

***Towards a Theory of Software Evolution
- And its Practical Impact***

ISPSE 2000

Japan

1- 2 November 2000

M M Lehman

Department of Computing

Imperial College of Science, Technology and Medicine

180 Queen's Gate

London SW7 2BZ

tel. +44 (0) 20 7594 8214

fax. +44 (0) 20 7594 8215

mml@doc.ic.ac.uk

<http://www-dse.doc.ic.ac.uk/~mml>

Genesis of an Insight

- Kyoto - IWSPE98 two day **Principles of Software Evolution** International Workshop
- Mixed bag of papers ranging from the **theoretical** to the **empirical**
- Presentations **remote** from **concepts** and **investigations** of evolution that have occupied me on and off for some 30 years
- **Penny** only **dropped** next day
- With one or two exceptions, papers addressed **theories**, **abstractions**, **models**, **languages**, **activities**, **methods**, **tools** for **effective**, **reliable software evolution**
- Focus on **evolution** as nounal form of **verb** - the **how** of system evolution, its **improvement** and **effectiveness**
- Little to do with my approach to study of software **evolution**
- Focus on **evolution** as a **noun** - the **what** and **why**

The 'How' of Evolution

- Studied, though not by that name, since **beginnings of programming** - Wilkes, Wheeler and Gill, *The Preparation of Programs for an Electronic Digital Computer*, Addison Wesley Press Inc., 1951, 167 pp.
- Concentration on **means** to meet **needs** - **process, activities, tools**
- Because of continuing **demand** and **pressure** for
 - functional **extension**
 - ever **more applications**
 - improved **quality**, especially **reliability**
 - improved **changeability, responsiveness**
 - software and business **integration**
 - etc., etc.
- Has remained main **focus of interest** to this day in **academic & industrial research & process improvement** communities
- Reason obvious: need to develop **rapid response** to **market demand** and **competitive pressure**, promise of **quick return**
- Also: **not** obvious or generally recognised that software evolution is sufficiently **disciplined** that it is a **legitimate** and **worthy object of study**

Is Such Absolute Concentration Justified?

- Total **concentration** on activity of evolution considered harmful
- Understanding **causes** of a phenomenon, its **nature** and **behaviour** can contribute significantly to its successful **pursuit, improvement, effective management** and **control**
- Some would say that such understanding is **essential** for major long term improvement
- If **identified** and **interpretable**, a **common pattern** of evolutionary behaviour over a range of systems, organisations, processes etc. provides indication of the source of evolutionary forces and, hence, suggestions for their **control**
- One must **understand** a phenomenon before one can hope or expect to **systematically** and **continually** control or improve it

Need to consider the **what** and **why** of **evolution**

The *What* and *Why* of Evolution

- **Continuing maintenance** and **evolution, universal experience** since start of computer age
- **Phenomenology** and underlying **causes** now reasonably well understood
- The **need** for and **reality** of **continuing evolution** an established **fact**
- Encapsulated in first and sixth **Laws of Software Evolution**
- Generally supported by **black box** and **white box** studies of the FEAST projects
- **Commonalties in behaviour** of several systems from a variety of **application domains, implementation environments, system characteristics** suggests some common cause, an underlying **phenomenon**

Identification and **analysis vital** for **further advances** in **software technology**

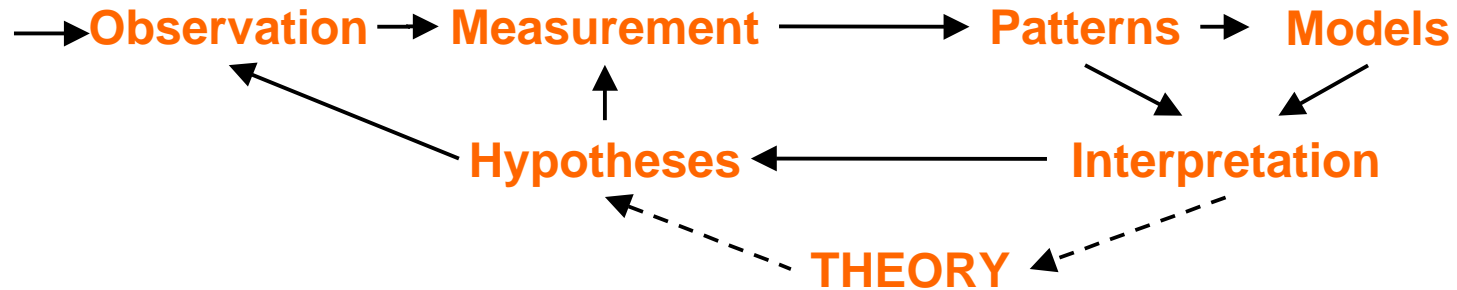
Conclusion

- Verbal and nounal views are **complementary**
- Both **legitimate**
- Both **important**
- Both **worthwhile**
- But **understanding why** and **what** of a **phenomenon** is of great help in seeking to **master** and **improve** it

