

FEAST Workshop II

24 - 25 October 1994

Department of Computing

Imperial College of Science, Technology and Medicine

180 Queen's Gate

London SW7 2BZ

+44 (0) 171 594 8213/4

fax +44 (0) 171 594 8215

Organisational Matters

General Information	2
Program	3
Attendees	4

Briefing Papers

Some Characteristics of S-, E- and P-type Programs Manny Lehman, ICSTM	7
Charts for the above	11
Evolution and Learning in the Software Process Aidan Ward, Coherent Technology Ltd.	27

FEAST Papers

F1 - Five Position Papers		32
F2 - Various Comments		41
F3 - Three-way Discussion Jose Fiadeiro, Vic Stenning, Manny Lehman		62
F4 - A Three way email Discussion Wlad Turski, Dewayne Perry, Manny Lehman		66
F5 - Manifesto and Minutes Dewayne Perry, Wlad Turski, Manny Lehman		78
F6 - Comments on Manifesto Bob Snowdon		82

- The workshop will be held on the 24/25 October in the Department of Computing, Imperial College, 180 Queen's Gate, London, SW7 2PH
- Registration fee of £55, cash or UK cheque or traveller's cheque, payable to Imperial College.
- The workshop sessions will begin both mornings at 9 45 in room 418 of 180 with coffee available from 9 15 in room 433.
- Hotel accomodation for those requesting it has been reserved at the Embassy House Hotel, 32 Queens Gate., SW7. Room charge £65 per night single.
- The nearest underground station to the hotel are Gloucester Rd. and South Kensington both on the District, Circle and Picadilly Lines. The College is equidistant from them both.
- Sessions will be in room 418 on the 4th floor.
- Tea, coffee breaks and lunches in room 433 on the same floor.
- Dinner on Monday evening will be at 6 30 for 7 00 at Physics Common Room, Level 8, Blackett Lab in Huxley Building.
- Barbara Claxton, room 554, front corridor, 5th floor, will be available to assist attendees as required.
- In her absence assistance can be obtained from the Department general office, room 437.
- Incoming telephone calls direct to +44 (0) 171 594 8213 (Barbara)
- Departmental fax number is +44 (0) 171 594 8301
- email messages marked for your attention to bpw@doc.ic.ac.uk.
- Outgoing email can be sent from Barbara's machine or from mml's in room 556.

Monday, 24 October 1994

09 15 - 09 45	Registration and Coffee		433
	Morning Chair	Nazim Madhavji	
09 45 - 10 00	Opening remarks	Manny Lehman	418
10 00 - 11 15	Characteristics of <i>S</i> -type and <i>E</i> - type programs	Manny Lehman	418
11 15 - 11 45	Coffee		433
11 45 - 13 00	The FEAST Manifesto and a Preliminary Model	Dewayne Perry	418
13 00 - 14 00	Lunch		433
	Afternoon Chair	Vic Basili	
14 00 - 15 30	Experimentation and Measurement in FEAST	Larry Votta	418
15 30 - 16 00	Tea		433
16 00 - 17 30	General Discussion		418
6 30 - 7 00	Cocktails		
7 00 - 9 30	Dinner	Physics Common Room level 8, Blackett	

Tuesday, 25 October 1994

09 15 - 09 45	Coffee		433
	Morning Chair	Colin Tully	
09 45 - 10 15	ICL Model	Bob Snowdon	418
10 15 - 10 45	AT&T Model	Dewayne Perry Larry Votta	418
10 45 - 11 15	Coffee		433
11 15 - 12 45	Discussion		418
12 45 - 14 00	Lunch		433
	Afternoon Chair	Bob Bishop	
14 00 - 15 30	Issues, Challenges and Opportunities	Bob Balzer	418
15 30 - 16 00	Tea		433
16 00 - 17 30	Commitments and Funding		418

NOTE: All topic titles and assignments are tentative and subject to the agreement of the individual concerned.

Alan Ainsworth,
77 High Road
Ickenham, Uxbridge,
Middlesex
tel: +44 0895 630529,
aa5@doc.ic.ac.uk

Dr. Bob Balzer,
Information Sciences Inst
4676 Admiralty Way, Suite 1001,
Marina del Rey, CA 90292-6695
tel:+1 310 822 1511
balzer@isi.edu

Professor Vic Basili,
Dept. of Computer Science,
University of Maryland,
College Park, MD 20742,
tel: +1 301 405 2668,
fax +1 301 405 6707
basili@mimsy.umd.edu

Professor Keith Bennett
Durham University
Science Labs
South Road
Durham
DH1 3LE
Tel: 091 374 2632
keith@uk.ac.dur.easby

Bob Bishop,
18 Aggisters Lane
Wokingham, Berks, RG11 4DN
tel: +44 (0)734 774017
fax: +44 (0)734 894 254
rb@gid.co.uk

Anthony Brigginsshaw, ICSTM,
180 Queens Gate, London, SW7 2BZ
tel: +44 (0)71 589 5111 ext 58248
fax: +44 (0)71 581 8024
awb@doc.ic.ac.uk

Dr Bill Curtis
TeraQuest Metrics & SEI
3644 Ranch Creek
Austin, Texas 78730-3701, USA
Tel: 1 512 346 2435 :
Fax: 1 512 346 4677
email: curtis@acm.org

Mark Dowson,
Marleston Software Tech. Inc.,
525K East Market Street 303
Leesburg, VA 22075,
USA tel: +1 703338 3951
Fax: +1 703 771 8413
dowson@marlstone.com
(2031 Chadds Ford Drive)
703 476 9006 Fax. 703 264 1427
evenings 703 264 1427

Dave Homan,
Northern Telecom Europe Ltd,
Oakleigh Road South,
London, N11 1HB
tel: +44 (0)81 945 3097
fax: +44 (0)81 945 3960
homan@annsgy41.northern.co.uk

City University,
School of Informatics, Dept of Bus. Comp.
Northampton Square, (Sep 94)
London EC1V OHB
tel: 477 8000 ex 3725
z.levy@city.ac.uk
Or: 48 Haparsa St,
Tel-Aviv 69085 (After Sep 94)
Israel
tel: + 972 3 6471 373
fax 972 3 566 1667

Prof. Meir M Lehman, ICSTM
180 Queens Gate,
London, SW7 2BZ
tel: +44 (0)71 594 8214
fax:+44 (0)71 594 8215
mml@doc.ic.ac.uk

Dr. Pertti Lounamaa
Nokia
P.O.Box 45, (Heikkiläntie7) FIN-00211
Helsinki, Finland
tel: +358 0 437 6594
fax:358 0 455 2091
lounamaa@research.nokia.fi

Professor Nazim Madhavji,
McGill University
School of Computer Science
McConnell Engineering Bldg.,
3480 University St, Montreal, Quebec
Ca. H3A 2A7
tel: 514 398 3740/514 284 6730
fax: 514 398 3883
madhavji@opus.cs.mcgill.ca

Dr. Bashar Nuseibeh,
ICSTM,
180 Queens Gate,
London, SW7 2BZ
tel: +44 (0)71 589 5111 ext 58286
fax:+44 (0)71 581 8024
ban@doc.ic.ac.uk

Dr Dewayne Perry,
AT & T Bell Laboratories
Room 2B 431,
600 Mountain Avenue,
Murray Hill,
New Jersey 07974, USA
tel: +1 908 582 2529,
fax: +1 908 582 7550
dep@research.att.com

David Redmond-Pyle
LBMS
Evelyn House
62 Oxford Street
London W1N 9LF
tel: 071 636 4213
fax: 071 636 2708
drp@lbms.co.uk

Suzanne Robertson,
Atlantic Systems Guild Inc.,
11 St. Marys Terrace,
London, W2 1SU
tel: +44 (0) 262 3395
fax: +44 (0)71 262 1378
100065.2304@compuserve.com

SFC,
West Avenue
Kingsgrove,
Stoke on Trent, ST7 1TL
tel: +44 (0) 782 771000
fax: +44 (0) 782 776667
ras@kid01pml.icl.co.uk

Prof. Vic Stenning,
Warilda, Soames Lane,
Ropley,
Hants SO24 OER
tel/fax: +44 (0)962 772531
vs6@doc.ic.ac.uk

Colin Tully
Colin Tully Associates
6 Meadow Hill Road
Tunbridge Wells, Kent. TN1 1SQ
Tel and Fax: 44 892 539125

Prof. Wlad M Turski,
Institute of Informatics
University of Warsaw,
Banacha 2,
00-903 Warsaw 59,
Poland
tel: +48 2 658 3522,
fax: +48 2 658 3164
wmt@mimuw.edu.pl

Dr Larry Vota
AT&T Bell Laboratories
Napierville, IL
votta@research.att.com

Aidan Ward,
Coherent Technology Ltd.,
Suite 10, Challenge House,
Sherwood Drive
Bletchley,
Milton Keynes, MK3 6DP
tel: +44 (0)908 274591
fax: +44 (0)908 274693
Aidan@CohTech.co.UK

Dr. Alan Whitfield
163 Sycamore
Wilnecote, Tamworth
Staffs, B77 5HF

Dr Tarek K Abdel-Hamid,
Administrative Sciences,
Naval Postgraduate School, Monterey, CA 93943
tel: +1 408 656 2686,
fax: +1 408 656 3407
3991P@NAVPGS.EARN

Barry Boehm CS Dept.,
University of Southern Cal.,
LA CA 90089
USA
Tel: 213 740 8163
boehm@cs.usc.edu

Valerie Downes, Ovum Ltd.,
1 Mortimer St., WIN 7RH
tel: +44 (0)71 255 2670
fax: +44 (0)71 255 1995
100350.1627@compuserve.com

Dr Anthony Finkelstein,
ICSTM,
180 Queens Gate, London, SW7 2BZ
tel: +44 (0)71 589 5111 ext 48261
acwf@doc.ic.ac.uk

David Freestone, BT,
Section B81, 160PP27
Martlesham Heath, Ipswich IP5 7RE
tel: +44 (0)473 646433
fax: +44 (0)473 640897
freestone_d@bt-web.bt.co.uk

Professor Carlo Ghezzi
Dipartimento di Elettronica e
Informazione, Politecnico di Milano
Piazza L. da Vinci, 32
20133 Milano, Italia
Tel: 010 39 2 239 93411
ghezzi@elet.polimi.it

Dr. Peter Gibbins,
Sharp Labs of Europe Ltd.
Edmund Halley Road,
Oxford Science Park
Oxford, OX4 4GA
tel: +44 (0)865 747711
fax: +44 (0)865 747717
gibbins@sharp.co.uk

Professor Peter Henderson
Dept. of Computer Science
Southampton University, University Road
Highfield, SO17 1BJ
Southampton University
tel:01703 595000

Dr. Marc Kellner,
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
tel: +1 412 268 7721
Fax: +1 412 268 5758
mik@sei.cmu.edu

Prof. Peter Lee,
Department of Trade and Industry
151 Buckingham Palace Road,
London, SW1W 9SS
tel: +44 (0)71 215 1967

Prof. Stuart Madnick
MIT Sloan School of Management
Room E53-321
20 Wandsworth Street
Cambridge, MA 02139, USA
email:smadnick@MIT.EDU

Prof. Tom Maibaum, ICSTM
180 Queens Gate, London, SW7 2BZ
tel: +44 (0)71 594 8283/84
fax:+44 (0)71 594 8285
tsem@doc.ic.ac.uk

Jean Pierre Moularde,
SEMA (France)
16, rue Barbes
F-92126 Montrouge Cedex,
France
tel: +33-1-40 92 43 16
fax: +33-1-46 56 96 53
moularde@metra.fr
Secr phone: 33-1-40924218
Secr. fax: 33-1-40924241

Brian Oakley,
120 Reigate Road
Ewell, Epsom,
Surrey, KT17 2BX
tel: +44 (0)81 393 4096
fax: +44 (0)81 393 9046

Dr Bob Rockwell,
Softlab GmbH.,
120, D-81677 Munich,
tel: +49 89 93 00 1475,
fax:+49 89 938 281
Home: Milloeckerstr. 34a,
D-81477 Munich, Germany
tel: 049 89 784 9002
email: roc@softlab.de
100317.1353@compuserve.com

Paul Rogoway
26 Haneseim St.
Petach Tikva 49550, Israel
Tel: 010 972 3 388 950
h - 3 922 7744
car 5206666 sub. 50351

Dr. Berc Rustem, ICSTM
180 Queens Gate, London, SW7 1BZ
tel:+44 (0) 71 594 8345
fax:+44(0) 71 5890 8024
br@doc.ic.ac.uk

Graham Samuel
39 Laurier Road, London, NW5 1SH
Tel: 071 485 7916
email: graham@livfoss.demon.co.uk

Gordon Scarrott, 34 Parkway
Welwyn Garden City, Herts. AL8 6HQ
tel: +44 (0)707 323 073

Owen Shallcross
19 Grahame Avenue
Pangbourne, Berkshire
RG8 7LF
Tel: 0734 843107
Fax: 0734 842126

Northern Telecom Europe Ltd.,
Oakleigh Road South,
New Southgate, London, N11 1HB
tel: +44 (0) 81 945 3199
fax: +44 (0) 81 945 3454
deps@lon4Q.nt.com

Carolyn Story,
BNR Europe,
London Road
Harlow, Essex, CM17 9NA
tel: +44 (0) 279 403648
fax +44 (0) 279 437830
carrie@bnr.co.uk

1 Introduction

The *S*-type, *E*-type and *P*-type classification for programs¹ (that is for software and computer applications) was first defined in the early seventies and has been further refined ever since. Most recently, as a result of discussions in the FEAST project the question of the classification and, in particular, of the distinctive properties of *E*-type systems has been raised. This note represents an initial statement of the currently recognised properties of the three types of program

2 Definition

2.1 S-type Program

An S-type program is a computer program that is required to satisfy some pre-stated specification.

That is, the specification is the sole, complete and definitive determinant of program properties. It is the *only* arbiter of correctness. Once a program has been developed correctness relative to that specification, in the strict mathematical sense, is the only criterion of success. The completeness, validity, relevance or aesthetics of the specification and any program properties not explicitly included in the specification nor (formally) inferable from it, are extraneous issues. *S*-type programs are mathematical objects and artifacts. They must, at all times, be developed and processed with the rigour, precision and discipline that this implies and permits. In their context *correctness* has an absolute meaning. It is a specific, formally demonstrable, relationship between a program and its specification.

2.2 E-type Program

An E-type program is a computer program that realises (models) a computer application in some real world domain.

For real world applications it is the *consequences of execution* as, for example, the information conveyed to human observers or the behaviour induced in attached or controlled artifacts that determines the acceptability of the program. Ultimately, one is concerned with the value or level of satisfaction execution of the program yields. The concept of *correctness* that defines the acceptability of *S*-type programs expresses a precise relationship based on calculable equivalence between a program or other formal representation and some higher level specification. For *E*-type programs the real world is both specification and arbiter. Because the real world domain is unbounded calculable equivalence with it and, therefore, correctness relative to it, are meaningless concepts. Moreover, once installed, the system and its software become part of the application domain. Together they comprise a feedback system that cannot, in general, be precisely or completely known or represented. The specification is intrinsically volatile.

For *E*-type systems the notions replacing the Boolean concept of correctness are, therefore, essentially fuzzy, based on both qualitative and quantitative measures. System acceptability depends on a process of subjective human judgment. *It is the detailed behaviour under operational conditions that is of concern.*

2.3 P-type Program

A P-type program is computer program that is required to produce an acceptable solution to some stated problem.

This was the definition of *P*-type programs as originally formulated. It was based on the observation that when *acceptability* of a problem solution rather than a relationship of *correctness* relative to a specification is of concern the program should not be classified as of type *S* even when though the problem statement can be formalised, its program instantiation rigorously developed and it can be proven correct. With this view if, when the program is analysed or executed, undesirable properties (side effects) not addressed in the problem statement are observed the problem statement must be modified to exclude them. A new version of the program may then be created to provide an acceptable solution. That is, this *P*-type program definition allows for changes in the specification to achieve a satisfactory outcome when the program is executed and the solution is applied.

More recently, an alternative view has emerged. This considers that if a complete and unambiguous statement of the problem has been provided so that the program may be completely specified it should be classified as of type *S*. despite the fact that program correctness relative to that specification is a means to the *real end*, achievement of a satisfactory solution. It requires one to accept that if, after implementation, the program product is unsatisfactory, for whatever reason, one produces a *new* problem statement (possibly by revision of the old) and infers a *new* program (possibly by reasoning based on the unacceptable program).

Which of these alternative views is the more useful, that is whether one adopts a two or a three member classification scheme requires further consideration. In any event, *P*-type programs are intermediate to the other two-types and it is not self evident that they possess any special properties. If such a program is completely specified it may and should be treated as an *S*-type otherwise as of type *E*. The binary classification schema that results from this later viewpoint bisects the universe of programs into those for which correctness, as defined above, is *sufficient* and those for which (at the whole system level) it is not. The present analysis is restricted to *S* and *E* types.

3 Some Characteristics

3.1 S-type Programs

- An *S*-type program is completely defined by a specification
- Even if, initially, informally expressed that specification can always be transformed into a formal notation
- By definition the specification is fixed - it may not, for example, be changed because of dissatisfaction with the derived program
- Where circumstances call for revision of the specification, the revision is viewed as a new specification, and the resulting program as a new program - even though it may have been derived from the earlier program

¹ For convenience the term "*X*"- type is normally used to categorise *programs*. It is equally meaningful to refer to "*X*"- type software or applications or to programs, software or applications of type *X*.

- The program may be derived from its formal specification by mathematical reasoning applying the logic associated with the notation - the specification is an extralogical axiom of which the program is a provable theorem
- Sole criterion of successful implementation is the program's correctness - with respect to the specification
- Correctness is absolute - the program will have all the properties required by the specification independently of the individual(s) who derived it
- The starting point of the S-type development process is uniquely defined and fixed
- The S-type development process is an instance of logical and mathematical inference - it is well established though the specific techniques used may vary from instance to instance
- It may have additional properties not addressed by the specification - as determined by the derivation process and the individual(s) executing it
- The specification / program relationship is recursive - that is, a specification may, itself, be a program that is derived from (is a theorem of) a higher level specification (some more abstract extralogical axiom)
- Similarly, a program may be taken as a specification (extralogical axiom) from which a lower level program may be derived - for example an executable program compiled from a high level language program
- S-type programming elements are appropriate as the unit of individual programming assignment - no individual should ever be given an implementation assignment that is not fully specified

3.2 E-type Programs

- By definition the real world is the operational domain - the application concept and its operational domain are initially unbounded and encompass the entire universe; any entity, process or phenomenon in that domain may influence execution
- User and usage satisfaction (acceptability?) determines the value of the system, it is the ultimate criterion of successful completion
- Any E-type system is a model of the application in its the real world domain
- As such, the the solution system contains a - possibly implicit - model of itself
- The domain of operation is, conceptually, unbounded
- It is also essentially continuous
- So is the application concept itself - there is no limit to the number or variety of the concepts, phenomena and behaviours that can be associated with it
- At least in so far as humans are involved in the processes and phenomena of the application and its computerised model, precise and complete theories/models of domain and application properties - including behaviour - cannot exist
- Neither application nor its operational domain can be precisely or completely represented/modelled by finite and discrete models - that is by software
- E-type software model can, therefore, not be complete - or precise
- To bridge the consequent gap between the real world application in its domain and its software model assumptions are made and embedded in the system - in the form of theories, algorithms, structures, parameters, sequencings etc
- The real world undergoes continuous change
- Since it involves human observation, interpretation, action, reaction, creative decision and so on such change is unpredictable
- Some of the assumptions embedded in the system will become invalid
- The consequences under execution of such invalidity are unpredictable
- However often an E-type program has been successfully, that is satisfactorily executed, it may fail to do so as a consequence of a, now, invalid assumption
- The existence, location or nature of such invalid assumptions cannot be known
- The outcome of any execution of an E-type program is uncertain, - unpredictable
- Fixing bounds for a program and its operational domain is an integral part of the evolution process² - elements contained in or excluded from the application or its domain will be referred to as application/domain included or excluded respectively
- Some bounds will be fixed explicitly others implicitly
- The fixing of the bounds includes pragmatic considerations - available expertise, limits of knowledge and understanding, financial resources, time to completion, available hardware, etc.
- As a consequence of changes in the real world - growth, extension, societal, technological, economic, perception, etc - included elements may become redundant and excluded elements may need to be included
- During execution of the evolution process the humans involved will acquire additional insight into the the application and its underlying theories, methods and techniques for its implementation and desirable characteristics of the solution system
- Such insights trigger changes to - generally, extensions of - the application and/or the domain and/or decisions reached in earlier stages of the evolution process
- The evolution process is a learning process with the acquired information applied to activities in the forward path of the process or used to modify earlier decisions
- Installation and operation (use) of an E-type system each change the operational domain - a further feedback effect
- All these phenomena/properties lead to pressure for continuing change

² Use of the term *evolution process* in this note encompasses conventional use of the terms *software development* and *software maintenance* as

applied

- Learning and feedback driven evolution is an intrinsic property of E-type systems
- Those involved in the evolution process are essentially multidisciplinary representing at the very minimum application expertise and software development skills
- For any but the smallest applications/systems, organisational, financial and marketing and other problems imply strong involvement and project influence of agencies responsible for such areas
- Evolution of E-type systems, in general, requires interactions between many human populated agencies involving a wide variety of knowledge, understanding, experience and authority
- This implies a need for learning, elucidation, clarification, compromise

4 Laws of Evolution

Seven laws of program evolution as originally formulated were inferred from observation and measurement of OS/360 as confirmed by measurements on other systems. Their interpretation was based on organisational and sociological behaviour. It is now possible to link them to the properties listed above with each law being related to certain of the properties. The first law *Continuing Change*, for example has to do *inter alia*, with closed loop nature of the process and the bounding of the domain. The sixth law, on the other hand while also associated with the closed nature of the process arises directly from the bounding of the application itself. Thus identifying and examining the E-type properties should prove a powerful means for achieving a sound underpinning for the laws and, perhaps, identifying additional laws. This work has, however, not yet been done in detail. For the present the laws are simply listed and some preliminary observations added.

