

Software Evolution - Its Source, Nature and Control

Computing Department Seminar
The Open University
Walton Hall, Milton Keynes
11 October 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 a **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
- **Present** - **FEAST** studies - **Feedback, Evolution And Software Technology**
 - **impact** of **feedback** on **global** software **process** and on **evolution** of its **product**
 - in collaboration with **ICL, Logica, MoD-DERA, Lucent Technologies, Matra-BAe, BT**
 - **evolution** of systems of **different: sizes, application areas, development environments, processes**
- **Future**
 - **explore** FEAST concepts and results in context of **re-use, component, COTS** based systems
 - **develop operational system dynamics models** of **global** software processes
 - **initiate** development of **formal theory** of software process evolution
- **Observations** over a 30 year period support generality of **continuous evolution** of **E-type** systems in distinction to **S-type** systems, **static by definition**

What are these software types?

S-type Systems

Definition, Properties

- Program properties **formally specified** to:
 - prevent **ambiguity**
 - facilitate **reasoning**
 - demonstrate **correctness**
 - etc., etc.
- Criterion of **acceptability** is **verified correctness** relative to specification
- **Unsatisfactory** behaviour/results in execution after verification implies **unsatisfactory specification** - **incorrect** or **inappropriate**
- System **validation** a matter of opinion/judgement assumption - **beauty contest**
- Fixing a program found to be **unsatisfactory** in use requires change to **specification** followed by derivation of **new program** - **new** by **definition** even when obtained by modification of original code
- **S-type** programs as **bricks** for construction of software systems

Programs in the **real world**

E-type Systems

Definition

- **Systems** solving **problem** or addressing **application** in real world

Properties

- **Correctness** w.r.t. **real world meaningless**
- **Behaviour** and/or **results** of **execution** determine **acceptability**
- **Satisfaction** of **stakeholders' needs** is ultimate **criterion** of **validity**
- System a **model like reflection** of real world of **interest**
- Real world **dynamic** with application and operational **domains, needs** and **desires** of user community, **changing continually**
- Requires **continuing** system **evolution** to **maintain** system **validity, value**

What is evolution?

General Definition

- **Evolution** is a **process** of **discrete progressive change** over **time** in the characteristics, attributes or properties of some **material** or **abstract, natural** or **artificial, entity** or **system** or of a **sequence** of **entities** or **systems**
- Changes are **progressive** when they result in **definable trend** of, for example, **increasing value**, **growing precision**, **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**

