An Approach to Modelling Long-Term Growth Trends in Software Systems

ICSM 2001
Florence
7 Nov 2001

M M Lehman
Dept. of Computing
Imperial College
180 Queen’s Gate
London SW7 2BZ, UK
mml@doc.ic.ac.uk
http://www.doc.ic.ac.uk/~mml

J F Ramil
Dept. of Computing
Faculty of Maths and Computing
The Open University
Milton Keynes MK7 6AA, UK
j.f.ramil@open.ac.uk
http://mcs.open.ac.uk/jfr46

U Sandler
Dept. of Mathematics
Jerusalem College of Technology
P.O. Box 16031
Jerusalem 91160, Israel
sandler@math.jct.ac.il

Introduction

- Software **evolution** is process of continual **fixing**, **adaptation**, **enhancement** to **maintain stakeholder satisfaction**
  - in response to **changes** in **domains**, needs, **expectations**
  - intrinsic to ***E*-type,** that is real world**,** applications
  - co-evolving domains, ***many*** concurrent **activities** *at many* levels
  - **subsumes maintenance,** which is said to absorb 60 - 80% of total costs over lifecycle

- **Dependence** on computers and **demands** on professional **resources** make **planning** and **management** of evolution **vital**

- **Rate** of evolution **critical** for business

- **Long-term viewpoint** contributes to **sustainable evolution**

- Not generally addressed in current **models** and **tools** for **effort estimation**

- In particular, there is **need** for empirical **models** of evolution phenomenon

Modelling **approaches**

Modelling Approaches

- **Many approaches** to modelling
  - **Visualisation,** including **plotting**
  - **Statistical representation** and **analysis**
  - General **simulation**
  - **System dynamics models** and **simulation**
  - **Analytic modelling - from first principles**
- List is **sequence** of techniques applied to study of **software evolution**

Presentation focuses on **work in progress**

Laws of Software Evolution as Background

- Studies in late 60s and 70s led to recognition of evolution **phenomenon**

- Plotting and visualisation led to identification of **behavioural commonalties** across systems - e.g. **laws of software evolution**

- Termed ***laws*** because they relate to **mechanisms** largely **independent** of **technology** and process **detail**
  - recently discussed in FEAST and in **publications** since 1974
  - provide framework for tools and management **guidelines** - paper to appear in Annals of Softw. Eng, v. 11

- FEAST assessed their **empirical support**
  - seven **supported** after minor modification
  - see FEAST **publications**

- As stated, laws are **qualitative** **descriptors** of **behaviour**

Framework for **behavioural** **process modelling**

Laws and their support

No.	Brief Name	Law	Support
I 1974	Continuing Change	An *E*-type systems must be continually adapted else it becomes progressively less satisfactory in use	yes
II 1974	Increasing Complexity	As an *E*-type system is evolved its complexity increases unless work is done to maintain or reduce it	indirect
III 1974	Self Regulation	Global *E*-type system evolution processes are self-regulating	yes
IV 1978	Conservation of Organisational Stability	Average activity rate in an *E*-type process tends to remain constant over system lifetime or segments of that lifetime	yes
V 1978	Conservation of Familiarity	In general, the average incremental growth (growth rate trend) of *E*-type systems tends to decline	yes
VI 1991	Continuing Growth	The functional capability of *E*-type systems must be continually enhanced to maintain user satisfaction over the system lifetime	yes
VII 1996	Declining Quality	Unless rigorously adapted to take into account changes in the operational environment, the quality of an *E*-type systems will appear to decline as it is evolved	theory
VIII 1996	Feedback System (Recognised 1971, formulated 1996)	*E*-type evolution processes are multi-level, multi-loop, multi-agent feedback systems	white box models

Evolving Complexity

- **Operational domains** undergo continual change and some **assumptions** reflected in software may no longer be valid - **principle of software uncertainty**
- Software maintenance as maintenance of **validity of assumption set**
- As system is evolved, **ageing effects accumulate** unless **counteracted**
- One of these is **increasing complexity** - stated as second law of software evolution
- Unless adequately **counteracted** - by for example, **anti-regressive** work - increasing complexity will increase effort required for evolution, ultimately to **stagnation**
- **Effort** needed to achieve evolution is related to system **size**, **complexity** however **defined**
- **Reference** to complexity of the system "***as a whole***"
- **Complexity** seen as **ratio** of **effort applied** and resultant **size increment** - where size is surrogate for **functional power**