I Law of Continuing Change

An E-type program that is used must be continually adapted else it becomes progressively less satisfactory

- universal experience
- intrinsic property of computer application
- though all systems evolve, software is distinct in its rate of evolution and in that both individual artifacts and generations of artifacts evolve
- software must be regarded as an organism
- evolution Implemented in feedback driven and controlled maintenance process

II Increasing Complexity - the second law

As a program is evolved its complexity increases unless work is done to maintain or reduce it

- Change upon change upon change
- Spreading interactions and dependencies
- Balance between progressive and anti-regressive activity determined by feedback

III Fundamental Law of Program Evolution - self regulation

Program evolution is self regulating with statistically normal distribution of measures of product and process attributes

- Pseudo independent managerial and implementation decisions
- Feedback stabilisation

IV Conservation of Organisational Stability - invariant work rate

Average global activity rate invariant over product life time

- Feedback regulation

V Conservation of Familiarity - perceived complexity

During active life of an evolving program, content of successive releases is statistically invariant

- Limited by rate of learning and acquisition
- Feedback controlled

VI Continuing Growth

The functional content of a program must be continually increased to maintain user satisfaction

- Specification bounds become performance bottleneck, user irritants
- Feedback generated pressure for change

VII Declining Quality

E-type programs will be perceived as of declining quality unless rigorously maintained

- Existing assumptions become invalid
- New assumptions, constraints become necessary
- Users become more perceptive and expectant
- Acceptance criteria change
- Application demands adaptation
- Feedback generated pressure for change

General Remark

Laws derive from organisational behaviour, describe E-type evolution dynamics and constrain management freedom

- Maintenance increasingly difficult, costly, error prone as program ages
- Management freedom increasingly constrained by system
- Complexity control significant maintenance concern
- Life time expenditure high relative to initial development costs
- Limit to growth of E-type systems via classical **route**

Remarks

Since the characteristics described by the laws derive from the behaviour of people and organisations and not from the technological specifics of computer science and software technology they must be regarded as Laws from within those disciplines and taken into account in activities such as process design and process improvement. To overcome limitations implied by laws, the organisation within which and process whereby software is developed must be changed. Organisational behaviour and management are important considerations in software engineering

Uncertainty Principle

In addition to the laws behavioural analysis, now confirmed by identification of the characteristics of E-type systems leads to the statement of an Uncertainty Principle as already identified in the characteristics listing. The principle is simply stated here together with its currently identified sources. Linking this phenomenon with the identified characteristics in detail has not yet been undertaken.

Principle

The real world outcome of E-type software execution inherently uncertain with precise area of uncertainty also not knowable

- Outcome of any execution not absolutely predictable
- No guarantee of satisfactory result even if previous operation has been impeccable

Types of Uncertainty

Three sources of uncertainty have been identified of which one may be further subdivided.

Embedded

- validity of embedded assumption set cannot be known nor responsively maintained

Sub-categorisation

Textual

- assumptions embedded in program text about:
 - operational environment
 - solution system and its environment,
 - algorithms, procedures, data etc.

Procedural

- assumptions about **validity** of development **process**, its steps, methods, tools

Heisenberg-like

- system development, installation and use changes application, solution and perception of both
- the more precise user knowledge and understanding of an application and its solution the more will views, expectations and criteria of satisfaction change
- validity of totality of assumptions about expectations or criteria of satisfaction cannot be known

Gödel-like

- An E-type program is a model of a model.....of a model of a real world application
- each model pair can be interpreted as a theory and a model of that theory or as a specification and its implementation
- hence every program is Gödel incomplete
- Gödel-like application uncertainty is **reflection** of Gödel incompleteness

5 Conclusion

The contents of this note are a summary of the observations, thoughts and conclusions of many years. As a first systematic attempt at detailed analysis and listing they must be taken as the input to a discussion rather than, in any way, conclusive. The note constitutes a working paper for the second FEAST workshop and, as modified by the discussion, for the FEAST activity beyond that. There is clearly there is scope for further thought. The last word has not been said.

Manny

mml535[papers/FEAST]

21 October 1994

Some Characteristics of *S*-type and *E*-type Software

Second FEAST Workshop

DoC

24 October 1994

M M Lehman

Department of Computing

Imperial College of Science Technology and Medicine

London SW7 2BZ

+44 71 589 5111 xt 5009

fax. +44 (0)71 581 8024

mml@doc.ic.ac.uk

- ***S*-type**
 - Program **defined** by fixed **specification**
 - **Starting point** of development process **absolutely defined**
 - Development **open loop** by definition
 - Unit of individual **programming assignment**
 - Criterion of successful implementation is **correctness** -
wrt specification

- ***E*-type**
 - Realises **real world** application
 - **Unbounded** starting point
 - Intrinsic **closed loop** development
 - **Multi-disciplinary, many person** development
 - Criterion of successful implementation user/usage **satisfaction**

- ***P*-type**
 - Solves **specific problem**
 - **Intermediate** between *S* and *E*-types

Is it the characteristic properties of *E*-type systems
that FEAST should be investigating?

- First attempt at detailed listing of both *S*-type and *E*-type characteristics provided in preprints
- Have also included statements of the Laws as derived in the past from observation, measurement, modelling of OS/360 growth as subsequently confirmed for other systems
- No systematic attempt yet to link the laws to the identified characteristics
- The first relationships have been identified
- Detailed analysis can greatly strengthen the theoretical foundations of FEAST
- And of software technology/process studies in general

A framework for process improvement

- Completely defined by a **specification**
- Specification can always be transformed into a **formal notation**
- Specification is **fixed**
- Where revision is required it is viewed as a **new** specification, and the resulting program as a **new** program - even though derived from earlier program
- Derivation (construction?) process is **open loop** - revision process is, by definition, not coupled to the mathematical process of program derivation
- Program **derived** from its formal specification by **mathematical reasoning** - specification an extralogical axiom of which program is a provable theorem
- Sole criterion of successful implementation is the program's **correctness** - with respect to the specification
- Correctness is **absolute** - program will have all the properties required by specification independently of individual(s) who derived it
- **Starting point** of *S*-type development process uniquely defined

- The *S*-type development process an instance of **mathematical inference** - well established though specific techniques used may vary
- Program may have **additional properties** not addressed by specification - determined by derivation process and individual(s) executing it
- The specification / program relationship is **recursive** -specification may, itself, be a program that is derived from (is a theorem of) a higher level specification (some more abstract extralogical axiom)
- Similarly, a **program** may be taken as **specification** for derivation of a lower level program - as in compilation for example
- *S*-type programming elements are appropriate as the unit of **individual programming assignment**

- By definition the real world is the **operational domain**
- User and usage satisfaction (acceptability?) determines **value** and **acceptability** of system
- Any *E*-type system is a **model** of the application in its the real world domain
- As such, the the solution system **contains** a - possibly implicit - model of itself
- **Domain** of operation is, conceptually, **unbounded**
- It is also essentially **continuous**
- So is application concept - no limit to number or variety of concepts, phenomena and behaviours that can be associated with it
- **Human involvement** in application processes and phenomena and its computerised model, **excludes** precise and complete theories/models of domain and application properties
- *E*-type applications and their operational domains cannot be precisely or completely modelled - by finite and discrete models, by software
- *E*-type software model can, therefore, not be complete - or precise

- **Assumptions** bridge **gap** between real world application in its domain and its software model are made, embedded in system
- The real world undergoes **continuous change**
- **Human involvement** makes such change unpredictable
- Some embedded assumptions in the system will become **invalid**
- **Consequences** under execution are **unpredictable**
- However often E-type program has been successfully executed, it may still **fail** as a consequence of a, **now**, invalid assumption
- **Existence, location** or **nature** of invalid assumptions not known
- **Outcome** of any execution of an *E*-type program **uncertain**
- **Fixing** bounds for program and operational domain an integral part of evolution **process**
- Some bounds will be explicit others implicit
- Design of bounds includes pragmatic considerations

- Changes in the real world may make included elements **redundant**, excluded elements may need to be **included**
- Humans involved in evolution process will acquire **additional insight** - application and its underlying theories, methods and techniques for its implementation, desirable characteristics of the solution system, etc.
- Such insights trigger **changes** to application, the domain and/or decisions reached in earlier stages of evolution process
- Evolution process a **learning process** - acquired information applied to activities in forward path of the process or used to modify earlier decisions
- Installation and operation (use) of an *E*-type system changes operational domain - a further feedback effect
- All these characteristics lead to pressure for **continuing change**
- *E*-type process is a **multi-loop feedback** process - inherently unstable unless adequate negative feedback controls are applied
- **Learning and feedback driven evolution** perhaps the intrinsic characteristic of *E*-type systems
- **Multidisciplinary** involvement essential in evolution process - at the very minimum, application expertise and software development skills

- For any but the smallest systems, organisational, financial, marketing and other problems imply strong **involvement** and project **influence** of **agencies** responsible in such areas
- *E*-type system evolution, in general, requires **interactions** between many human populated agencies involving a wide **variety** of knowledge, understanding, experience and authority
- Demands **learning, elucidation, clarification, compromise**

**Systematic characteristics may be formalised
as Laws**

I Law of Continuing Change

An *E*-type program that is used must be **continually adapted** else it becomes progressively **less satisfactory**

- ☞ **Universal** experience
- ☞ **Intrinsic property** of computer application
- ☞ Though all systems evolve, software is distinct in its **rate** of evolution and in that both **individual** artifacts and **generations** of artifacts evolve
- ☞ Software must be regarded as an **organism**
- ☞ Evolution Implemented in **feedback** driven and controlled maintenance process

II Increasing Complexity - the second law

As a program is evolved its **complexity** increases unless **work** is done to **maintain** or **reduce** it

- ☞ **Change** upon change upon change
- ☞ Spreading **interactions** and **dependencies**
- ☞ **Balance** between progressive and anti-regressive activity determined by **feedback**

III Fundamental Law of Program Evolution - self regulation

Program evolution is **self regulating** with statistically **normal** distribution of measures of product and process attributes

- ☞ Pseudo **independent** managerial and implementation **decisions**
- ☞ **Feedback** stabilisation

IV Conservation of Organisational Stability - invariant work rate

Average global **activity rate invariant** over product life time

- ☞ **Feedback** regulation

V Conservation of Familiarity - perceived complexity

During active life of an evolving program, **content** of successive releases is **statistically invariant**

- ☞ Limited by rate of **learning** and **acquisition**
- ☞ **Feedback** controlled

VI Continuing Growth

Functional content of a program must be continually **increased** to maintain user satisfaction over its lifetime

- ☞ Specification **bounds** become performance **bottleneck**, user **irritants**
- ☞ **Feedback** generated **pressure** for **change**

VII Declining Quality

***E*-type programs will be perceived as of declining quality unless rigorously maintained**

- ☞ Existing **assumptions** become **invalid**
- ☞ New **assumptions, constraints** become necessary
- ☞ Users become more **perceptive** and **expectant**
- ☞ Acceptance **criteria** change
- ☞ Application **demands** adaptation
- ☞ **Feedback** generated **pressure** for **change**

- Laws derive from **organisational behaviour** and describe *E*-type evolution **dynamics**
- **Constrains** management freedom

p Corollaries

- Maintenance increasingly **difficult, costly, error prone** as program ages
- Management **freedom** increasingly **constrained** by system
- **Complexity control** significant maintenance concern
- **Life time** expenditure **high** relative to **initial** development costs
- **Limit to growth** of *E*-type systems via **classical route**

Remark

Phenomena addressed are **outside** realm of **computer** science and software technology since they derive from **behaviour** of **people** and **organisations**

Interpretation of **observation** and **measurement** in terms of that **behaviour**

From **within** technology phenomena must be accepted as governed by laws

To overcome **limitations** implied by laws, the **organisation** within which and **process** whereby software is developed must be **changed**

Organisational behaviour and **management** are important considerations in software engineering

Laws provide elements of **process theory**

Complemented by **Uncertainty Principle**

*Real world **outcome** of E-type software execution inherently **uncertain** with precise **area of uncertainty** also not **knowable***

- Outcome of any execution not **absolutely** predictable
- No **guarantee** of satisfactory result even if previous operation has been impeccable

Types of uncertainty

- **Embedded**
 - **validity** of embedded **assumption** set cannot be known nor responsively maintained

- p **Sub-categorisation**
 - **Textual**
 - **assumptions** embedded in program text about:
 - ⇒ **operational** environment
 - ⇒ **solution system** and its environment,
 - ⇒ **algorithms, procedures, data** etc.

 - **Procedural**
 - assumptions about **validity** of development **process**, its **steps, methods, tools**

- **Heisenberg-like**
 - system development, installation and use **changes** application, solution and perception of both
 - the more **precise** user knowledge and understanding of an application and its solution the more will **views, expectations** and **criteria** of satisfaction **change**
 - **validity** of totality of **assumptions** about expectations or criteria of satisfaction **cannot be known**

- **Gödel-like**
 - *E*-type program is a **model of a model . . . of a model of a real world application**
 - each model pair can be interpreted as a **theory** and a **model** of that theory or as a **specification** and its **implementation**
 - hence every program is **Gödel incomplete**
 - Gödel-like application uncertainty is **reflection** of Gödel incompleteness

Management **control** and software process **constraint**

- Laws and principle as element of process **theory**
- **Observation** ^{fi} **Phenomenology** ^{fi} **Measures** ^{fi} **Models** ^{fi} **Theory**
- **Fixed specification** of *S*-type development rules out development of a global dynamics
- For *E*-type software feedback ever present **driving, controlling** - and hence **constraining** - mechanism

Effect on process improvement

- Characterisation a **first stab**
- Need for **discussion, clarification, completion**
- Systematic identification and analysis a significant part of the **FEAST investigation**

Major input to process improvement process

1) Introduction

As Pat Hall has recently shown it can be useful to take a postmodern stance and refuse to believe any longer in grand utopian designs for the "right" way to conceive a town or a building or to run a society, to refuse faith in progress and a better tomorrow. Refusing faith in progress may seem a strange way to begin a discussion of the software process and how to improve it, but we will see that such faith may well block our understanding of what it takes to improve. We must take a limited and local view of how technology can be used in an unobtrusive way before we can understand the necessary radical commitment to respect the needs of the social fabric of a business rather than fitting flexible and dispensable people around some software system that is a totem for progress.

Before the role of learning and evolution in the software development process can be understood, the concept of software must be reframed. Software may appear to be a consumer product, a commodity that is selected, used and discarded: there clearly is software in mature application areas where this model is useful. The software that makes up a typical information system in a business organisation, however, has to be deconstructed in a different way. Its criteria for correctness and usefulness lie entirely in the unstable realm of business values and consensus within the organisation. The expectation is that such a system will reflect and capture the unique insights of the organisation into how to perform aspects of its business.

A business organisation has a purpose, even if it is not made explicit and even if it defaults to economic survival. Even a one person organisation is a complex system of interdependencies with suppliers and customers where any rational congruence between its actions and its purpose is hard to find. The people in an organisation and their shifting alliances are both the substance and hope of the organisation and perpetually in the way, restricting access to its goals. Organisations are dysfunctional in this sense, consuming much of their energy in internal difficulties rather than harnessing it to their purposes. The degree to which this dysfunction can be detected or acknowledged from inside an organisation is very variable. Unfortunately, but quite inevitably, both software systems and their development process are caught up in this dysfunction. The postmodern view is important because it respects this contingency, relativity of values, fallibility of conception and does not sweep away the dysfunction with some grand design for a better future. How many information systems are built not as support but as a cure for business dysfunction that there is not the political will or common purpose to sort out?

The significance of evolution and learning is that they emphasise the dynamic situation that form a context for software development. It is not simply that the software product can usefully evolve over time, it is that the very values that system will be judged against are dynamic and contingent. Indeed we will find it useful here to recognise at least four realities we will have to deal with, and which are not even connected unless we make the connections:

- 1) System creation is often based on a mythic reality of inventing the future. If the future is not sufficiently distinct from the present it is not worth building the system. This reality does not necessarily have any continuity at all: it is subject to artistic temperament.
- 2) System use is based in a social reality of the values and meanings of the groups interacting with it. A system is only useful if it is used and this is the world of fashion, of prejudice, of the group defensive behaviour at the root of much business dysfunction.
- 3) System operation is grounded in a sensory reality of scientifically observed events and structures. The system has to work in some engineering sense that does not relate to the above realities. Systems are a graveyard for corporate ambition because the system cannot be commanded to work, money will not buy out bugs.
- 4) There are unitary realities that may impinge on the system as well. The tax and legal systems are unitary realities, accepting no relativity or contingency. Senior management may come from a reality where values may not be questioned or compared.

System development is performed by a small group of people as part of a larger organisation. The literature on software process starts by drawing a dividing line and insisting that the development group can only develop what they have been asked to, and that it is therefore important to develop a detailed set of requirements and get them signed off. This attempt to control the relationship with the host organisation by formalising the boundary is misplaced and naive. It focusses on the software product as a commodity to be judged independently of its social context, assumes a single reality. To try to close the boundary, to police the exchanges across it with change control and work authorisation procedures is to promote failure.

The relationships between groups within an organisation are never straightforward. Groups get caught up in a need to defend themselves and in so doing they attribute to other groups some of their own problems. The anxiety of trying to do real work in an imperfect world makes it inevitable that the good aspects of the group get split off from the bad aspects which can then be attributed elsewhere. Sometimes we have a need for enemies! One of the defences which development groups have at their disposal is the technical nature of development. Maths, logic, specialist technical skills are needed to be able to understand systems: business managers cannot and do not understand what it is they are asking for. The issue reduces to this: if I attribute a problem of mine to someone else I can no longer solve the problem. To the extent that a development group projects its problems out into the business, onto suppliers or wherever, it cannot overcome them. Similarly, if the business sponsoring

This paper is about the software development process. The key questions we will illuminate are these:

- 1) The industry question: Is the present high (and grossly under-reported) failure rate of software projects going to converge to zero by a progressive tightening up of the technical transformations in software process?
- 2) The organisation question: What is process engineering or process improvement the answer to?
- 3) The practitioner question: To what degree is the application of engineering techniques in the software process appropriate and legitimate?
- 4) The management question: What is the motivation behind process improvement, engineering control over technical tasks or management control over people?

We will address these questions by developing a holistic description of process that allows us to situate engineering as a technique.

2) The role of engineering in the software process

The culture among the Information Technology fraternity often rejects a role for emotion and intuition in the formation and debate of judgement. Legitimacy stems from explicit, logical argument, from detailed investigation by scientific methods from tight control of expenditure and effort. Indeed my own straw polls would indicate that IT people are self-selected as preferring cold logic to warm people. Here we want to open up the process question of what types of approach and activity are effective in the development process.

The problem with the cultural bias is that it skews the collection of data on this question. Other fields such as economics have plainly allowed the development of formal models to take place at the expense of relevance and engagement in the underlying task. On a smaller scale it is clear that accountancy techniques very often throttle the goose that lays the golden eggs.

First we want to draw an important distinction between task and process. Task looks at a single objective and is directly concerned with its achievement. In the software development world tasks, such as the testing of a software module, can always be improved by applying engineering principles and advancing the technical techniques and procedures involved. To some extent the association of tasks with engineering approaches is tautologous because tasks are typically defined by objectives that lie in the engineering domain. The process is a different beast. It looks at end-to-end effectiveness in the higher level task (i.e. system development) and asks questions about what factors in the whole environment make a difference to performance. Here the role of engineering is less clear.

