

SETh - Approach to a Theory of Software Evolution Case for Support Part 2: Proposed Research and Context

A Software Evolution

The phenomenon of software *evolution*, first identified in the early 70s [1,2], is now widely recognised as a topic worthy of serious investigation [3]. Studies undertaken have been largely *ad-hoc*, lacking a unifying conceptual framework, focussing mainly on the *how* of evolution [4] and on mechanisms for process improvement. The goal has been to increase productivity, reliability, adaptability and predictability of the software process, to decrease development time and to improve the quality of its product. Methods and tools making extensive use of formalisms and other mathematical techniques have been developed with the approach to their development and use primarily based on experience and intuition. The impetus for such investigation has come from the ubiquity of computers, and hence of software, in all aspects of human activity. Understandably, this has led to widespread, active, interest in *process improvement* in the context of software and, more generally, business processes as evidenced respectively by the SEI CMM model [5] and a recent EPSRC initiative [6]. A small number of groups have adopted a complementary approach [4], seeking to determine and understand the underlying *causes*, *attributes* and practical *impact* of evolution on the software process and its products. Their focus is on the *why* and *what* of software evolution; their goals, to address the practical nature of the phenomenon as experienced in industry and by users and to derive practical improvements. Such studies require observation of and data acquisition from industrial software processes and reliance on application of empirical methods [7]. Recently concluded, EPSRC supported, FEAST studies were based on this second approach and on a hypothesis that the software process is a complex, multi-level, multi-loop, multi-agent feedback system. The concepts underlying this and earlier work and their results have been widely disseminated [8].

Computer operations are ever more closely integrated. Desktop computers and the Internet have intensified demand for software that can be maintained compatible with a changing world and that satisfies a variety of needs and growing ambition. As society becomes ever more software dependent, software evolution [3] processes must be reliable, timely and cost-effective. To achieve this, software development and evolution processes require significant improvement. As the demand for this increases and spreads, various *ad hoc* methods for achieving *maturity* of industrial software development and maintenance organisations and processes have been developed [5]. But advances in methodology primarily focussed on *programming in-the-small* [9], have not, in general, scaled up to wide industrial applications. The absence of a supporting and guiding scientific base and framework has long been noted [e.g. 7,10,11]. The focus of this proposal is development of a theoretical framework addressing the *process* of transforming a computer application concept - in the widest sense - into an operational system, and maintaining the latter satisfactory in an ever changing execution domain, constitutes a potentially major contribution to ensuring increasingly reliable computer usage.

B Theory of the Software Process and of Software Evolution

Theory development [12], formalisation of key concepts and their wide exposure to falsification via experiment are key to well-founded process improvement and to successful assessment of the practical value of such improvements. Achieving this for the *total*, that is *global*, software process [13] is not yet in sight. Deep understanding of its various parts and of their interactions simply does not exist. A *theory of software evolution* is, however, a different matter. Thirty years [1] of software process studies has yielded a body of empirical knowledge and coherent, albeit incomplete and informal, understanding of the observed evolutionary behaviour of key attributes of software system evolution processes and their invariants. Thus, we are confident that the time is ripe for development of such a theory, with some immediate and practical application in process management and improvement. Moreover, wider holistic implications and properties are expected to become apparent from the development. The theory, and techniques employed in developing it, may well find application in the study of other complex systems and processes [14]. Results from a number of sources [e.g. 11,15,16,17,18], together with those obtained from the FEAST projects have greatly strengthened earlier insights, widened the domain of relevance and potential applicability of the concepts and observations, and strengthened confidence in their validity. The concepts and experience available in the groups submitting this proposal and their external advisors, in the fields of software process, evolution, and formal theory formation, gives us confidence that, with resources as requested in this proposal, foundations of a formal theory of software evolution and a base for its future extension can be achieved in a period of three years. Establishing a theory of software evolution to facilitate understanding, reasoning, analysis, refinement and extension is essential for reliable planning and management of software evolution. It is equally necessary as a basis for scientific experimentation.