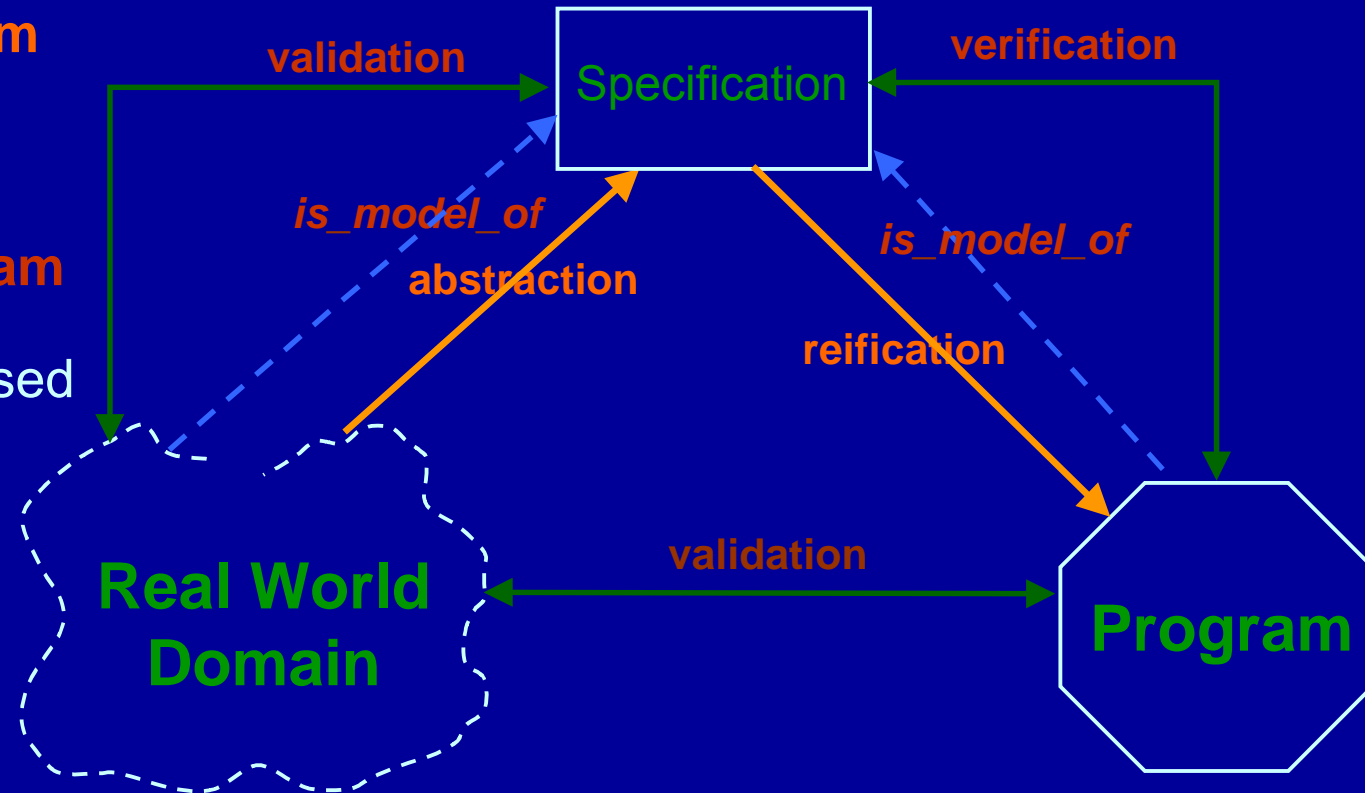
Areas of Evolution

- Many **areas** of evolution in **software context**
 - I **Ab initio development** of a system or, for example, **implementation of changes** to produce a **maintenance** or **upgrade release**
 - II **Sequence of releases** or **versions** of, for example, a software system over its lifetime
 - III **Applications**
 - IV **Operational domain** of an application - impact on relevant evolution process
 - V Software evolution **processes** in, for example,:
area I - for **system development**, area II - for **release planning and management**
 - VI **Interacting domains** - e.g. **organisational**, **market place**, **technology** - their impact on relevant evolution process
 - VII Software process **models**
- List a **simplified structure**

We focus on area 2 **sub-area** - **evolution** of **E-type system** in **active use**

Programs in the Real World

- **Real world operational domain** is **model** of a **specification** derived by process of **abstraction**
- **Program** reached by a process of **reification**, also a **model** of that **specification**
- **Verification** of a **program** may be possible in **whole** or in **part**
- Both **domain** and **program** have, in general, **properties not** addressed by **specification**
- **Specification, program must** be **valid** relative to **real world**



The real world - **E-type system** relationship

Some of their Properties

- Universe, with its **unbounded** number of attributes, is ultimate **operational domain** of *E*-type systems
- Actual **operational domain** is a **sub-domain of the universe** and also has an **unbounded** number of **attributes**
- **Application** is equally **unbounded** in the number of its **potential attributes**
- **System** is **finite**

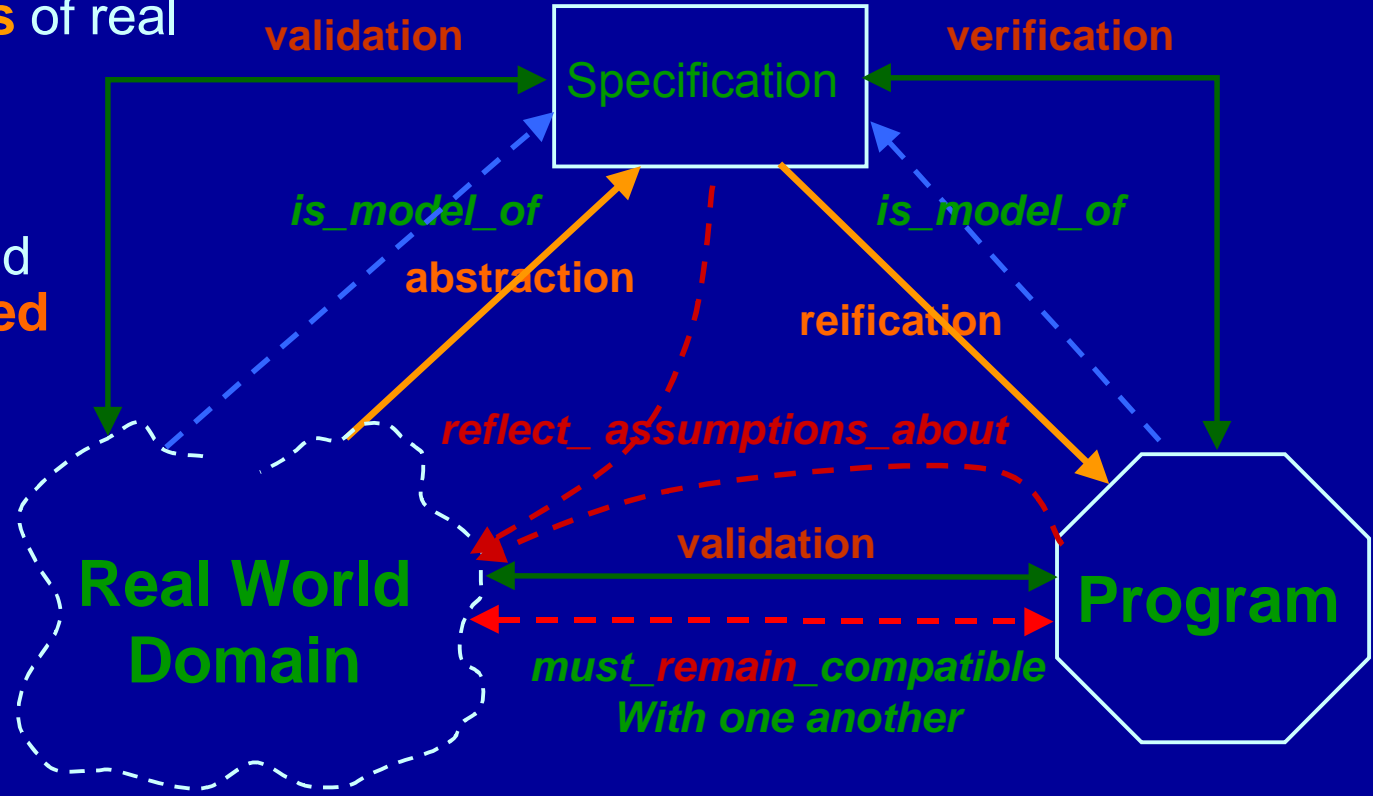
Gap between **finite** *E*-type system and its **unbounded operational domains** bridged by **assumptions**