The reductionist view of process believes that a (balanced) approach to improving the quality of individual task performance, perhaps with the introduction of new tasks, will always increase the quality of the process as a whole. If it does not, it can only mean that engineering understanding of the interconnectedness of tasks in the process is faulty. The sophisticated extreme of this view of process is the construction of systems dynamics models that animate the interdependence of process factors.

The holistic view espoused here says that there is not a task to be performed nor an objective to be achieved that can not be carried out in a mechanistic way that does not build to overall success. This argument from "work to rule" says that the formal process structure always fails to capture the essence of the overall task. The holistic view keeps as primary the joint group aggression needed to perform any real work in an organisation, and insists that engineering is a tool to support the effective deployment of that aggression.

The reductionist view is scientific, believes a priori that thought processes that are not explicit and logical cannot deliver the goods. If there is not a process definition connecting tasks they will not connect properly, if there is not a detailed plan (schedule?) then coordination must be lacking. The idea that communication might be interfered with by formality is anathema. The idea that formality might be a displacement activity designed to shield developers from the anxiety of facing the underlying task is heresy and would get you sacked in many organisations.

To understand the holistic view we need to turn the tables somewhat and use the attitudes displayed by developers and their host organisations as diagnostic. Choose an organisation at random. Ask the developers and the business about the engineering process they use and about the suitability of the information systems produced. You will get defensive answers, maybe referring to the ideal state already achieved, maybe referring to the impossibility of doing anything else given the constraints. You would be amazed to find an attitude of open enquiry into better places to be. So at the root of the software process we find defensive and often bitter relations between the development groups and the business. In the few organisations I have observed where there was no hostility across this boundary there was acquiescence and a truce: we know, however, that there will need to be aggression and managed conflict to get work done.

Systems development is about organisational change. The group of people responsible for a development need to be authorised to drive the organisational change: hence the conflict. Organisational change is the making concrete of a vision, a fantasy of how things might be. It is creative and imaginative in nature. It cannot be achieved without sufficient authority and autonomy. The

underlying observations. Three decades of trying to make strategic planning explicit and logical have only emphasised the underlying non-rational elements.

The perspective on process that we need to carry forward here is:

- 1) All the important aspects of process involve the communication of complex and difficult information that cannot be assumed to be welcome or even acceptable.
- 2) The engineering tasks involved in the process modulate the underlying primary group dynamics, both within the group and between the group and its organisational context.
- 3) The underlying tasks of organisational change with associated systems change are fundamentally creative, involve imagination and fantasy, take their values from vision.

Evolution and Learning

Machine models of mechanically interconnecting tasks in a process are inappropriate for the reasons dealt with above. In their place we want to use biological models and in particular models of:

- 1) educational development and
- 2) evolution in the sense that an ecosystem evolves.

An ecosystem is a dynamically balanced system that has homeostatic properties but which also changes over time owing both to the wider environment and to internally driven change. Groups of species, often pairs, may engage in a co-evolutionary dance where evolutionary changes in each spur corresponding changes in the others. The developers in a group, the way they behave and the roles that they play are illuminated by using an ecosystem evolutionary model.

Mintzberg is keen to point out that most strategy evident in the behaviour of companies has a major element of emergence, that is the organisation adopts a strategic approach in a bottom-up, snowball fashion. The roots of this emergent rather than planned organisational behaviour lie in the rates of change driven by the environment (planning may be too slow) and in harnessing the learning potential of people at all levels in the organisation.

These concerns map directly onto the software development process where organisational change goals can never be certain and increasingly the technical systems base is beyond adequate engineering control for economic reasons. What happens in these situations is that technical development groups learn and evolve strategies for coping with the job. It is possible to list with some accuracy at the start of a project the things that are not known that will need to be known by the end of a project. The interconnectness of these not-knowns often precludes an engineering approach (examples from Fuzzy Thinking and from Design Thinking).

Why is any of this problematical? It is contentious because it indicates the limits of control. In bringing up their children parents are educated rather quickly in the limits of their control over the children's intellectual development. Most realise that hands off in a stimulating environment is a better strategy than close monitoring with lots of control actions. When children start exposing the inadequacies of their parents world view, there is a great divide between a virtuous spiral of growth and a vicious spiral of reaction and rejection.

Learning and evolution are fundamental to the software development process: the very word development indicates this. Learning and evolution can be straightjacketed and diverted and suppressed by control actions but this always harms the primary task first: development will not be successful without the ability of the development team to grow into a capability of achieving the goals. We need to stress again here that development goals are always contentious and need to evolve with the process: this degree of autonomy makes typical business management uncomfortable and suspicious. The underlying concept of development as the concrete aspects of organisational change is never really bought into.

This is the battlefield that process finds itself in. It is all too easy for management consultants to sell the idea that if developers did the job right the development would be successful. The elision comes in equating "right" with an appropriate engineering approach. This allows the primary task of authorising an appropriate arrangement between management and development team for an anxiety provoking change to be diverted into a sterile battle over the engineering approach to be used. Why do we find finance directors mandating technology for their companies on the basis of inadequate information and understanding? (Market research by Process Communications). It can only be to deflect the underlying issues of power and control that these directors do not want to face.

The second issue around evolution also falls naturally out of the model. The organisational changes and system changes that development creates are changes in the real world that was the source of the requirements for the change. (Manny Lehman) This is evolution at work: any change is the ground for and a stimulus for, further change. All systems are the coagulation of past business practice, the detritus of the business process even when new. Their significance is in the concrete platform they provide for envisioning new places to be. This view of systems as naturally evolving under their own dynamic emphasises the learning

System 30 was managing them rather than vice versa. He parted company then and we still haven't learnt the lesson of where control lies and how to gain influence over outcomes.

Engineering and control

Software technology is a difficult medium for organisations. It is sold on the basis of cutting out labour costs and increasing control over complex information issues. It brings a level of centralisation and dependency that are hard to manage. Many organisations do not survive a 24 hour crash of their central systems. Tiny errors in systems produce wildly disproportionate effects. The cost of modest changes to the functionality of a system can be arbitrarily high and even impossible to effect. A high proportion of spreadsheet programs in use contain undetected errors that make the results significantly misleading. Given these observations many organisations have software development the way individuals have a habit.

Engineering, and often very sophisticated engineering, is necessary to deal with these effects. Engineering, however, produces information which can be unwelcome and moves organisational control into the hands of the engineers. One report I submitted in an engineering company circulated happily until it reached the technical director who ordered all copies to be found and destroyed. The thesis of this paper is that process is crucial to the effective development and use of software because it situates the communication necessary for engineering to be applied successfully. It is not that engineering skills are not widely enough disseminated or that the medium is too complex to gain engineering control. It is that most organisations do not have an open enough culture to be straightforwardly experimental and enquiring about what is possible. It is that the reaction to incipient loss of control over technical systems is the imposition of management controls over people, which tends to worsen the outcomes. Management projects their impotence over technical systems onto their IT staff.

Engineering techniques are capable of producing, are designed to produce, information in the management domain. This information is about issues that are capable of having business impact, even critical business impact. I may be informing senior management that they should not launch the shuttle in cold weather. I may be saying that security on the network is dangerously lax. The information isn't guaranteed to be correct just because it has an engineering source, and its significance is a management issue not a technical one, but the exchange of information with other groups in the business and particularly with management must convey something of the content of the message. Defensive behaviour on either side or both sides means that the content of the message fails to get through, and probably that engineering effort gets put into discrediting management and management effort into discrediting engineering: an issue of dealing with external sensory reality gets turned into an issue of power and control.

It is perfectly possible to detect when group behaviour is off task, concentrated on some group fantasy such as the professional standing of the group rather than the job at hand. The usual management response to such behaviour when the group concerned is involved in software development is to increase project management controls. The behaviour of the group is framed as "ineffective" or just "poor" and requiring discipline, rather than as being involved in a defence in a wider system. Management's uncertainty and their own defensiveness cannot be handled because the problem is framed as purely a development team question. If the information that comes out of the extra project control actions is not valued and used constructively, then it is likely that the underlying group dysfunction will be given an extra twist: here is evidence of the need for defensive behaviour!

There are no quick answers to the inability on the part of individuals, groups and the the organisation as a whole to confront aspects of the reality they generate. The learning and development process is largely one of gradually gaining trust and confidence, allowing a more enquiring attitude, a growing refusal to be bound by the rules and traditions of "best practice". A narrow scientism that tries to take case studies of theory and determine what practice to apply is likely to nip such growth in the bud. Because the issues are contentious, ability to engineer requires first an ability to handle the issues emotionally.

Business Planning and Process Improvement

Process is a strategic issue. The intent in examining process is usually to try to make strategic decisions about the adequacy of a process, or the contribution that process factors have made to a project success or failure. One of the decisions that can be made is to invest in the capability of the process by education and training or by support from a methodology.

All too often planning and learning are opposite poles in practice. Planning believes that the factors that which will determine future performance are measurable, adjustable, that it is in the detailed interactions and scheduling interdependencies that success is to be found. Learning believes that many important factors are unknown or poorly understood, that the motivation to address challenges is vital, that sensitivity to the significance of unexpected outcomes outweighs constancy of approach.

Mintzberg has brought out the degree to which the nature of the planning as an activity militates against a learning approach. Planning tends to disregard individuals, to regard them as interchangeable, and it has an inbuilt centralisation tendency. Coordination is only available via control and a single plan. The combination of these biases tends to repress motivation and ownership of problems, the management system takes over from the contribution of individual aggression and group common purpose.

Experience with process improvement initiatives shows up this dilemma. Many initiatives are heavy-handed and cynical dislocations of the current process to get rid of present dysfunctions. Others are theoretical, methodology based impositions of a "right" way to do engineering. The very fact that these are planned from outside the the existing process raises questions about

compensate for a perceived lack of (growth in) process capability? Framing the question this way lets us see that it is likely that part of the capability problem lies outside the process and that by imposing change this aspect is denied. The roots of process improvement failure lie in the way the problem is framed as an engineering and skill question.

The role of consultants in organisational change is subtle. It is possible to describe a role that is a resource in the unlocking of process dysfunctions so that internal learning rates are improved. This is a matter of the system of which the process is an enactment healing itself once blocks are removed. The consultant has to have his own support system in order to proactively avoid introducing new dysfunctions and dependency around his role.

Interpretation of Research Data

We are now in a position to cast new light on some important practical findings in the software industry. Prototyping is still the most important single development in engineering practice in terms of the effect it has on success rates. By putting early systems into operation with other groups in an organisation, information is spread without it having to be fully conceptualised and without the development group having to own it. This may, and does, spark some real communication about important issues before everything is cast in stone. Basically prototyping starts the co-evolution of the business development and the system development, replacing a naturally confrontational situation where this does not occur.

The major body of empirical research around the software process is that surrounding the Capability Maturity Model. It was always made plain by the SEI that the measures in the CMM were indicators, not first order measures. The CMM doesn't say that you must have configuration management to be doing software engineering effectively, it says that if you aren't doing CM you're probably not coping with the engineering task, you're not thinking straight. The root problem is not identified by the assessment, only the symptoms indicating its existence. We would speculate here that the likely root problems are group dynamics dysfunctions: adding the missing engineering techniques is not about to improve things and may well make them worse. The CMM must be interpreted holistically and not as a process improvement checklist.

Conclusions

Process improvement, the attention to the congruence of activities in a development process to the goals of that process, can be tackled in a wide variety of ways. Because the development group is a self-regulating system in terms of its behaviour, change can be introduced in a number of ways with similar end results. The health of the process needs to be understood before change actions can be designed however. The health of a process can only be diagnosed by looking at the process in context, that is evaluating its interactions with the rest of the organisation.

Software development is a creative design task that cannot be reduced to routine work. An old-style scientific management approach to its optimisation will therefore backfire. Ownership of the process needs to stay with the process actors: there is evidence that internally driven process improvement is much more effective.

There is no such thing as an effective process in isolation from its context. Often perceived process dysfunctions have their roots outside the process itself so that process improvement cannot address them directly. Examining the health of the wider organisation, its ability to support difficult creative work of any kind, is the best place to start an evaluation.

Page	Author	Action No.	Action
1	Colin Tully	1	<p>“Establish a common process modelling capability to permit intra-project communication, subject to clearly defined needs for process modelling capabilities, which may in part arise from other actions” - Interpreted as - “consider two questions relating to action:</p> <p>a: why should FEAST undertake the task of establishing a process modelling capability?</p> <p>b: what modelling capabilities exist already, and what new ones (if any) should FEAST develop?”</p>
5	Tarik Abdel-Hamid	2	<p>“Develop the application of control theoretic and systems dynamics modelling techniques, and any other appropriate techniques, to the software process” - Interpreted as: “Application of Control Theory/System Dynamics”</p>
7	Carolyn Story	3	<p>“Examine impact of innovations on software process, both in individual organisations and more widely, paying special attention to effects of ignoring or misusing feedback mechanisms” - Interpreted as .”Evaluating past failures to have promised impact”</p>
9	Suzanne Robertson	5B	<p>“Abstract and generalise the results of action 5A, at least for one problem domain and eventually (if possible) across domains” - Interpreted as: - “Study at least one existing real-world software process, and its interactions with a real-world (E-type) problem domain), at least for one problem domain and eventually (if possible) across domains”</p>
10	Gordon Scarrott	9	<p>“Review existing doctrines.” - Interpreted as:“Review established shibboleths in system design”</p>

Dear Manny

The following is the position paper which I undertook to produce by today. Dewayne agreed to review it (and I his).

PROCESS MODELLING CAPABILITY

POSITION PAPER BY COLIN TULLY

0 INTRODUCTION

The main function of this paper is to consider two questions relating to action

- 1: why should FEAST undertake the task of establishing a process modelling capability?
- 2: what modelling capabilities exist already, and what new ones (if any) should FEAST develop?

Additionally, following Manny's comment in relation to action 1, that "many of the concepts and terms we use ... need clarification and an agreed definition", I have added the subsidiary function of offering a set of definitions to get us started. (My objection during the workshop was only to spending workshop time on definitions, not to the need for them.)

The form of the paper is accordingly that it comprises three sections:

- (1) WHY MODEL?
- (2) WHAT MODELS?
- (3) DEFINITIONS.

Each section comprises a set of main statements, numbered from 101.0, 201.0 and 301.0 respectively. Each statement may be followed by one or more notes, labelled in each case from .1 upward. The rather detailed labelling system is designed to enable comments to be precisely targeted.

The aim is to be as brief as possible, to generate discussion rather than to include it.

1 WHY MODEL?

101.0 It is necessary to distinguish between (a) a software process, (b) a software process description, (c) a software process model.

102.0 A software process is a set of human activities undertaken in the making of software.

102.1 It occurs in real time and (like all human activities) is transient. We may sometimes for emphasis call it an ACTUAL SOFTWARE PROCESS.

103.0 A software process description is information about a software process.

103.1 At one extreme (eg unrecorded human conversation) it may be very informal and as transient as the process it describes. At the other extreme it may be very formal.

103.2 It is derived from an actual process by description processes which may include observation, remembering, hearsay, imagining, fantasising. The nature of these description processes, which we usually ignore, is of considerable importance, and has a big effect on the value and validity of the resulting description: that is well understood elsewhere in science.

104.0 Software process models form a class of software process descriptions, toward the formal end of the spectrum.

104.1 All software process models are information models (including, say, mathematical and graphical models, and excluding physical models); that is to say they are descriptions.

104.2 Some people would say that all descriptions, not just the more formal ones, are models of reality. I prefer the reasonably accepted usage which would distinguish between, say, a description of an aircraft (by say a journalist or a poet) and a model of an aircraft (by say a modelmaker or a designer).

104.3 A description need only, as a minimum, follow the rules of natural language. The criteria for a model are more demanding: it should follow the rules of a modelling formalism (a subset of which may be natural language rules).

104.4 The observation in 103.2 applies with even greater force to process models: the nature of the modelling process has great influence on the value and validity of models. So does the nature of the modelling formalism.

process, from model to real world, has great influence on the quality of the actual process.

105.0 Software process models are essential (a) if we are to carry out scientific study of actual software processes, (b) if we are to manage their complexity.

105.1 Scientific study: all science is based on models of reality.

105.2 Managing complexity: the actual process which develops a software product is arguably more complex than the product itself (and we are accustomed to say that software products, as a class, are the most complex artifacts mankind has yet attempted to design). Faced with a human system of that complexity, arguably the only thing to do is to treat it as a system to be engineered. Engineering is modelling. Models allow us freedom in deploying structure to control complexity.

105.3 Models are thus essential both for academics and for practitioners. We will only get a payoff, however, if we attend to the relationship between models and reality, and to the nature of the modelling formalisms we use.

2 WHAT MODELS?

201.0 Software process models as currently proposed mostly fall into four classes: (a) macroprocess models, (b) assessment models, (c) microprocess models, (d) CSCW models.

202.0 Macroprocess models describe the top-level structure of the general software process at an inter-organisation ("universal") level..

202.1 Life cycle models belong to this class. They typically identify a small number of high-level processes, and propose very simple relationships between them. They are often in the nature of fantasies, bearing little relationship to reality. Their main role is as comforters for management.

203.0 Assessment models are also macromodels, describing the structure of the general software process at a universal level, but with the purpose of assessing actual processes in specific organisations.

203.1 They differ in two main ways from the simpler macromodels described in 202. First, they usually define many more high-level processes (often called key processes), and usually subdivide them to at least a second level. Second, since their purpose is assessment, (a) it is essential for them to have some recognisable relationship to reality, (b) they define a number of measurable attributes of key processes and subprocesses. This class includes SEI-CMM, Trillium, Bootstrap, STD etc.

204.0 Microprocess models describe the fine-grained structure of fragments of the software process, at a universal, organisational, team or personal level.

204.1 The bulk of the modelling formalisms proposed by the software process research community are designed to produce micromodels. They are typically used, and are suitable, for describing methods in great detail, probably in conjunction with tools which will interpret the model to provide human guidance.

205.0 I know little about the state of practice of CSCW models, though I guess they are - and certainly should be - used in the software process domain.

206.0 Those four main classes of software process models in current use need to be augmented by two other classes which (as far as I know) are rarely considered, and which may be called (e) business impact models and (f) project-specific models.

207.0 Business impact models describe software processes within the setting of a specific organisation, showing how they relate to other business processes and to each other, and how these relationships impact the success of the business as a whole.

207.1 These are essentially cause-and-effect models on the scale of the business as a whole. Successfully used, they would provide strong demonstrations to management of where process improvement is necessary and the benefits to be gained from it. In principle they are an application of business process modelling, although it is likely that few if any business process models yet claim to offer this level of explanatory power.

207.2 Feedback mechanisms would be essential elements in such modelling formalisms. Systems dynamics would be an obvious candidate for business impact modelling. Weinberg's cause-effect diagrams can be seen as a less formal but related approach.

208.0 Project-specific models describe projects in the large, that is to say not just project fragments and not just the aspects of whole projects customarily addressed by project management models (which are mainly orderings, dates, durations, staff allocations, costs etc).

208.1 The need is for modelling formalisms that have the power to represent the often great complexity of large software projects, that can take account of the fact that project structure develops concurrently with the development of the product design, and that can take account of the continual backtracking that occurs in real projects. The formalisms should be able to disarm complexity through

granularity through the project.

208.2 My personal view is that this is the most important class of models to have available if we are to make scientific studies of what actually happens in the real world, and if we are to engineer better actual processes.

209.0 Different classes of models correspond to different modelling purposes (though not with a one-to-one mapping). Since many purposes for software process modelling have been identified, it would be worthwhile analysing them to see how many of them lack appropriate classes of models.

DEFINITIONS

NOTE. For brevity, after the initial entry the phrase 'software process' is usually shortened to 'process'.

301.0 **Software process.** (1) A system of individual or group tasks used to develop software products. (2) The generality of processes as in (1), within an individual organisation or more widely.

301.1 Software processes are a class of business processes.

301.2 Tom's definition, as extended by Wlad in his comments circulated in Manny's e-mail of 30 June, should be added.

302.0 **Process model.** A model (representation, abstraction) of an actual or proposed process or class of processes.

303.0 **Specific process model.** A model of an individual process, actual or proposed.

304.0 **General process model.** A model of features common to a class of processes.

305.0 **Specialised process model.** A general or specific process model which has been generated by a process of specialisation from a generic process model.

306.0 **Generic process model.** A general process model from which other general or specific process models can be generated by a process of specialisation.

307.0 **Process type.** A definition of a class of processes, following some accepted type definition rules.