Software evolution phenomenon *per se* **candidate** for **serious study**

Phenomena as Objects of Study

- Apply **scientific method** using both **empirical** investigation and **theory formation**



- Followed by **FEAST** project in its **empirical studies** of the **role** and **impact** of **feedback** in **industrial** evolution **processes**
- Theory** now **within reach**
- Two levels
 - **Observation** level - reflection of **phenomenology**
 - **Theory** level - mathematical **abstraction**

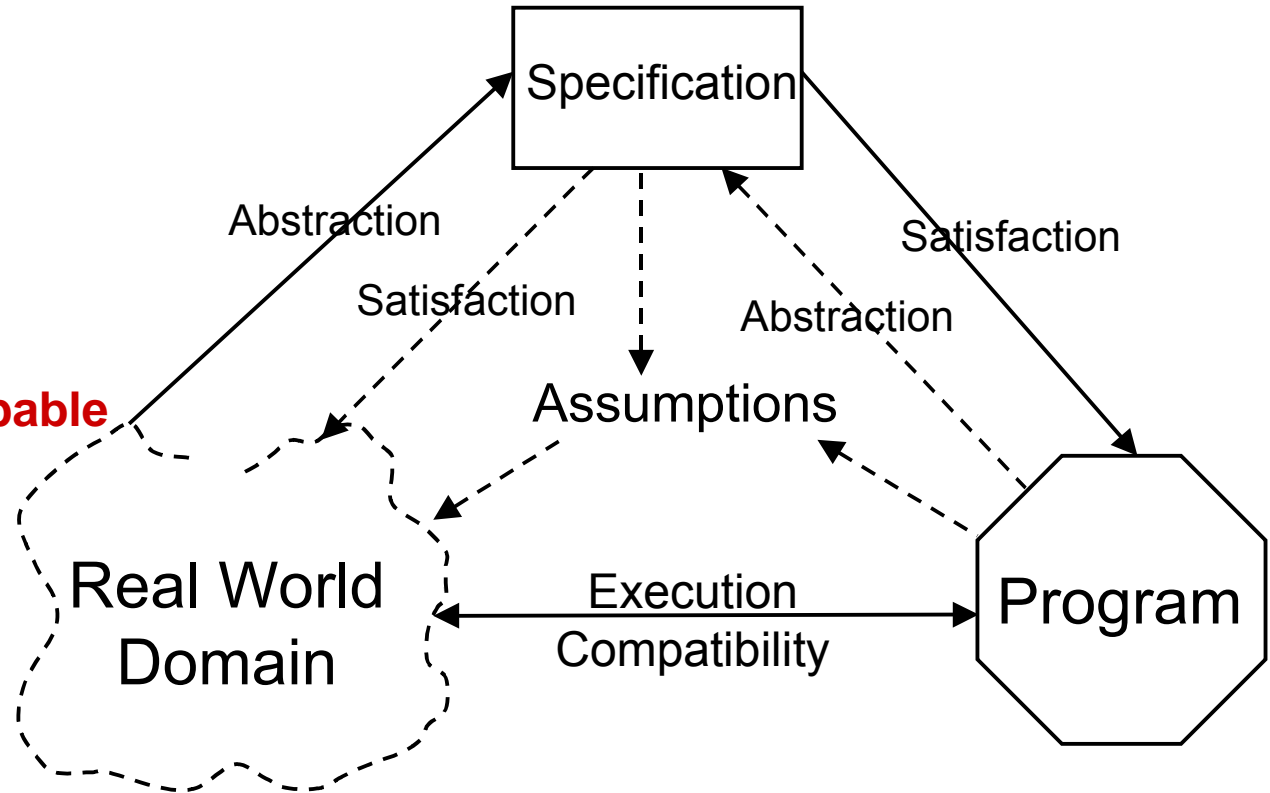
Brief History of Observation Level Study

