

Software Evolution

Keynote Lecture
IWPSE 2001
Vienna
10 September 2001

M M Lehman

Dept. of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ
mml@doc.ic.ac.uk
<http://www.doc.ic.ac.uk/~mml>



Introduction

- **Past** - some highlights
 - 1968: IBM software **process** study - including **empirical data analysis**
 - 1969: **The Programming Process** report
 - 1971: First **discussion** of software process as **feedback system**
 - 1974-86: **Laws** of Software Evolution
 - 1979: **SPE-type program classification schema**
 - 1989: Principle of **Software Uncertainty**
 - 1993: **FEAST** hypothesis
 - 1996 to 2001: **FEAST/1** and **/2** projects
- Current **understanding** reflects >30 years of study
- Assumes that software evolution is phenomenon **worthy of study**, and - as proven by results - **can** be studied
- Today's objective: **high level view** of software evolution as a **phenomenon**
 - what is it?
 - why is it important?
 - one or two conclusions
- Discussion limited to **E-type applications** and **systems** - outline of **main properties**, more **detail** in pre-prints, elsewhere

What are **E-type systems**?

E-type Systems

- **Solve problem, address application** in **real world**
- Definition of *E-type applications* follows
- *E-type software is model like reflection* of **real world** of **interest** and must remain so as both the **real world, software change**
- **Criterion** of **acceptability** for *E-type* systems is **satisfaction** of **stakeholder needs** - in contrast to *S-type* systems where **correctness** is **criterion**
- **Correctness meaningless** in **real world context**
- **Behaviour** during and **results** of **execution** determine **acceptability**
- **Overwhelming number** of applications and programs in **regular use** of type *E*

Intrinsic property of *E-type* systems that they must be **continually evolved** to remain **satisfactory**

General Definition of "Evolution"

- **Process** of **discrete progressive change** over **time** in characteristics, attributes, properties of some **material** or **abstract**, **natural** or **artificial**, **entity** or **system** or of a **sequence** of these
- Changes are **progressive** when they result in **definable trend** of, for example, **increasing value**, **growing precision** or **better fit** to a changing **domain** or **context**
- **Entities** include **objects** or **population** (collection) of objects such as **natural species**, **societies**, **cities**, **artefacts**, **concepts**, **theories**, **ideas** or **systems** of these
- Whether any pair of these entities share **common evolution features** apart from those listed in definition, is not addressed here
- **Change process** will, in general, be **continuous** with relatively **slow** rate of change, or **discrete** with **individual incremental** changes, **small** relative to entity as a whole

Further discussion limited to **software context**

Concerns Arising from Study of Evolution in Software Context

- The **what** and **why** - "**evolution as a noun**" interpretation - studies its **nature** and that of evolution **process** and its **products**
 - **what** evolves?
 - **why** does it evolve?
 - what **drives, controls, constrains** evolution process?
 - do **process** and its **product**, the **evolving system**, have identifiable **attributes** or **patterns**?
 - what is **impact** of the above on **cost, effectiveness, timeliness, value** of evolution **processes** and their **products**?
 - etc., etc
- The **how** - "**evolution as a verb - to evolve**" interpretation - considers **implementation** of evolution process
 - what is **goal** of evolution activity?
 - **where** or which **part** must be evolved to attain **goal**?
 - what **methods, techniques** are likely to be **appropriate, effective**?
 - what **tools** would be **appropriate, effective**?
 - what **resources, skills** are required?
 - etc., etc.
- Each is important
 - **how** reflects, for example, **interests** of **business world**
 - mastery of **what** and **why** critical for **disciplined, directed how**
 - without that understanding **how advances** tend to be **ad hoc**