308.0 **Process type model.** A general process model of processes of a certain type.

309.0 **Process model type.** A definition of a class of process models.

310.0 **Software metaprocess.**

(1) A process used to develop software processes.

(2) The generality of metaprocesses as in (1), within an individual organisation or more widely.

310.1 **Software metaprocesses** are a class of business processes.

311.0 **Process** (or process modelling) **language** (or notation or representation or formalism). A language (or notation etc) used to record process models.

312.0 **Process** (or process modelling) **metalanguage.** A language used to describe process languages (or notations etc).

313.0 **Process metamodel.** A model of the concepts underlying a process (or process modelling) language (or notation etc).

314.0 **Public process model.** A process model in the public domain.

315.0 **Universal process model.** A process model of a widely used process.

316.0 **Organisational process model.** A process model of a process as used in an individual organisation.

317.0 **Team (or group) process model.** A process model of a process as used by a team (or group).

318.0 **Individual** (or personal) **process model.** A process model of a process as used by an individual.

318.1 Watts Humphrey is currently focussing his efforts on models of "the personal software process".

319.0 **Software process engineering.** The activity of explicitly designing software processes by developing process models.

whereas software process engineering is predominantly about the design process.

320.0 **Software process tool** (or toolset). A tool (or toolset) that supports the development and use of software process models.

321.0 **Method**. A general (and often public and universal) process model, with the following parameters: a theory T; a domain of usefulness D; a set of work product definitions {W}; a set of notations {N}; a programme of steps {S}; a collection of heuristics {H}.

321.1 Methods are important process resources.

322.0 **Technique**. (1) A way of doing things, independent of any method or process context, with the following parameters: a set of notations {N}; a set of operations {O}; a collection of heuristics {H}. (2) The skill of applying (1).

322.1 Techniques are the raw material of methods.

I have concentrated on preparing this paper, which is why I have not commented on the fascinating discussion that has been going on. I hope to have time to do that next week: since I shall be preparing to spend the summer at the European Software Institute in Bilbao, however (I leave on 12 July), I may not have the time to do so until after I have settled in there.

Best wishes - ColinTue Jun 28 09:39:47 1994
From: "Tarek K. Abdel-Hamid" <3991P@NAVPGS.EARN>
Subject: FEAST's objective 2 write-up
To: Manny Lehman <mml@doc.ic.ac.uk>

Manny:

I would like to thank you for inviting me to participate in the FEAST workshop. Obviously, I strongly feel that FEAST is on the right track, that this is very important work, and that it has the potential to make significant contributions to the software engineering field.

As promised, I am enclosing a write up on **objective # 2**.

Note: I will be out of town until July 18th.

OBJECTIVE 2: APPLICATION OF CONTROL THEORY / SYSTEM DYNAMICS

BACKGROUND/INTRODUCTION:

System dynamics is the application of feedback control systems principles and techniques to model, analyze, and understand the dynamic behavior of complex systems, that is, the behavioral patterns they generate over time. Its origins trace back to the pioneering work of Jay W. Forrester into industrial dynamics. The system dynamics approach views organizations, economies, societies --- in fact, all human systems --- as feedback systems. Feedback processes are, thus, seen to hold the key for structuring and clarifying relationships within such systems and, indeed, in understanding their dynamic behavior.

In the thirty years since the publication of Forrester's "Industrial Dynamics," the scope of application of the method has become extremely wide, including R&D management, urban stagnation and decay, commodity cycles, economic fluctuations, community drug policy, energy life cycles and transitions, dynamics and management of ecosystems, and software engineering.

WHY IMPORTANT:

The software development process, like most organizational systems, is characterized by a complex conglomerate of interconnected feedback loops (Abdel-Hamid and Madnick (1991)). The behavior of such systems often confounds common intuition and analysis, even though the dynamic implications of isolated loops may be reasonably obvious. The behavior that such complex feedback systems generate over time can usually be traced only by simulation.

In addition to handling dynamic complexity, computer-based system dynamics/simulation models can serve as laboratory tools for experimentation. Engineers turn to laboratory experiments to learn about the behavior of complex engineering systems. Social systems are far more complex and harder to understand than many technological systems. Why, then, not use the same approach of making computer models of social systems, such as a software development organization, and conduct laboratory experiments on these models?

The answer is often stated that our knowledge of social systems is insufficient for constructing useful models. But what justification can there be for the apparent assumption that we do not know enough to construct models but believe we do know

systems. Conversely, we do not know enough to design the most effective social systems directly without first going through a model-building experimental phase (Forrester, 1971).

WHAT TO DO --- PRE-PROPOSAL

Demonstrate the viability of the system dynamics approach by referring to a selected set of papers/findings by Abdel-Hamid/Madnick in the software project management area. (NOTE: Manny, I already sent you a list of recent publications.)

WHAT TO PLAN TO DO

Two things:

1. Develop a system dynamics model of an "interesting" dynamic phenomenon. The focus should be on a well defined dynamic problem (e.g., software reuse). Validate model in one or more industrial settings. Exercise model to generate (hopefully interesting) insights on the software engineering phenomenon studied.
2. Identify set of generic feedback archetypes that might serve as building blocks for the development of system dynamics models in software engineering domain. (NOTE: such a set has been developed for general management ... see Peter Senge's "THE FIFTH DISCIPLINE.").

References:

Abdel-Hamid and Madnick, S.E Software Project Dynamics: An Integrated Approach, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1991.

Forrester, J.W. "Counterintuitive Behavior of Social Systems." Technology Review, Vol. 73, No. 3, January, 1971. Mon Jun 27 14:54:52 1994

To: mml@doc.ic.ac.uk
From: Carolyn Story
Subject: Feast Objective 3

Manny,

Here is my contribution to Feast **Action 3** on evaluating past failures of technology to have the impact promised.

Activity:

Look at some innovation which either:

- a) some companies have used with significant positive impact on their S/W development process.

or

b) has shown in well documented realistic case studies that it has the potential for major impact; but in practice has not delivered that promised improvement in order to discover possible inhibitors to the delivery of the expected benefits.

Possible Innovation topics:

- use of declarative approaches
- use of Formal Methods

(could also take a look at a wider phenomena and examine why the move to workstations from mainframes succeeded so well, and why all inhibitors were overcome so quickly and effectively).

Criteria for choice of innovation topics:

where there are well documented successes

or

where there are well documented case studies genuinely indicating that major gains are to be expected.

Note: avoid "type" technologies where all is expected with little proof as yet of ability to deliver (e.g. Object Oriented approaches).

Reason for doing activity in 1994:

- a) as background to FEAST proposal indicating that there is significant payback in understanding (and eliminating) the inhibiting

b) to determine whether investigations into past innovation failures and successes as part of the main FEAST workplan will yield insights into the nature of these inhibitors and promoters.

How to Approach the Activity:

a) literature (and other) searches for real well documented successes

b) interviews (if possible) with the successful users, to gain their perception of why they were successful. Areas to consider include:

- success factors
- environment/context
- method of introduction
- potential (and actual) blocking factors which were avoided or overcome (and how this was done)

c) literature search (or other) for examples of failures to introduce the same technology into another (equivalent?) domain/context/environment

d) interviews with the people who did not find it possible to gain benefits from the technology

e) brief analysis trying to pinpoint the key differences and list potential reasons for the failure.

Future Work:

3-4 Case studies. Abstract fundings to identify key inhibitors and promoters.

Question:

May be useful to include other disciplines where innovation has succeeded?

Carrie Story

Date: 05 Jul 94 11:55:32 EDT

From: Suzanne Robertson <100065.2304@compuserve.com>

To: mml <mml@doc.ic.ac.uk>

Subject: Action 5B

Manny,

With apologies for the delay (I have been out of town until today). Here is my write up on Action 5B: Abstract and generalise the results of action 5A (Study at least one existing real-world software process, and its interactions with a real-world (E-type) problem domain), at least for one problem domain and eventually (if possible) across domains.

1. INTRODUCTION

At its most abstract level, the software development process can be thought of as an application system whose domain of knowledge is the system for building systems. A model of this domain of knowledge would provide input to the study of planned and unplanned feedback loops. However, due to factors like: type of industry, organisational politics and cultural norms, every software development project implements the development process in a different way and quite often, especially over a period of time, the essential purpose and the existing feedback loops are hidden by the fragmentation of the implementation procedures. Observation of a number of instances of the development process (action 5A) provides the input for building an abstract model (action 5B) that defines the essential characteristics of the process.

2. PRE-PROPOSAL ACTIVITIES

Search literature for examples of models of the software development process, especially from the point of view of feedback. Some References: Abdel-Hamid, Boehm, DeMarco, Robertson, Weinberg Use findings to illustrate the usefulness of building a model to simulate the software development feedback loops.

3. FUTURE ACTIVITIES

The overall objective of this activity is to test the ideas of FEAST by building a simulation model of part of the software development process. I say part of the process because the model needs to be extremely detailed and trying to simulate every aspect of the development process is too large a task. It is not our objective to build a brand new software development process, we are concerned with understanding how feedback works/does not work/ could work, within the context of the software development process.

(Develop the application of control theoretic and systems dynamics modeling techniques) to determine the best way to model systems development feedback loops.

3.2 Software development is becoming more of an E-type process. Some say this is due to multi-technology projects lacking the disciplined control processes necessary to monitor their interfaces. Another view is that multi-technology has made it more necessary to have a software development process which looks for unplanned feedback opportunities rather than expecting planned feedback. Investigate this.

3.3 Define the essential components of a model of the software development process. We might find that an existing model (or some combination) contains the essential components we require, if that happens then we can use it as a template for the FEAST simulation model.

3.4 Identify an area of the software development process that would provide the most advantages as a test bed for FEAST. E.g: Project Planning or Maintenance or Reuse.

3.5 Use output of action 5A plus results of 3.1, 3.2, 3.3 and 3.4 to build a working simulation model. This is an iterative task.

3.6 Find an organisation willing to do a project using the simulation model, monitor and record the results. Find consultants willing to use the model as a working tool.

This is a long term plan using a gradually growing simulation model as evidence of progress.

Suzanne Robertson

Received: via fax

Date: Wed, 29 Jun 1994 11:45:12 +0000

From: Gordon Scarrott

Review established shibboleths in system design

The system designs of the first computers were shaped mainly by the intuition of the pioneers. Since then system objectives and implementation techniques have evolved by a process of mutation and natural selection over a period of 50 years so that it is quite possible that some design objectives and techniques have survived whose validity is now questionable. Analogous situations have arisen in biological evolution. For example we have an appendix that was of value to our remote ancestors but is now unnecessary and sometimes a focus of disease.

The first computers were designed by a "bottom up" process so that all the techniques used to make it serve a useful purpose for its human users were enshrined in the software. We can now recognise that an artificial information system should be regarded as a sub system to human society which operates as a natural information system. Hence ideally the functional foundations of an artificial information system should be derived from its interface with the natural people system by a "top down" analytic design process.

People are not determinate machines so that our social information system is characterised by the interplay of recursively defined order and intrinsic disorder. Consequently the hardware functions should be designed to exploit the recursively defined order in human information and respect the disorder by providing a cost effective technique for accessing information without reference to its structure.

The design process to achieve this should therefore be a combination of top down and bottom up design which has seldom been used in commercial systems and this is the fundamental reason why the software development process for current information systems is such a shambles.

Throughout recorded history our natural information system has been shaped and operated by statesmen who have accumulated some principles of classic wisdom that might be of value in the design of an artificial sub system. For example no competent statesman believes that he can define or achieve functional perfection in his bailiwick. Consequently he endeavours by a technique of divide and rule to partition his responsibilities in such a way that if a fault arises in one section the consequences of the fault are limited to that section and do not lead to general failure. His design objective is not perfection but to control the spread of imperfection. There is no doubt that this principle can be directly applied in the design of an artificial system. Some obvious shibboleths in present system designs are:-

(1) There are two distinct kinds of information, operands and instructions, structural information must be implied by the sequence of instructions ie. in the software.

In fact there are three kinds of information - operands, instructions and structure. It is quite practicable to represent structure by appropriate symbols in the hardware and very advantageous to do so.

(2) In a multi program time sharing system, it is necessary and sufficient to ensure that a fault in one program cannot sabotage any

without adequate consideration by commercial system designers.

In fact a typical program incorporates other programs as sub contractors which need to be protected from faults in the main program. It is quite practicable to arrange such intra program protection and it then provides inter program protection for no further complication since each working program is in effect a sub contractor to the operating system program.

I was aware of these cherished fallacies before I retired in '81. Probably some more have accumulated since then that could be identified by sceptical reconsideration of current system design practise but I am no longer in touch with the tactical situation.

Page	Source
1	Aidan Ward
6	Bob Snowdon
8	Colin Tully - Workshop Decisions
14	Wlad Turski
17	Bob Rockwell
199	Berc Rustem
22	Suzanne Robertsonl
23	Dave Homan

Manny,

Thank you for facilitating my involvement in the FEAST workshop which I thoroughly enjoyed. The organisation of the workshop was excellent and the catering of a very high standard. The cross-section of the attendees was very stimulating.

I would like to contribute some initial comments of a general nature before submitting my official input to the planning process. My perception is that the workshop evaded its overt task.

I probably talked to half the attendees during breaks. At the risk of imposing upon their goodwill, I attempted in depth discussion of the philosophical questions around your proposal. It seems to me that the genie you have persuaded from its bottle is that the delivery of *E*-type systems perturbs the reality into which they are delivered in way which cannot be completely predicted or determined. I would add that current management fashion for organisational and cultural change puts high value on that sort of dislocation. The reason why this is a genie is that it disturbs the rationalist sensory reality of relying on the observed "facts". The new understanding of (for example) business priorities introduced by the view from an existing business process and its IT support cannot be limited by the goals, objectives, meanings and values associated with the current system.

During these discussions I found much openness and enthusiasm for exploring these avenues which I suspect are largely untrodden. In the workshop sessions, however, there was a businessness-like push to get definitions and agreements in place and make "progress". This of course is an example of the "disease" we had on the table before us: a concentration on the forward process rather than the feedback. My hypothesis is that feedback into the process and the meta-processes from what is after all an attempt to gain new perspectives and new realities is an interesting sort of a beast; we must study the feedback in a process which desires discontinuity and dislocation of meaning and value! I think this probably maps to your Godel-like uncertainty.

So my observation amounts to a thirst on the part of attendees for a sounder philosophical base for discussion which did not translate into the public agenda. This is not unheard of as present systems for research put pressure for maximal defence of the value of work to date on the part of researchers. And of course the research your proposal implies requires a re-evaluation of the worth of past work whether this is tackled explicitly or not. In my understanding, Wlad's views are that computer scientists should withdraw from the interactions of the system evolution process with the business process. This at least recognises the seriousness of the problem. For the record I would classify myself as someone trying to bridge the gap and spell out the real implications of IT to businesses.

On the matter of funding, I have another problem with the public stance of the meeting. Coherent Technology's recent experience is that going looking for industrial problems to study causes the candidate problems to evaporate or self-destruct. There is an extreme case here of the attempt at observation and measurement disturbing the problem in a Heisenberg-like fashion. I think the relevant methodologies for capturing some meaningful data are soft-systems approaches such as action-research. That is, the processes (and their dysfunctions/opportunities) that we wish to study are not suitable without management commitment to get to root causes of the dysfunctions/opportunities. The usual sign of such commitment is readiness to pay! My feeling is that classical funding by public bodies is a blunt instrument which often damages the research itself.

Without building it up to be something it is not, I am pursuing my own agenda in trying out ideas with current clients and with trying to put together small consortia to do specific consultancy work around the management of change/systems evolution. My gut feeling is that this is a viable route to collecting data for a project like FEAST (without in any way ruling out other activities). My need for FEAST is to have a community for the development of ideas and as a pool of expertise for meeting commercial opportunities as they come up. I realise that in saying this I am posing a serious question about the nature of academic work in this field; the question stems from an understanding of how much is at stake commercially in potential industrial study subjects, and how closely what is at stake ties into the management agenda and cultural issues.

I hope these comments are of value to you.

Many thanks, Aidan

To Aidan Ward

From: mml@doc.ic.ac.uk (Manny Lehman)

Subject: Re: FEAST workshop impressions

>Manny,

>

>Thankyou for facilitating my involvement in the FEAST workshop which I >thoroughly >enjoyed. The cross-section of the attendees was very stimulating.

Thanks and your welcome

My perception is that the workshop evaded its overt task.

I would accept that. Both Vic and I had already agreed this same point which has also been made in their own inimitable way by Wlad and Bob S. Briefly I accept full responsibility. In my own mind and that of Vic the basic and by far most important task of the workshop was to examine and develop the conjecture or genie - as you call it - to identify its critical elements, what is important, significant or of potential theoretical or practical interest and what must be studied and developed in what direction to provide a solid basis for further work and development. (wow what a sentence I shall not go back and edit it). I am very conscious of the fact that in studying these ideas I am going to need support and collaboration from some worthy individuals and organisations to exchange and polish ideas, obtain data, provide a sounding board and realistic test beds and ultimately to obtain funds to support "me" and those working with me in so far as they need it. With this in mind my plan was for the Friday morning to be spent in the sort of exploration we all would have liked and then use the afternoon for some more mundane planning to ensure 1) commitment to the project 2) a proposal or proposal that would help us get the funds which are going to be needed for the long term study which I believe is needed, possible and justified. that plan misfired in that the "objectives" which we had set up as straw men to be shot at, discarded, simplified with one or two resultant activities defined and selectively adopted, became instead an object that was extended ever wider to cover the software engineering universe - or almost thereby preparing the ground for precisely the same error of "expectation" that I was identifying with other innovators. I shall try my best to see that that error is corrected and that we will get back on track while also searching out the support that I and others (individuals and organisations) are going to need for the investigation.

>I probably talked to half the attendees during breaks. At the risk of imposing upon their goodwill, I attempted in >depth discussion of the philosophical questions around your proposal. It seems to me that the genie you have >persuaded from its bottle is that the delivery of e-type systems perturbs the >reality into which they are delivered in a >way which cannot be completely predicted or determined.

I've been saying that for at least 15 years though perhaps not always so succinctly That is, of course, only part of the story though by no means the least important. Possibly though - and this comes to me as I sit and type - one could regard ALL the loops - or many of them - as of this type as perturbing the dynamic environment in which they are embedded and in this way you may indeed have encapsulated the entire concept in your genie.

>I would add that current management fashion for organisational and cultural change puts high value on that sort of >dislocation. The reason why this is a genie is that it disturbs rationalist sensory reality of relying on the observed >"facts". The new understanding of (for example) business priorities introduced by the view from an existing business >process and its IT support cannot be limited by goals, objectives, meanings, values associated with current systems.

Yes but needs more thought, exploration and discussion.

>During these discussions I found much openness and enthusiasm for reexploring these avenues which I suspect are >largely untrodden. In the workshop sessions, however, there was a businessness-like push to get definitions and >agreements in place and make "progress".

I've already accepted indirect responsibility for that. The way to hell is paved with the best intentions. I should also have been a little more specific in my directions or suggestions to Val.

>This of course is an example of the "disease" we had on the table before us: a concentration on the forward process >rather than the feedback. My hypothesis that feedback into the process and the meta-processes from what is after all >an attempt to gain new perspectives and new realities is an interesting sort of a beast; we must study the feedback in >a process which desires discontinuity and dislocation of meaning and value! I think this probably maps to your >Godel-like uncertainty.

>So my observation amounts to a thirst on the part of attendees including, may be most of all, me for a sounder >philosophical base for discussion which did not translate into the public agenda. This is not unheard of as present >systems for research put pressure for maximal defence of the value of work to date on the part of researchers. And of >course the research your proposal implies requires a re-evaluation of the worth of past work whether this is tackled >explicitly or not. In my understanding, Wlad's views are that computer scientists should withdraw from the >interactions of the system evolution process with the business process.

No I think you are wrong there. He was amongst the first - in a 1980 paper - to recognise this phenomenon. Only he talks in his own terms and in particular made the assertion that theories to completely describe business processes, all processes involving people - cannot be formulated hence etc.

>This at least recognises the seriousness of the problem. For the record I >would classify myself >as someone trying >to bridge the gap and spell out the >real implications of IT to businesses.

>On the matter of funding, I have another problem with the public stance of the meeting. Coherent Technology's >recent experience is that going looking for industrial problems to study causes the candidate problems to evaporate or >self-destruct.

True absolutely so, but my dilemma was and remains what to do about it. I need funding for myself ("to make a living"), I need people I can talk to, collaborators and they need to make a living or to justify the time to their employers and employers need revenue or at least cover for resources employed. Ideally we should interest a few companies in the concept and get their support for the sake of *long term* support. But can we and how to set about it.