- 1968/9 - **IBM programming process** study
- ~ 30 years of **observation** and **interpretation** have led to:
 - 8 **Laws** of Software Evolution
 - SPE program **classification**
 - Principle of **Uncertainty**
 - **FEAST hypothesis**
 - FEAST/1 and FEAST/2 **projects** and **findings**
 - Inverse square **growth model** - example
 - System dynamics **feedback models**
- **Some** evolution **drivers** identified
- **Rules, guidelines** for long term evolution **planning, management** formulated
- Together with the work of others there now exists a significant **body of knowledge**

Basis for **observation level theory**

Background to Observation Level Study

- **Real world domain** and **program** as **models** of the **specification**
- **May have properties not addressed by specification and incompatible with one another**



- **Assumptions inescapable**

- **Gradual invalidation**

- **Evolution inevitable**

E-type System Evolution Process

- **Trigger**

Need/Demand/Opportunity not satisfied by current system
leads to

Preliminary statement of system **purpose** or required **change**

- **Yields**

Application or **change concept** - high level **requirement**
Domains of application

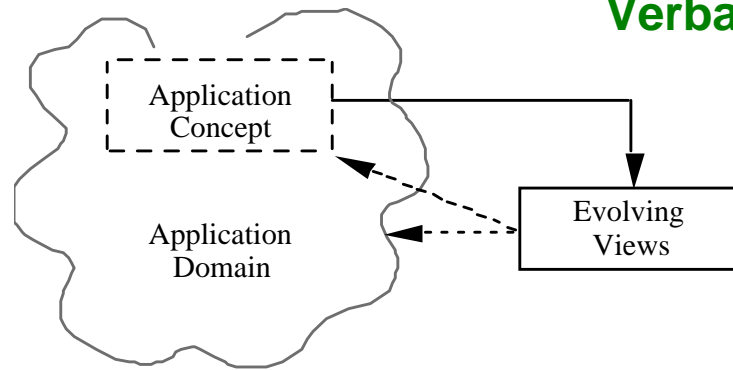
- Both possess **unbounded** number of **attributes**

- **Initially both:**

- **ill defined**
- **not** fully or precisely **verbalised**
- not explicitly **bounded**

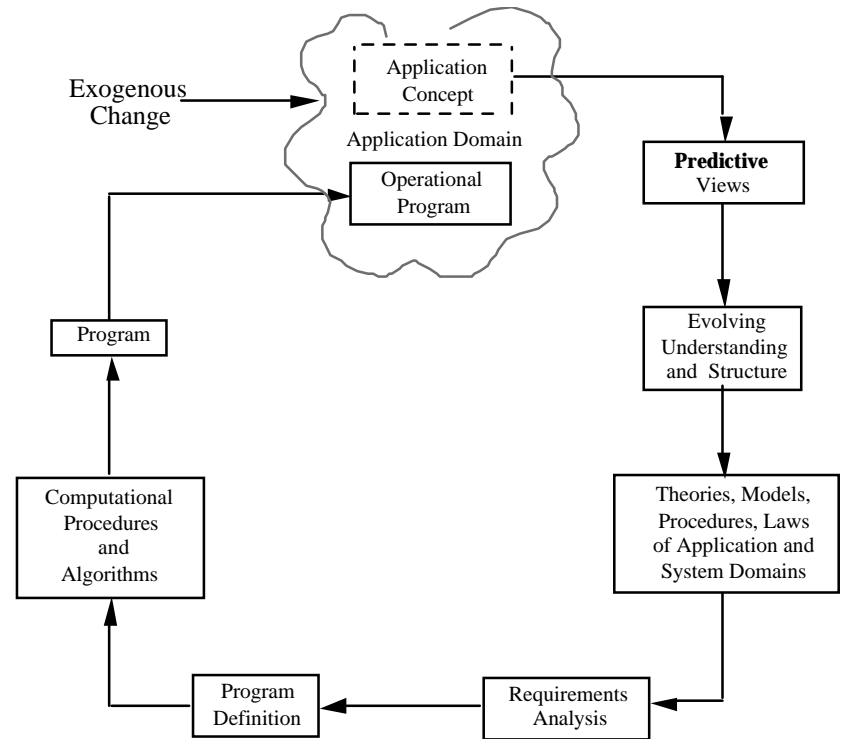
- Ultimate operational domain - **universe**