Areas of evolution

Most Widely Studied Example of Evolution in Software Context

- **Sequence** of **releases** of a software system
- **Satisfies** stated **definition** of evolution, **evolving** as discrete **sequence of entities**
- Evolution **concepts** and **phenomenological detail** emerged over thirty years period from **analysis, modelling, interpretation** of **empirical data**
- Data related to a number of **industrially developed** and **evolved** systems from variety of **application, evolution** environments
- Results provide **compelling empirical evidence** that the *why* and the *what* can be **usefully** studied, revealing **intrinsic** properties of evolution **phenomenon**

But release level is **not** only **area of interest**

Areas of Evolution in E-type Software Context - renumbered

- I **Ab initio** development of a system or **implementation of changes** to, for example, an **individual maintenance** or **upgrade release**
- II **Sequence of releases** or **versions** of a software system over its lifecycle
- III **Application**
- IV **Operational domain** of an application - as it impacts relevant evolution process
- V Software evolution **processes** as at:
area I - for **system development**, area II - for **release planning and management**
- VI **Interacting domains** - e.g. **organisational**, **market place**, **technology**
- VII Software process **models**

List a simplified structure

Simplification

- **Sub-areas** - examples indicated
- **Interdependencies** e.g.:
 - **software changes** drive **application** and **application domain changes**
 - **application** and **application domain changes** drive **software changes**
 - **domain, application, technology** evolution **drive** each other
 - etc., etc.
- **Other areas?**

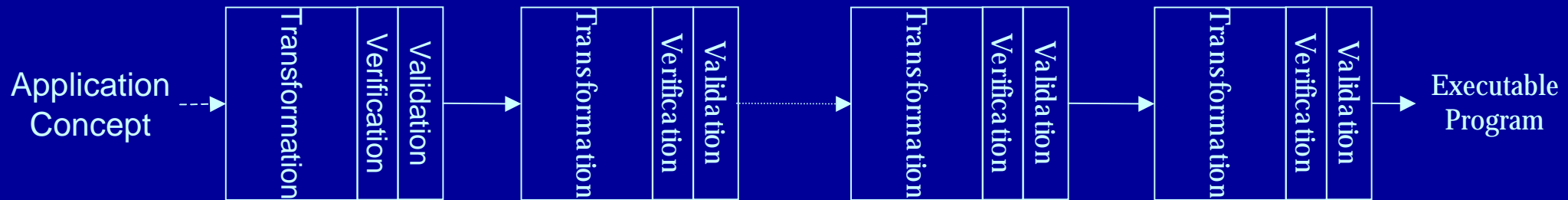
• **Area I - Development ab initio**

- For other than *trivial* systems, development is a **process** consisting of a series of **disparate steps** that transform **statement of objectives** - the application concept - into a shippable **product**
- **Progress** through sequence will, normally, include **iteration** within and over groups of steps
- **Process** described some 30 years ago by Wirth as **successive refinement** - expressing more limited concept than that recognised here
- Step sequence constitutes gradual **evolution** by successive **transformation** of starting **concept** into solution
- **Refines** increasingly detailed **descriptions** of desired **solution** into an executable program by processes of **abstraction, reification**
- Essentially a **learning** process. **Insight** acquired **during development** is applied, **iteratively** to **evolve detail** since one cannot, in general, devise fully **detailed solution** in advance

Clearly an evolution process

Example of Area I Paradigm

- **LST** software development paradigm comprise a sequence of **transformation steps** each of which, on completion, is followed by **verification** and **validation**



- Diagram provides a **top-level view**, that **hides** transformation and **structural detail** including **iteration** and **feedback loops** within or between steps
- Input to first step is **poorly defined, high level**. It is a statement of **purpose** of the **development**
- **Input** to a step is - as per accepted mathematical terminology - termed **step specification** and **output** a **model** or **implementation** of that specification
- **Output** of each step is **accepted**, after **verification** and **validation**, as **specification** of next step