C The Practical Impact of a Theory of Software Evolution

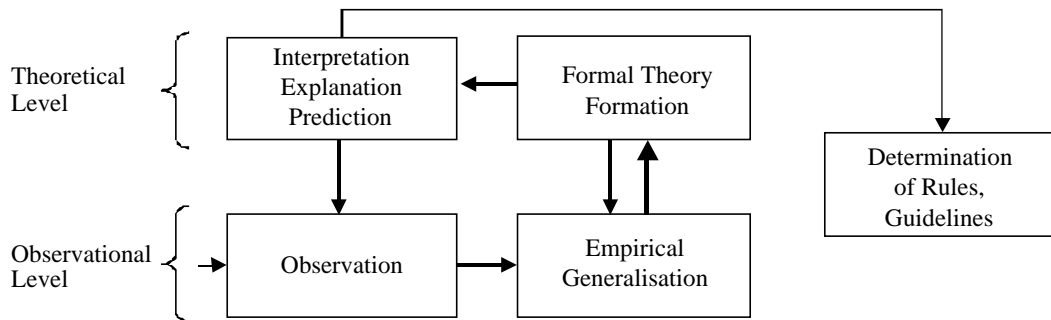
Application of Carnap's two level theory [12] discussed below pioneers new ground in software engineering. It is an *adventurous* development with potential, in the longer term, for providing tools for software process improvement. Current interest in software architectures [19], the search for *reuse*, pressures for moves to component and COTS based systems [20], all reflect the fact that software must, simultaneously, be made cheaper, more reliable, of higher quality, more adaptable and more evolvable. Software maintenance has been widely identified as a major source of lifetime costs (50 - 95%), increasing complexity and, hence, deterioration of adaptability, evolvability, reliability and quality. A theory identifying sources of evolutionary pressures, attributes of evolutionary behaviour, and controls and constraints managing/controlling that behaviour will identify means to control the deterioration. It will significantly advance the ability to architect and design systems for faster, more reliable and timely evolution. It is not being proposed primarily for the intellectual interest and challenges it presents, but for its potential to develop, support and spread *good practice*. Introduction into industry and achieving widespread acceptance requires one to overcome inbred scepticism to convince managers and practitioners of legitimacy and efficacy of the practice to be introduced. To date, little if any effort has been invested to demonstrate such legitimacy. Case studies have been the primary, if not the only, tool. *Systematic* progress demands empirical and theoretical work that feed into one another [21]. A theory will provide, for example, the *why* of good practice to help justify deployment costs by explaining why they work. Its explanations of

process behaviour will also provide an improvement catalyst, suggesting, for example, hypotheses for empirical investigation.

In summary, an explanatory theory provides a framework for process reasoning, permitting derivation of qualitative and quantitative management and implementation guidelines. Other illustrations derived from FEAST observations, other current sources and earlier work are discussed elsewhere [22]. Confidence in them, their acceptability, integration, extendibility and tool support will be greatly enhanced if they are shown to be part of a coherent, explanatory theory, more so if they follow from a formal, scientific, theory providing sound concepts and a rationale for good practice.

D Theory Development

The proposed development will be rooted in the methods of deductive theory [12,23] and formal reasoning [9,29] with phenomenology input based on years of observation, measurement, interpretation of industrial software processes. Observations include measures and patterns of behaviour, revealing behavioural invariants, suggesting *empirical generalisations* directly interpretable in terms of the phenomenology. Accumulated observations and understanding must be assembled and organised in an *observational theory*, as in the figure below. As this evolves, it provides a rationale for existing observations and generalisations. This will permit modest predictions about systems with properties that conform to the base domain of the empirical generalisation. Similar organisations providing similar software evolution services should have characteristics similar to those forming the basis for the generalisation. Real insight comes when a *theory* has developed. Evolutionary behaviour predicted, explained and interpreted from the theory should reflect fundamental insights; consistent behaviour across organisations, systems and sub-sets of systems in the real worlds of computer application, software development and its evolution. Observations restricted to a subset of systems reflecting common domain characteristics will also be of great interest. Experimental validation for inferences from the theory, and of conclusions reached by others [e.g. 24,25,26] must await formulation of the theory. IC and King's PIs, respectively, will lead the observational and theoretical level developments. Practical implications for engineering and management will follow as illustrated by a report provided to the FEAST referees [22]. The *simplified* figure below illustrates basic stages of this procedure [12,23].



