

Metrics and Models of Software Evolution

University of Sannio
2 May 2001

Juan F Ramil

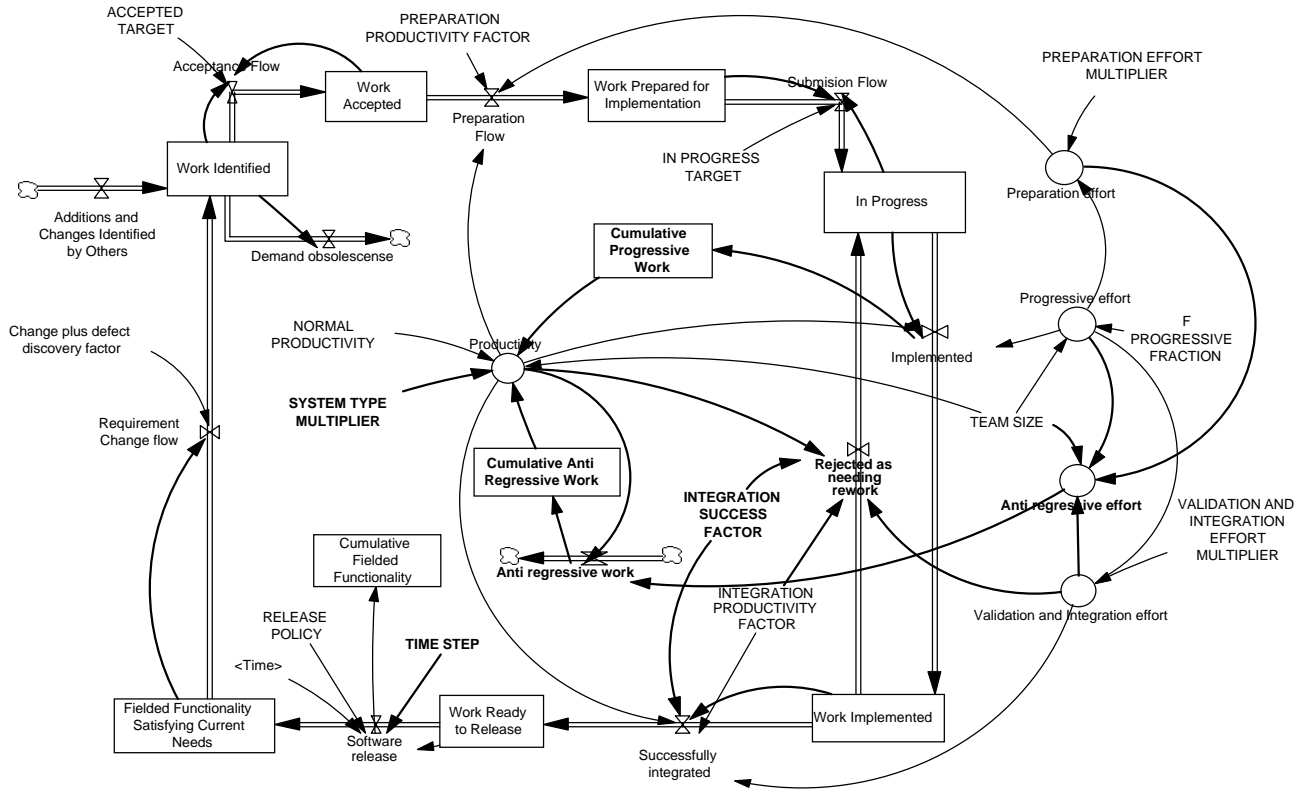
Department of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ
tel. +44 (0) 20 7594 8216
fax. +44 (0) 20 7594 8215