Software **development process** an **evolution process**

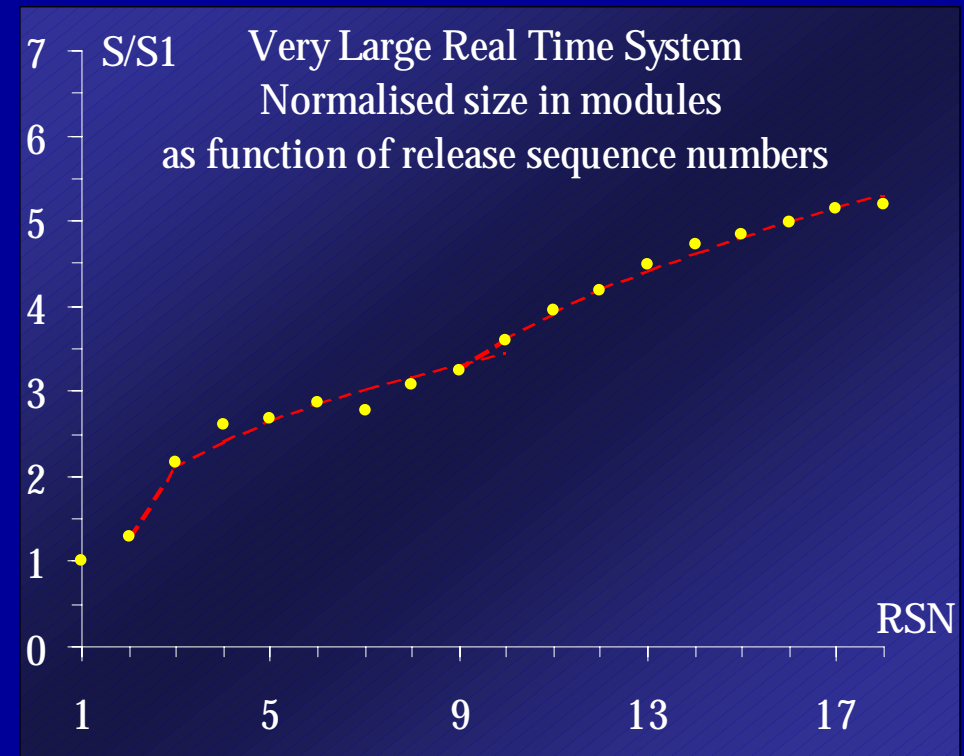
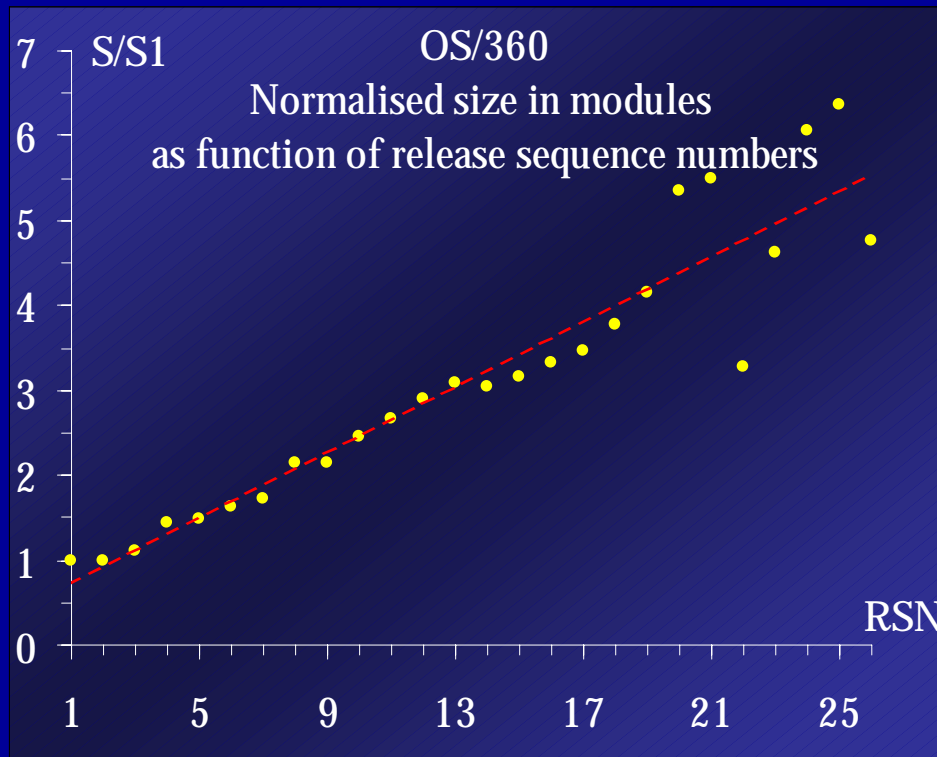
• **Area 2 - post first release evolution**