Initially the work will exploit an existing body of knowledge to systematically explore and refine candidate theories. The investigators will guide the two RA and the PhD students in definition of properties and relationships that lie at the root of the formal theory. External overseas advisors, Professors Fiadeiro and Haeberer of the U. of Lisbon, C Potts of the Georgia Institute of Technology and Dr R Balzer of Teknowledge have expressed strong interest in the proposed development. Regular discussions to seek their independent views are included in the proposal plans and costings.

E An Example of the Phenomenology and of Related Observational Theory

The proposed theory relates specifically to the evolution of *E*-type systems [7]. Such systems include "programs that address an *application* (or solve a *problem*) defined in and part of a real world domain". The domain of interest (henceforth termed the *execution domain*), the *application* as part of that domain and the *program implementation* can all be considered as *models*, in the usual mathematical meaning of the term, of a common *requirements* specification. For an *acceptable* solution, the specification states those execution domain and putative application properties that must be reflected in the implementation to achieve an acceptable program. Properties not included are, by commission or omission, declared to be of *no concern* for an acceptable solution. The specification is an *abstraction* derived, by means of a process of iterative bounding [19] from an understanding and vision of the application being pursued within, and as part of, the execution domain. Both domain and program also possess other properties, the former because the abstractor focuses on the required properties and may or may not exclude others, the latter because implementation details are introduced during development. All properties must be, and must remain, compatible with one another (validation) and with the specification (correctness) as the world changes. If, as a result of changes, a specification is no longer a valid abstraction of the execution domain or if it still is but properties of the latter and the program become incompatible (because what was a valid implementation decision is no longer valid), the *E*-type program addressing it may display unacceptable behaviour. Evolution is essential to maintain acceptability. Since the real world is always changing, *E*-type program behaviour in execution is inherently uncertain, can never be *guaranteed* to remain acceptable [27,28]. This property is a formal property of the (software) engineering process *framework* described elsewhere [9,29].

The nature and levels of discourse to be used in formation of the theory are illustrated by a brief example. A relationship $size(i+1) = size(i) + (e / (size(i))^2)$, where $size(i)$ is the *size* of the i th release of a specific software system in number of modules and e a parameter determined from the size of previous releases, proposed by Turski closely models (mean magnitude of relative error (mmre) < 10 %) the growth of one specific system over the releases for which historical data was available [30]. The model was subsequently fitted to other systems with available release data, with the parameter e separately estimated for each one. For some systems, because of inflexion points in the behavioural trends two segment fits are more appropriate. The observation yields an empirical generalisation of form: *the sizes of*

future releases of a system are predictable by the inverse square model, possibly segmented, with a high level of precision.

Consider now the second law of software evolution, also a candidate for adoption as an empirical generalisation in the proposed theory. It may be stated as "A real world software system that is in regular use increases in complexity unless work is done to control and reduce that complexity" [7]. The inverse square growth pattern as outlined above and other related models [31] are compatible with this law. The inverse square model implies an assumption that *complexity control* activity is negligible and that the effort applied to evolution is approximately constant. Its parameter may vary from system to system because of differences between systems and, from time to time in a given system, because of major changes (stages) in it as it evolves [11]. Similarly, major changes in organisations, applications, external forces such as market factors and so on may be reflected in process changes and explain why inflexion points occur, why model segmentation is in some cases beneficial. Local deviations from predicted values, on the other hand, are primarily due to different release content requirements or changing release interval. System dynamics models [e.g. 8,32] provide a further source of empirical generalisations derived from the mathematical relationships they reflect. The SD models developed within the FEAST projects provide tools for exploration of dynamic forces and constraints, many feedback based, that are likely to exercise powerful, possibly dominating, influence on evolutionary behaviours and trends of *E*-type system evolution.

Together with other observations, phenomena and invariants, the above provide the concepts and phenomenology from which the proposed theory of software evolution will initially develop and be formalised. That theory will provide a framework for a further development, as more data and observations become available. This will permit refinement to relate observed phenomena. Size and complexity growth, for example, may be related to process, organisational and environmental attributes and to control activity. From this one will seek to identify application and software process attributes to which the value of *e* in the above model might be related [31] and the detail of a further model $e = F(X_p, \dots, X_n)$, where the X_j are values of relevant application and process attributes. Given sufficient data on a variety of systems from different implementation domains one may eventually be able to determine relationships between various attributes and the impact of changes in them on values of *e*; that is, on the growth rate as the system ages, and on the restoration of growth rate by appropriate system, process or organisational changes. The models will also suggest management and engineering guidelines for improved management of the evolution process.