Analytic extension

Basic Formulation

- Previous observations suggest complexity is defined as:

$$C = dE/dS$$

where S = Size

E = Cumulative Effort

C = Complexity

- Also C may increase with size of total system, hence:

$$dC/dS = f(C)$$

where f(C) is a non-decreasing function

$$0 <= f(C) <= A\ C^{\beta}$$

$$0 <= \beta < 1$$

where A and β are constant parameters

Basic Formulation

- Assuming that f(C) **-->**A C $^{\beta}$, it follows that:

$$C \sim S^k$$

$$S \sim E^{1/g}$$

where $k = (1 - \beta)^{-1}$

$g = 1 + k$

$k >= 1 \quad g >= 2$

- where interpretations of g and k are linked to metrics used

- Provides basis for **models of long-term growth** based on **size/effort relationship**

e.g., **Size(t) ~ [Cumulative_effort(t)] $^{1/g}$**

What measures of size and effort?

System Size Indicators

- Size:
  - lines of code (loc), function points (fp), not widely available from historical records
  - numbers of elements - such as modules, sub-systems - **likely** to be correlated to functional power and posses a level of functional integrity

- For a system of with **n elements**, maximum number of element **interconnections** is ***n(n-1)/2***

- Reflected by: $C \sim S^2$ and $S \sim E^{1/3}$ - k = 2, g = 3

- Linked to **inverse square model** [Turski 1996]:

$$S_i = S_{i-1} + e/(S_{i-1})^2$$

Effort Indicators

- **Direct** indicators, such as person-months, always **desirable**

- But **detailed** records **not** normally **retained**

- **Meaningful study** nevertheless **possible**

- Moreover, observations suggest that effort applied is **approximately constant** over time **intervals** or **sequence of releases**

- For time period over which this holds one may assume :

cumulative effort applied ~ system age or alternatively

~ release sequence number

- ***Work-rate*** as an **indirect** **effort indicator**
  - example: **elements handled**
  - see paper for definition
  - see also ICSM 2000 paper

Three Analytical Models

$$\text{Relative Size} = (\text{Release Sequence Number})^{1/g}$$

$$\text{Relative Size} = (\text{Relative Cumulative Work-Rate})^{1/g}$$

Inverse Square Model

Model A Normalised size as a function of the release sequence number	Model B Normalise size as a function of the normalised work-rate	Model C Normalised size as a function of the release sequence number
$\underline{S}_i / S_1 = (i)^{1/g}$ for i >= 1	$\underline{S}_i / S_1 = (H_i/H_1)^{1/g'}$ for i >= 1	$\underline{S}_i / S_1 = \underline{S}_{i-1}/S_1 + [e\,(S_1)^2/(\underline{S}_{i-1})^2]$ for i >= 2

S_i and $\underline{S}_i$ stand for actual and estimated size at release i, respectively

g, g' and e are model parameters

- Examples of what can be achieved with analytical models

Some Remarks Relating to Empirical Assessment

- Data from four **industrially evolved** systems
- Representing 70 to 90s **technology**, progressively **adapted**
- Three different **organisations** (two UK, one US)
- **Three** different **application domains**
- Size measured in in ***modules*** - i.e. element counts,
- **No** direct **effort indicators** available at time of study
- **Work-rate** measured in ***modules handled***

Data availability leads to three model candidates

Systems Included in Study

System ID (symbol used in plots)	Application Domain	Main Source Code Language	Technology of First Operational Release	Number of Major Releases in Data Set	Period of System Evolution Studied (approx. in years)
A (■)	Real Time	C	1980s	19	20
B (▲)	Real Time	C	1980s	16	20
C (?)	Operating System	S3 (*)	1970s	21	15
D (+)	Information System	COBOL	1980s	13 (**)	14

(*) proprietary language of the 70s

(**) data aggregated per year

Estimated Parameters (least squares fit)