- For reasons elaborated elsewhere *E*-type systems will inevitably require **continual change**, to maintain compatibility with **external changes**
- **Implemented** as an **area I** activity but requires **separate** release **planning, management, control** activity or its equivalent - a **release process**
- Focus of **extensive empirical studies** over the years

System evolving in (possibly multiple) **series** of **usable steps**

• Example

- Two systems, one an **operating system** of the **70s**, other a very large **real time system** of the **90s**



- **Common features** are **striking**. Appear to reflect same **general phenomenon**
- **Differences** require **explanation** which, when achieved, will assist **development of theory of software evolution**

• **Area 3 - application evolution**

- E-type **application** evolution is driven by **several forces**
- **Internal**
 - unforeseen **consequences** of implementation **detail**
 - usage reveals **weaknesses** and **opportunities** for **functional** and **performance** improvement
- **External**
 - impact of **installation** and **usage changes domain** and, therefore, **requirements**
 - program **embedded** in **operational domain**, becomes **part of application** and, hence, contains implicit model of itself, a **recipe** for **continuing change**
 - **interactions** between users and **integration** of their **usage** reveals **improvement opportunities** in pursuit of, for example, **cost-effectiveness** and **extension of application**
- **Societal**
 - human **learning, competition, creativity, ambition** will suggest **changes, additions** and **justify** their **implementation**
- Hence, application evolution will also be **step incremental** triggered, in part, by **system** and **domain** evolution but with **human imposed delay**

• **Area 4 - evolution of operational domain**

- On being **installed** and **used** in its operational domain, application **system** becomes part of that domain and **application** an activity in it, hence, both become **part** of the real world
- Hence, as each evolves, the real world **changes with** them
- Human **interest** will **ensure** that operational domain derives **benefits** from usage, in some sense, hence its change **is** progressive, hence **evolutionary**

• **Area 5 - process evolution**

- More commonly termed **process improvement**
- Striving for increased **effectiveness, productivity, response time**, etc., leads, in absence of a theoretical base and framework, to process changes, in general **individually conceived**, that produce **incremental improvements**
- Occasionally, **paradigm changes** may make **wider impact** at **some level** and, at that level, **appear** more than incremental
- Their impact on the **full industrial process**, that **transforms** a **concept into an effective and reliable product**, is however, relatively small and, whence process improvement is also evolutionary

• **Area 6 - interacting domains**

- **Computers** used in the real world **operate** in a number of **different domains**
 - IT department
 - business domain
 - market place
 - technology domain
 - etc., etc.
- They **interact** with them, **provide information** to them, are subject to **pressure from them**
- **Mutual interactions**, as well as **external forces**, lead to **co-evolution** of all the domains though not, necessarily, at same **rate** or same **instance in time**
- **Interactions, dependencies, evolution** must be **understood, planned, managed**

• **Area 7 - evolution process models**

- **Process models** must be **evolved** with the **processes** they **reflect**
- Self evident but fundamental when adopting process **models** as **tool** for **process management, improvement**
- Raises issue whether models can be **reliably adapted, calibrated** to maintain them **consistent** with processes they are intended to **reflect, support**