F The Proposed Approach

The observational level theory is derived from empirical generalisations, based on data, observations, models and interpretations of industrial software evolution as developed over the years [e.g. 1,6,11,15,19], as illustrated by the figure in section D. The above example of theoretical issues to be addressed exemplifies the intended approach to theory formation. What has been included will change, and the theory extended, as ideas are formalised, outstanding issues resolved, more observed phenomena captured and represented, explanations derived.

Inputs to theory formation will be enriched, whenever possible, by addressing questions such as "can characteristics of non-monotonic behaviour of program attributes (e.g., size) that permit prediction of observed behaviours be found?" This, and similar, questions may be addressed by control, optimisation and decision theories, Professor Rustem's area of expertise. Control theory, in particular, has played a role in past software process studies [33,34] and, may, in general, provide other ingredients for theory development [35,36], a potential indicated by informal analysis. Control and optimisation theory indicate, for example, that increasing constraints on a system (e.g. resource, pricing) reduce performance. A potential application of the proposed theory could help decide whether open source software is better in some sense than conventionally developed software since it is developed with fewer constraints [37].

Once existing observations and empirical generalisations have been structured and absorbed, theory level activity may also explore computational procedures as described, for example, in [23]. Candidate theories will be defined in terms of domain attributes. Discovery of these attributes and which of them best explains empirical observations will provide clues to key evolution process characteristics. Initial approximations that ignore phenomena, as when friction is ignored in physics or engineering, will lead to results and predictions to be validated against existing and new observations. Successive refinement and validation will extend and strengthen confidence in the emerging theory and in its domain of relevance. The accuracy of predictions will guide the search for refinement and investigation of second order phenomena.

Over a three-year period, this innovative project will establish foundations from which a fuller theory will progressively emerge, as the topic is more widely studied. Development of satisfactory definitions and initial formalisation are amongst the first challenges to be met. Subsequent theory formation based on observed behavioural invariants and phenomenology, will yield a significant contribution, providing solid foundations and a framework for further progress in process improvement. Abstracting insights, interpretations and understanding of evolution and its successful formation will also make a fundamental contribution to the development of evolvable systems. This is crucial for a world ever more reliant on software. Equally, it has important implications for general business process improvement.

G Objectives and Tasks

G.1 Objectives

- (a) develop initial explanatory, observational level, theory of software evolution based on identified empirical generalisations and evolutionary patterns of industrially developed, marketed and maintained systems
- (b) derive and extend a theoretical level explanation in a selected formalism
- (c) derive and prove inferences that follow from both levels of the theory as they develop
- (d) interpret and validate inferred consequences against already held and newly acquired empirical real world data
- (e) determine their practical implications and potential for exploitation

- (f) explore industrial exploitation and support industry introduction of practical implications of the theory
- (g) document and disseminate results of the project to encourage critical examination, refinement of results, extension of theory and, above all wide take up and extension of the approach and its practical application

G.2 Tasks

The following steps, overlapped where possible, iterated as necessary and with involvement of collaborators as appropriate, are required to meet these objectives:

- (a) identify, classify, structure patterns, invariants and other observations as basis for empirical generalisations
- (b) develop outline definitions and initial observational level theory and validate against industrial observations
- (c) explore, refine and extend observational level theory by further empirical generalisation
- (d) explore the emerging observational level theory in the context of control and other candidate theories
- (e) select and evaluate candidate formalisms and test viability by application to initial observational level theory
- (f) on the basis of a fuller observational level theory and selected formalisms develop initial theory level theory
- (g) refine, extend and continually validate both levels of theory and derive inferences and proofs at the theory level
- (h) formally derive industrial guidelines (rules and tools) for practical application and develop full documentation