- **Initial** process **steps**
Verbalise, bound



Closing the Loop

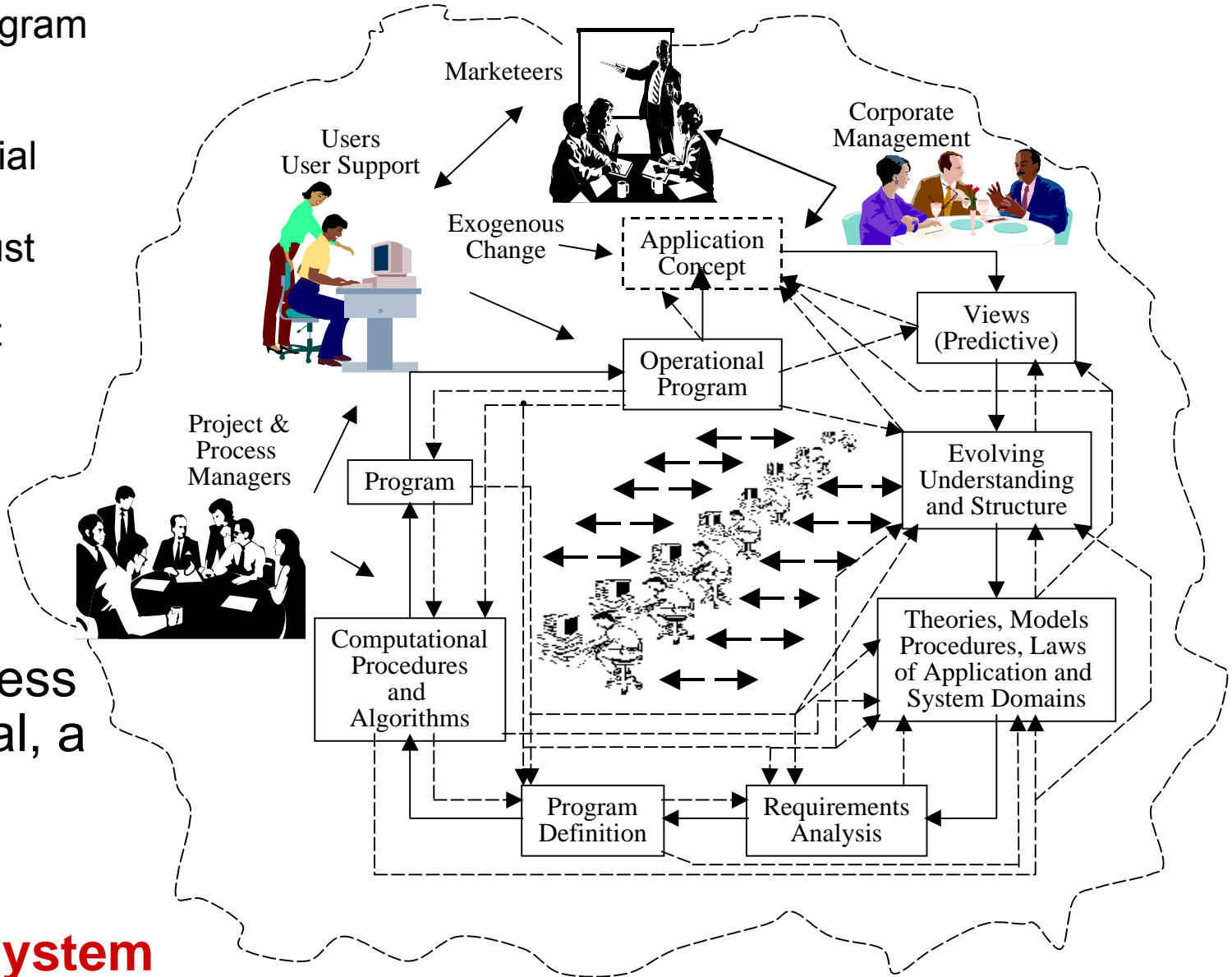
- Installation of system **changes domain**
- **Use** triggers **changes** in application **concept, behaviours**
- System **contains** implicit **model of itself**
- **Installation**, introduction into **usage** closes major **feedback loop**
- **Fact of life** for process **improvement**



Driver of continuing software system evolution

More Realistic Picture

- Previous diagram a **fiction**
- **Not** sequential
- **More** than just technical development activity