• **Final Word**

- Have introduced seven **areas of evolution** in software context
- List presented a **simplification** in that there are **sub areas, relationships, linkages, interdependencies** between them
- **Alternative** breakdowns, perhaps based on other criteria, are certainly **possible**
- **FEAST** studies **focussed** on **area II** evolution and extent to which its **identified attributes** are shared by other areas is **not known**
- On the contrary, **reasoning** indicates, for example, **stark contrast** between manner of **area I, II** evolution on the one hand and **area VII** on the other

• **Brief Summary**

- FEAST studies focused on **area II**
- Message has been of **interdependencies** between all areas
- Need to develop **outline agreement** on **definitions, concepts, analysis**
- Then achieve degree of **understanding** of **all** areas and their **interactions**
- Growing **dependence** on computers makes society ever more dependant on **up to date, continually changed** software which, despite growing **complexity**, must remain compatible with **world as it is** and with co-evolving **domains**
- Theoretical **base** and **framework, theory of software evolution**, one **key** for success

Some FEAST Publications - See <http://www.doc.ic.ac.uk/~mml/feast>

A complete listing of FEAST papers is provided at and some may be accessed via <http://www.doc.ic.ac.uk/~mml/feast>

Rules and Guidelines

MM Lehman, *Rules and Tools for Software Evolution Planning and Management*, pos. paper, FEAST 2000 Workshop, Imp. Col., 10 - 12 Jul. 2000, DoC, Res. Rep. Nov 2000, a revised version to appear in *Annals of Software Engineering, Spec. Issue on Software Management*, vol. 11., 2001

MM Lehman, *The Future of Software - Managing Evolution*, inv. contr., IEEE Software, Jan-Feb. 1998, pp. 40-44

Theory

MM Lehman and JF Ramil, *Towards a Theory of Software Evolution - And Its Practical Impact*, invited talk, ISPSE 2000, Intl. Symposium on the Principles of Software Evolution, Kanazawa, Japan, Nov 1-2, 2000

Modelling of Evolution Processes

G Kahen, MM Lehman, JF Ramil and PD Wernick, *Dynamic Modelling in the Investigation of Policies for E-type Software Evolution*, ProSim 2000, International Workshop on Software Process Simulation and Modelling, 12 - 14 Jul. 2000, London, UK

BW Chatters, MM Lehman, JF Ramil, P Wernick, *Modelling a Software Evolution Process*, ProSim'99, Softw. Process Modelling and Simulation Workshop, Silver Falls, Oregon, 28-30 Jun. 1999, also as *Modelling a Long Term Software Evolution Process* in *J. of Softw. Proc.: Improv. and Practice*, 2000 v. 5, iss. 2/3, Jul. 2000, pps. 95-102

MM Lehman, DE Perry and JF Ramil, *On Evidence Supporting the FEAST Hypothesis and the Laws of Software Evolution*, Proc. Metrics'98, Bethesda, MD, 20-21 Nov. 1998, pp. 84-88

P Wernick and MM Lehman, *Software Process White Box Modelling for FEAST/1*, ProSim '98 Workshop, Silver Falls, OR, 23 Jun. 1998. As a revised version in *Journal of Systems and Software*, Vol. 46, Numbers 2/3, 15 Apr. 1999

MM Lehman and P Wernick, *System Dynamics Models of Software Evolution Processes*, Proc. Int. Wrkshp. on the Principles of Software Evolution IWPSE-98, ICSE-20, 20-21 Apr. 1998, Kyoto, Japan, pp. 6-10

MM Lehman, DE Perry, JF Ramil, WM Turski and P Wernick, *Metrics and Laws of Software Evolution - The Nineties View*, Proc. Fourth International Symposium on Software Metrics, Metrics 97, Albuquerque, New Mexico, 5-7 Nov. 97, IEEE Comp. Soc. or. n. PR08093, pp 20-32. Also as in K El Eman and N H Madhavji (eds.), *Elements of Software Process Assessment and Improvement*, IEEE CS Press, 1999

WM Turski, *A Reference Model for the Smooth Growth of Software Systems*, IEEE Trans. Softw. Eng., v. 22, n. 8, Aug. 1996, pp. 599 - 600