H Related Work

The IC PI [38] and his colleagues have recognised and been discussing the need and potential for a theory of the software process for some time. There are also scattered references elsewhere to the absence of a theoretical basis and framework for software engineering and to the role that such a theory could play. Bennett and Rajlich, for example, state "... A major challenge for the research community is to develop *good theoretical understanding and underpinning* for maintenance and evolution, which scales to industrial applications ..." [11]. Other than thoughts recently outlined [21], we are, however, not aware of any work in theory level theory development, as defined in this proposal, whether in the wider arena of software engineering or of constituents such as software process or evolution. Informal elements at the observational level can be found in system dynamics and other process models [e.g. 8,32]. Ontology and taxonomy work has been pursued [17] and will provide inputs to the proposed study. Mathematical theory relating to formal representation, formal methods and programming languages does exist and may prove important in supporting it. But such theory is qualitatively different to that being proposed here. Despite differences due to human involvement in software evolution, as a phenomenological descriptor the theory contemplated is more akin to the theories of the physical sciences [12]. It has some relationships with the work outlined in [29], which the King's PI is particularly keen in exploring in the context of SETh. The application of *formal methods* and of the many representations and logics in computer science is also relevant here. The proposed study will have access to the necessary knowledge, understanding and experience to these approaches through Imperial and King's investigators and the individuals named in section D.

The change in paradigm implied by emergence of technologies such as OO, open source, the Internet, component-based architectures, COTS etc. and its implications are clear, as illustrated by studies of Linux evolution [37]. Similarly, hypotheses about COTS usage [20] lie at the heart of a research proposal now in preparation [39]. The proposed development based on empirical generalisations appears to be only weakly related to software technology factors. Thus one may expect, and strive to achieve, a theory independent of specific evolution paradigms or process models. Whether this can be done is an issue to be resolved. The importance of such investigations cannot be over stressed.

I Research Issues

The research hypothesis is that the behaviour of key attributes of evolution processes may be described by a formal scientific theory. Furthermore, the strength of *feedback* in driving and steering system evolution suggests that behaviour typical of such systems is to be expected. Control theoretic concepts may well find application in the proposed theory, though the nature of the feedback structure and the intimate and creative involvement of people in the process may prove a major challenge to such application. If and how this is to be achieved remains to be determined. The FEAST and other studies provide the necessary conceptual foundations, empirical data and generalisations to explore both hypotheses. Issues that arise, relate to the selection of appropriate formalisms, approaches to theory formation and means to exploit the developing theory as a source and justification for rules, guidelines, tools for software evolution.

The proposed theory will reflect the aggregate behaviour of individuals. Such a focus is established practice in software engineering with evidence of successful application, as illustrated by simulation of software projects and process modelling [32,40], resource estimation [41] and formalisation [42], including the FEAST studies [8]. Of course, the individual human element in software engineering cannot be ignored. Its investigation must be pursued by other means [43]. However, the theory development will contribute to the systematic identification of the potential, bounds and limits for aggregated process modelling. In this regard, our proposal is expected to open up a new direction that will permit process modelling to be undertaken in an innovative way. As an adventurous break into new ground, success for the SETh project cannot be guaranteed, may take years to achieve, but a high degree of feasibility is indicated, its innovative character and potential undeniable. The plans have already been exposed to the research community [21,44].

J Work Plan - The work plan pictured in the appendix describes and expands on the tasks listed in section G 2.

K Collaborators

IBM - Hursley, ICL, Isocra and QinetiQ (formerly DERA) have confirmed their participation as per attached letters. Technical personnel at Deutsche Bank and Ericsson Systems Expertise Ltd. but commitment awaits management approval. Matra-BAE Dynamics, a long time collaborator, has also expressed strong interest and their decision is awaited. The role of all collaborators will be: provision of data as available, interpretation of findings, critique of the emerging theory, support for progressive validation, exploitation and so on.

L Relevance to Beneficiaries

The absence of a software evolution theory is believed to be constraining advances in software technology, mastery of feedback in the software process and process improvement, all matters of major concern, and expenditure, to industry.

Long-term benefits will come from rationalisation, integration, validation and extension of the existing knowledge in the area. All will play an important role in creating the confidence required to achieve improvements in quality, productivity, response time, predictability, resource forecasting, planning and so on. The theory will facilitate reasoning about evolution, the process of evolution and its modelling. Advancing insight and inferences from the developing theory provides opportunities for improving software evolution practice. This is illustrated by a report "Rules and Tools for Software Evolution Planning and Management" issued to FEAST collaborators and shortly to be published [22] and in the example of section E. The report provides guidelines ready for application. Much of this material may appear self-evident. Its originality, greatly to be strengthened by development of a theory, lies in the fact that the practical guidelines are provided within a unifying conceptual framework. Even in the short term, it has significant potential for contributions to software practice and, more generally, for business and organisational process improvement.