Crucial Role of Assumptions

- Assumptions are **properties** of specification, program accepted as **valid reflections** of real world properties

- Must be **captured** and periodically **reviewed**

- Program maintained **compatible** with **real world** over system life time



Assumptions and, hence, **program** must reflect real world as it **is now**

Adoption of Assumptions

- Adopted **throughout global** development and usage processes
 - by**
 - corporate management
 - local management
 - marketeers
 - vendors
 - users at all levels
 - etc., etc.
 - during**
 - conception
 - verbalisation
 - requirements statements
 - specification
 - designs at all levels
 - validation at all levels
 - integration
 - release
 - installation
 - usage
 - etc., etc.
- Adoption may be by **commission** or **omission**, **explicit** or **implicit**, **conscious** or **unconscious**, **recorded** or **unrecorded**
- Individual assumptions **become invalid** as **changes** occur in **operational domains, application, technology** etc.
- Software **maintenance maintains validity** of **assumption set** and, thereby, **stakeholder satisfaction**

Such maintenance at **core** of system **evolution process**

E-type System Evolution Process

- **Ab initio** development or **change** for whatever reason

Trigger - **Need/Demand/Opportunity** not satisfied by current system



Preliminary, generally broad, **statement** of required **system** or **change**



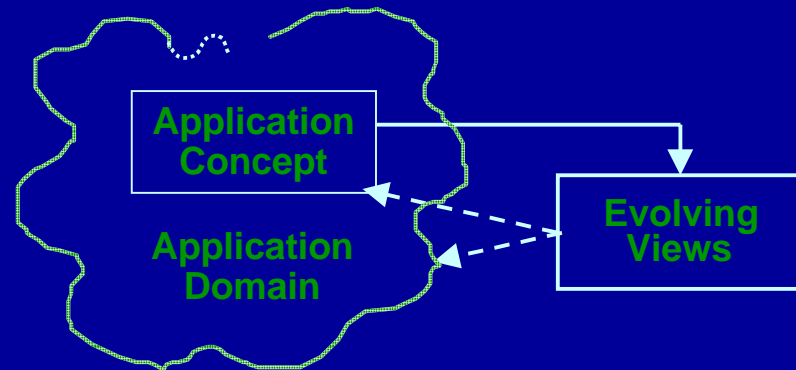
Yields - **new** or **changed application concept** and/or **domain**

- In general, **both concept** and **domains** initially:
 - **ill defined**
 - **not** fully or precisely **verbalised**
 - **not** explicitly **bounded**
- **Authoritative statement of purpose** provides **base** for process evolving proposed change, providing, *inter alia*, initial functional and **domain bounds** for **application**
- **Bounding** a **critical** and **continuing** process **activity**

Continuing revision throughout **process, system lifetime**

Bounding Process

- **Elicit, reconcile, merge, limit** restricted, biased views of individual stake holders:
 - all levels of **management**
 - organisational and individual, direct or indirect **users**
 - **marketeers, suppliers, procurement** personnel
 - domains and application **experts**
 - system, software **engineers, developers**
 - etc., etc.
- **Process** blends **viewpoints**, determines **broad goals**, agrees **assumptions**
- **Convergence** to consensus changes **application, bounds**