ramil@doc.ic.ac.uk
<http://www.doc.ic.ac.uk/~ramil>

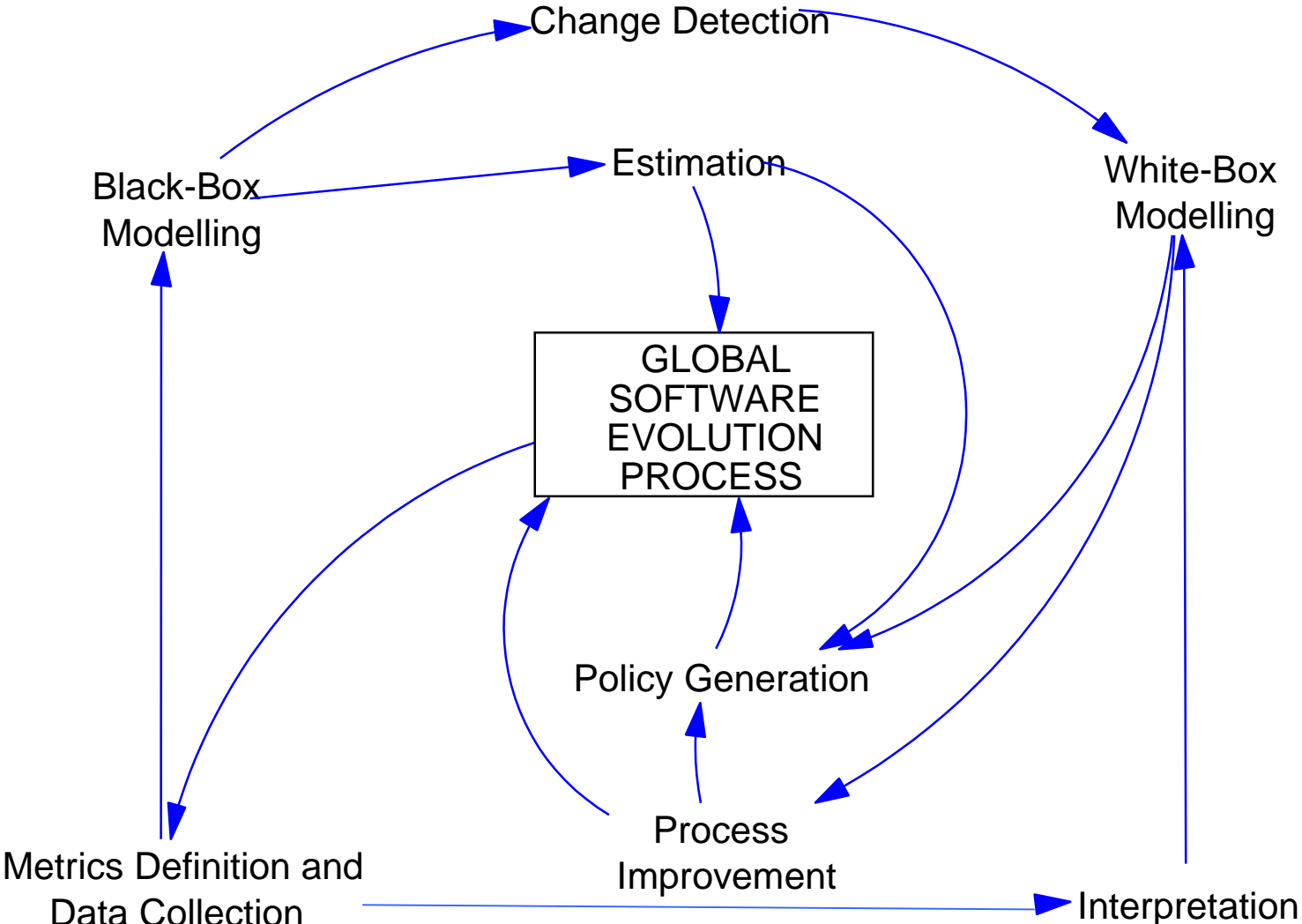
Professors Lehman's lectures provided conclusions reached from **studies**, **models** and **interpretations** of **metric** data

Process Models in FEAST

- **Black-box models** to encapsulate regularities in evolutionary attributes
 - e.g. Turski's inverse square model of system growth: $Size_t = Size_{t-1} + E / Size_{t-1}^2$
- **White-box models - system dynamics** - to investigate policies and process improvements
 - e.g. a model to examine division of effort between **progressive** and **anti-regressive** work



Case Study - Resource Estimation



Measurement Issues in the Context of Evolution

- Metrics will be **useful** if **interpretation** of the **behaviour** they reflect leads to understanding that facilitates **explanation, control, prediction** of process and/or product attributes
- In the context of a real life situation, **practical application** of specific metrics requires, however, that the **data** available can be **related** to **questions in hand**
- Facts of life:
 - the study of **evolutionary trends** requires **historical data**
 - **collection**, storage & preservation in readily available records **not widespread** in industry
 - in practice, one must **rely on** data sources such as **configuration management** and/or **change history** records, etc, which were designed and collected for **purposes other than modelling**

Possible **approaches** to metrics definition and collection
in evolution context