>There is an extreme case here of the attempt at observation and measurement disturbing the problem in a Heisenberg->like fashion. I think the relevant methodologies for capturing some meaningful data are soft-systems approaches >such as action-research. That is, the processes (and their dysfunctions/opportunities) that we wish to study are not >suitable without management commitment to get to root causes of the dysfunctions/opportunities. The usual sign of >such commitment is readiness to pay! My feeling is that classical funding by public bodies is a blunt instrument >which often damages the research itself.

I could not agree more and would love to take the non-public funding route. Is it feasible and how do we start?

>Without building it up to be something it is not, I am pursuing my own agenda in trying out ideas with current >clients and with trying to put together small consortia to do specific consultancy work around the management of >change/systems evolution. My gut feeling is that this is a viable route to collecting data for a project like FEAST >(without in any way ruling out other activities).

If you see any way of involving me in that (does CT make use of "independent consultants?") I would be delighted to join you in appropriate circumstances/studies.

>My need for FEAST is to have a community for the development of ideas and as a pool of expertise for meeting >commercial opportunities as they come up. I realise that in saying this I am posing a serious question about the >nature of academic work in this field; the question stems from an understanding of how much is at stake >commercially in potential industrial study subjects, and how closely what is at stake ties into the management >agenda and cultural issues.

I'm not even so certain that it represents a stab at academic research. I would prefer to think of it as the providing the sort of real - world laboratories that the studies we have in mind demand

>I hope these comments are of value to you.

>They most certainly are.

Manny

From: Aidan Ward
Subject: FEAST tasks
Mon, 11 Jul 94 09:01:44

Manny,

Thankyou for your comments on my initial reactions and also for similar comments from others. I include below some rushed notes nominally to address action 9 about doctrines but actually to try to show the direction from which I would address the human factors aspects of FEAST.

I am progressing both brainstorming activities about subjects close to FEAST and commercial projects which might present opportunities to you. I am happy, time permitting, to try to coordinate these with FEAST activities if that is useful to you. Let me know how to progress this.

Introduction

This is a preliminary position on action 9, to review existing dogma/doctrines. As indicated in my previous comments, I think that the stance of FEAST is corrosive of traditional software engineering arrangements: this note explores some of the doctrines that are undermined by the FEAST core proposition.

A lot of "common sense" engineering practice is brought into question by the proposition that the systems it plays a role in are stable in some homeostatic sense, so that the evolution of software systems follows "laws" which are true despite attempts to overcome their implications by engineering effort. That being the case FEAST can either:

\begin{itemize}

\item Revisit the role of software engineering as a business activity on the grounds that the evolution of software systems is far from understood by their consumers/purchasers. "Understand what the job is"

I propose that the first choice is reductionist and against the spirit of FEAST although it is far more manageable in a narrow sense. The implications of the second choice are uncomfortable and are explored further here.

\section{The relation of business to IT}

>From a business perspective, most businesses are highly constrained by their information systems. They may see this as a matter of the cost (both financial and political) of getting the right system. They may see it as a power struggle over access to information. They may perceive that any system is about yesterday's way of doing business, and that keeping systems current is difficult.

Much of the emphasis in business management is now on ability to change, on responsiveness and on potency. Whatever the perceived root cause, existing systems are almost always in the way of business change. All too often planned new systems become the repository for hopes of future business change, especially the avoidance of today's unmanageable problems. This conflict between past experience and optimism about the future seems to be maturing into a cynical attitude towards IT.

The increasing use of user programming and user configuration of systems raises other issues. It moves the issue of evolution of systems into the business domain rather than the engineering domain while at the same time increasing the pressure for s-type specification of the system components.

The implications for FEAST are that the notion of system is changing and the notion of the role of engineers in creating systems is changing. My guesses are that:

\begin{itemize}

\item The system of interest when looking at evolution dynamics is always socio-technical, always distributed and only partial information is ever available.

\item There are multiple realities involved with any system: there is no consistency available in descriptions of what it does, what it means, why it is significant.

\item Many systems issues are inevitably the locus of power struggles in the organisation. Managed conflict is an important source of new ideas as well as helping to define current realities.

\item Nobody can track the movement of realities and understandings well enough even to learn the important lessons. Systems are archaeological artefacts in helping to track these issues.

\item Any attempt to closet the development of systems from the chaos of the rest of the organisation sets in train a divergence of values which must result in conflict.

\end{itemize}

\section{A reframing}

In the important dimensions there is no separation between information system and organisation. It is a reductionist mistake to try and manage them separately. What we have is a situation where the growth in understanding on the part of the business about how to go about its business must be mirrored in a very intimate way in the information systems on which it depends. In mature and simple domains there are sometimes abstractions which become commonplace (one could argue the case for spreadsheets) but in general there is little understanding about how to separate out the stable concepts from the ephemera.

The business value in the information systems lies in its ability to track the realities the business is involved in. This is a positive statement of the Lehman hypothesis: not only is evolution forced on us rather against our engineering judgement, but in terms of the full lifecycle return on investment it is the overwhelming factor. What we need to be able to do is understand and influence the drivers of evolution so that this return can be maximised.

Not only is this about feedback and stability, it is about effective functional communication. The development process is part of an organisational change process. Many adjustments will need to be made in areas of the organisation not obviously affected by the systems under development.

To treat the development as isolated from the organisation is a reductionist mistake. It insists that there is separation where in fact there are many poorly understood connections. In practice, successful developments stay close to their constituencies and maintain open communications.

Models of communication emphasise that:

- \item The content of communications undergoes an extensive filtering and interpretation process on the input side (Saini model).
- \item Far more and more reliable information is conveyed by non-verbal than by verbal communication.
- \item There is rarely a single recipient for a message so that decoding the intentions behind a message is defeatingly complex in real cases.
- \item The assumptions behind even the most formal communication are too numerous ever to be documented.

These communications {\bf are} the feedback paths, and no matter what the content of the messages, keeping them open is known to be non-trivial. In particular, a major block to communication in the normal case is the overvaluing of the technical system, its conception as a solution {\bf by definition} not by evaluation.

\section{A suggestion}

The reframing of the problems (as suggested above) would have considerable impact on a real development project. In fact the very impact would make the reframing difficult if not impossible to achieve: it changes the power relations.

The usual systemic approach in this situations is to evaluate the relations involved by using observations of the way the observer is caught up in the situation. Basically the assumption I am making is that system development is caught up in a dysfunctional system in the normal case, and that facilitating functional communication in key places will on the technical plane open the feedback paths and on the organisational plane allow the dysfunction to heal itself.

Intervention cannot tackle the problem head-on because that is to take a power position which is not available and also to assume that a dose of top-down design is what is required. The more systemic approach is to allow the control/management system to learn to use the information feedback suggests.

Cheers, Aidan

From: ras@kid01pml.icl.co.uk (Bob Snowdon)
 Subject: Thoughts from last week.

A few thoughts following last week's FEAST workshop.

It seems to me that what is needed is some devoted intellectual attention to the nature of the problem space in which E-type systems occur. The traditional S-type problem is one which can be neatly encapsulated and for which a computer program can be devised as a "solution". Software development generally has been "forced" into this model and thus created many of our present problems. The E-type problem space is different and evidence suggests that we should perhaps not expect current practices to be very satisfactory. (This is what I was trying to suggest in my talk by characterising S-systems by the phrase "computer as calculator" and E-systems by "computer system as partner").

I guess I was a little disappointed by the direction the workshop seemed to take on the second day, when, after Wlad's sound advice, the major outcome was a "plan" to develop a proposal for some project in and around the FEAST space to be submitted (in all likelihood) to the CEU Framework 4 programme. This might be OK, but somehow the sort of things being discussed seemed rather a long way from the original, obviously not completely thought out, ideas. I certainly feel unsure at this stage that such a project (or even proposal) would be likely to produce the sort of results or emphases I think you set out to address. I certainly have considerable unease about devoting (too) much attention to the software development process or to software process engineering before having a firm grip on the nature of the problem. I think that the relationships amongst all the various systems involved are an essential feature of E-type situations and that understanding these relationships is therefore rather important (hence the value of studying examples of the genre rather than simply studying the software development element).

Am I over reacting? Maybe. However I feel that during the coming months it would be extremely valuable to devote some time to the heart of the problem which, as I have said, I believe lies in understanding the various inter-system relationships inherent in E-type situations. This can be done "in theory" and in pilot case studies of existing situations, with the potential for more significant projects and spin offs later. One way could, of course, be to submit a proposal under Framework 4, though there would be others. However I am suspicious of "tails wagging dogs"!

Anyway, thank you for arranging things for the meeting last week and for inviting an interesting group of people. It is a long time since I last met Gordon Scarrott for example and I enjoyed listening to his thoughts again. Let's now look forward to some fruitful work in what I think is a very interesting area.

Bob.To: ras@kid01pml.icl.co.uk (Bob Snowdon)

Dear Bob

Thanks for the quick response. Herewith my initial reaction and right away the comment that I share many of your reactions and concerns. Many things could evolve from our two day effort and I believe that I was preoccupied or over concerned with arousing interest, obtaining commitment and, believing that neither Industry nor Academia is prepared (or able) to do very much without funding, ensuring that we developed a plan which would ensure adequate funding for all those wishing to collaborate in the investigation (still to be clarified and defined - unfortunately we did not do this on Friday, quite the opposite).

>It seems to me that what is needed is some devoted intellectual attention to the nature of the problem space in which >E-type systems occur. The traditional S-type problem is one which can be neatly encapsulated and for which a >computer program can be devised as a "solution". Software development >generally has been "forced" into this model >and thus created many of >our >present problems. The E-type problem space is different and evidence suggests that >we should perhaps not expect current practices to be very satisfactory. (This is what I was trying to suggest in my >talk by characterising S-systems by the phrase "computer as calculator" and E-systems by "computer system as >partner").

>I guess I was a little disappointed by the direction the workshop seemed to take on the second >day, Frankly so was I >as already indicated above when, after Wlad's sound advice, the major outcome was a "plan" to develop a proposal for >some project in and around the FEAST space to be submitted (in all likelihood) to the CEU Framework 4 >programme. This might be OK, but somehow the sort of things being discussed seemed rather a long way from the >original, obviously not completely thought out, ideas.

Exactly. I am very conscious of only having some preliminary thoughts, insights or what you will, for the need for focussed but wide ranging - if this is not a contradiction - discussion with like minded people to clarify, refine and solidify what it is I am trying to observe. I hope we can get it back onto track in one way or another, that we can get the right people together, individually or as a group (Vic, Jose and I will be meeting next week).

>I certainly feel unsure at this stage that such a project (or even proposal) >would be likely to >produce the sort of results or emphases I think you set >out to address.

I suspect it might not unless we take some firm action - though I am not necessarily the best person to steer the ship back on course (pardon the mixed metaphor) considerable unease about devoting (too) much attention to the software >development process >or to software process engineering before having a firm >grip on the nature of the problem.

Yes, you are 100% correct. The trouble is that in the materialistic world of today one has to demonstrate a high probability of "real" impact on industry (Ha Ha - since, if I am correct, the ultimate impact of a slower and more carefully directed start concentrating on concepts, phenomena, interpretation theory generation and so on would have far greater impact). Hence my formulation of the "objectives" which with the (incorrectly?) perceived need for funding (I certainly need it for myself after next March) led to the direction that Friday took. - It was not as I had intended it that part was for the afternoon only. I gave insufficient direction to Val, Wlad gave a limited gut reaction to one or two points and didn't lay the foundations or trigger a fundamental discussion

>think that the relationships amongst all the various systems involved are an essential feature >of E-type situations and that understanding these relationships is therefore rather important >(hence the value of studying examples of the genre rather than simply studying the software >development element).

Of course. But to do that requires a "real" guinea pig - on second thoughts one could create a hypothetical organisation. Maybe one needs to do both. This is precisely the sort of issue Vic and I hoped to discuss when we set up our 8 objectives as straw men which we wanted thrown out, cut down to size and trimmed to provide the framework for an initial study. But then, how, without funds, do you get people to join you and, in this day and age, how do you get funding bodies to support you if you don't promise the dawn of the Golden age, the solution to their (or industry's) pet problem. Or have I got it all wrong.

>Am I over reacting? Maybe. However I feel that during the coming months it >would be >extremely valuable to devote time to the heart of the problem >which, as I have said, I believe >lies in understanding the various >inter-system relationships inherent in E-type situations..

Agreed yet once again, by far the most important element of the foundations for a thorough investigation of the whole issue and the development of the much desired theory framework I and others am looking for. But we need some more thinking and discussion to really identify the things to be done and the order in which they should be done.

>This can be done "in theory" and in pilot case studies of existing situations, >with the potential >for more significant projects and spin offs later. Yes but while one person (me for example??) might kick this off he (or she) would need to be triggered and guided both to kick off and during the development.

suspicious of tails wagging dogs !

Not quite sure of which tail and which dog. But clearly we have to be on our guard. But frankly would you/ICL be prepared to get involved actively on a continuing basis - at least for say one year to start with - a) if it just involved time b) +some money c) if there was promise of a funded project (or at least an application) at the end of the road?

>Let's now look forward to some fruitful work in what I think is a very >interesting area.

Thanks. I take that as your personal committment to remain involved, even to intensify that involvement. That, by itself would have made the workshop worthwhile.

Manny

I shall send this without reading it through again so you may well find some less than carefully thought out remarks. But at least its entirely spontaneous.

From: C.Tully@herts.ac.uk
Subject: workshop report

SUMMARY OF DECISIONS AT FEAST WORKSHOP
FRIDAY 17 JUNE 1994

Decisions, below, are labelled D01, D02 ...

The group in most cases was not concerned with the precise wording of decisions, which therefore is my responsibility. My apologies where people are unhappy with the outcome.

** D01 Definitions **

Workshop time should not be spent in search of an agreed definition of process.

Tom Maibaum offered a simple diagram, in which a process has input and output, resources and waste. This was accepted as showing a necessary characteristic of all processes.

It was also accepted (a) that processes are multilayered in the sense shown by Marc Kellner, (b) that any process may be part of a larger process and decomposable into subprocesses.

Except where explicitly stated otherwise, the strict (control engineering) definition of feedback, in terms of a controller sampling process outputs and using them to adjust process behaviour, should be assumed.

** D02 Assertion 1 revised **

The software process for E-type systems, which includes both development and maintenance, constitutes a complex system, for the understanding of which the concepts of feedback, evolution and learning are essential.

** D03 Assertion 2 revised **

Lack of attention to the global software process, and to the role of feedback in it, explains at least in part the limited impact of even successful innovations.

** D04 Objectives **

Not all the items listed as objectives are strictly objectives. All of them, however, are proposed actions. They are therefore renamed as actions.

The elements of the set were also changed. Action 4 was deleted after being merged with action 3. Action 5 was split into actions 5A and 5B. An action 9 was added. Revisions to the wording of the nine actions are shown as D05 to D13 below.

** D05 Action 1 **

Establish a common process modelling capability to permit intra-project communication, subject to clearly defined needs for process modelling capabilities, which may in part arise from other actions.

** D06 Action 2 **

Develop the application of control theoretic and systems dynamics modelling techniques, and any other appropriate techniques, to the software process

Examine the impact of innovations on the software process, both in individual organisations and more widely, paying special attention to the effects of ignoring or misusing feedback mechanisms.

** D08 Action 5A **

Study at least one existing real-world software process, and its interactions with a real-world (E-type) problem domain.

** D09 Action 5B **

Abstract and generalise the results of action 5A, at least for one problem domain and eventually (if possible) across domains.

** D10 Action 6 **

Investigate the use of metrics, and their effects (beneficial and/or harmful) on understanding and improving the software process.

** D11 Action 7 **

Search for global process improvement opportunities and ways of evaluating them.

** D12 Action 8 **

Manage the project, including the preparation of funding proposals.

** D13 Action 9 **

Review existing doctrines.

** D14 Priority evaluation **

Three rounds of voting took place on the nine actions, as follows.

Vote 1: is the action important?

Vote 2: are you prepared to work on the action?

Vote 3: should work be done in 1994?

Each participant had up to nine votes per round, of which not more than one vote could be given for each action. Further, volunteers were sought to write an initial position paper (maximum three pages) on each action. The results were as follows.

	Vote 1	Vote 2	Vote 3	Author
Action 1	12	9	9	Tully (plus Perry)
Action 2	14	5	11	Abdel-Hamid
Action 3	6	5	4	Storey
Action 5A	21	13	22	Moularde
Action 5B	17	11	0	Robertson
Action 6	7	4	5	Perry (plus Tully)
Action 7	16	7	0	Kellner
Action 8	all	5.5	all	Downes
Action 9	4	3	2	Scarrott

** D15 Target dates **

The nine position papers should be produced by Friday 1 July. All participants are invited to submit comments on any actions (other than the actions on which they may be writing position papers) by the same date. Position papers and comments should be sent by e-mail to Manny. They will then be circulated to everyone (form to be determined).

A further workshop will take place at Imperial in late October.

From: C.Tully
Subject: comments

Dear Manny

Here are the comments I threatened on the correspondence to date.

They fall into four sections. The first relates to the plethora of gloomy breast-beating about supposed deficiencies of the workshop. The second relates to Manny's comments on the minutes. The third relates to a variety of other comments. The fourth relates to proposed dates.

The following are the main instances of self-criticism

>My perception is that the workshop evaded its overt task. [Ward]

> n the workshop sessions, however, there was a business-like push to get definitions and agreements in place and >make "progress". This of course is an example of the "disease" we had on the table before us: a concentration on the >forward process rather than the feedback. [Ward]

>So my observation amounts to a thirst on the part of attendees for a sounder philosophical base for discussion which >did not translate into the public agenda. [Ward]

>I was, however, disappointed with the amount of time (admittedly difficult in view of the limited duration) spent on >examining the fundamental issues and challenges that FEAST represents. [Lehman 28/06]

>I came away from the workshop with the feeling that we had not done justice to laying solid technical foundations for >the proposed investigation. [Lehman 28/06]

>As commented yesterday, in common with Aidan and Bob, I believe that, in a sense, we evaded our task to some >extent by concentrating on Objectives and Actions rather than on the fundamental issue(s). [Lehman 29/06]

>I have already addressed our common concern as to the direction of the discussion on the second day. This was shared >by Aidan and I have also now received important comments from Berc Rustom who expresses related concerns. I >certainly would like to see us get back to main concentration on the original >observation/hypothesis/conjecture/assertions etc. And to achieve real progress in this (and in, in my view, in software >engineering in general) requires us to "have a firm grip on the nature of the problem". [Lehman 30/06 11:28]

> Hence the 8 objectives (now actions) intended as strawmen to be shot down or modified by workshop participants to >leave an attainable target for the coming months. The end nature and size of the selected actions would have to depend >on the commitments obtained to participate. Instead we adopted all 8 plus one additional one. [Lehman 28/06]

I should like to comment on the whole bundle as follows.

(1) I think we should not be so hard on ourselves (or, by implication, on Val as the chairman of the second day). The design of the workshop allocated 6 hours and 5 minutes to presentations and immediately relevant questions/discussion, and 4 hours 45 minutes (less than 45%) to "real" discussion sessions. It is unrealistic (dare I say academically romantic?) to imagine that, with 33 people, 4.75 hours and a subject which everyone agreed is ill-understood and ill-defined, one could make much progress on "fundamental issues", "a sounder philosophical base", "solid technical foundations", "a firm grip on the nature of the problem" and other such good things.

(2) In any case it is mistaken to imply that the time spent on objectives/actions was all of a project planning nature. I interpreted it as a first step in trying to clear some of the undergrowth that we can expect to impede our subsequent progress, and to start exposing some of the variations in viewpoint which exist (necessarily and desirably) among us.

(3) In any case (again!) I don't think anyone would seriously propose that we do not have to devote a considerable proportion of effort up-front to deciding what we should actually do and how that relates to funding opportunities. Otherwise there is no way we can in future indulge the desire most of us no doubt have (certainly I do) for the academic luxury of philosophising and abstract speculation.

(4) With particular reference to Aidan's nice self-referential observation about "concentration on the forward process rather than the feedback": I suspect it is true that you can exercise feedback until at least some forward process has occurred. I think some other people (reading between their lines) felt in contrast that there was not enough forward process. As I have suggested in (1) and (2) above, there was a good deal of forward process given the context of the meeting and the topic. We must understand the complexity and the (human) system dynamics of the process on which we are embarking.

(5) Manny is worried that we did not kill more objectives/actions. We did, however, make a preliminary expression of priorities. Given the constraints to which I have been drawing attention, and given that the objectives (coming from Manny and Vic!) were all good prima facie, we would have been rash to trash any quite so quickly.