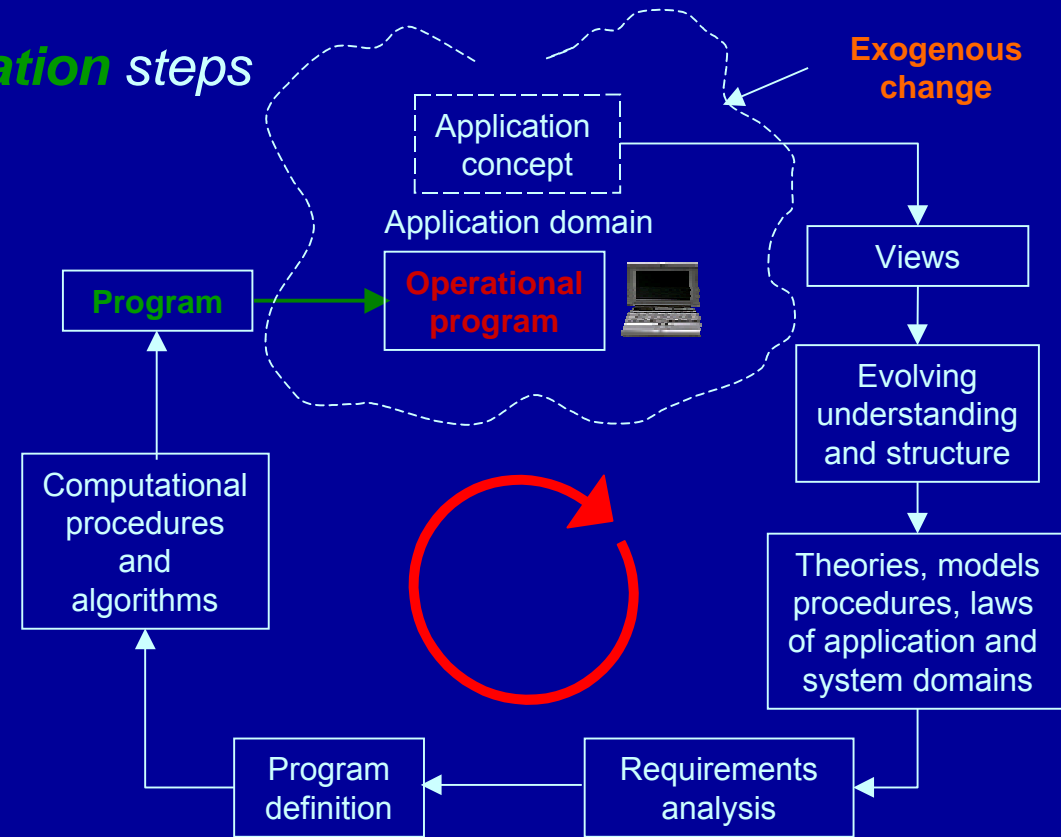
- **Convergence** to consensus changes **application, bounds**

First step in *E*-type **evolution** - development and maintenance - **process**

High Level View of E-type Process

Series of **development** and **implementation** steps

- Culmination: **validated program**
- Final step: **installation** which changes **domain** and **operation** which changes **application**
- All in **changing real world** domain
- **Installation** and introduction into **usage** closes major **feedback loop** that drives **iterative process** to **release** sequence of **upgrades** to users