Resource Estimation

JF Ramil and MM Lehman, *Exploring Cost Estimation Models in the Context of Continuing Software Evolution*, FESMA-AEMES Software Measurement Conference 2000 "Management Excellence through IT Measurement", Madrid, Spain Oct. 18-20, 2000 Lessons Learnt (e.g., Modelling Methodology)

JF Ramil and MM Lehman, *Metrics of Software Evolution as Effort Predictors - A Case Study*, Proc. ICSM 2000, Int. Conference on Software Maintenance, 11-14 Oct. 2000, San Jose, CA Component-based and Integration-intensive Processes

JF Ramil, *'Why COCOMO Works' Revisited or Feedback Control as a Cost Factor*, pos. paper, FEAST 2000 Workshop, Imp. Col., London, 10 - 12 Jul. 2000, 5 pps. Also as Res. Rep. 2000/3, Dept. of Comp. Imp. Col., Feb. 2000

Other

MM Lehman and JF Ramil, *Software Evolution Phenomenology and Component Based Software Engineering*, to appear in *IEE Proc. Softw.*, sp. issue on Component Based Software Engineering, scheduled for Dec. 2000, earlier version as Tech. Rep. 98/8, Imperial College, London, Jun. 1998

JF Ramil (ed.), *Preprints of FEAST 2000 International Workshop on Feedback and Evolution in Software and Business Process*, Imp. Col., London, 10 - 12 Jul. 2000, 124 pp.

JF Ramil, MM Lehman and G Kahen, *The FEAST Approach to Quantitative Process Modelling of Software Evolution Processes*, Proc. PROFES'2000 2nd International Conference on Product Focused Software Process Improvement, Oulu, Finland, 20 - 22 Jun. 2000, in Frank Bomarius and Markku Oivo (eds.) LNCS 1840, Springer Verlag, Berlin, 2000, pp. 311 - 325.

MM Lehman, *Feedback in the Software Evolution Process*, Keynote Address, CSR Eleventh Annual Workshop on Software Evolution: Models and Metrics. Dublin, 7-9 Sept. 1994, Workshop Proc., Information and Software Technology, sp. is. on Software Maintenance, v. 38, n. 11, 1996, Elsevier, 1996, pp. 681 - 686

General Bibliography

A listing of Prof Lehman's papers and other material is provided at and some may be accessed via <http://www.doc.ic.ac.uk/~mml>

References indicated with an '*' have been reprinted in Lehman MM & Belady LA, *Program Evolution—Processes of Software Change*, Acad. Pr., London 1985