(6) Finally, a comment for future workshops. Please don't underestimate the problems in getting real results out of working groups of this size. It demands (at least) setting very clear (and attainable) objectives, preparing material well in advance, very careful design of the meeting itself, and outstanding chairmanship. I should be sad if inattention to those prerequisites led to even greater breast-beating second time round.

Section 2: comments on the minutes

From now on, I will comment on each point in turn.

I cannot "agree" with that! I was punctilious in trying to check agreement on every point I recorded. I hope what Manny means (and I would "agree") is that they should only stand as interim points of agreement, which undoubtedly will be changed as we do further work. Manny indeed has quite properly already started that process.

>Clearly one cannot go into detail in a summary but I believe that the present wording is misleading and could be >improved.
[Lehman 28/06: D01 para 3]

My comment above applies. In any case, I personally think that the wording is only misleading, if at all, through being insufficient rather than wrong.

>I do not believe that this was agreed nor did we discuss the definition of feedback in any great detail. The present >wording reflects the view expressed by Colin at the meeting but did not receive adequate discussion or "agreement". I >believe it is inadequate for a FEAST definition and will require further consideration, probably after more work has >been done to agree what FEAST is all about, what ground it is to encompass and what issues can and should be >explored. [Lehman 28/06: D01 para 4]

Before recording the decision, I went out of my way to ask the specific question, "Do we want to use feedback in FEAST with the loose meaning it has in common usage, or with the technical meaning it has in control engineering?" No one urged the former; those who said anything urged the latter. If there is any problem it is in the interpretation of what feedback means to control engineers, and then how (if at all) that may need to be adapted to our rather wider purposes. There is of course need for a lot of discussion on this --- it is central to FEAST.

> Agree that we agreed to change term "objectives" to actions. However I stress again that not all these (previously) >objectives were visualised or intended for immediate action. That depends on many factors (some >discussed above) >and on feasibility and worthwhileness. At best Colin's wording should have been "actions to be >explored for >feasibility and reasonableness" (clumsy but will do for moment - I want to get this out). [Lehman 28/06: D04]

I hope Manny will accept my phrase "proposed actions" as shorthand for "actions to be explored for feasibility and reasonableness"!

>Problem with current wording that we need to start on this immediately so that modification by perceived need calls >partly for some sort of (brief) preliminary discussion and then for "adaptation" as additional needs are identified or rear >their head. Thus instead of "subject to etc. I would say "based on initially identified needs and reviewed and updated as >additional needs arise from or are identified by other actions.

>In addition, though this was not discussed as part of this action at the workshop, as Tom pointed out many of the >concepts and terms we use (seeD01) need clarification and an agreed definition for common usage. It seems to me that >such definition is, in any case, a necessary precursor to "establishing a common process modelling facility" and >therefore fits naturally into this action.
[Lehman 28/06: D05]

I wholly agree with both these paragraphs.

> Study domain identifying, in particular feedback paths and their characteristics. [Lehman 28/06: D08]

It is critically important to treat feedback as a complete systemic mechanism, not just to focus on the paths that carry measurements and control signals.

> remainder OK except that Storey should be Story [Lehman 28/06]

and Collin should be Colin!!! I had Story in my database, and only changed it to conform to the entry in the attendance list. Sorry Carrie --- or is it Carry??!!

Section 3: other comments

> It seems to me that the genie you have persuaded from its bottle is that the delivery of e-type systems perturbs the >reality into which they are delivered in way which cannot be completely predicted or determined. [Ward]

The word "completely" is absolutely critical. It is correct, BUT we should not take it to mean that partial prediction is impossible, should not be attempted, and cannot be greatly improved. I take it as axiomatic that a major task for systems engineering is to investigate predictable and possible effects of the system being engineered on the contextual systems in which it will be used.

>New understanding of, for example, business priorities introduced by the view from an existing business process and >its IT support cannot be limited by the goals, objectives, meanings and values associated with current system. [Ward]
"Cannot be limited by": again, correct BUT we should not jump to the conclusion that we should ignore current systems. They are an important component of what we have to analyse in setting requirements.

process with the business process. [Ward]

Have computer scientists ever got anywhere near such interactions (let alone withdrawing from them)? Such interactions are instances of the kinds of "soft" real-world problems that the majority of computer scientists have always been proud to scorn. Thank heavens, mind you, they have not tried to tackle them, because the discipline is wholly inadequate to enable them to do so. What they are wrong to do is to scorn such problems, and to emasculate the emerging software engineering discipline which attempts to handle real-world issues.

>I think the relevant methodologies for capturing some meaningful data are soft-systems approaches such as action->research. [Ward]

Yes yes yes! Here it is the phrase "such as" that is the critical one. Many research methods may apply. That was the point of my intervention on day one about the nature of history, science and engineering (and Aidan added philosophy).

> It seems to me that what is needed is some devoted intellectual attention to the nature of the problem space in which >E-type systems occur. The traditional S-type problem is one which can be neatly encapsulated and for which a >computer program can be devised as a "solution". Software development generally has been "forced" into this model >and thus created many of our present problems. [Snowdon]

Yes, with a caveat. The software process is a human process (with support from e-type and s-type systems). This type of system is even more "wicked" than an e-type system: let us call it an h-type system. The point now is that h-type systems are necessary for the development of s-type systems --- not just of e-type systems. So let's not suppose that we can drop s-type systems wholly from our consideration.

> I certainly have considerable unease about devoting (too) much attention to the software development process or to >software process engineering before having a firm grip on the nature of the problem. [Snowdon]

While these things are certainly not the whole of the problem, they surely remain an intrinsic part of it. We cannot hope to get our firm grip BEFORE thinking about them. I guess Bob means we must be sure to think about them in their wider contexts.

> All this is reasonably well-understood and investigated, although it may be worthwhile to provide a systematic, >succinct description (e.g. emphasizing that "backtracking" is a feedback loop!). [Turski]

I am not at all certain that this is right. I regard backtracking as retracing a backward path in the logical space of a project to discover the point(s) of impact of a currently discovered defect or change of mind. This is followed by reactivating some of the (suspended) intervening processes to change their existing deliverables. This doesn't fit my concept of feedback. But it is something that needs careful thought.

> But this is just iteration - always a feedback phenomenon! [Turski]

Again (I'm getting fearful!) I don't agree. Iteration is a repetition phenomenon, using an equality test for completion. Feedback is a corrective phenomenon, using measurements of deviation to control the strength of corrective action. Not all loops are feedback loops.

> I am, for example, worried by the term "controller". In what way is a user discovering and reporting an error, a >controller? Similarly the term "process behaviour". Why view the rework necessitated by an error report as "adjusting >process behavior"? [Lehman 28/06: D01]

The user herself is not a controller. I would say that the process in which a user detects and reports a deviation from expectation is a control process. You could of course reasonably object that it was not set up, or engineered, as a control process: the user does not think of herself as executing a control process, like a tester would. The discovery of the defect, in other words, is accidental rather than essential. But we rely on such detection processes, and it is crucial to have a connection to which the user can make her deviation report, and the control mechanisms for subsequently handling it. So I would provisionally call it a control process, even though I would recognise that we need to classify control processes: they are not all the same.

> One may, or should perhaps, regard many or even all of the loops as perturbing the dynamic environment in which >they are embedded. [Lehman 29/06]

The loops do not perturb the environment: they are just information paths, which are essentially pieces of static structure. What may perturb the environment is the changed behaviour resulting from the existence and use of the feedback mechanism as a whole.

> Let me add that I, for one, am firmly convinced that a technology that does not have a firm philosophical and >scientific base and framework is limited in its growth capability and I regard potential FEAST contributions in this >area as the most significant. [Lehman 29/06]

Just in case any earlier comments may be misunderstood, I am in total agreement with Manny's position here.

and backward (in process not time of course) - is crucial to making progress in controlling and directing such information flow and hence in controlling and directing the process, process improvement and the >process improvement process. [Lehman 30/06 11:28]

This is I think a very important observation. It directs attention to all the information flows, not just the information flows occurring on feedback loops (even though we are right to focus a lot of attention on them).

Section 4: dates

> As date I suggest Tuesday 25 - Thur 27th October with a Mon 24 or Wed 26th starts as alternatives. My preference is for the 25 - 27 with 24 - 26 second best. Tues - Thur appears to minimise costs for overseas participants. Please let me have your reaction to these dates as soon as possible. [Lehman 28/06]
All of these dates are so far OK.

> Dewayne will be spending a week here from the 18th - 22nd of July and again from the 24th - 30 August. Anyone who wishes and is able to join us during that time for a longer or shorter period will be most welcome and appreciated. [Lehman 30/06 11:28]

I am currently available on 28 and 29 August and would love to join in.

Colin

From: "Wladyslaw M. Turski" <wmt@mimuw.edu.pl>
Subject: Feedback on feedback

Dear Friends,

I've been thinking about the blasted feedback, trying to get some simplifying view. The results of this thinking process follow. Do what you like with them (e.g. trash them!). I'll be away from my office rest of this week (another conference, in Poznan), then for 10 days at the desk, then - until the end of July - on holidays.

Cheers, Wlad

First, consider Elementary SW Process (ESWP):

Takes Input: Need (for a computer application)
Uses Resources: Method (or a work procedure)
 Manpower*Time
 Tools (e.g. SADT, Ada compiler, IPSE)
Produces Output: Code |
 Documentation | Collectively known as
 User training | Product
 |

Within a class of applications the ESWP is "reusable as is": given an Input it produces an Output consuming some Resources (Manpower*Time is expanded, while Method and Tools are worn out, i.e. become progressively more obsolete each time the process is run).

In an ESWP there usually are several built-in feedback loops (internal to the ESWP): e.g. the Method (think of LST) may be iterative (backtracking), the application of a compiler may result in a list of syntactic errors, thus forcing a modification to the program text (previously constructed). Also internal to the ESWP are feedback loops resulting from walkthroughs, code inspection etc. as long as they cause backtracking before Product delivery.

All this is reasonably well-understood and investigated, although it may be worthwhile to provide a systematic, succinct description (e.g. emphasizing that "backtracking" is a feedback loop!).

There are two ways in which an ESWP may be extended. For simplicity I shall treat them as orthogonal:

- horizontal extension (HSWP)
- vertical extension (VSWP)

HSWP obtains when the Product of a given ESWP is made into (a part of) its Input, i.e. the need is expressed relative to an application and an existing product. Thus we have a horizontal external feedback loop (output loops into input).

Assuming that the Method and Tools can cope with such a mixed Input (and it is not at all clear that this is the case!) we have the first opportunity for a feedback control: the amount of resources (Manpower*Time) consumed will vary depending on the "size" of

exactly what Lenman/Belady studies of OS360 have shown, except they used Manpower and Time components separately.

Here the scope for research is unlimited - but I am sceptical about the possibility to make all necessary ingredients measurable over a range of products. The best one can hope is to get data on a given product over its consecutive releases or versions (again, as for OS360). Note however that contemporary products do not have so many releases.

VSWP obtains when we consider a process of "improving" an existing ESWP. Now the Input is the old ESWP, the Output a new ESWP, Resources are Manpower*Time, Tools (I doubt the current availability of suitable tools of this category!), Method (Procedure for improving ESWP).

Examples of (individual runs of) VSWP are provided by history (your "forward path innovations"): an ESWP with the Assembler coding for a tool supplied as Input to (a specialized) VSWP is transformed into an ESWP with Ada programming as a corresponding tool (I am referring here to a tool inside ESWP!). This new ESWP is the Output of the VSWP. Note that both the old and new ESWPs can be used to make products (when run on their common Input: a need each will produce a Product). The difference is not necessarily in products of these two ESWPs, but - presumably - in that the new ESWP is "better" than the old one.

How can one measure the goodness of an ESWP is a nice problem; one easy assumption is to consider the amount of Manpower*Time resource used by an ESWP (on a given input) to make a satisfactory product. The ESWP which uses less Manpower*Time for the same input (need) is better (as long as its product is OK).

Where is the feedback loop in VSWP? Of course it obtains when we take its output as input. But this is just iteration - always a feedback phenomenon! To get a feedback control in thus looped VSWP we need to extend its Input by a "need" for ESWP and agree on a measure of satisfaction (wrt this need) provided by an ESWP.

In reality we are probably facing the problem both horizontal and vertical extension of ESWP at the same time, which makes the research combinatorially more complex.

Yet, my recommendation would be to start with a more detailed study of the constituents of the model: ESWP and its internal feedback loops and controls (if any), HSWP and its internal feedback loops and controls (here they sure exist!), VSWP and its internal feedback loops and controls (very dubious). As I can see the field today, it is hopelessly confused with internal loops of ESWP held "responsible" for both horizontal and vertical evolution (i.e. for sequences of horizontal and vertical products of entirely different processes).

From: mml@doc.ic.ac.uk (Manny Lehman)
Subject: Wlad's comments
30/6/94, 2.35

Wlad

§1 - 4. Minor typo "expanded" should presumably be "expended" (though one of the classical software process problems is that so very often the manpower time product is, indeed, expanded.

Yes the feedback loops you mention are well recognised though I am not confident that their properties and effects have been adequately studied, modelled or quantified) or should I put that positively? I suspect that their properties and effect have not been adequately studied, modelled or quantified). Thus as Wlad says "it may be worthwhile to provide a systematic, succinct description"

As, I believe, we implicitly agreed following Wlad's comments re high level languages and compiling and Bob Balzer's remarks re the need for a Global approach we recognise the unquestionable real benefits that the feedback mechanisms Wlad mentions as examples bring. In view of the relatively small fraction of total process activity that these "local" actions and their feedback controls represent it is perhaps not surprising that their impact on global properties may be limited. We are nevertheless justified in asking whether

1. They could be reasonably expected to have more impact?
2. Do the control and feedback mechanisms currently in use in relation to these activities impose constraints on benefit?
3. Can we identify mechanisms, feedback or otherwise that could increase their impact?

The main question in relation to the straightforward ESWP is what information flow and feedback exists outside the range of activity implied by Wlad's definition, that is arising from the domain within which the ESWP is conducted and what impact, beneficial, constraining, otherwise, it has on product and process and process improvement ?

Extension of ESWP

loop picture, or at least closely related to it. In this the completed program is installed and becomes part of the application domain. It was this picture that led to the observation that every E-type program contains an implicit model of itself. Study of this HSWP model is (as is my third question above), I believe, closely related to Bob Snowdon's suggestion that we study the nature of the E-type problem (and solution) space. I too wonder the extent to which it may be made measurable in itself, never mind over a range of products. But I have hopes that some aspects can be quantified, vide the Lehman/Belady studies. Incidentally though for some of the latter we separated out time and resources, in fact we did very little that involved manpower and found the most useful measure of "time" to be release sequence number (Wlad - time considered irrelevant - though not, of course, for resource consumption). The RSN is a pseudo-time measure that proved powerful because the point of release are or are assumed to be the stable points in an otherwise flux of changing design, code and documentation.

As a "tool" for process extension (Wlad's term) the VSWP is "a process of process improvement", whereas the ESWP is a tool for software evolution and the HSWP appears, in Wlad's description to be a "situation" rather than a tool though one can envisage tools that would support the transition. Typing this and rereading Wlad's comments makes me wonder whether he is thinking VSWP too as a situation rather than a process or tool. In any event the three SWP terms represent different types of "beast" not just different level and I think need some elaboration to firm them up and facilitate their use in building up understanding and, dare I say it, a theory of software evolution.

Wlad's concept clearly has potential but needs considerable clarification and refinement. For example §9 opens with the remark that "VSWP obtains" suggesting a state (what I called above a "situation". In the next § he talks of VSWP "runs" and "outputs". Are these terms compatible? This is not a criticism but an indication that I, at least, require clarification of the terms. What are they, what do they mean. However the concept of the output of one ESWP being "better" than that of another while very, very broad is clear. How to determine it is a different, but crucial, question. I'm afraid, however, that Wlad's first attempt in terms of Manpower (or should that be Person Power?) is not very satisfying seems it is based on an assumption of OKness which cannot be divorced from the concept of better. A circular situation?

Yes - in practice one seeks both H and V extension simultaneously but perhaps that is precisely one of the aims of the FEAST exercise - to separate them.

And so to the final §. The more detailed study - discussion, clarification, extension and definition throughout these processes is a way, a useful way, forward. The field is unquestionably confused, where people think about it at all. I would certainly want to be involved in such investigation and hope that others will join me. Some discussion is needed before more is written down so I stop at this point.

Manny

From: Wladyslaw "M." Turski <wmt@mimuw.edu.pl>
To: mml@doc.ic.ac.uk
Subject: FEAST discussion

Dear Manny,

As I was collecting various comments that you fed to us electronically and combined them into a "folder" I realized that there was much more interesting stuff in these notes than there was at the discussion day 2. (My fault? - wrong tone set at the opening? perhaps!) One thing transpires quite clearly: Without better definition of what we are talking about no real progress is possible. There is no controversy between philosophical and technical approaches better definition - a technical point - arises from better understanding of a phenomenon and its setting - a philosophical point!). Wlad

To: Wladyslaw "M." Turski <wmt@mimuw.edu.pl>
From: mml@doc.ic.ac.uk (Manny Lehman)
Subject: Re: FEAST discussion

>Dear Manny,

>As I was collecting various comments that you fed to us electronically and combined them into a "folder" I realized >that there was much more interesting stuff oin these notes >than there was at the discussion day 2. (My fault? - >wrong tone set at the opening? perhaps!)

Yes, in part at least. You just raised one issue which rapidly became a red herring. Partly my fault in not instructing Valerie as to the objectives of the morning at least. She, Collin and Jean Pierre obviously have strong interest, the first two personal, Jean Pierre corporsate in getting funds hence she drove the discussion the way it went. I realised what was happening but did not feel able to intervene and redirect. We learn from our mistakes. At least, I try to.

>One thing transpires quite clearly: without better definition of what we are talking about no >real progress is possible >(and there

understanding of the phenomenon and its setting - a philosophical point!).

Agreed and we shall see how to get that going. Collin tries some definition in his position statement which I shall send out later today but I have not read it yet.

Manny

>From mml
>Reply to Bob Rockwell
>Subject: Re: Evolution, Entropy

Bob

Delighted to hear from you, wherever you are. Thought you were in the States but it appears not.

> Manny, it has been too long (for me) since last we talked. Yes, too long for me too > I do hope that both you and >Chava (sp?) are well, and life stays interesting.

Fine and busier than ever b'h.

> The June 1994 issue of IEEE Computer has 3 articles that deal with themes we have talked much about in the past. >The title stories are on "genetic algorithms" -- program structures that mimic the process of evolution to locate >ever-improving >solutions to complex problems. As I noted in that long Email to you a year or so ago, that puts >another spin on your discussion of "software evolution." The same >issue also contains an article by a young project >engineer at the Northrop >corporation on "conceptual entropy." His citations are few (an IEEE policy) and purely >technical and/or self-referential. He ought to be aware of your work, but I suspect is not. It is none- the less >intriguing to see people gradually stumbling onto concepts you derived from first principles long years ago.

Will keep a look out for it and then comment be'h

>What are you doing currently?

That question embarrasses me because I should have been in contact with you about some surprising developments. Some months ago two things happened. 1. 1. We got an indication that if we came up with a sensible proposal we might be able to obtain an extension to our ESF funding. We came up with three appropriate proposals and after a tough fight and some bouncing backward and forward to demonstrate "Industrial Relevance" obtained the extension funds 1 April 1994 - 31 March 1995. Of the three one was my FEAST proposal. To explain that I switch over to point 2.

2. Many months ago I asked myself the question if, over the past twenty years, so many innovations in the programming process have failed to deliver the anticipated benefits instead of providing individual explanations for each one perhaps we should be looking for a common cause. And having asked the question the answer was obvious. In fact I don't understand why it took 20 years to recognise. Scattered throughout my writings over that period I find all the evidence and assertions to have come to the same conclusion long ago. Anyway I formulated the conjecture that since it is universally recognised that the software process is a learning process that relies heavily on multi-space feedback it comprises a multi-loop feedback system with both positive and negative feedback. (whatever that means in this context). That being so the "system" must be expected to display the common property of feedback systems (given enough negative feedback) of external stability or invariance, that is its externally visible global properties will not change as a result of changes to forward path mechanisms alone. Global changes require changes to the feedback paths and mechanisms. End of brief outline of conjecture. I therefore established a project FEAST, Feedback, Evolution And Software Technology. To explore the conjecture (which is "obviously" correct) and see if and how it may be exploited.

This is a difficult investigation if only because the feedback mechanisms heavily involve humans and clearly demands a multi-disciplinary approach and several years of investigation. The potential benefit if successful is a "pot of gold". I therefore immediately fished around for potential partners and 1.5 weeks ago we had a very successful 2 day "First FEAST Workshop".