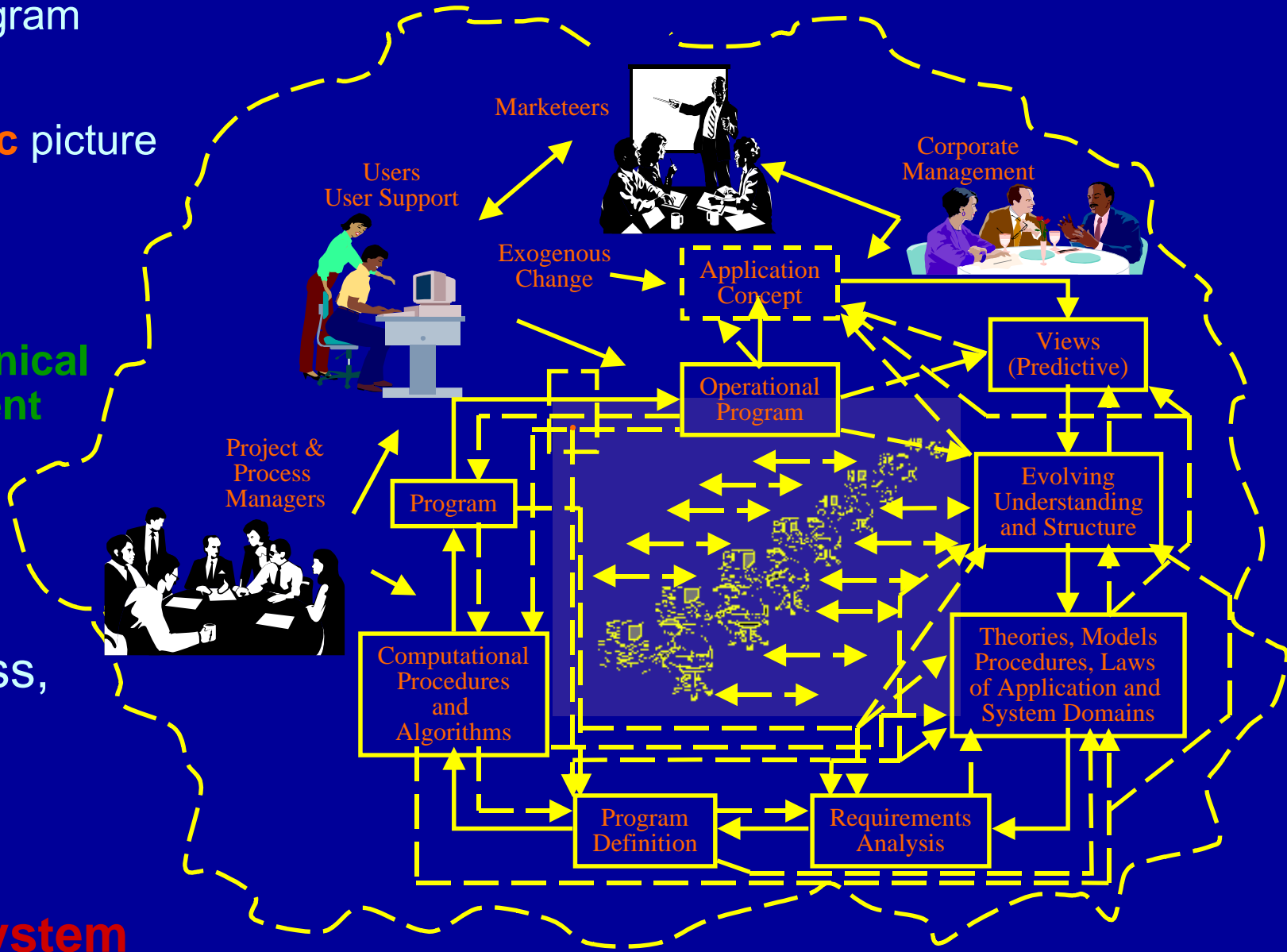


Driver of **continuing** software **evolution**

More Realistic Picture

- Previous diagram a **fiction**
- More **realistic** picture
- Process **not sequential**
- Not just **technical development**

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



The Global Process

- **Aggregated** effect of **all activities** that **transform concepts, ideas, needs, resources** into executable **code** and other **deliverables**
- Involves many **activities, groups, people** with diverse **responsibilities**
- **Complex, non-linear**, possibly **time varying**, loop **structure**
- Satisfies engineering **definition** of **feedback** - *output from part of system or process modifies one or more of its inputs*
- All these **factors** influence **process & product properties & behaviour** and must be considered when **planning, executing, controlling** evolution
- These and other **observations encapsulated** in **generalisations** relating to **software evolution**

Laws of software evolution

The Laws and Their Observational Base

No.	Brief Name	Law	Support
I 1974	Continuing Change	An <i>E</i> -type system must be continually adapted else it becomes progressively less satisfactory in use	yes
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	yes (indirect)
III 1974	Self Regulation	Global <i>E</i> -type system evolution processes are self-regulating	yes
IV 1978	Conservation of Organisational Stability	Average activity rate in an <i>E</i> -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 <i>E</i> -type systems tends to decline	yes
VI 1991	Continuing Growth	The functional capability of <i>E</i> -type systems must be continually enhanced to maintain user satisfaction over system lifetime	yes
VII 1996	Declining Quality	Unless rigorously adapted to take into account changes in the operational environment, the quality of an <i>E</i> -type system will appear to be declining as it is evolved	theory based
VIII 1996	Feedback System (Recognised 1971, formulated 1996)	<i>E</i> -type evolution processes are multi-level, multi-loop, multi-agent feedback systems	yes (indirect)

What **empirical observations** support these conclusions?