- *Belady LA & Lehman MM, *An Introduction to Program Growth Dynamics*, in Statistical Computer Performance Evaluation, W Freiburger (ed), Academic Press, New York, 1972, pp. 503 - 511
- Boehm BW, *Software Engineering*, IEEE Trans. on Comp., v. C-5, n. 12, Dec. 1976, pp. 1226 - 1241
- *id.*, *A Spiral Model of Software Development and Enhancement*, Computer, v. 21,, May 1988, pp. 61 - 72
- *id.*, *Software Engineering Economics*, Englewood Cliffs, N.J, Prentice-Hall, 1981
- Brooks FP, *No Silver Bullet - Essence and Accidents of Software Engineering*, Information Processing 86, Proc. IFIP Congress 1986, Dublin, Sept. 1-5, Elsevier Science Pubs. (BV), (North Holland), pp. 1069 - 1076
- *Lehman MM, *The Programming Process*, IBM Research Report RC 2722, IBM Research Centre, Yorktown Heights, NY, Sept. 1969, also in *Program Evolution—Processes of Software Change q.v.*
- **id.*, *Programs, Cities, Students - Limits to Growth?.*, Imperial College. Inaugural Lecture Series, v. 9, 1970 - 1974, also. in [GRI78], pp. 42 - 69, Lehman MM & Belady LA, 1985, pp. 133 - 163, also in *Program Evolution—Processes of Software Change q.v.*
- **id.*, *Laws of Program Evolution - Rules and Tools for Programming Management*, Proc. Infotech State of the Art Conf., Why Software Projects Fail, - Apr 9 - 11 1978, pp. 11/1 - 11/25
- **id.*, *Programs, Life Cycles and Laws of Software Evolution*, Proc. IEEE Sp. Iss. on Softw. Eng., v. 68, n. 9, Sept 1980, pp. 1060 - 1076
- Lehman MM, Stenning V & Turski WM, (1984). *Another Look at Software Design Methodology*, ICST DoC Res. Rep. 83/13, Jun 1983. Also, *Software Engineering Notes*, v. 9, no 2, Apr 1984, pp. 38 - 53
- Lehman MM & Belady LA, *Program Evolution,- Processes of Software Change*, Academic Press, London, 1985, 538 p.
- *id.*, *Evolution - The Cause of Iteration*, Third Process Workshop, Breconridge, CO, Nov. 1986. In *Iteration in the Software Process - Proc. 3rd Int. Process Wrkshp.*, Dowson M (ed), IEEE Comp. Soc. Press, Mar. 1987, pp. 29 - 32
- Lehman MM, *Process Models, Process Programs, Programming Support*, Inv. Resp. To A Keynote Addr. By Lee Osterweil, Proc. 9th Int. Conf. on Softw. Eng., Monterey, CA, 30 Mar. 2 Apr 1987, IEEE Comp. Soc. pub. n. 767, IEEE Cat. n. 87CH2432-3, pp. 14 - 16
- *id.*, *Uncertainty in Computer Application*, Comm. of the ACM, Vol. 33, No. 5, May 1990, pp. 584-586
- *id.*, *Uncertainty in Computer Application and its Control through the Engineering of Software*, J. of Softw. Maint., Res. & Pract., v. 1, 1 Sept 1989, pp. 3 - 27
- *id.*, *Models and Modelling in Software Engineering*, Ency. of Softw. Eng., J Marciniak (ed), Wiley and Co, 1994, vol. 1, pp. 698 - 702
- *id.*, *Software Evolution*, loc cit, vol. 2, pp. 1202 - 1208
- Osterweil L, *Software Processes are Software Too, Iteration in the Software Process*, Proc. of the 3rd Int. Proc. Worksh., Breckenridge, CO, 17 - 19 Nov. 1986, IEEE cat. n. TH0184-2, IEEE Comp. Soc. order n. 709, 1987, pp. 79 - 80
- Perry DE, *Policy and Product-Directed Process Instantiation*, Proc. of the 6th Int. Softw. Process Workshop, 28-31 October 1990, Hakodate, Japan
- Rajlich VT and Bennet KH, *A Staged Model for the Software Life Cycle*, Computer, July 2000, pp. 66 - 71
- Turski WM, *And No Philosophers' Stone Either*, Inf. Processing 86, Proc. IFIP Congr., Dublin, Sept. 1 - 5, 1986, Elsevier Sci. Pubs, London, pp. 1077 - 1080
- Wilkes M V, Wheeler D J & Gill S, *The Preparation of Programs for an Electronic Digital Computer*, Addison Wesley Press Inc., 1951, 167 pp.
- Wirth N, *Program Development by Stepwise Refinement*, CACM, v.14, n.4, Apr 1971, pp.221-227
- *Woodside CM, *A Mathematical Model for the Evolution of Software*, J. of Sys. and Softw. vol. 1, no. 4, Oct 1980, pp. 337 - 345 Zurcher FW and Randell B, *Iterative Multi-Level Modelling - A Methodology for Computer System Design*, IBM Res. Rep. RC 1938, Nov. 1967, IBM Res. Centre, Yorktown Heights, NY 10594. Also in *Information Processing 67*, Proc. IFIP Congr. 1968, Edinburgh, Aug. 1968, pp. D138 - 142