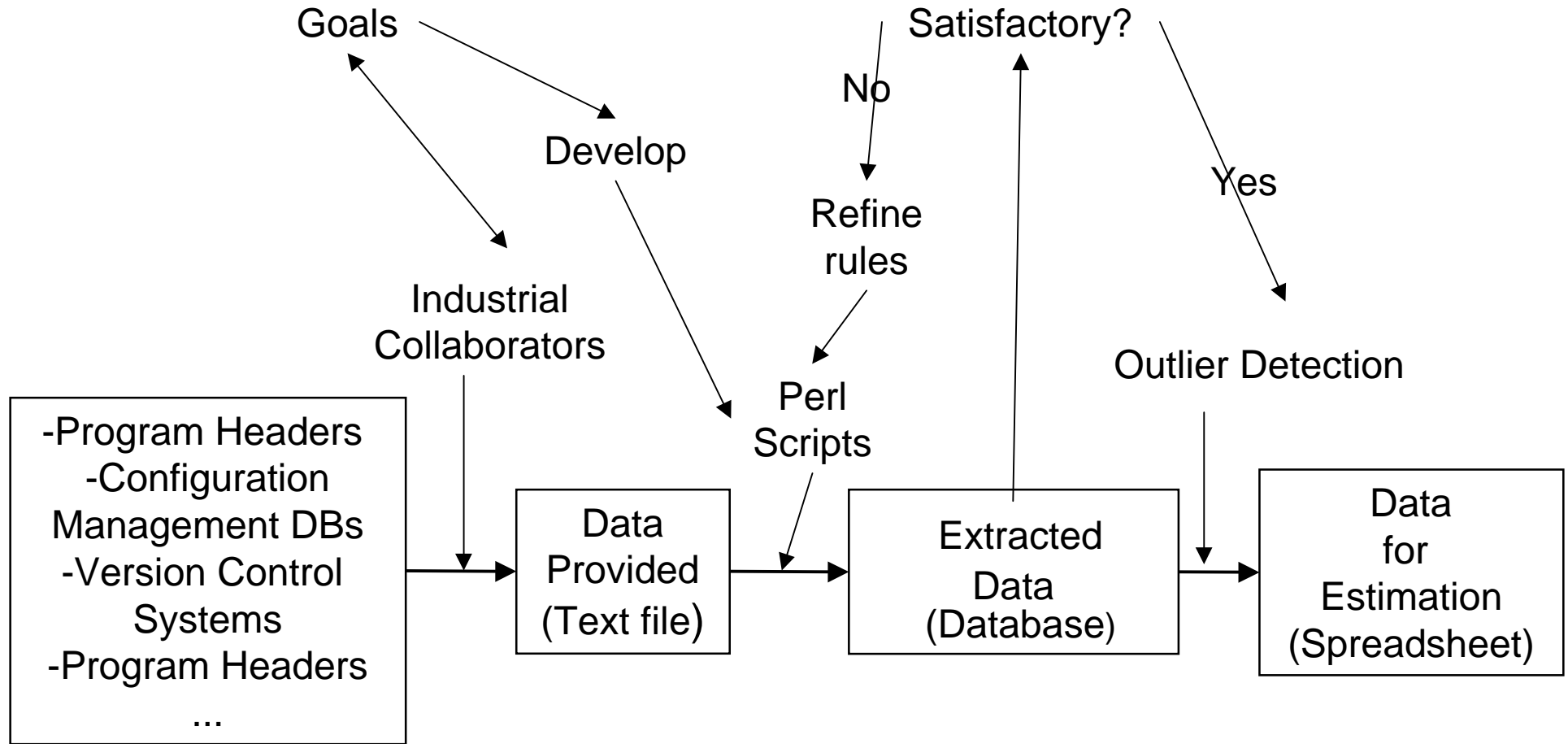
Sources of Metrics in the Evolution Context

- **Ad Hoc:** use of existing records - default
 - data **collected** with other purposes, such as **configuration management** and **change history records**
 - tends to **reflect code** and code-related work **only**
 - raises quality issues such as interpretation, missing or incomplete records, changes in meaning
- **Top-down**
 - start with small set of attributes, defined by, for example, GQM
 - successive refinement of metrics set
 - may involve a long delay until sufficient data is accumulated and patterns can be identified
 - relevant aspects of phenomenon may be left unmonitored
- **Bottom-up**
 - **store** a large number of **attributes defined a priori** with hope of capturing a **useful** sub-set
 - e.g., store snap-shots of process state and product on a regular basis, similar to a census
 - high costs of data collection and their storage
 - resultant flood of data thwarts analysis
- **Mixed**
 - selective mixing of all three approaches may offer a useful trade-off, if thoughtfully pursued
 - define set of **attributes** of direct interest
 - identify set of **metrics from which** the **attributes can be derived** or inferred
 - offers potential to derive metrics not initially foreseen

Example of Data Source: Change-log in Program Header

```
; Change log:-  
;  
;  
; 20NOV85          HJ  
; Modified to properly support reprint of alarms  
;  
;  
; 22NOV85          DV  
; Modified to use LOGINT module to interpret alarms, as alarms are  
; just logs in VAX TMAN, and virtual address space is not limited!  
;  
;  
; 01DEC85          DV  
; Modified to use printer sequence number proof accumulator  
;  
;  
; 04DEC85          PC  
; Modified to use VMS port names instead of RSX port numbers  
; 31Jul87          HC  
; Made size of lines area to be driven off the symbol  
; sok.APPFIL_size from FW$:OPTIONS.MAR.  
;  
;  
; 20th November 1988          HJ  
; Changed ln:w.flags to ln:l.flags  
;  
;  
; 20th November 1988          JC  
; (implemented by HJ)  
; If the output sequence number gets out of sync with the proof so that  
; the proof is larger of the two, then the alarms printer will print  
; all the alarms from the time the system was cold started. If this  
; situation occurs then the proof will be reset to the output sequence  
; number.
```

Data Extraction



Indicators of Evolution Activity Used in Study

Indicator Based on	Abbreviation (Full name)	Description
Modules	<i>ModifHandlings(t)</i> (Modification Handlings)	Symbol '#' to be read as 'count over interval t to $t+1$ ' # of changes to modules as reflected by number of change log modification entries, referred to as <i>handlings</i>
Modules	<i>ModulesChanged(t)</i> (Modules Changed)	# of modules modified
Modules	<i>ModulesCreated(t)</i> (Modules Created)	# of modules added to the system
Modules	<i>TotalHandlings(t)</i> (Total Number of Handlings)	# of total change log entries, that is, including both creation entries and modification entries
Modules	<i>ModulesHandled(t)</i> (Modules Handled)	# of modules, either added to the system or modified, or both (if both, module is counted once)
Subsystems and Modules	<i>SubsysChanged(t)</i> (Subsystems Changed)	# of subsystems that underwent modifications to their modules
Subsystems and Modules	<i>SubsysHandled(t)</i> (Subsystems with Modules Handled)	# of subsystems that underwent either additions or changes to their modules, or both (if both, subsyst. is counted once)
Subsystems and Modules	<i>SubsysInclCreations(t)</i> (Subsystems that include Modules Created)	# of subsystems which underwent module additions

Analysing Evolutionary Attributes - Size, Change Rate,...