Sources of Empirical Observations

- **FEAST/1** and **/2** long delayed follow up of **1968 IBM-process** study

Objectives

- **Identify, assess, model, analyse presence and impact of feedback loops and mechanisms**

Approaches

- **Black and white box** modelling of **evolution** attributes - such as growth, change rates, incremental growth - of **several industrial systems**
 - **empirical** investigation
 - **metrics based**
 - product evolution **trend indicators**
 - **system dynamics** models

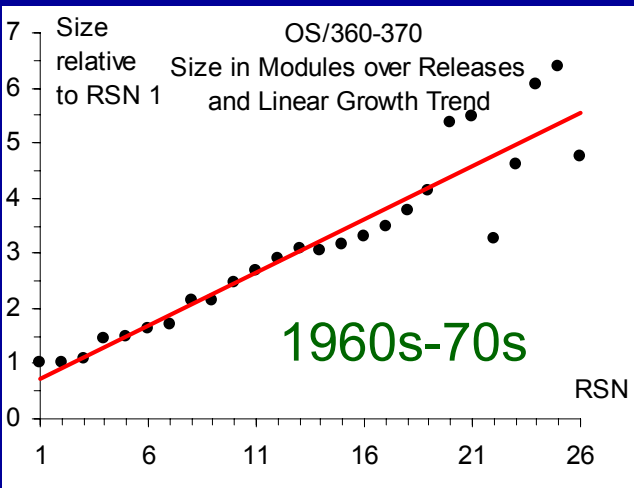
Industrial Collaborators - BT, DERA, ICL, Logica, Matra- BAe Dynamics

- Provided data on a **variety** of **systems** differing in **application** and **market areas, size, evolution domains, organisational structure** and so on

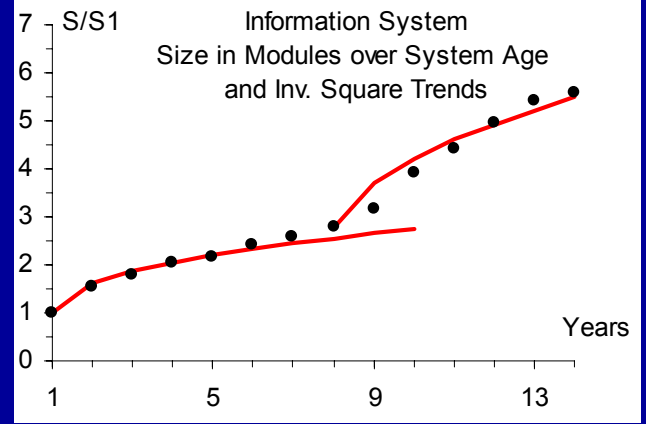
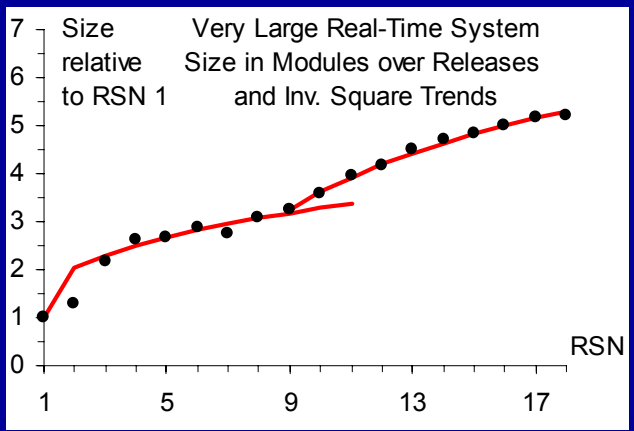
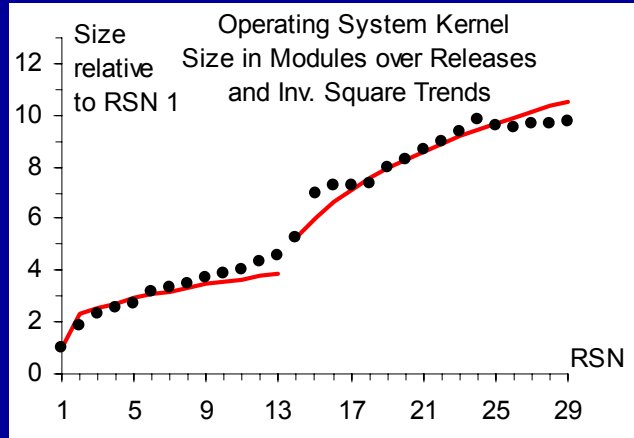
Some of the **results obtained**

Continuing Growth - Linear or Inverse Square?

- OS/360



- Linear growth to release 19
- End result - instability leading to fission
- Cause - linear growth beyond sustainability?



- Segmented inverse square growth : $S_{i+1} = S_i + \hat{E}/S_i^2$ ($1 \leq i \leq n-1$)
 - where i are sequence numbers (rsn) of the n releases for which data is available

- Indication of complexity constrained growth?