M Dissemination

The importance of continually bringing the results of this project to the widest possible audience, to encourage others to join in development and investigation and to exploit the results, is clearly recognised. As in the FEAST projects that produced over 50 publications, the SETh results will be energetically disseminated. The widespread interest in the results obtained speaks for the effectiveness of the group in this regard. The SETh project will build on these results, extend and strengthen confidence in them and may be expected to equal this record. The proposed budget includes an amount to cover the cost of such dissemination including conference and workshop participation.

N Allocation of Personnel to Tasks

All investigators will be actively involved in the project. Prof. Lehman expects to devote no less than 50% of his time to the project. The IC located RAs and a doctoral student will work, jointly and individually at the observational level, progressively getting involved at the theoretical level. The doctoral student at King's will focus on the theoretical level. The senior IC RA will address critical project tasks, liaise between the IC and King's teams and, in consultation with the IC PI and the collaborators, be responsible for the identification, formulation and documentation of rules and guidelines and for monitoring activities to ensure that project objectives are being met. All this activity will be under joint supervision of the IC and King's PIs. As individual topics and tasks are defined, the PhD students will be guided by their PI supervisors to each achieve identifiable contributions subject to appropriate contribution to project objectives. On the basis of their experience and other current research, Profs. Hankin (IC) and Maibaum (King's) will supervise selection of appropriate formalisms and formal theory formation. Prof. Rustem will be involved in directing resolution of the various control theoretic issues mentioned above or that arise as the project progresses. Using email, telephone and visits every six months or so (as opportunities permit), the overseas advisors will contribute to concept formation and planning, and review progress. It is hoped that Profs. W M Turski (U. of Warsaw) and D E Perry (U. of Texas at Austin) and Dr V Stenning (Anshar Ltd.) will also be able to continue their long-standing involvement with the Department and group. Industrial collaborators will contribute from 10 to 45 person-days of technical effort, participating in workshops and by direct contact with the academic groups. They will critically examine progress, review documents and contribute to development, interpretation and validation of concepts, models and the evolving theory in relation to their industrial criteria and individual software evolution and organisational processes.

O Management and Resources

O.1 Project Personnel

One RA1A with wide interests in the software process and evolution, formal methods and/or theory formation and mathematical skills to apply, interpret and explore modelling techniques and models, will work at IC on the basic development under the guidance of the Investigators and the external advisors. A second, RA1B will bring together and organise the data and observations required for and work with the senior RA on theory development, developing tools required for the project, be responsible for a SETh web page and carry out other project related duties. It is also planned to attract and involve two full time doctoral student, one each at IC and King's.

O.2 Project Management

Direction and day to day project management of the IC contribution, including overall progress monitoring of the project as a whole and timely provision of deliverables will be the responsibility of the IC PI. The King's PI will be responsible for management of the King's team. The two PIs will be jointly responsible for liaison between the two teams and with the external advisors. The co-investigators will provide critical appraisal of approaches being used, of project plans, conclusions drawn, draft reports and papers. Seven workshops attended by all those involved collaborators and guests will review results to ensure technical soundness and that plans for the remainder of the project will lead to objectives being met.

O.3 Travel and Subsistence - supporting both IC and King's Teams

Seven project workshops with some 20 participants each are planned over the three year project at a total cost of £1200. Growing interest in the topic and plans for continuous, widespread dissemination requires active participation in software engineering conferences and workshops such as ICFCS, ICSE, ICSM, ESEC/FSE, ETAPS, ISPSE, EWSPT, WESS. Support (including registration fees) is requested for 12 UK/European (£8000), 8 US and other destinations *person-trips* (£11,200), 6 visits to the named advisors and interested organisations in industry and government, timed to coincide with these events, (£4800), 30 UK *person-trips* for work sessions with the collaborators (£1900).

O.4 Equipment, Consumables, Support

Desktop computers and a printer are required for the PI, and two RAs at IC and for the PI and student at King's. The level of expected travel and presentations requires a laptop for use by all project personnel. Costs include computer consumables, system support, connection to departmental networks, reproduction costs, new software, upgrades.

O.5 Other Resources

As indicated in the attached letters of support, the industrial collaborators will provide significant staff effort to work on the project, travel and subsistence for those staff to attend meetings, and on-site facilities as required. If other groups involved in related work can be found, links to build on common interests will be established.