- Many factors may affect software evolutionary **trends, behaviours**
 - **aging** effects, performance **constraints**
 - **superposition** of **changes** and their **orthogonality** to architecture and installed features
 - increasing **remoteness** from **original architecture**
 - increasing **complexity** and other consequences of **system growth**
 - loss of **familiarity** with totality of system, personnel **turnover**
 - impact of process **improvement**, management **policies**
- Appropriate process models permit study of these factors and their interactions
 - tendency to expect **complex models** because of complexity of process
 - models need not be large or complex to be useful
 - **simple** models often the best
- **Process improvement** and **effort estimation** closely related to concepts and results to be outlined
- Time does not permit inclusion of more than a **sample** of the **behaviour** observed in **FEAST**, the **models** developed, the **implications** derived and the **metrics** on which all are based

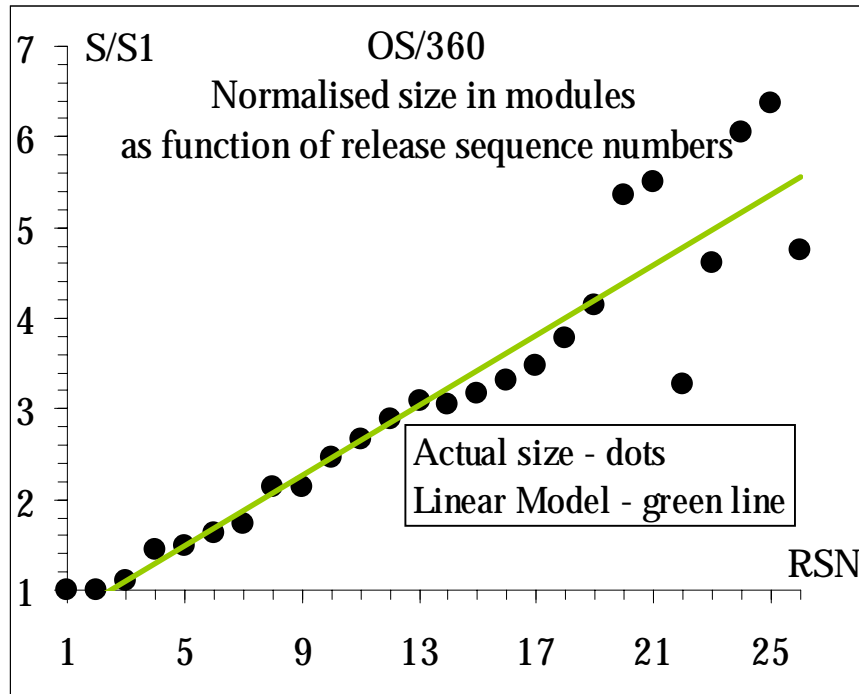
What **behaviour**, which **models**, which **metrics**?

Behavioural Patterns

- **Systems** studied:
 - one **operating system** kernel
 - two **real time systems**
 - one **defence system**
 - one **financial transaction system**
 - one **operational support system**
- Evolutionary behaviour compared to that of **IBM OS/360**, as studied in late sixties
- Some results:
 - **behavioural patterns** have been identified
 - qualitative and quantitative **empirical generalisations** have emerged
 - **elements** for **theory formation**
 - suggestions for **decision support** models and tools
- As explained in Lehman's charts, and supported, in general, by the **FEAST** studies, the **laws** provide high level qualitative statements of **behavioural patterns**

Some examples...