Model Precision

- Mean magnitude of Relative Error (MRE) and pred(T%)
 - $MRE = |actual_size - predicted_size| / actual_size$
 - $pred(T\%) = k/n$, where T is selected threshold for MRE, k is number of points for which MRE is less or equal to T and n is total number of data points

System	Model	MRE %	Pred[10%] %	Pred[20%] %
IBM OS/360	Linear	9.6	73	92
Operating Syst. Kernel - Seg. 1	Inv. Sq.	8.6	58	92
Operating Syst. Kernel - Seg. 2	Inv. Sq.	3.7	93	100
V. L. Real Time Syst - Seg. 1 *Starting at rsn 2	Inv. Sq.	10.0 (3.5*)	88 100	88 -
V. L. Real Time Syst - Seg. 2	Inv. Sq.	9.2	100	-
Information System - Seg. 1	Inv. Sq.	4.2	100	-
Information System - Seg. 2	Inv. Sq.	5.7	83	100

- Remarkable precision for process involving significant human intellectual and mechanical activity for decision, control and execution
- Indicates presence of driving, stabilising, constraining system dynamic forces?

How many releases are required for system dynamics to dominate?

Model Precision

- Mean magnitude of Relative Error (MRE) and pred(T%)
 - $MRE = |actual_size - predicted_size| / actual_size$
 - $pred(T\%) = k/n$, where T is selected threshold for MRE, k is number of points for which MRE is less or equal to T and n is total number of data points

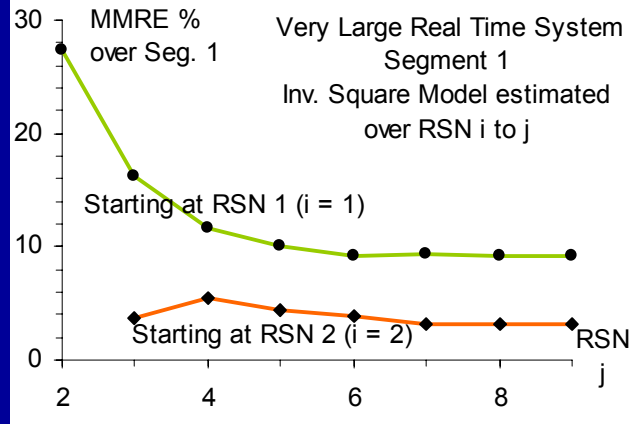
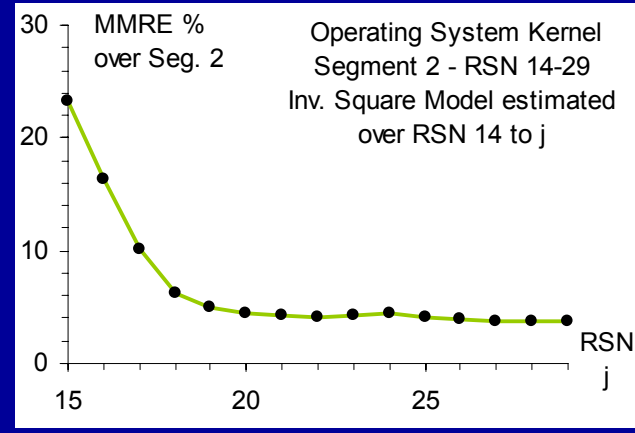
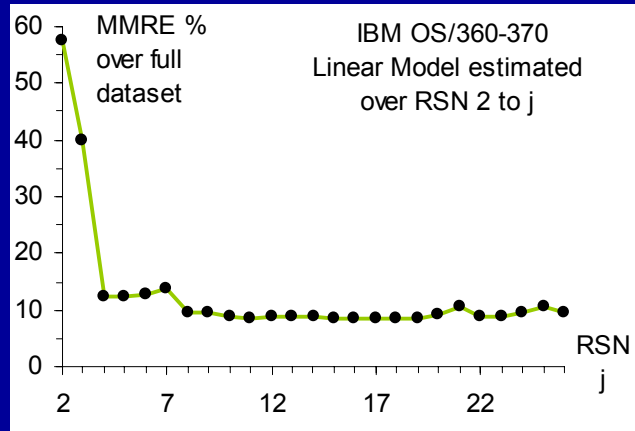
System	Model	MRE %	Pred[10%] %	Pred[20%] %
IBM OS/360	Linear	9.6	73	92
Operating Syst. Kernel - Seg. 1	Inv. Sq.	8.6	58	92
Operating Syst. Kernel - Seg. 2	Inv. Sq.	3.7	93	100
V. L. Real Time Syst - Seg. 1 *Starting at rsn 2	Inv. Sq.	10.0 (3.5*)	88 100	88 -
V. L. Real Time Syst - Seg. 2	Inv. Sq.	9.2	100	-
Information System - Seg. 1	Inv. Sq.	4.2	100	-
Information System - Seg. 2	Inv. Sq.	5.7	83	100

- Remarkable precision for mechanical activity for $Pred[x\%] = y$ indicates that prediction is within $\pm x\%$ of actual size for y% of all data points
- Indicates presence of driving, stabilising, constraining system dynamic forces?