Rather than say more now I will 1. Send you a copy of the ws Preprint which will tell you more AS SOON AS YOU LET ME HAVE YOUR SNAIL MAIL ADDRESS.

2. I will put you on the bcc circulation list of "interested but not (yet?) involved people/organisations. Should you and SoftLab or who ever you work for decide to become involved I will then switch you to the main FEAST list

>Did you find a publisher for the "long version" of your big history/review article on >process/evolution?

No I never have had the chance to finish it and now the ideas and insights are b'h coming too fast to make the time right.

This is just as I typed it off the cuff. Hope it reads alright but I really must not take more time right now. My first contribution to the preprints provides a (I hope) better introduction though it already needs revision.

From: ROC%softlab.uucp@Germany.EU.net (Bob Rockwell)
To: mml@doc.ic.ac.uk (Manny Lehman)
Subject: Mail arrivals

Manny, this is just to let you know that the copy of Aidan Ward's comments and your response arrived here in my softlab mailbox *twice* -- did you perhaps send them to both my softlab aliases? If so, you can drop the longer one. Our new permanent address is the "softlab.de".

On content: I'm sure I don't have to tell you that I agree totally with Aidan on the poisonous effects of public money. Sadly, I must also accept your observation that current business realities make the funding of your work by industry a shaky proposition at best. Especially since I also agree with you and Aidan and the others that the philosophical issues are utterly critical: perhaps of more practical value in the long run than the short-term "objectives and actions" you refer to!

So much keeps coming back to what Fred Brooks tellingly identified as people's persistent belief in the possibility of silver bullets. Management must learn to accept that developing software is just plain hard, in a way that will never go away, for the same reasons that writing a good contract is hard: because there are many imponderables, and the parties must discover together what risks they are willing to accept, and how much they are willing to pay for different things, etc.

It sounds like you have a good group. Who else besides Ward, Snowden and Turski are involved?

Thanks for including me. Bob

From: Berc Rustem <br@doc.ic.ac.uk>
Subject: FEAST (or BEAST?)

Dear Manny I meant to send you a message earlier but the present activity prompted me to action.

I enjoyed most of the meeting, though was thoroughly frustrated sometimes as I heard the same arrogance coupled with ignorance that I used to hear in the seventies about the applicability of control and dynamic systems theory to economic systems.

I do not mind you treating me as a some kind of resource to be unleashed (while under a chain all the time) if and when it is necessary. It does not bother me. However, people should know at least what control and systems theory is, what it does and what it requires as input. I hope you will allow me to summarise this either directly or through you (I have no particular ambition to talk at this stage - at least because I suspect I might put a number of people off, including Bob Bishop).

Systems theory requires the specification of a process. That was thoroughly evaded. People mentioned learning and adaptation. These are really advanced concepts relative to the specification of the underlying process.

Another thing is feedback. I found great enthusiasm to specify the feedback rather than the process it was supposed to be feeding back to. When Tom drew his diagram, I wanted to add to it that whether you look at the process incorporating the feedback or without the feedback should be clarified.

The question that you change the system that you are studying/controlling is discussed widely in economic systems. This is called rational expectations. There is a huge literature on it which is quite nice and helpful. The basic idea (in its simplest form) is this: you know that people have expectations, you know that they try to guess what will happen in the future. You ask the question, if people's actions today is based on their expectations of the future and if the expectation happens to be exactly the future value (in the extreme case) what will happen?

The answer is that the future influences the present. Physicists dealing in time travel consider this. so do economists. It involves the determination of a fixed point that ensures that even if you know what the future is, your actions today are taken accordingly so the future stays the same (since your action depends on future expectation and that is at its future value). The future, in turn is decided by your action today. But as you had determined your actions based on the correct guess, the future is the same as your initial guess.

All the best Berc

Reply: mml
From: Berc Rustem
Subject: Re: FEAST (or BEAST?) Cc: Bcc: X-Attachments:

>Dear Manny I meant to send you a message earlier but the present activity prompted me to >action.

Thank you for reacting and I apologise for the delay in replying. Life is simply too hectic these days.

>I enjoyed most of the meeting, though was thoroughly frustrated sometimes as I heard the same arrogance coupled >with

I'm sorry that you should feel so strongly about comments that were made. I must admit that I do not recollect any reactions that I would have described as arrogance. Still if, as the expert on the subject, if you felt that way I'm pleased you felt able to say so.

>I do not mind you treating me as a some kind of resource to be unleashed if and when it is necessary.

I don't understand that remark at all. I approached you and asked whether you were interested in the project/investigation and consider you a full member of the team. I am sorry that you feel suppressed or otherwise under some sort of constraint but to the best of my knowledge I have not said or done anything to make you feel that way. If I feel I did then that is most certainly a misunderstanding.

>(while under a chain all the time) It does not bother me.

What chain who restricted you in any way I certainly did not notice that and, by the way, Vic mentioned that at one point in the discussion he was signalling to you to chip in and respond but that you did not react so please let's try and clarify the situation
>However, people should know at least what control and systems theory is, what it does and what it requires as input. >I hope you will allow me to summarise this >either directly or through you

Certainly directly I would be delighted if you make a contribution, at this point in writing, in October verbally. Certainly not through me. The essence [of a successful project and investigation is that every participant feels free]and says whatever he or she considers appropriate. So go ahead.

>(I have no particular ambition to talk at this stage - at least because I]suspect I might put a number of people off, >including Bob Bishop).

What has Bob done to you or said that triggered that remark? He is a very bright, knowledgeable and experienced individual and I do not remember his saying anything outrageous.

>Systems theory requires the specification of a process. That was thoroughly evaded. People mentioned learning and >adaptation. These are really advanced >concepts relative to the specification of the underlying process.

Clearly so as others have also commented in the remarks I have circulated. Several of us felt that we did not pay enough attention to definition and fundamentals and that is something I hope to see rectified in current interchanges and at the 2nd workshop.

>Another thing is feedback. I found great enthusiasm to specify the feedback rather than the process it was supposed to >be feeding back to. When Tom drew >his diagram, I wanted to add to it that whether you look at the process >incorporating the feedback or without the feedback should be clarified.

You obviously have more experience/insight in this area than any of us and I really hope that you will give us the benefit of that in the future so that we can move forward together.

>The question that you change the system that you are studying/controlling is >discussed widely in economic >systems. This is called rational expectations. There is a huge literature on it which is quite nice and helpful. The basic >idea (in its simplest form) is this: you know that people have expectations, you know that they try to guess what >will happen in the future. You ask the question, if people's actions today is based on their expectations of the future >and if the expectation ?happens to be exactly the future value (in the extreme case) what will happen? The answer is >that the future influences the present. Physicists dealing in time travel consider this. so do economists. It involves >the determination of a fixed point that ensures that even if you know what the future is, your actions today are taken >accordingly so the future stays the same (since your action depends on future expectation and that is at its future >value). The future, in turn is decided by your action today. But as you had determined your actions based on the correct >guess, the future is the same as your initial guess.

If your guess was correct. Do we need to get involved in such considerations at this stage or are they, from our current standpoint second order effects. > Thanks again

Manny

Mon, 11 Jul 94 11:37
From: Berc Rustem
Reply-To: Manny Lehman, Jul 4, 94 04:27:20 pm

Dear Manny I think you took my remarks more negatively than intended.

A number of issues that I raised have begun to be addressed in the recent emails that have been circulated. The coauthored book by Tarek Abdulhamed provides a good beginning.

model evolves from the discussion. Rightly or wrongly, I thought that was your view too and that it was wise, even though frustrating in the short run.

As to affecting the system you observe, this has been an objection to every exercise that has to do with managing activities that involve humans. It is an important consideration that needs to be taken into account in calculations but would not, barring extreme circumstances, invalidate or cancel management action.

Berc

Mon Jul 11 12:47:33 1994

To: Berc Rustem

From: mml

Subject: Re: FEAST (or BEAST?)

>Dear Manny >I think you took my remarks more negatively than intended.

Thanks. I am delighted that I was wrong on that score

>A number of issues that I raised have begun to be addressed in the recent emails that have been circulated. The >coauthored book by >Tarek Abdulhamed provides a good beginning. Perhaps I should have talked as you say. >However, despite my complaints, I believe I should wait a while longer until a prototype model evolves from the >discussion. Rightly or wrongly, I thought that was your view too and that it was wise, even though frustrating in the >short run.

No that was not my view and I would have welcomed your contribution. Never mind, that is behind us. However I wonder whether you should wait or whether you have not a role to play in evolving a prototype model or in helping us select/develop methods and formalisms for constructing such models. > >As to affecting the system you observe, this has been an objection >to every exercise that has to do with managing activities that involve >humans. It is an important consideration that needs to be taken into account >in calculations but would not, barring extreme circumstances, invalidate >or cancel management action.

Of course not but it must be taken into account. Any good manager will carefully consider the possible effects of his words/actions and may, according to circumstances use his analysis or observation to modify the action or adjust, for example, the rate or way the are put into effect. AN example of real or virtual feedback. Thanks again, Manny

Mon Jul 11 13:07:10 1994

From: Berc Rustem

Subject: Re: FEAST (or BEAST?)

To: mml

Date: Mon, 11 Jul 1994

In-Reply-To: "Manny Lehman" at Jul 11, 94 12:47:36

I would have loved to be involved right away. I failed to pick an individual with whom I could have heated dialogues about this at the meeting. I must be careful to say that "I" failed, not that there wasnt one... There is a recent email by Susan someone. That looks a helpful email. I am afraid I deleted it but I shall try to get in touch with her. Please let me have her surname if you know her.

Just now my coeditor on JEDC resigned (actually some months ago, but I am feeling the delayed effect now) so I have to carry the entire journal so until I get someone to help me as co-editor, I am buried in files. But that will pass, soon I hope.

I also have not explained properly that economists use this "managing those who can observe the management" to argue against macroeconomic policies of governments. There are (quantitative) techniques to work around this most of the time, as far as dynamic economic models are concerned.

Regards Berc

From: Suzanne Robertson

Subject: Feast Feedback

Manny,

Here (sorry it's late) is my feedback as requested at the end of the Feast Workshop 16/6/94 to 17/6/94.

Thank you for organising the workshop, it was very well run and the mixture of participants' backgrounds and interests is ideal. The presentations and discussions, while informative in themselves, did not place enough emphasis on the idea of feedback. Improvement of the software development process by using feedback is the core idea of the Feast project, it is a unique selling point

focus is probably because there are so many different types of feedback loops in the software development process and, at this early stage, we are talking about them in a very general way. Our proposal for funding should discuss different types of feedback and explain what effect they have/could have on the process of software development. Here are some of the issues I would like to examine further, along with my ideas so far:

What is feedback? - Information about the state of a processor. - Evidence that can be acted on to tune the operation of a system.

What Types of Feedback? - Planned feedback - is produced and reacted to as the result of implementing an S type specification. This type of feedback is produced by processors (automated or human) within the boundary of the system being studied. - Unplanned feedback - is produced and reacted to as the result of implementing an E type specification. Can be caused by processors either within or external to the boundary of the system being studied.

How does Feedback affect the Software Development Process? - Management reliance on planned feedback contributes (causes?) project failure - Obviously, unplanned feedback is more difficult to recognise because of its unexpected forms and (often) exogenic sources. Also, projects are often set up so that any unplanned feedback is ignored because it interferes with the schedule

How can FEAST benefit the Software Development Process? - Models and definitions of unplanned feedback, how to recognise/find it, how to react to it and how to make it an integral part of software development. - Methods for making unplanned feedback obvious - A tuning approach to running projects

I hope these comments are useful to you.

Many thanks Suzanne

From: D.J.Homan
Subject: FEAST - feedback

Manny,

Firstly I must thank you for inviting Dave Smith and myself to a most stimulating and thought provoking two days.

As we stated at the end of the meeting TXE Engineering very willing to be a source of software engineering process data and to act as guinea pigs in the trialing and evaluation of new software development techniques.

We have great sympathy with the ideas that you postulate in the FEAST conjecture in particular that it is often reasons beyond the immediate scope of the software engineer that dictates if process change or progress is made or not. For example, I would have great difficulty in justifying the expenditure on a new CASE tool simply on the return on investment. I would probably have to justify it on the grounds of greater efficiency which if successful would mean in the end a reduction in staffing levels.

The main thing that concerns us as software engineering practitioners who have been delivering high quality software to a very demanding customer for the last 10 years is that much of what we heard during your workshop we had great difficulty in relating to what I would describe as a real world situation. We came away from the workshop wondering how we ever managed to ship a successful product out the door.

I look forward to seeing you again.

Dave Homan TXE Engineering Quality Manager

To: D.J.Homan
From: Manny Lehman
Subject: Re: FEAST - feedback

>Manny, Firstly I must thank you for inviting Dave Smith and myself to a most stimulating and thought provoking two days.

It was a pleasure to have your participation and now I look forward to active collaboration.

>As we stated at the end of the meeting TXE Engineering very willing to be a source of software engineering process >data and to act as guinea pigs in the trialing and evaluation of new software development techniques. We have great >sympathy with the ideas that you postulate in the FEAST conjecture in particular that it is often reasons beyond the >immediate scope of the software engineer that dictates if process change or progress is made or not. For example, I >would have great difficulty in justifying the expenditure on a new CASE tool simply on the return on investment. I >would probably have to justify it on the grounds of greater efficiency which if successful would mean in the end a >reduction in staffing levels. The main thing that concerns us as software engineering practitioners who have been >delivering high quality software to a very demanding customer for the last 10 years is that much of what we heard >during your workshop we had great difficulty in relating to what I would describe as a real world situation. We came >away from the workshop wondering how we ever managed to ship a successful product out the door.

is no question that reputable companies such as BNR and (in the UK) its predecessors have been able to deliver. The question is how much faster, how much higher quality, how much more responsively, how much more release content, what cost reductions etc. etc. should one be able to achieve?

I too look forward to continuing and active contact

Manny

1. FEAST itself must be much clearer and sharper than presently defined. As we see it, it is not simply intended to meet the objectives as summarised in the preprints or the actions as worked out at the workshop.
2. What level of feedback are we talking about.
3. At that level attempt to define one feedback path, define it.
4. Take it to some one who understands control theory and ask does that identified path represent the concept of feedback in the classical sense.
5. Difference between iteration and feedback, if any.
6. The real problem is the nature of the challenges of designing and evolving E-type systems.
7. One issue is the definition of their boundaries and domains.
8. How do we design the process when we know that there will be continuous pressure for product evolution?
9. How do we improve the process to cope with ever larger and more complex applications requiring collaboration with ever more people?
10. What product architectures are appropriate for more complex and evolving applications?
11. How do we cope with the situation in which we have to change an existing (and operational) system rather than starting from ab initio?
12. Can the FEAST conjecture be of any help in answering (or tackling) these questions?]

19/7/94 Dewayne, Jose, Vic, Manny

A wide ranging meeting, unfortunately unrecorded, discussing various technical issues and initial plans for the second workshop.

20/7/94 Dewayne, Manny

1. The most important comment yesterday (discussion between Vic, Jose, Dewayne and Manny was not recorded) was the fact that, in general, global system characteristics are function of many, not just one, high level feedback loops - see 4/DO6 below
2. In relation to the difference between iteration and feedback we note two types of iteration
 1. As in iteration over a set of objects eg. an array as in matrix multiplication
 2. Iteration with feedback as in Newtonian solution to a polynomial equation with iteration, yes or no, determined by size of residue
3. Discussion points.
 1. Produce a high level model of process, suggest we start with my chart no. 522/9-[charts]-15
 2. FEAST 1 and 2 documents and possible responses
4. Walk thru. PPs.

Action 1 - ct D05: Establishing a common process modelling capability. Colin's paper primarily concentrates on definitions. Propose to accept his analysis for the moment leaving it to individuals to address and possibly question or modify any issues they wish. Has covered interesting points but does not appear to address the issue of what modelling capabilities we need, what formalism might be appropriate and so on. **That question must be addressed with some priority.** Priorities? Available symbolisms, languages etc?

Action 2 - ct D06: Application of Control Theoretic and System Dynamics modelling and other appropriate techniques.

software process is characterised by a *complex conglomerate* of interconnected feedback loops. This is probably the source of the difficulty in providing an example of feedback as a constraint on deriving innovative benefit.

2. In "What to Plan to Do" 1 Tarik refers to reuse as an example of a dynamic problem. We suspect that reuse is not a good example since it appears to have only limited feedback aspects. In 2 he refers to Generic Feedback types. We really need to understand specific rather than generic types. Suggest that a study of the generic properties of *E*-type systems (a la Bob Snowdon) would be a potentially more fruitful approach.

Action 3 - ct D07: Impact of innovations on the software process. There appears to be a discrepancy of sorts between Colin's formulation of the action and Carrie's interpretation with the latter rather closer to the original formulation by Vic and mml. Carrie's methodology appears sound but we are not optimistic about obtaining positive results whereas the formulation of D07 seems more likely to produce results. This requires some tweaking of the approaches proposed in the PP, a slight shift of emphasis.

Action 5b - ct D09: Abstract and generalise the study (D08) of at least one real-world process. It came out at the workshop (Bob Balzer), and is supported by Tarik's statement re interacting conglomerate of feedback loops, that FEAST concern is, in the first instance, with the global process though, if successful, it will ultimately move down in the process structure, ie a top down approach. In that sense the PP's approach, while very convincing, should be applied to a global model rather than to a sub process area as proposed in Suzanne's section 3.4. With this minor modification the PP defines a challenging activity which should be undertaken as soon as possible.

Action 9 - ct D13: Review existing doctrines. An interesting document with important insights but not clear of its relevance - for the moment at least - to FEAST

21/07/94 Dewayne, Manny continued

The above reactions to the position papers are cursory and represent a first trawl through the statements. We look forward to a wider reaction and above all to some action. Also we await with interest the remaining 5 PPs.

Now over to Comments 1 - document F2 - including Dewayne's reactions to my reactions.

Aidan - page 4

\begin{itemize}

\item Concentrate on the notion of feedback in the engineering process and try at this technical process design level to make process improvement happen. "Do the job better"

\item Revisit the role of software engineering as a business activity on the grounds that the evolution of software systems is far from understood by their consumers/purchasers. "Understand what the job is"

\end{itemize}

I propose that the first choice is reductionist and against the spirit of FEAST although it is far more manageable in a narrow sense.

Comment Aidan's first choice we agree with him that actual process improvement not a focus of FEAST activity though we may have to be diplomatic in expressing this to avoid switching off potential funders. The point is that FEAST investigations should improve **the process improvement process** though part of this and the search for **understanding** will lead us into exploring process improvements and how they might be improved, ie "global benefit" increased.

We both agree with Aidan's basic thesis that the technical, organisational and sociological are inseparable or rather inextricably mixed. It is, in fact, implicit in all that has been written on FEAST and explicit in Dewayne's *People, Organisations and Process Improvement* paper in the July issue of Software Magazine, that in studies of process from almost any point of view one should not divorce or isolate technical studies from organisational and sociological issues. It has also always been explicit in Manny's seven laws. Their statement has always been in technical terms while their "explanation", interpretation of the phenomenology they reflect or model is organisational or sociological.

As already indicated in the immediate comments as above we both agree with much, if not all, of Bob's comments and approaches. We draw particular attention to his remark

This is what I was trying to suggest in my talk by characterising *S*-systems by the phrase "computer as calculator" and *E*-systems by "computer system as >partner"

which I (Manny) did not take in during Bob's talk and glossed over in my reactions to his comments. It seems to us that this remark reflects a fundamental insight that is not unrelated to some discussions we are having in the context of FEAST with some of the Logic/AI people in the Department on the meaning and role - even appropriate definition - of their concept of (multi) agents. It is worthy of further consideration, discussion and refinement.

Colin page 12

Overall we accept/agree with Colin's observations as already commented on. We draw attention to his characterisation on page 12 of iteration vs feedback which agrees with ours as noted above on page 1, point 2.

Wlad page 14

We question Wlad's use of the word "backtracking" in sentences such as

Also internal to the ESWP are feedback loops resulting from walkthroughs, code inspection etc. as long as they cause backtracking before Product delivery.

Wlad appears here to have used the term "backtracking" informally without consideration of its precise meaning. The process he describes is clearly one of feedback. (We realise that we have not yet as a project adopted a firm view or definition of feedback or identified its essential properties.) We claim, however, that that process is not to be termed backtracking. Backtracking we see as a process of unrolling or backing up followed by redoing or regeneration. These activities constitute a mechanism for giving the feedback information the required characteristics and applying it. There is a level of process analysis at which backtracking is an important technique for, for example, correcting a product property by undoing and repeating a portion of the process (but not necessarily unrolling and redoing the entire product). This is quite different from a fault repair process where fault information from an internal organisation or from users is injected into an ongoing evolution process. The distinction may be subtle or fine but it is vital.