Continuing Growth - the OS/360 case



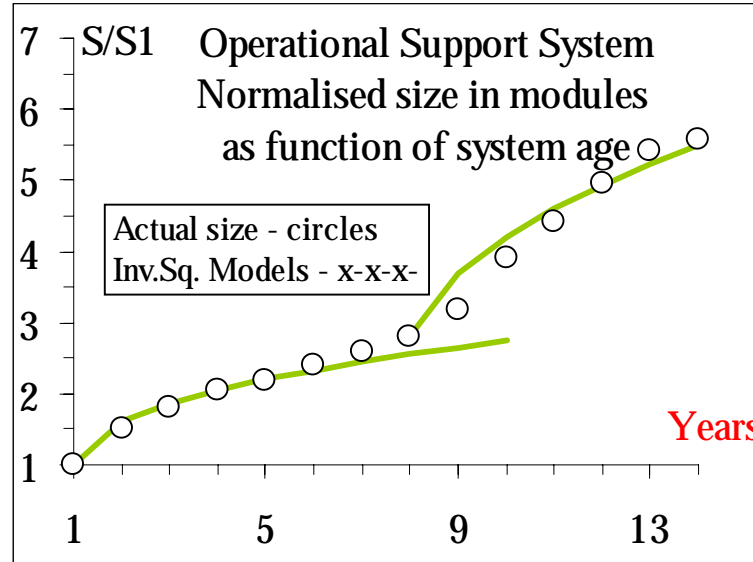
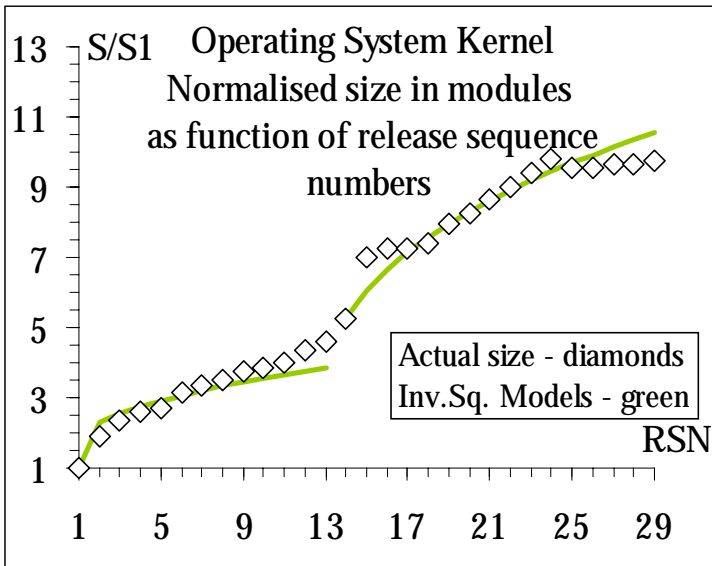
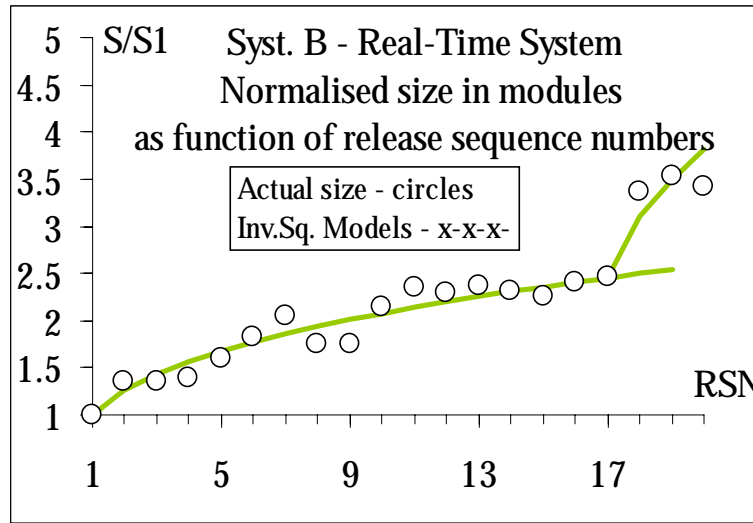
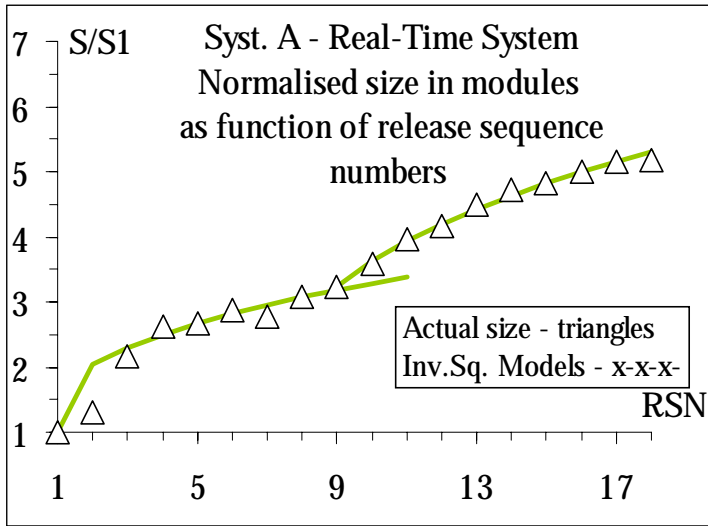
- **Linear** growth trend
- **End result** - **fission**
- **Cause?** - **excessive growth rate?**

Other **evolutionary attributes** of OS/360 evolution were also studied

Led to first three laws

Linear growth trend **atypical** when compared to other systems

Examples of Black-Box Modelling



- **Growth trend model:** inverse square, $Size_t = Size_{t-1} + E / Size_{t-1}^2$
- **Decaying** growth trends with **discontinuities** in growth rate
- Likely to reflect **regeneration** points
- **Trend model** fitted to individual segments

Development as Evolution

- A defence system
 - an example of **development** as **evolution**
- Despite physical constraints, performance and functionality have increased over 5 years or so - illustrates possible benefits of **anti-regressive** work
- An anomaly with respect to sixth law?, a **different** type of system?
- System **not installed** in the **operational domain** until development finishes
- Process, a form of **evolutionary development**
- Evolution takes place over **generations** of the defence system