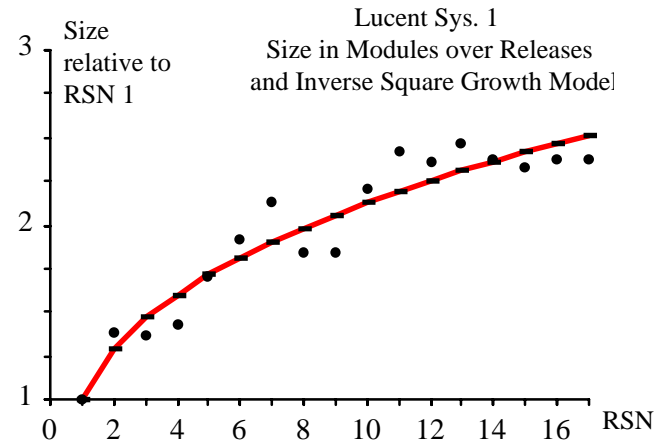
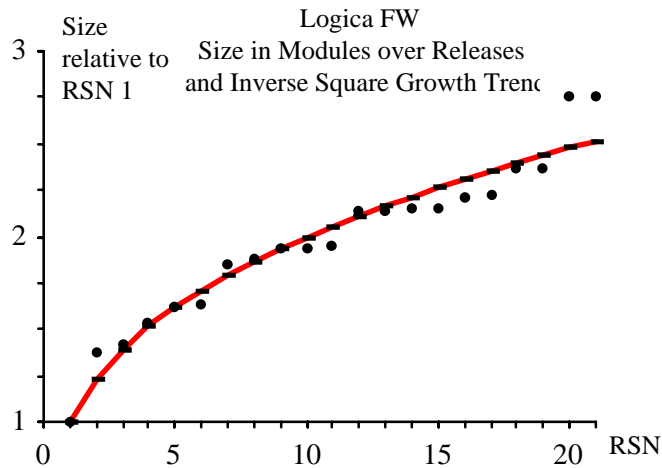
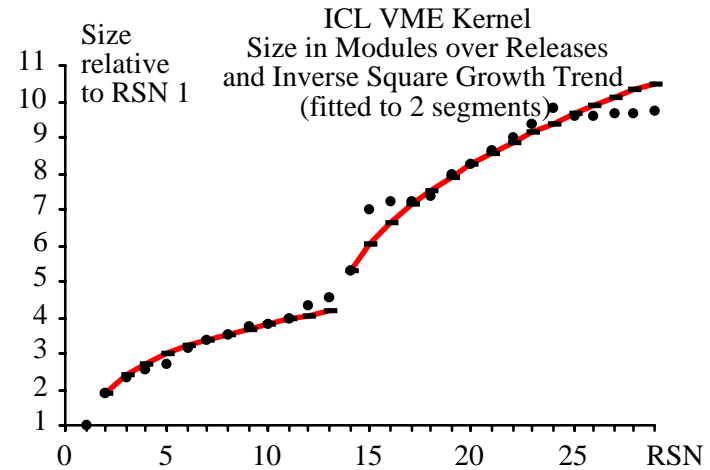
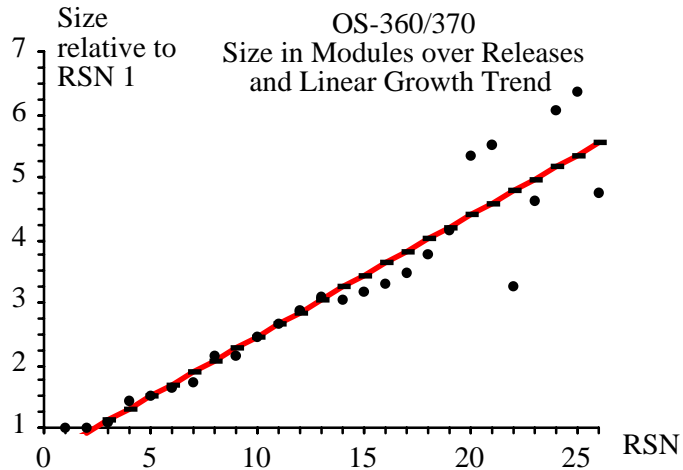
Global process is, in general, a multi-level multi-loop multi-agent feedback system

From IBM Process Study to FEAST - inputs to theory development

- **Growth dynamics** of **OS/360**
- **Feedback** driven and controlled - **momentum/inertial** effects
- **Phenomenon** of software evolution
- **Laws** of software evolution
- **SPE** program classification
- **Software uncertainty principle**
- **FEAST hypothesis**
- **FEAST projects** - extension, insight, understanding

Empirical support for, and extension of, past results
Trigger for theory formation