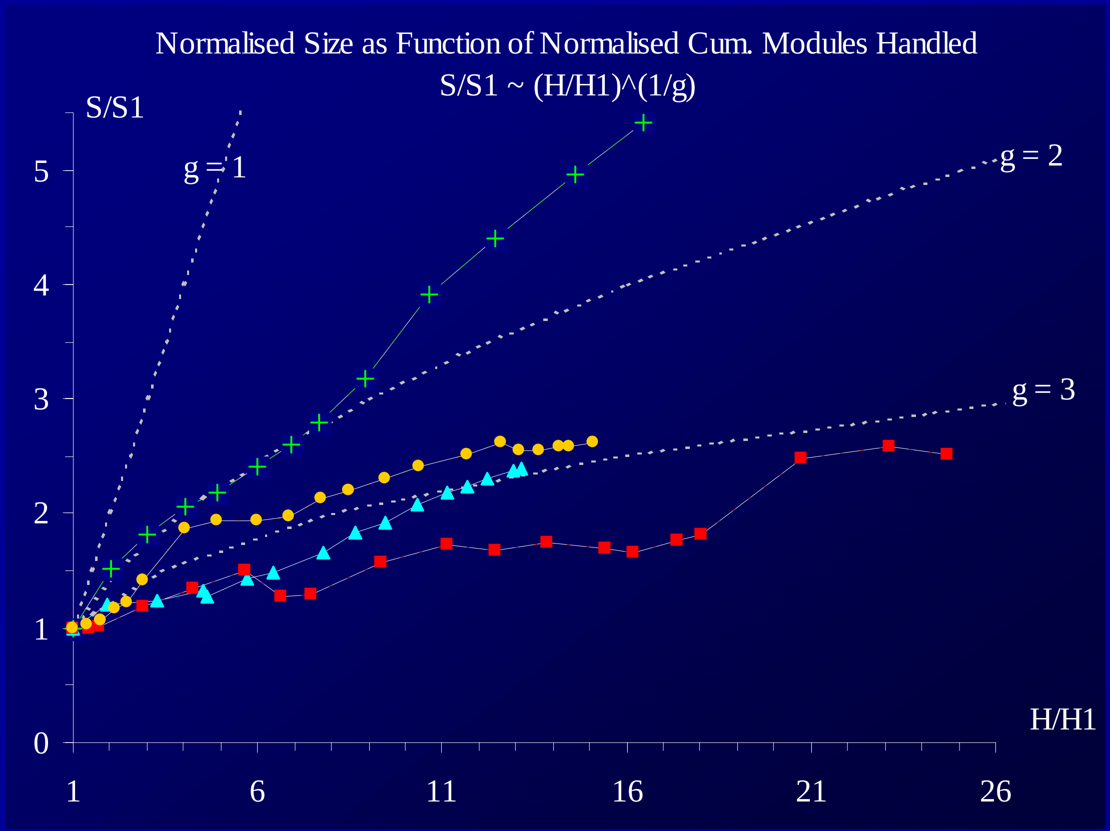
System	Model 1 Parameter g	Model 2 Parameter g’	Model 3 Parameter e
A	4.74	3.33	0.11
B	3.51	4.42	0.21
C	3.14	2.72	0.27
D (entire data set)	1.89	1.86	0.74
D (segment 1: 1 = i = 8)	2.04	2.02	0.64
D (segment 2: 9 = i = 13)	1.61 (*)	1.68 (*)	0.26 (**)

(*) without data re-normalisation (**) after data re-normalisation at i = 9

- In all systems g > 1, that is, growth at a slower rate than accumulation of effort

Growth patterns

Growth as Function of Cumulative Effort Applied



- Note closeness in 3 of the systems to "S = $E^{1/3}$ trajectory"
- One system starts following S = $E^{1/2}$ trajectory, though mid-life change requires further analysis

Modelling Precision

System	Model 1 MMRE %	Model 2 MMRE %	Model 3 MMRE %
A (*)	9.1	10.1	8.4
B (*)	11.1	8.6	9.7
C (*)	9.4	5.0	9.8
D (*)	11.7	8.4	16.8
D (segment 1: 1 <= i <= 8)	2.8	2.7	5.7
D (segment 2: 9 <= i<= 13)	5.6	2.3	1.7