Continuing Growth - summary

- **Decaying growth rate**, over one or more **segments**, has been observed in all available data - with exception of OS/360 and “development as evolution”
- **Discontinuities**
 - growth rate **discontinuities** have been observed in several of the trends
 - may be due to a **degree** of **regeneration** achieved by, for example, **restructuring**, process change, etc.
 - may also be due to **consolidation** of parallel versions, etc.
 - consistent with Bennett and Rajlich’s view of **staged** software evolution
- **Inverse square growth models** have been fitted to all of relevant data over individual segments

$$S_{i+1} = S_i + \hat{E}/S_i^2$$

where $i \in (1, 2, \dots, n)$ are **release sequence numbers** (rsn) and n is **number** of **releases**

- Interpreted, in general, as an **indicator** of **growing complexity**
 - but other interpretations not to be excluded

Continuing Change - the OS/360 case

- **Elements handled** per release H: number of elements involved in work, e.g. cardinality (#) of the union of sets of elements added, changed and removed if the latter is available

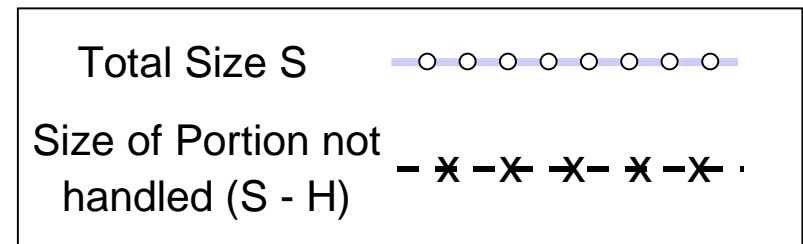
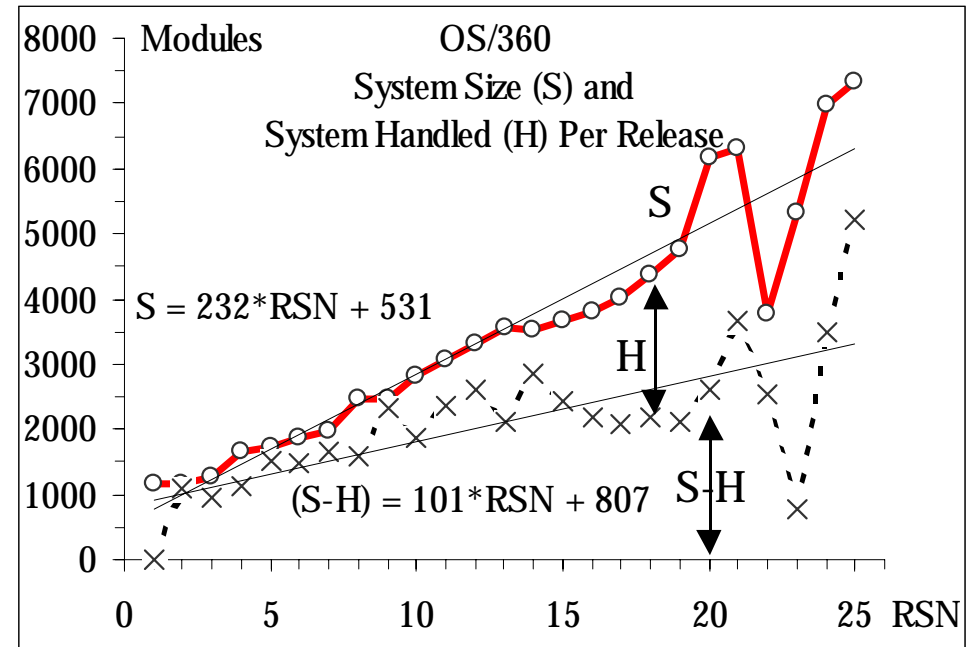
$$H_i = \# (Added_i \cup Changed_i \cup Deleted_i)$$

- **Modules handled**, distance between upper and lower curves

- **Diverging curves** indicates that an **increasing fraction** of system is being **handled**

- **Cause?** - growing complexity?

