Daniel Klauser

Homework 1

How can the growth oft he IBM OS 360 system be interpreted?

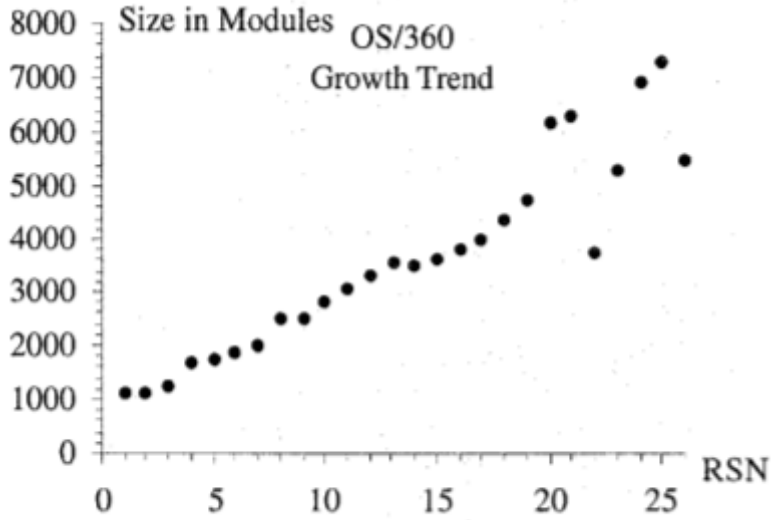

Fig. 1 OS/360 growth trend by rsn

The figure shows the software releases from the OS/360. RSN means Release Sequence Number. Up to RSN 21 the modules of the software is increasing. After release 21 the paper says it can be interpreted differently. We could say that it behaves chaos like because there is no linear ascend or the fission process has begun. Certain is that we cannot predict further releases.

KROPP : It's impressive how constant this increase is over the time

- In my opinion it looks like they might made a big improvement of the code. Because it could do the same as before with less Modules. That implies that they did not really care about the size of the software until rsn 21.

KROPP : A good possible interpretation. A major refactoring might have lead to less modules, keeping the same functionality. This hypotheses could be concluded from the two previous releaes. Where a lot of new modules (more than in the previous) have been added. It looks that the team had to add a lot of new functionality in a short time (RSN 19 and 20), then did a major refactoring for clean up (RSN 21). Other option could also be, that they indeed removed functionality.

What is the special challenge with E-Type system, compared with S- and P-Type systems?

I havent read anything about the S and P Type systems but I guess the E Type system is never complete. If you developted an E Type software it has just started. The software will evolve with further requirements.

KROPP: Indeed, you can say, that such systems (if used) are always changing. E-Type systems are what we call end-user software, while S and - type systems are embedded software. This context is much more closed, specifications are often very clear (at least they where at the time of the writing of the paper)

Daniel Klauser

Why did Lehman add new laws – do you agree with these?

Several studies had similarities so Lehman started to write down the laws of software evolution to achieve a generalisation in software behavior. The rules have to be applied that the software will survive.

No.	Brief Name	Law
I 1974	Continuing Change	*E*-type systems must be continually adapted else they become progressively less satisfactory.
II 1974	Increasing Complexity	As an *E*-type system evolves its complexity increases unless work is done to maintain or reduce it.
III 1974	Self Regulation	*E*-type system evolution process is self regulating with distribution of product and process measures close to normal.
IV 1980	Conservation of Organisational Stability (invariant work rate)	The average effective global activity rate in an evolving *E*-type system is invariant over product lifetime.
V 1980	Conservation of Familiarity	As an *E*-type system evolves all associated with it, developers, sales personnel, users, for example, must maintain mastery of its content and behaviour [leh80a] to achieve satisfactory evolution. Excessive growth diminishes that mastery. Hence the average incremental growth remains invariant as the system evolves.
VI 1980	Continuing Growth	The functional content of *E*-type systems must be continually increased to maintain user satisfaction over their lifetime.
VII 1996	Declining Quality	The quality of *E*-type systems will appear to be declining unless they are rigorously maintained and adapted to operational environment changes.
VIII 1996	Feedback System (first stated 1974, formalised as law 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.

Table 1 Laws of software evolution

Do I Agree with these?

1. Continuing Change : yes, software demands increase of time
2. Increasing Complexity : yes, due to rule 1.
    a. KROPP: Unless you invest effort to keep complexity low. e.g. by an very modular and extendible design.
3. Self Regulating : yes, if someone observe the measures.
    a. KROPP : Measuring it makes the regular behavior visible. He derives the regularity from the many observations over time. Having such a regularity allows you to make predictions for the future, how the system may evolve.
4. Conservation of Organisational stability : yes due to development of the surroundings of the software.
5. Conservation of Familiarity : yes, they do have to work with the software so they are getting along with these. If a bigger update will follow people might not getting along with it.
6. Continuing Growth : Yes due to rule 1.
7. Declining Quality : Yes
    a. KROPP if you don't take measure to avoid it.
8. Feedback System : ?
    a. KROPP : You get constant feedback on various levels. From the system itself, by measuring it, when crashing, or showing misbehaviour, from the user who reports errors or request for new features,

Daniel Klauser

Discuss the consequences of the laws for your own development work and organization when you are a maintainer of a just released software system.

I am responsible for the image processing framework in our company. On every new machine we build, I have to adapt the software on certain tools (law four). Those tools not always fit into the software so I have to adapt it. Gladly I have time to refactor the framework to meet the requirement for the tools. Law 1 ,2 and 6 are thereby fullfilled. Law five is also fullfilled because my ability to code is increasing over software changes. Due to the time I have law seven will hopefully not occur. Furthermore people are giving feedback while using the application in the testphase. Neverthless I have to admit that the quality might decrease over time until a refactoring happens due to time constraints or simply not beeing carefull (law 7).

KROPP : a good transfer of the theory to your own work. Yes, I completely agree, "law 7" can be avoided by taking measures,

As conclusion

- Time is an important factor in software evolution.
- To have a certain testphase will make the feedback construtive and the bugs fewer.
- Maybe double check code to avoid quality decrease.

KROPP : Good suggestions.
And reserve enough resources to maintain and evolve the system.