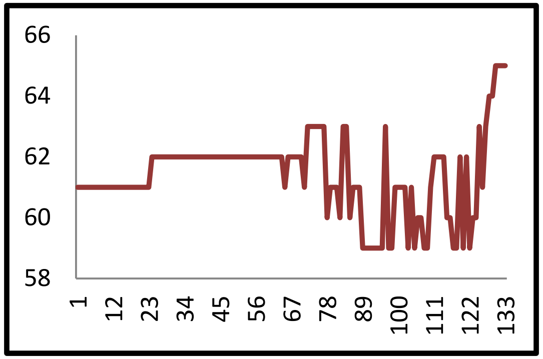


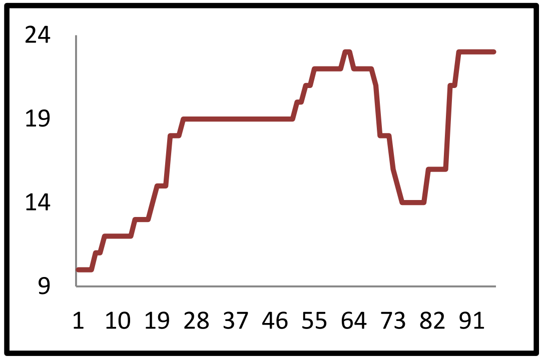


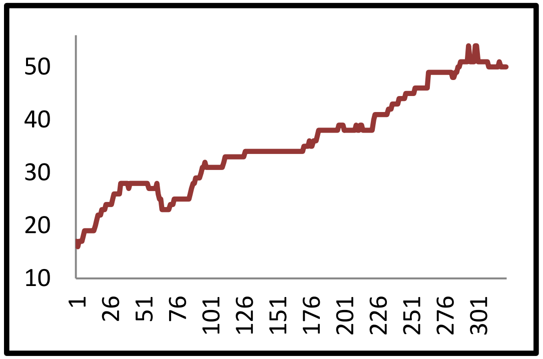












http://www.cs.uoi.gr/~pvassil/publications/2014_CAiSE/

Our goal…

We focus on amazon webservices™

one of the **most successful stories** of the service-oriented paradigm in industry

We perform a principled **empirical study** that detects evolution **patterns** and **regularities**, based on **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.”

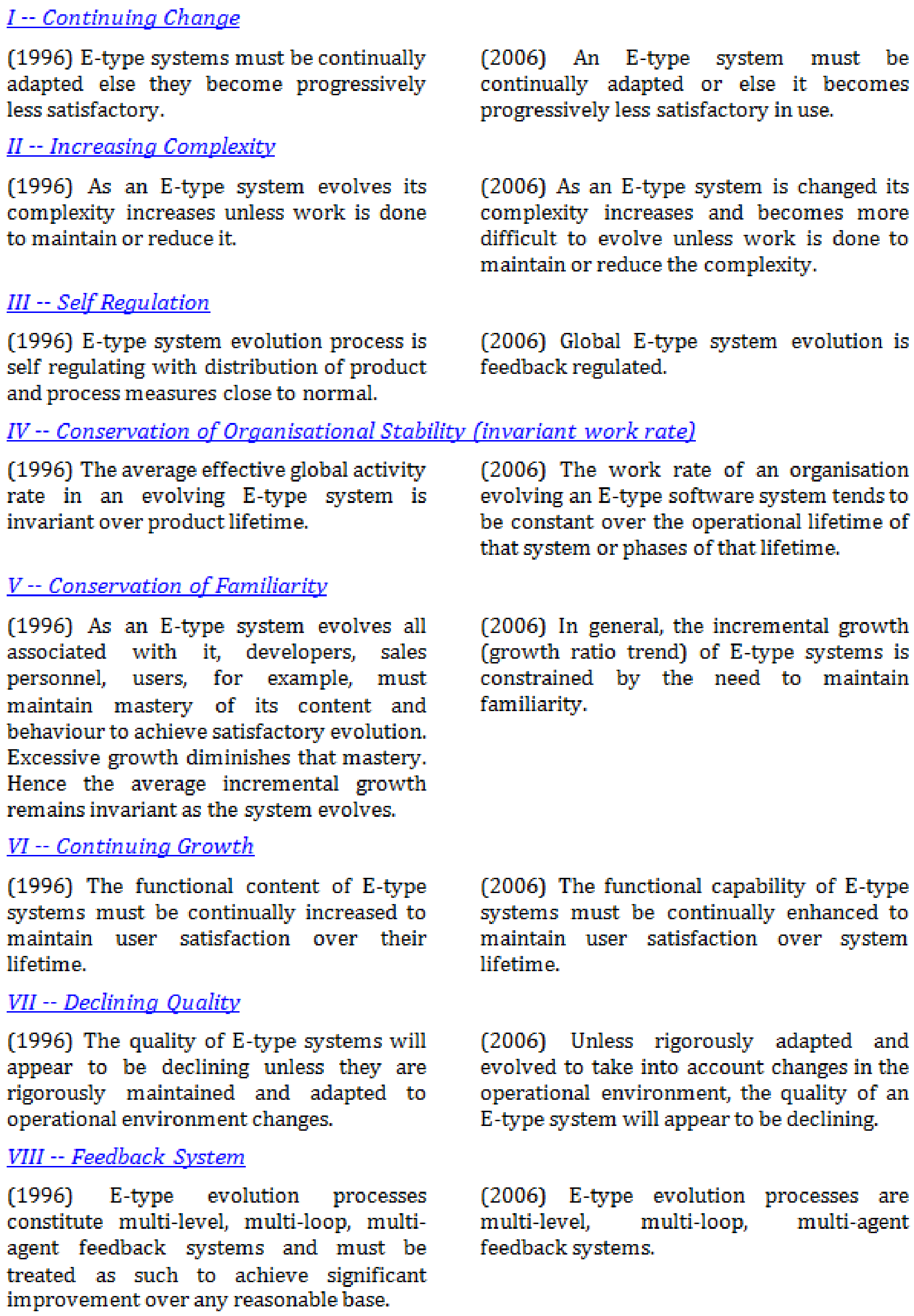
VII. Declining Quality

“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.

Keep Calm & Wait for the Spike!