(*) MMRE calculated over entire data set

- Remarkable precision for such simple models and for a process that involves human activity at all levels
- Suggests that system dynamics constraints/drive evolution
- Models likely to be useful - but see discussion of anomalies in paper

Final Remarks

- Approach provides means for **evolution planning**, **management and control**
  - tools
  - long term prognosis
  - indicator of process or organisational change

- **Extension**
  - additional measures extend models
  - measures may be obtained from historical data - e.g. developer **names** in **change log records**
  - model refinement

- **Comparison** of evolution under different paradigms
  - Object Oriented, Open Source, COTS and component intensive processes

- Input to development of **theory of software evolution**

References

- [ant01] Anton A and Potts C, *Functional Paleontology: System Evolution as the User Sees It*, Proc. 23rd ICSE, Toronto, 12-19 May 2001, pp. 421-430
- [bel71] Belady LA and Lehman MM, *Programming System Dynamics or the Metadynamics of Systems in Maintenance and Growth*, IBM Res Rep T J Watson Res. Centre, Yorktown Heights, NY, RC 3546, Sept. 71, p 30. Reprinted in [leh85]
- [bel72] Belady LA and Lehman MM, *An Introduction to Program Growth Dynamics*, in Statistical Computer Performance Evaluation, W Freiburger (ed), Academic Press, New York, 1972, pp. 503 – 511. Reprinted in [leh85]
- [ben00] Bennett KH and Rajlich VT, *Software Maintenance and Evolution: a Roadmap*, in A. Finkelstein (ed.), The Future of Software Engineering , ICSE 2000, June 4-11 Limerick, Ireland, ACM Order Nr. 592000-1, pp. 75 - 87
- [boe81] Boehm B, *Software Engineering Economics*, Prentice Hall, 1981
- [boe00] Boehm B *et al*, *Software Cost Estimation with COCOMO II*, Prentice Hall, 2000
- [cha01] Chapin N et al, Types of Software Evolution and Software Maintenance, J. of Softw. Maint. and Evolution: Res. and Practice, Vol. 13, Issue 1, Jan-Febr, 2001, pp. 1 - 30
- [cox66] Cox DR and Lewis PAW, T*he Statistical Analysis of Series of Events*, Methuen, London, 1966
- [fen00] Fenton NE and Neil M, *Software Metrics: Roadmap*, in A. Finkelstein (ed.), The Future of Software Engineering , ICSE 2000, June 4-11 Limerick, Ireland, ACM Order Nr. 592000-1, pp. 357 - 370
- [for61] Forrester JW, *Industrial Dynamics*, Productivity Press, Cambridge MA, 1961
- [fri00] Friedman Y and Sandler U, *Scaling Laws and Universality in Large Software System Development*, working paper, Dept. of Maths., JCT, Jerusalem, July 2000
- [gdf00] Godfrey MW and Qiang Tu, *Evolution in Open Source Software: A Case Study*, ICSM 2000, 11-14 Oct., San Jose, CA, pp. 131 - 142
- [gil76] Gilb T, *Software Metrics*, Chartwell-Bratt, Cambridge MA, 1976
- [ieee93] IEEE Std. 1219, *Standard for Software Maintenance*, IEEE Computer Society Press, Los Alamitos CA, 1993
- [kah01] Kahen G *et al*, *Modelling of Software Evolution Processes for Policy Investigation: Approach and Example,* to appear in J. of Systems and Software, Vol. 15, 2001
- [kem99] Kemerer CF and Slaughter S, *An Empirical Approach to Studying Software Evolution*, IEEE Trans. on Softw. Eng., vol. 25, n. 4, July/August 1999, pp. 493 - 509
- [leh74] Lehman MM, *Programs, Cities, Students, Limits to Growth?*, Inaug. Lect., in Imperial College of Science and Technology Inaugural Lecture Series, Vol. 9, 1970, 1974, pp. 211 -- 229. Also in Programming Methodology, (D. Gries. ed.), Springer Verlag, 1978, pp. 42 – 62. Reprinted in [leh85]
- [leh85] Lehman MM and Belady LA*, Program Evolution - Processes of Software Change*, Academic Press, London, 1985
- [leh96] Lehman MM, *Laws of Software Evolution Revisited*, Proc. EWSPT 96, Nancy, 9 - 11 Oct. 1996, LNCS 1149, Springer Verlag, 1997, pp. 108 - 124
- [leh97] Lehman M *et al*, *Metrics and Laws of Software Evolution - The Nineties View*, Metrics 97, Nov. 5-7, Albuquerque, NM, pp. 20 - 32

References (cont.)