An Example of Empirical Analysis



- **Inverse square** model: $S_i = S_{i-1} + \hat{E}/S_{i-1}^2$ ($2 \leq i \leq n$) where **i** represents **release sequence number** and **n** is the total **number of releases** considered
- **Ripple** - suggested **feedback stabilisation**

The Laws

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

Observation Level Theory

Includes

- Basic **concepts** and **terms**
- **Abstractions** of **behavioural patterns**, **process invariants** from observations
- **Common behavioural descriptors**, **models**, **interpretations**
- Suggested **extraction**, **formation** of behavioural **axioms**
- And **inferences** from from these, in turn, as potential **theorems**
- **Develop proof outlines**
- **Iterate, extend**
- **Derive, formulate initial** practical **implications**
- **Initiate theory level theory formation**

To produce an initial theory of software evolution!
and of the **software process?**

Theory will ...

- **Provide** an **encapsulated world view**
- Facilitate **reasoning** about the **software process** and **software evolution**
- **Framework** for **empirical studies**
- **Provide explicit formal, informal** links between **observation** and **good practice**
- And **justification** for **investment** in **good practice** - combat **inbred scepticism**
- **Identify** sources of **evolutionary pressures**
- **Predict** behaviour of evolutionary **attributes** and **identify means** for their **control**

Goal

Derivation of **qualitative** & **quantitative** rules & **guidelines** for **evolution management, control, direction**

Observation Level Theory - three potential axioms

- *Note*
 - preliminary **definitions** determined, need **refinement**, **completion**
 - **real world domain** abstracted by a specification and program solving defined problem in that domain are each a **model** of the **theory** that the specification constitutes
- **Observation (axiom 1)**: Real world is **dynamic**, properties may **change** at any time
- **Observation (axiom 2)**: May be partitioned in an **unbounded** number of ways into domains that, in general, each possess an unbounded number of **properties**
- **Observation (axiom 3)**: *E*-type **programs** as such (as distinct from such programs in execution) are **finite**
- Defn. 9: Property displayed by *E*-type **program** that is **not** a property of specification reflects an **assumption about the real world**

Observation Level Theory - five potential inferences

- **Inference (step 1): Assumptions** reflected in specification **may become invalid** so causing it to become invalid as abstraction of operational domain
- **Inference (step 2): Real world** from which specification of E -type program is abstracted has **unbounded** number of properties
- **Inference (step 3):** The **number** of *stated distinct* behavioural properties of an E -type program implied by its specification is **finite**
- **Inference (step 4):** An E -type program is **essentially incomplete** since it cannot reflect an **unbounded** number of real world **properties**
- **Inferen. (theorem): Behaviour** of an E -type program when executed is **inherently uncertain** and cannot be guaranteed to be acceptable

Outline proof of **Principle of Software Uncertainty**

Practical Implications

Some examples

- Explicitly specify **initial bounds** for operational **domain** and, for program **functionality** and for non functional characteristics as **early** development step
- **Capture, document, structure, retain assumptions** during all process **activity**
eg. bounding, specification, design, implementation, integration, validation
- **Validate implied assumptions with** validation **of specifications**
- **Develop tool support** for **recording assumptions** in **structured form** classified by appropriate categories, throughout the process
- **Document initial** and **change assumptions, design, implementation rationale**
- Periodically **review** and **validate assumption set**

Management of assumptions illustrates many direct practical lessons to be derived from **theory**

Theory - Level Theory

- To be addressed under **current funding application** - the **TheSE** project
- Proposal to use **modal action logic** to develop **abstract** theory
 - cannot discuss further today
- **Co-investigators** - **Chris Hankin, Tom Maibaum**
- **Collaborators** - **Deutsch Bank, Ericsson, IBM, ICL, MoD - DERA**
- **Funding** - **UK EPSRC**

Final Remarks

- Accumulation of **knowledge** and **understanding** derived from the FEAST and related studies of evolution as an **Object of Study** is leading to the development of a **theory of software evolution**
- Could lead to a **theory of software process**
- **Theory** intended in **strict sense**
- Development of such a theory would certainly **demonstration** that is an object worthy of study
- **Extension** of study to cover **component based** and COTS intensive SE
- That the **'what'** and **'why'** of **evolution** are **meaningful** questions that can be systematically answered
- That its study is **worthwhile**