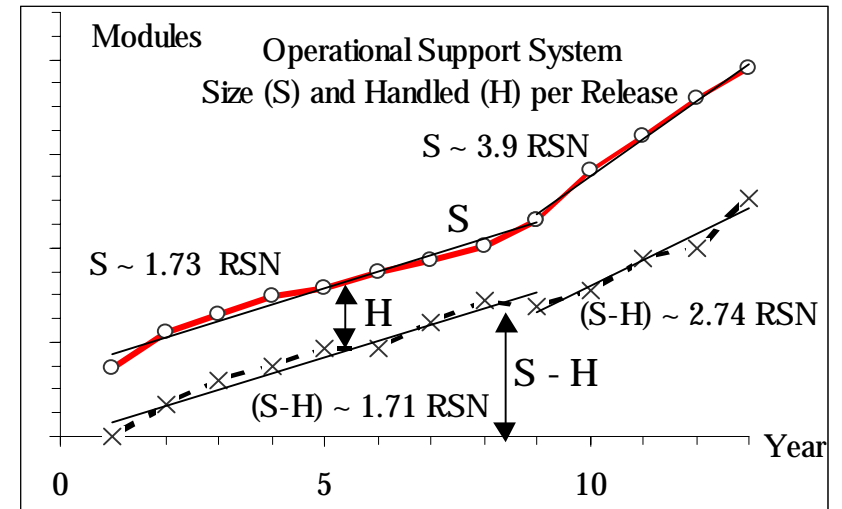
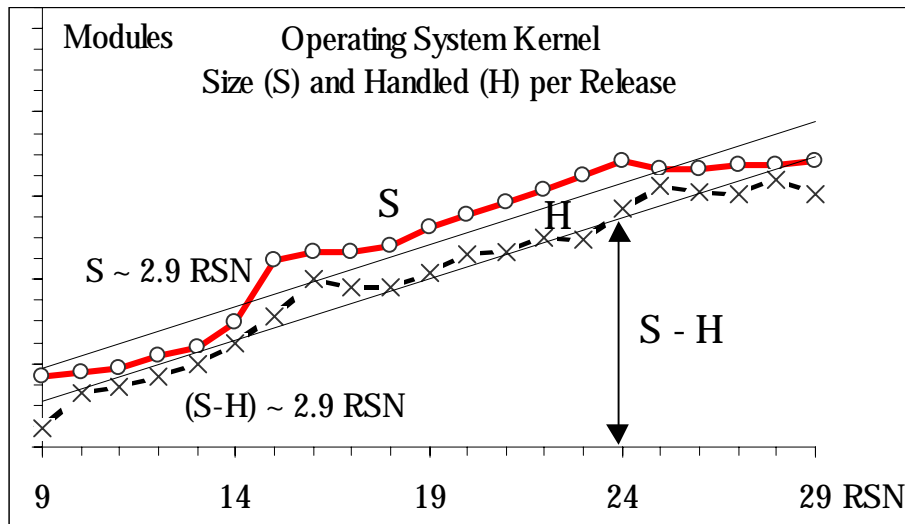
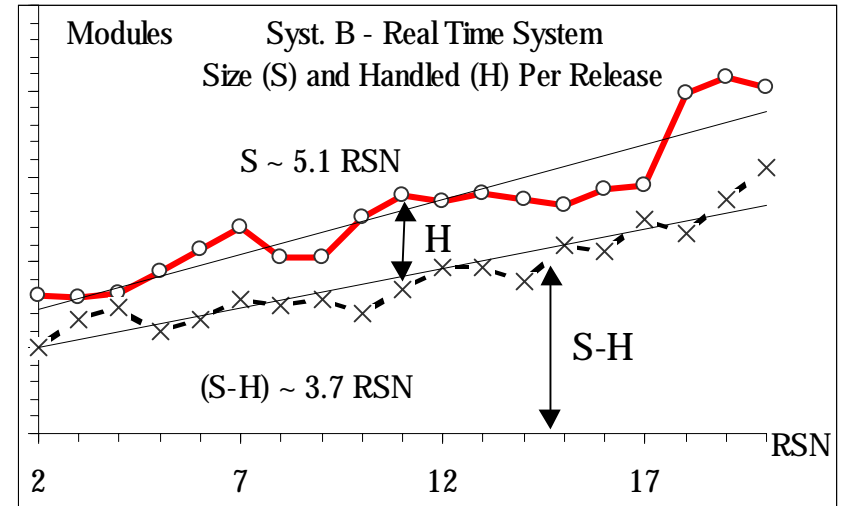
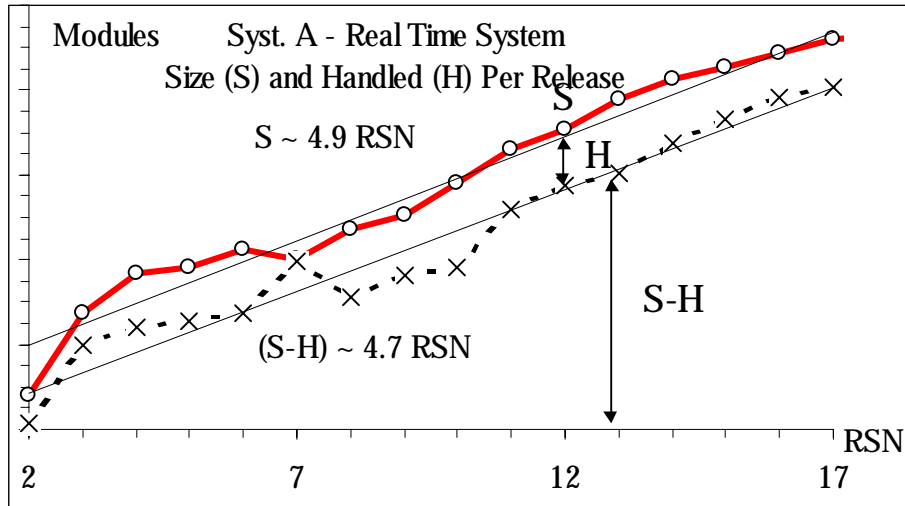
- **End result** - end of **operational life**



Example of unstable evolution

Continuing Change - more recent systems

- **Elements handled** per release **constant** or not increasing significantly, a characteristic of **stable evolution**?