How many releases are required for system dynamics to dominate?

Predictive Power of Growth Models

- Only a small number of releases required to achieve a good predictive model



System	Settling Point	MMRE stabilises at
Operating System Kernel	6	4 %
Very Large Real Time System	6	9 % (3 %*)
Information System	4	4 %

*starting at RSN 2

- Initial knee indicates model settling period - first thought to be build-up of dynamics but later recognised that this is, at least in part, inherited

White box modelling to identify sources of dynamics

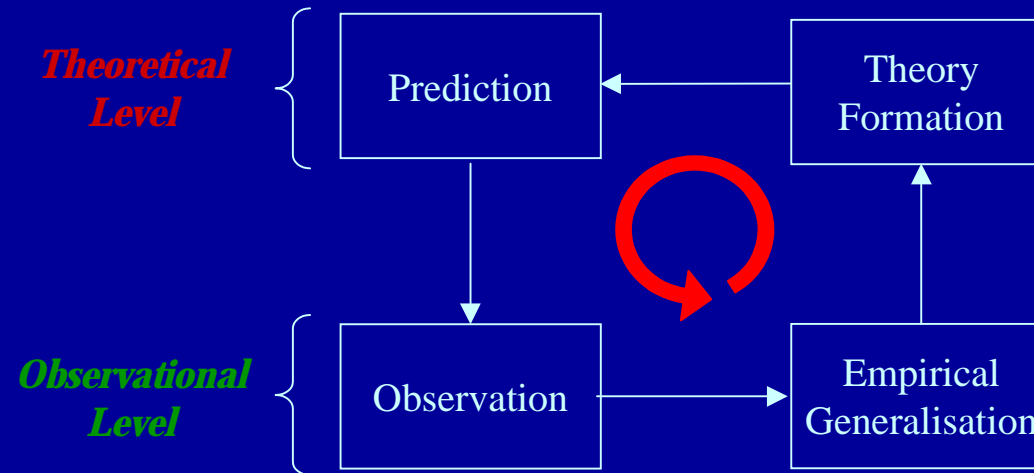
Some FEAST Results

- **Evolution characteristics** and **behavioural patterns** of release-based systems studied **qualitatively similar** to those observed in the 70s
- **Support** for at least 6 of the 8 **laws of software evolution** as modified over the years
- **Extended** and **new insights** into and **conclusions** about **process**
- Study of **influential** feedback **loops** in long-term **evolution context** and **assessment** of their **impact**
- Black box and system dynamics **models** to support **process planning, management, control, tuning** and **improvement**
- Some 35 **management** and **planning** rules and guidelines

Inputs to **formal theory of software evolution**

Development of a Formal Theory

- Based in **observations**, **behavioural invariants**, **wider phenomenology**
- Provides basis for **formation** of informally expressed **observational level** theory
- As a set of derived **empirical generalisations**



- From **axioms suggested** by the **empirical generalisations** one derives a **theoretical level** theory stated in a **formal notation**
- **Predict behaviours** on basis of **emerging theory** and **validated** against **observations**
- **Iterative extension**, yields **theorems** - further **generalisations** and corresponding **axioms** may also emerge from **observations** suggested by **evolving theory**

SETh proposal to **initiate theory development** submitted

FEAST Summary

- In addition to providing **process management tools**, **empirical studies** and their **interpretations** largely support **hypothesis** that software process is a **feedback system**
- Have led to **identification** of process **patterns**, **trends** and **invariants** that can be reflected in **empirical generalisations**
- Net **conclusion** is that **disciplined evolution intrinsic** to computer **applications**, to **software** and to many **other areas** in **software context**
- **System dynamics models** provide a **tool** for its study and **tools** for **evolution process planning, management, control, improvement**
- **Studies** and **results** also provide basis for **development** of **theory of software evolution** at **phenomenological** and **theoretical** levels
- **Wealth of results** ripe for **further investigation, extension** to provide theoretical **base** and **framework** so long missed in software engineering

Practical illustration of application of **scientific method**

The FEAST Project as a Scientific Endeavour

- Triggered by **observation** of **empirical data** - in early seventies
- **Patterns** and **trends** recognition - some results
- **Analysis, inferencing** - some results
- **Generalisation, hypothesis formation** - some results
- **Modelling** and validation, **rejection** or **trigger** for further **experimentation** - some results
- **Theory generation** and **validation** - planned
- **Iteration** and **extension**

Expectation that **approaches** and **results** can be **extended** to **wider systems studies**

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, ol. 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, Maryland, 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