P **References**¹ - for a full bibliography and copies of recent papers follow links at [8]

- [1]* Lehman MM, *The Programming Process*, IBM Res. Rep. RC 2722, IYorktown Heights, NY, Sept. 1969.
- [2]* Belady LA and Lehman MM, *An Introduction to Program Growth Dynamics*, in *Statistical Computer Performance Evaluation*, W. Freiburger (ed.), Acad. Press, NY, 1972, pp. 503-511.
- [3] Lehman MM and Ramil JF, *Software Evolution*, Invited Keynote Lecture, IWPSE 2001, 2nd Intl. Workshop on the Principles of Software Evolution, Vienna, Sept. 10-11, and in *Ency. of Softw. Eng.*, 2nd. Edition, Wiley, 2002
- [4][#] Lehman MM *et al*, *Evolution as a Noun and Evolution as a Verb*, SOCE 2000, 12-13 Jul. 2000, Imp. Col., London
- [5] Paulk MC, Curtis B *et al*, *Capability Maturity Model, Version 1.1*, IEEE Software, v. 10, n. 4, 1993, pp. 18 – 27
- [6] *SEBPC Systems Engineering for Business Process Change*, EPSRC Managed Research Programme
- [7] Lehman MM and Belady LA, *Program Evolution - Processes of Software Change*, Acad. Press, London, 1985
- [8] *FEAST Projects Web Site*, Dept. of Comp., Imp. Col., Aug. 2001, <http://www.doc.ic.ac.uk/~mml/feast/>
- [9] Maibaum TSE, *Mathematical Foundations of Software Engineering: a Roadmap*, in A. Finkelstein (ed.), *The Future of Software Engineering*, ICSE 2000, June 4-11 Limerick, Ireland, ACM ord. no. 592000-1, pp. 161-172
- [10] Naur P and Randell B (eds.), *Software Engineering*, Report on a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Jan. 1969, 231 pps.
- [11] Bennett KH and Rajlich VT, *Software Maintenance and Evolution: A Roadmap*, in Finkelstein, A. (ed.), *The Future of Software Engineering*, 22nd ICSE, Limerick, Ireland, Jun. 2000, ACM ord. no. 592000-1, pp. 73-87
- [12] Carnap R, *An Introduction to the Philosophy of Science*, Gardner M (ed.), Dover, Toronto, 1995, 300 pps.
- [13] Lehman MM, *Feedback in the Software Evolution Process*, Keynote Address, Proc. CSR Eleventh Annual Wrksh. on Softw. Ev. - Models and Metrics. Dublin, 7-9th Sep. 1994, Also in *Info. and Softw. Tech.*, spec. iss. on Software Maintenance, v. 38, n. 11, 1996, Elsevier, 1996, pp. 681-686
- [14] Drucker PF, *The Theory of the Business*, Harvard Business Review, Sept.- Oct. 1994, pp. 95-103
- [15] Kemerer CF and Slaughter S, *An Empirical Approach to Studying Software Evolution*, IEEE Trans. Softw. Eng., v. 25, n. 4, Jul./Aug. 99, pp. 493-509
- [16] Turski WM, *Specification as a Theory with Models in the Computer World and in the Real World*, Infotech State of the Art Report, P. Henderson (ed.), se. 9, n. 6, 1981, pp 363-377
- [17] Kitchenham B, *et al*, *Towards an Ontology of Software Maintenance*, J. of Softw. Maint., v. 11, 1999, pp 365-389
- [18] Anton A and Potts C, *Functional Paleontology: System Evolution as the User Sees It*, ICSE 2001, pp. 421-430
- [19] Shaw M and Garlan D, *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, 1996
- [20] Lehman MM and Ramil JF, *Software Evolution Phenomenology and Component Based Software Engineering*, IEE Proc. Softw., sp. issue on Component Based Software Engineering, v. 147, n. 6, Dec. 2000, pp. 249 - 255
- [21] Lehman MM, *Towards a Theory of Software Evolution - And Its Practical Impact*, inv. lect., Proc. ISPSE 2000, 1-2 Nov. 2000, Kanazawa, Japan, IEEE Comp. Soc. Press, pp. 2 - 11
- [22][#] Lehman MM, *Rules and Tools for Software Evolution Planning and Management*, to appear in *Annals of Softw. Eng.*, v. 11