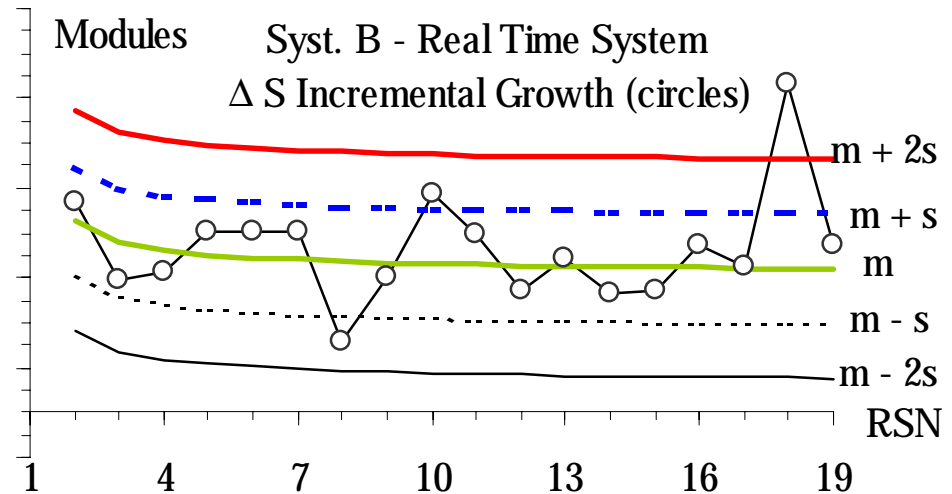
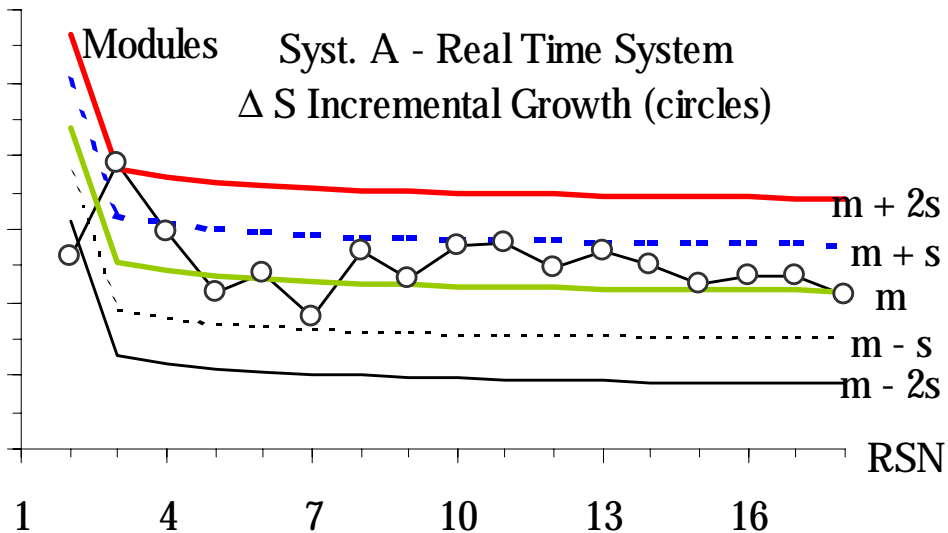


Self regulation and Conservation of Familiarity

- Three levels of release incremental growth identified
 - **small** increment of growth can be adequately validated release to users and safely followed by a further satisfactory release
 - **somewhat larger** growth is likely to lead to delays in release or high fault levels after release and therefore to one or more follow-up releases
 - **too large** a growth is likely to lead to serious problems, in the limit to "catastrophe"
- We have here deliberately refrained from defining small, somewhat larger, and too large
- In our analysis we term the value of the **long-term trend model** at RSN i as $m(i)$
- **Standard deviation** of residuals relative to the long-term trend is termed s
- Some evidence that $\ll m(i), \sim m(i), > m(i)+2s$ thresholds frequently observed and used in statistical process control are relevant here

Self regulation and Conservation of Familiarity

- Levels $\ll m(i)$, $\sim m(i)$, $m(i) + 2s$ define management guidelines
- Level of $m(i)$ provided, for example, by fitted inverse square trend



- Incremental growth **outside thresholds** is likely to follow by a small or even negative increment - behaviour interpreted as self-stabilisation of **global process**, a **system dynamic effect**

Summary of empirical support

No.	Brief Name	Law	Support
I 1974	Continuing Change	An <i>E</i> -type systems 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
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 the 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 systems will appear to decline 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)

- Laws are, in general, supported by **empirical data** and theoretical reasoning
- **Range** of process paradigms, organisations, applications and project sizes to which they apply requires further investigation

Related Work

- Antoniol G et al - Sannio U.
 - Maintenance effort dynamics
- Bennett K and Rajlich V - Durham U. and Wayne State U.
 - Stages in software evolution, case studies
- Godfrey MW and Qiang T - U. of Waterloo
 - Open source systems evolution (LINUX)
- Harrison R and Cook S, U. of Reading
 - Software evolution, software evolvability and architectural issues
- Kemerer C and Slaughter S - Pittsburgh U. and CMU
 - Analysis of patterns in evolutionary attributes
 - Discipline in examining historical records
- Mockus A and Harvey Siy - Lucent Technologies
 - Change records as data source
 - Impact of tools and technology on productivity
- Shepperd M *et al* - Bournemouth U.
 - Dynamics of software maintenance