References

- [1] Belady LA and Lehman MM, "An Introduction to Program Growth Dynamics", in *Statistical Computer Performance Evaluation*, W. Freiburger (ed.), Acad. Press, New York, 1972, pp. 503 - 511.
- [2]* Bennett KH and Rajlich VT, "Software Maintenance and Evolution: A Roadmap", in Finkelstein, A. (ed.), *The Future of Software Engineering*, 22nd ICSE, June 2000, Limerick, Ireland, pp. 73 - 87
- [3] Box G, and Luceño A, *Statistical Control by Monitoring and Feedback Adjustment*, Wiley, New York, 1997, 327 pp
- [4] Cook S, Ji H and Harrison R, "Software Evolution and Software Evolvability", working paper, Univ. of Reading, August 2000
- [5] Dijkstra ED, "Go To Statement Considered Harmful", *Comm. ACM*, v. 11 March 1968, pp 147 - 148
- [6] *Preprints of the three FEAST Workshops*, Lehman MM (ed.), Dept. of Comp., ICSTM, 1994/5
- [7] *Preprints of FEAST 2000 International Workshop on Feedback and Evolution in Softw. and Business Processes*, Ramil JF (ed.), Dept. of Comp., Imperial College, London, 10-12 July 2000, 124 pp. <http://www-dse.doc.ic.ac.uk/~mml/f2000>
- [8] *FEAST Projects Web Site*, Dept. of Comp., Imp. Col., London, UK, <http://www-dse.doc.ic.ac.uk/~mml/feast/>
- [9] Fordham RG, "Software Development Challenges for the 2000's", Keynote Address, Proc. *PROFES'2000 2nd International Conf. on Product Focused Softw. Process Improvement*, Oulu, Finland, 20 - 22 Jun. 2000, in Frank Bomarius and Markku Oivo (eds.) LNCS 1840, Springer Verlag, Berlin, 2000, p. 3
- [10] Gilb T, "Evolutionary Development", *ACM Softw. Eng. Notes*, April 1981
- [11] *Proc. Int. Wrkshp. on the Principles of Software Evolution IWPSE-98*, ICSE-20, 20-21 Apr. 1998, Kyoto, Japan
- [12] *Int. Symposium on the Principles of Software Evolution*, this volume.
- [13] Kemerer CF and Slaughter S, "An Empirical Approach to Studying Software Evolution", *IEEE Trans on Softw. Eng.*, v. 25, n. 4, Jul./Aug. 99, pp. 493 - 509
- [14]* Lehman MM, "The Programming Process", IBM Res. Rep. RC 2722, IBM Res. Centre, Yorktown Heights, NY, Sept. 1969.
- [15]* Lehman MM, "Programs, Cities, Students, Limits to Growth?", Inaugural Lecture, May 1974. Publ. in *Imp. Col of Sc. Tech. Inaugural Lect. Series*, v. 9, 1970 - 74, pp. 211 - 229. Also in *Programming Methodology*, (D Gries, ed.), Springer, Verlag, 1978, pp. 42 - 62
- [16]* Lehman MM, "Human Thought and Action as an Ingredient of System Behaviour", in *The Encyclopaedia of Ignorance*, R Duncan and M Weston-Smith (eds.), Pergamon Press, London, 1977, pp. 347 - 354
- [17] Lehman MM, "A Further Model of Coherent Programming Models", in, *Proc. of the Software Process Workshop*, Potts C (ed.), Egham, Surrey, UK, Feb. 1984. IEEE cat. no. 84CH2044-6, Comp. Soc., Washington D.C., order n. 587, Feb. 1984, pp. 27 -35
- [18] Lehman MM, and Belady LA, *Program Evolution - Processes of Software Change*, Acad. Pr., London, 1985
- [19] Lehman MM., "Uncertainty in Computer Application and its Control through the Engineering of Software", *J. of Software Maintenance, Research and Practice*, v. 1, 1 September 1989, pp. 3 - 27
- [20] Lehman MM, "Uncertainty in Computer Application", Technical Letter, *Comm. ACM*, v. 33, n. 5, pp. 584, May 1990
- [21] Lehman MM, "Feedback in the Software Evolution Process", Keynote Address, *CSR Eleventh Annual Wrksh. on Softw. Ev. - Models and Metrics*. Dublin, 7-9th Sep. 1994. Also in *Info. and Softw. Tech.*, spec. iss. on Softw. Maint., v. 38, n. 11, 1996, Elsevier, 1996, pp. 681 - 686
- [22] Lehman MM and Stenning V, "FEAST/1: Case for Support Part 2", Department of Computing, Imperial College, London, UK, Mar. 1996. Available from links at [8]
- [23] Lehman MM, "Laws of Software Evolution Revisited", *Proc. EWSPT96*, Oct. 1996, LNCS 1149, Springer, 1997, pp. 108 - 124