- [23] Shrager J and Langley P (eds.), *Computational Models of Scientific Discovery and Theory Formation*, Morgan Kaufmann Publishers, Inc, San Mateo, CA, 1990, 498 pps.
- [24] Dijkstra ED, *Go To Statement Considered Harmful*, CACM, v. 11 Mar. 1968, pp. 147-148
- [25] Wirth N, *Program Development by Step-wise Refinement*, CACM, v. 14, n. 4, Apr. 1971, pp. 221-227
- [26] Parnas D, *On the Criteria to be Used in Decomposing Systems into Modules*, CACM, v. 15, Dec. 1972, pp. 1053-8
- [27] Lehman MM, *Uncertainty in Computer Application and its Control through the Engineering of Software*, J. of Software Maintenance, Research and Practice, v. 1, 1 Sept. 1989, pp. 3-27
- [28] *id.*, *Uncertainty in Computer Application*, Technical Letter, CACM, v. 33, n. 5, pp. 584, May 1990
- [29] Haeberer AM and Maibaum TSE, *Scientific Rigour, an Answer to a Pragmatic Question: a Linguistic Framework for Engineering*, ICSE 2001, Toronto, Canada, May 12-19, 2001, IEEE Comp. Sc. Press
- [30] Turski WM, *A Reference Model for the Smooth Growth of Software Systems*, IEEE. Trans. Softw. Eng., v.22, n.8, Aug. 1996, pp. 599-600
- [31] Lehman MM, Ramil JF and Sandler U, *An Approach to Modelling Long -Term Growth Trends in Software Systems*, to appear, Proc. ICSM 2001, 6-10 Nov., Florence, Italy
- [32] Abdel-Hamid TK and Madnick SE, *Software Project Dynamics - An Integrated Approach*, Prentice-Hall, Englewood Cliffs, NJ, 1991, 264 p.
- [33] Riordan JS, *An Evolution Dynamics Model of Software Systems Development*, in *Software Phenomenology - Working Papers of the (First) SLCM Workshop*, Airlie, Virginia, Aug 1977. Pub ISRAD/AIRMICS, Comp. Sys. Comm. US Army, Fort Belvoir VI, Dec 1977, pp 339 – 360
- [34]* Woodside CM, *A Mathematical Model for the Evolution of Software*, ICST CCD Res. Rep 79/55, Apr. 1979. Also in *J. Systems and Software*, v. 1, No 4, Oct 1980, pp 337 - 345
- [35] Darlington J, Pantelides CC, Rustem B and Tanyi BA, *An Efficient Approximate Algorithm for Optimal Decisions Under Uncertainty*, European J. of Ops. Res., v. 121, 2000, pp. 343-362
- [36] Rustem B, Becker R and Marty W, *Robust Min-Max Portfolio Strategies for Rival Forecast and Risk Scenarios*, J. of Economic Dynamics & Control, v. 24, 2000, pp. 1591-1623
- [37] Godfrey MW and Tu Q, *Evolution in Open Source Software: A Case Study*, Proc. Intl. Conf. on Software Maintenance, ICSM 2000, 11-14 Oct. 2000, San Jose, CA, pp. 131-142
- [38] *Publication listings* available from links at <http://www.doc.ic.ac.uk/~mml>
- [39][#] Lehman MM and Ramil JF, *EPiCS – Evolution Phenomenology in Component-intensive Software*, WESS 2001, Florence, Nov. 9, 2001, 5 pps
- [40] Kellner MI, Madachy RJ and Raffo DM, *Software Process Simulation Modelling: Why? What? How?*, J. of Systems and Software, Vol. 46, No. 2/3, April 1999, pp 91 –106

¹ "*" indicates that the publication is reprinted in [7], "[#]" indicates that a version of this reference is available via [8]

- [41] Boehm B *et al*, *Software Cost Estimation with COCOMO II*, Prentice Hall, 2000
- [42] Fenton NE *et al*, *Software Measurement: Uncertainty and Causal Modelling*, submit. to IEEE Softw., Mar. 2001
- [43] Hoc JM, *et al* (eds.), *Psychology of Programming*, Academic Press, London, 1990, 290 pp.
- [44][#] Lehman MM and Ramil JF, *An Approach to a Theory of Software Evolution*, IWPSE 2001, Vienna, Sept. 10-11