- [leh98] Lehman MM *et al*, *On Evidence Supporting the FEAST Hypothesis and the Laws of Software Evolution*, Proc. Metrics 98, Bethesda, MD, Nov. 20 - 21, pp. 84 - 88
- [leh00] Lehman MM, *TheSE - An Approach to a Theory of Software Evolution*, UK EPSRC research proposal, Dept. of Computing, Imperial College, London, Dec. 2000, pp. 11
- [leh01a] Lehman MM and Ramil JF, *Rules and Tools for Evolution Planning and Management*, to appear in Annals of Software Eng., spec. issue on Softw. Management, vol. 11, 2001
- [leh01b] Lehman MM and Ramil JF, *Software Evolution*, Invited Keynote Lect., IWPSE 2001, Vienna, Sept. 10-11, 2001
- [mcc76] McCabe TJ, *A Software Complexity Measure*, IEEE Trans. Softw. Eng., v. 2, n. 4, 1976, pp 308 - 320
- [par94] Parnas DL, *Software Aging*, Proc. 16th ICSE, May 16-21, 1994, Sorrento, Italy, pp 279 – 287
- [per94] Perry DE, Staudenmayer N and Votta LG, *People, Organisations and Process Improvement*, IEEE Softw., July 1994, pp. 36 - 45
- [pgk97] Pigoski TM, *Practical Software Maintenance*, Wiley Computer Publishing, NY, 1997
- [pir88] Pirzada SS, *A Statistical Examination of the Evolution of the UNIX System*, PhD Thesis, Imperial College, London, 1988
- [raj00] Rajlich VT and Bennett KH, *A Staged Model for the Software Life Cycle*, Computer, July 2000, pp. 66 - 71
- [ram00] Ramil JF and Lehman MM, *Metrics of Software Evolution as Effort Predictors - A Case Study*, Proc. ICSM 2000, San Jose CA, 11-14 Oct. 2000, pp. 163 - 172
- [rio77] Riordan JS, *An Evolution Dynamics Model of Software Systems Development*, in Software Phenomenology - Working Papers of the (First) SLCM Workshop, Airlie, VA, Aug 1977. Pub ISRAD/AIRMICS, Comp. Sys. Comm. US Army, Fort Belvoir VI, Dec 1977, pp. 339 - 360
- [she00] Shepperd M, *Dynamic Models of Maintenance Behaviour*, Intl. Workshop on Empirical Studies of Software Maintenance, WESS 2000, 14 October, San Jose CA, http://members.aol.com/_ht_a/geshome/wess2000/ <as of Oct. 2000>
- [tur96] Turski WM, *Reference Model for Smooth Growth of Software Systems*, IEEE Trans. on Softw. Eng., v.22, n.8, Aug. 1996, pp. 599 - 600
- [som01] Sommerville I, *Software Engineering*, Sixth Edition, Addison-Wesley, Harlow, England, pp. 693
- [suc01] Succi G, Paulson J and Eberlein A, *Preliminary Results from an Empirical Study on the Growth of Open Source and Commercial Software Products*, EDSER-3 Workshop, co-Located with ICSE 23, May 14-15, 2001, Toronto, Canada
- [woo79] Woodside CM, *A Mathematical Model for the Evolution of Software*, CCD, Imperial College, Res. Rep. 79/55, Apr. 1979, also in, J. Sys. and Softw., Vol. 1, No. 4, Oct. 1980, pp. 337 – 345
- [www] FEAST: Feedback, Evolution And Software Technology, http://www.doc.ic.ac.uk/~mml/feast <as of Aug.2001>
- [zus90] Zuse H, *Software Complexity Measures and Models*, de Gruyter, NY, 1990