References

- [24] Lehman MM, "FEAST/2: Case for Support Part 2", Department of Computing, Imperial College, London, UK, Jul. 1998. Available via links at [8]
- [25] Lehman MM, "Rules and Tools for Software Evolution Planning and Management", *FEAST 2000 Pre-prints*, Imperial College, London, 10-12 July 2000. Available via links at [7]
- [26] Lehman MM, "Approach to a Theory of Software Process and Software Evolution", *FEAST 2000 Pre-prints*, Imperial College, London, 10-12 July 2000. Available via links at [7]
- [27] Lehman MM, "Evolution as a Noun and Evolution as a Verb", *SOCE 2000 Workshop on Software and Organisation Co-evolution*, 12 - 13 Jul. 2000, Imperial College, London. Available via links at [8].
- [28] Lehman MM, Proposal in Preparation, Department of Computing, Imperial College, London, UK, Sept. 2000.
- [29] Publication listings available from links at <http://www-dse.doc.ic.ac.uk/~mml>
- [30] Naur P and Randell B (eds.), *Software Engineering*, Report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7th to 11th Oct. 1968, Jan. 1969, 231 pps
- [31] Parnas D, "On the Criteria to be Used in Decomposing Systems into Modules", *Comm. ACM*, v. 15, Dec. 1972, pp 1053 - 1058
- [32] Pfleeger SL, "The Nature of System Change", *IEEE-Softw.* v. 15, n. 3; May-June 1998; pp. 87-90
- [33] *SEBPC Systems Engineering for Business Process Change*, EPSRC Res. Prog. <http://www.ecs.soton.ac.uk/~ph/sebpc/>
- [34] Shaw M and Garlan D, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, 1996
- [35] Shrager J and Langley P (eds.), *Computational Models of Scientific Discovery and Theory Formation*, Morgan Kaufmann Publishers, Inc, San Mateo, CA, 1990, 498 pps.
- [36] *SOCE 2000 Workshop on Software and Organisation Co-evolution*, Imperial College, 12 - 13 July, 2000
- [37] Turski WM, "Specification as a Theory with Models in the Computer World and in the Real World", in P. Henderson (ed.), *System Design, Infotech State of the Art Report*, se. 9, n. 6, 1981, pp 363 - 377
- [38] Turski WM and Maibaum T, *The Specification of Computer Programs*, Addison Wesley, London, 1987, p. 278
- [39] Turski WM, "An Essay on Software Engineering at the Turn of the Century", Invited Talk, *ETAPS*, Berlin, March, 2000, pp. 108 - 124
- [40] Wirth N, "Program Development by Step-wise Refinement", *Comm. ACM*, v. 14, n. 4, Apr. 1971, pp 221 - 227

General Bibliography

- 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
- Brookes 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, *Programs, Cities, Students - Limits to Growth?.*, Imperial College. Inaugural Lecture Series, v. 9, 1970 - 1974, also. in [GRI78], pp. 42 - 69, Lehman and Belady, 1985, pp. 133 - 163
- *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.
- 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.*, *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
- *id.*, *Software Engineering, the Software Process and Their Support*, IEE Softw. Eng. J. Spec. Iss. on Softw. Environments. and Factories, Sept 1991, v. 6, n. 5, pp. 243 - 258
- *id.*, *Software - Promise and Threat*, Inv. paper, Proc. Software Engineering in the Nineties., Software Eng. Res. Centrum, Utrecht, The Netherlands, Oct. 1988. Also, inv. paper, Shell Conf. on Logistics, Appeldoorn, Neths., Nov. 1988, Pergamon Press, 1989, pp.172 - 183
- *id.*, *Uncertainty in Computer Application and its Control through the Engineering of Software*, J. of Softw. Maint., Res. and Pract., v. 1, 1 Sept 1989, pp. 3 - 27
- *id.*, *Uncertainty in Computer Application*, Technical Letter, CACM, vol. 33, no. 5, pp. 584, May 1990
- *id.*, • *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
- Turski WM, *And No Philosophers' Stone Either*, Information Processing 86, Proc. IFIP Congr., Dublin, Sept. 1 - 5, 1986, Elsevier Sci. Pubs, London, pp. 1077 - 1080
- [wil51] Wilkes M V, Wheeler D J and 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 and in Lehman and Belady 1985, pp. 339 - 354