Wlad introduced EWSP as a high-level software production process and extended it in two directions: HSWP and VSWP. HSWP is ESWP in which the product is fed back; VSWP is EWSP modified to take the process ESWP instead of a software system to produce a new and improved version of EWSP. It is our claim that only one such model is required to express this intent.

We propose the following EvSWP model as an alternative evolutionary process for the evolution of *products*: in a format following Tom's proposal but omitting feedback connections and mechanisms.

Input:	<i>Needs</i>	
	<i>Product</i> ,	(initially, empty - or loosely constrained)
Uses:	<i>Resources</i>	(eg, manpower, time, ...)
	<i>Process</i>	(eg, Methods, tools, ..., their linkages and interactions,...)
Output:	<i>Product</i>	(often made up of various components, etc)
	<i>Waste</i>	(scaffolding, leftovers, ...)

With this model, one description takes care of both ESWP and HSWP. Moreover, given the level of generality here (and the purposeful use of generic terms), it does not matter what kind of product is being produced. In particular, either a software system or a software process system may be introduced and evolved. Hence the model, as it stands covers VSWP as well.

Of course, in detailing this process to deeper levels the reification reaches a stage, level of detail, where product-specific matters must be considered. At that point the distinction between, for example, evolving a

will emerge.

Suzanne page 22

In her feedback Suzanne states:

The reason for this lack of focus is probably because there are so many different types of feedback loops in the software development process and , at this early stage, we are talking about them in a very general way."

She then goes on to distinguish planned and unplanned feedback. We agree that there are many different kinds and, further, that we need a clear and crisp definition or characterization of feedback mechanisms and their various attributes. In addition to planned and unplanned feedback, distinction that are made include

positive, negative feedback - the sign of feedback is critical in determining its effect.

recognised, unrecognised feedback

used, ignored feedback

useful, useless feedback

explicit, implicit feedback

immediate, delayed feedback

17 August 1994

I have simply reproduced the entire contents of the various message without removing anything except purely "social" comments that are clearly totally irrelevant to the technical discussion.

From: "Wladyslaw M. Turski" <wmt@mimuw.edu.pl>
Subject: Feedback on feedback

Dear Friends,

I've been thinking about the blasted feedback, trying to get some simplifying view. The results of this thinking process follow. Do what you like with them (e.g. trash them!). I'll be away from my office rest of this week (another conference, in Poznan), then for 10 days at the desk, then - until the end of July - on holidays.

Cheers, Wlad

First, consider Elementary SW Process (ESWP):

Takes Input: Need (for a computer application)
Uses Resources: Method (or a work procedure)
 Manpower*Time
 Tools (e.g. SADT, Ada compiler, IPSE)
Produces Output: Code |
 Documentation | Collectively known as
 User training | Product
 |

Within a class of applications the ESWP is "reusable as is": given an Input it produces an Output consuming some Resources (Manpower*Time is expanded, while Method and Tools are worn out, i.e. become progressively more obsolete each time the process is run).

In an ESWP there usually are several built-in feedback loops (internal to the ESWP): e.g. the Method (think of LST) may be iterative (backtracking), the application of a compiler may result in a list of syntactic errors, thus forcing a modification to the program text (previously constructed). Also internal to the ESWP are feedback loops resulting from walkthroughs, code inspection etc. as long as they cause backtracking before Product delivery.

All this is reasonably well-understood and investigated, although it may be worthwhile to provide a systematic, succinct description (e.g. emphasizing that "backtracking" is a feedback loop!).

There are two ways in which an ESWP may be extended. For simplicity I shall treat them as orthogonal:

- horizontal extension (HSWP)
- vertical extension (VSWP)

HSWP obtains when the Product of a given ESWP is made into (a part of) its Input, i.e. the need is expressed relative to an application and an existing product. Thus we have a horizontal external feedback loop (output loops into input).

Assuming that the Method and Tools can cope with such a mixed Input (and it is not at all clear that this is the case!) we have the first opportunity for a feedback control: the amount of resources (Manpower*Time) consumed will vary depending on the "size" of change {defined as the "ratio" (new need - old need)/product}, i.e. on the magnitude of the control part of the input. I think this is exactly what Lehman/Belady studies of OS360 have shown, except they used Manpower and Time components separately.

Here the scope for research is unlimited - but I am sceptical about the possibility to make all necessary ingredients measurable over a range of products. The best one can hope is to get data on a given product over its consecutive releases or versions (again, as for OS360). Note however that contemporary products do not have so many releases.

VSWP obtains when we consider a process of "improving" an existing ESWP. Now the Input is the old ESWP, the Output a new ESWP, Resources are Manpower*Time, Tools (I doubt the current availability of suitable tools of this category!), Method (Procedure for improving ESWP).

coding for a tool supplied as input to (a specialized) VSWP is transformed into an ESWP with Ada programming as a corresponding tool (I am referring here to a tool inside ESWP!). This new ESWP is the Output of the VSWP. Note that both the old and new ESWPs can be used to make products (when run on their common Input: a need each will produce a Product). The difference is not necessarily in products of these two ESWPs, but - presumably - in that the new ESWP is "better" than the old one. How can one measure the goodness of an ESWP is a nice problem; one easy assumption is to consider the amount of Manpower*Time resource used by an ESWP (on a given input) to make a satisfactory product. The ESWP which uses less Manpower*Time for the same input (need) is better (as long as its product is OK).

Where is the feedback loop in VSWP? Of course it obtains when we take its output as input. But this is just iteration - always a feedback phenomenon! To get a feedback control in thus looped VSWP we need to extend its Input by a "need" for ESWP and agree on a measure of satisfaction (wrt this need) provided by an ESWP.

In reality we are probably facing the problem both horizontal and vertical extension of ESWP at the same time, which makes the research combinatorially more complex.

Yet, my recommendation would be to start with a more detailed study of the constituents of the model: ESWP and its internal feedback loops and controls (if any), HSWP and its internal feedback loops and controls (here they sure exist!), VSWP and its internal feedback loops and controls (very dubious). As I can see the field today, it is hopelessly confused with internal loops of ESWP held "responsible" for both horizontal and vertical evolution (i.e. for sequences of horizontal and vertical products of entirely different processes).

From: mml@doc.ic.ac.uk (Manny Lehman)
Subject: Wlad's comments
30/6/94, 2.35

Wlad

§1 - 4. Minor typo "expanded" should presumably be "expended" (though one of the classical software process problems is that so very often the manpower time product is, indeed, expanded.

Yes the feedback loops you mention are well recognised though I am not confident that their properties and effects have been adequately studied, modelled or quantified) or should I put that positively? I suspect that their properties and effect have not been adequately studied, modelled or quantified). Thus as Wlad says "it may be worthwhile to provide a systematic, succinct description"

As, I believe, we implicitly agreed following Wlad's comments re high level languages and compiling and Bob Balzer's remarks re the need for a Global approach we recognise the unquestionable real benefits that the feedback mechanisms Wlad mentions as examples bring. In view of the relatively small fraction of total process activity that these "local" actions and their feedback controls represent it is perhaps not surprising that their impact on global properties may be limited. We are nevertheless justified in asking whether

1. They could be reasonably expected to have more impact?
2. Do the control and feedback mechanisms currently in use in relation to these activities impose constraints on benefit?
3. Can we identify mechanisms, feedback or otherwise that could increase their impact?

The main question in relation to the straightforward ESWP is what information flow and feedback exists outside the range of activity implied by Wlad's definition, that is arising from the domain within which the ESWP is conducted and what impact, beneficial, constraining, otherwise, it has on product and process and process improvement ?

Extension of ESWP

HSWP - when the "product" of ESWP becomes a part of the input "need",. This is essentially the situation modelled in my closed loop picture, or at least closely related to it. In this the completed program is installed and becomes part of the application domain. It was this picture that led to the observation that every E-type program contains an implicit model of itself. Study of this HSWP model is (as is my third question above), I believe, closely related to Bob Snowdon's suggestion that we study the nature of the E-type problem (and solution) space. I too wonder the extent to which it may be made measurable in itself, never mind over a range of products. But I have hopes that some aspects can be quantified, vide the Lehman/Belady studies. Incidentally though for some of the latter we separated out time and resources, in fact we did very little that involved manpower and found the most useful measure of "time" to be release sequence number (Wlad - time considered irrelevant - though not, of course, for resource consumption). The RSN is a pseudo-time measure that proved powerful because the point of release are or are assumed to be the stable points in an otherwise flux of changing design, code and documentation.

software evolution and the HS WP appears, in Wlad's description to be a "situation" rather than a tool though one can envisage tools that would support the transition. Typing this and rereading Wlad's comments makes me wonder whether he is thinking VSWP too as a situation rather than a process or tool. In any event the three SWP terms represent different types of "beast" not just different level and I think need some elaboration to firm them up and facilitate their use in building up understanding and, dare I say it, a theory of software evolution.

Wlad's concept clearly has potential but needs considerable clarification and refinement. For example §9 opens with the remark that "VSWP obtains" suggesting a state (what I called above a "situation". In the next § he talks of VSWP "runs" and "outputs". Are these terms compatible? This is not a criticism but an indication that I, at least, require clarification of the terms. What are they, what do they mean. However the concept of the output of one ESWP being "better" than that of another while very, very broad is clear. How to determine it is a different, but crucial, question. I'm afraid, however, that Wlad's first attempt in terms of Manpower (or should that be Person Power?) is not very satisfying seems it is based on an assumption of OKness which cannot be divorced from the concept of better. A circular situation?

Yes - in practice one seeks both H and V extension simultaneously but perhaps that is precisely one of the aims of the FEAST exercise - to separate them.

And so to the final §. The more detailed study - discussion, clarification, extension and definition throughout these processes is a way, a useful way, forward. The field is unquestionably confused, where people think about it at all. I would certainly want to be involved in such investigation and hope that others will join me. Some discussion is needed before more is written down so I stop at this point.

Manny

From: Wladyslaw "M." Turski <wmt@mimuw.edu.pl>
To: mml@doc.ic.ac.uk
Subject: FEAST discussion

Dear Manny,

As I was collecting various comments that you fed to us electronically and combined them into a "folder" I realized that there was much more interesting stuff in these notes than there was at the discussion day 2. (My fault? - wrong tone set at the opening? perhaps!) One thing transpires quite clearly: Without better definition of what we are talking about no real progress is possible. There is no controversy between philosophical and technical approaches better definition - a technical point - arises from better understanding of a phenomenon and its setting - a philosophical point!). Wlad

To: Wladyslaw "M." Turski <wmt@mimuw.edu.pl>
From: mml@doc.ic.ac.uk (Manny Lehman)
Subject: Re: FEAST discussion

>Dear Manny,

>As I was collecting various comments that you fed to us electronically and combined them into a "folder" I realized >that there was much more interesting stuff oin these notes >than there was at the discussion day 2. (My fault? - >wrong tone set at the opening? perhaps!)

Yes, in part at least. You just raised one issue which rapidly became a red herring. Partly my fault in not instructing Valerie as to the objectives of the morning at least. She, Collin and Jean Pierre obviously have strong interest, the first two personal, Jean Pierre corporsate in getting funds hence she drove the discussion the way it went. I realised what was happening but did not feel able to intervene and redirect. We learn from our mistakes. At least, I try to.

>One thing transpires quite clearly: without better definition of what we are talking about no >real progress is possible >(and there is no controversy between philosophical and technical >approaches: a better definition - a technical point! >arises from a better understanding of the phenomenon and its setting - a philosophical point!).

Agreed and we shall see how to get that going. Collin tries some definition in his position statement which I shall send out later today but I have not read it yet.

Manny

Extract from F3 - minutes of discussions involving Dewayne, Jose Vic and Manny

Wlad page 14

We question Wlad's use of the word "backtracking" in sentences such as

backtracking before Product delivery.

Wlad appears here to have used the term "backtracking" informally without consideration of its precise meaning. The process he describes is clearly one of feedback. (We realise that we have not yet as a project adopted a firm view or definition of feedback or identified its essential properties.) We claim, however, that that process is not to be termed backtracking. Backtracking we see as a process of unrolling or backing up followed by redoing or regeneration. These activities constitute a mechanism for giving the feedback information the required characteristics and applying it. There is a level of process analysis at which backtracking is an important technique for, for example, correcting a product property by undoing and repeating a portion of the process (but not necessarily unrolling and redoing the entire product). This is quite different from a fault repair process where fault information from an internal organisation or from users is injected into an ongoing evolution process. The distinction may be subtle or fine but it is vital.

Vlad introduced EWSP as a high-level software production process and extended it in two directions: HSWP and VSWP. HSWP is ESWP in which the product is fed back; VSWP is EWSP modified to take the process ESWP instead of a software system to produce a new and improved version of EWSP. It is our claim that only one such model is required to express this intent.

We propose the following EvSWP model as an alternative evolutionary process for the evolution of products: in a format following Tom's proposal but omitting feedback connections and mechanisms.

Input:	<i>Needs</i>	
	Product,	(initially, empty - or loosely constrained)
Uses:	<i>Resources</i>	(eg, manpower, time, ...)
	<i>Process</i>	(eg, Methods, tools, ..., their linkages and interactions,...)
Output:	<i>Product</i>	(often made up of various components, etc)
	<i>Waste</i>	(scaffolding, leftovers, ...)

With this model, one description takes care of both ESWP and HSWP. Moreover, given the level of generality here (and the purposeful use of generic terms), it does not matter what kind of product is being produced. In particular, either a software system or a software process system may be introduced and evolved. Hence the model, as it stands covers VSWP as well.

Of course, in detailing this process to deeper levels the reification reaches a stage, level of detail, where product-specific matters must be considered. At that point the distinction between, for example, evolving a software system - the software process - and evolving a process system - process of process improvement - will emerge.

Mon, 8 Aug 94 09:43:14 +0200

Date: Mon, 8 Aug 94 09:43:14 +0200

From: Wladyslaw "M." Turski <wmt@mimuw.edu.pl>

To: mml@doc.ic.ac.uk

In-Reply-To: Manny Lehman's message of Fri, 5 Aug 94 13:34:45 BST <3779.9408051234@gummo.doc.ic.ac.uk>

Subject: your message

Dear Manny,

As promised, a few thoughts. It may be useful to distinguish between

- feedback (loop) and
- feedback control (loop).

The former - a more general situation - is when we are dealing with two (or more, but see below) mutually influencing parts of a "whole". The latter - an important particular case - when (at least) one of the parts plays the role of a "controller".

In a classic book "An Introduction to Cybernetics" (Chapman and Hall, Ltd., London, 1956), W.R.Ashby defines feedback (section 4/11) as any interaction between two parts of a system in which part P influences the behaviour of part R and part R influences the behaviour of part P.

Thus for example, given a set of differential equations

$$\begin{aligned}x' &= 2xy \\ y' &= x - y^2\end{aligned}$$

there is a feedback between x and y, whereas in the system

$$\begin{aligned}x' &= 2x \\ y' &= x - y^2\end{aligned}$$

several parts each influencing all others. With just four such parts there are 20 feedback loops and - according to Ashby - the system is far too complex for rigorous investigation.

In a feedback control loop we are considering a well-defined feedback loop in which (a part of the process) output is directed to a clearly identified controller which - in a well-defined manner - changes (some well-defined part of) the process.

This is the situation with the classic Watt controller for steam engines: the process converts the heat (unevenly supplied to the boiler) into steam of uneven pressure. The steam drives the piston, piston pushes the rod, movement of the rod performs some useful work (e.g. turn the wheel) *and* turns the controller shaft. The faster revolves the controller shaft (and, eo ipso, the wheel), the higher shift the weights driven by the centrifugal force. The higher go the weights, the higher goes the attached to them plunger, enlarging side opening in the steam duct (an escape valve). Consequently, less steam goes to the piston, it slows down, the rotation of the controller shaft is reduced, the weights

drop, the plunger closes the valve, more steam pushes the piston, the rod picks up its speed, etc. As I said, a classic negative feedback control loop!

In the software-making process we can identify many feedback connections, very few are clearly identifiable as feedback control! In fact we can identify so many feedback loops that the system becomes untractable if we want to treat them all at the same time.

Apparently, apart from general "philosophy", the only way to exploit the feedback paradigm would be to concentrate on simple, mutually dependent pairs of process-constituents and - preferably! - on simple, well-identified control loops. To do so, we need to identify the controllers in the process, determine the process-output components that feed the controllers, identify the nature (and laws) of controllers action on the process.

I do not think we can get simple closed formulae very easily (or at all!), but I hope that if we succede in identifying (at least some) of the controller elements and information paths through which they are fed and through which they exert control on the process, then even some early, qualitative work will be interesting, and - who knows! - may be useful in practical terms.

Let me repeat: it appears that the research agenda should concentrate on identification of controllers in the software process and on (at least) qualitative description of feedback control loops: process<==>selected_controller.

To: Wladyslaw "M." Turski <wmt@mimuw.edu.pl>

From: mml@doc.ic.ac.uk (Manny Lehman)

Subject: Re: your message

Cc: Dewayne,fgm,klc

Bcc:

X-Attachments:

Date: Mon, 8 Aug 94 09:43:14 +0200

From: Wladyslaw "M." Turski <wmt@mimuw.edu.pl>

To: mml@doc.ic.ac.uk

>Dear Manny

Dear Wlad

And herewith my off the cuff response to your interesting and thought provoking remarks.

>It may be useful to distinguish between

> - feedback (loop) and

> - feedback control (loop).

>The former - a more general situation - is when we are dealing with two (or more, but see >below) MUTUALLY INFLUENCING parts of a "whole". The latter - an important particular case - when >(at least) one of the parts plays the role of a "controller".

I understand what you say but wonder how real the distinction is or is it merely a question of roles. ie a feedback source is a controller if I say it is a controller. Surely it is no more than a matter of de facto (df) and de jure (dj). Consider element A feeding information to element B which processes it and feeds back some or all of its intermediate or final output back to A. I

behaviour in any way, even if only to reprocess so as to produce a different result, then surely de facto B must be considered to have "controlled" A? Is the distinction between an assigned (dj) and an incidental (df) role or is it between an ongoing (dj) and an occasional or when appropriate or when significant (df) role or have I missed something.

>In a classic book "An Introduction to Cybernetics" (Chapman and Hall, Ltd., London, 1956), >W.R.Ashby defines feedback (section 4/11) as any interaction between two parts of a system in >which part P influences the behaviour of part R and part R influences the behaviour of part P.

>>Thus for example, given a set of differential equations

>

$$> \quad x' = 2xy$$

$$> \quad y' = x - y^2$$

>there is a feedback between x and y, whereas in the system

$$> \quad x' = 2x$$

$$> \quad y' = x - y^2$$

>x dominates y (and there is no feedback, as x is not influenced by y).

Fine - but that is because x and y are not MUTUALLY influencing (see your definition above)

>Ashby goes on to state specifically that a precise definition of feedback is not needed because in >simple situations it is obvious, and in more complicated ones the notion itself becomes useless >He exemplifies such more complicated situations by a system consisting from several parts each >influencing all others. With just four such parts there are 20 feedback loops and - according to >Ashby - the system is far too complex for rigorous investigation.

Yes but that was some 40 years ago and there has been a lot of progress since. I don't underestimate the problems and the difficulties but we must re-explore to discover what is possible and what not.

>In a feedback control loop we are considering a well-defined feedback loop in which (a part of >the process) output is directed to a clearly identified controller which - in a well-defined >manner - changes (some well-defined part of) the process.

Ah - I see you do not consider B (above) as a controller but a "special mechanism in the path. But that surely is also a question of definition or viewpoint. In audio amplifier feedback one typically takes the output and feeds it back to the input via a resistor/capacitor attenuator that determines the fraction of signal feedback and hence the boost or attenuation at various frequencies. That same attenuator network may also be playing a role in the forward signal path so I would be hesitant in calling it either a feedback path mechanism or even a controller.

>This is the situation with the classic Watt controller for steam engines: the process converts the >heat (unevenly supplied to the boiler) into steam of uneven pressure. The steam drives the >piston, piston pushes the rod, movement of the rod performs some useful work (e.g. turn the >wheel) *and* turns the controller shaft. The faster revolves the controller shaft (and, eo ipso, >the wheel), the higher shift the weights driven by the centrifugal force. The higher go the >weights, the higher goes the attached to them plunger, enlarging side opening in the steam duct >(an escape valve). Consequently, less steam goes to the piston, it slows down, the rotation of the >controller shaft is reduced, the weights drop, the plunger closes the valve, more steam pushes >the piston, the rod picks up its speed, etc. As I said, a classic negative feedback control loop!

>In the software-making process we can identify many feedback connections, very few are >clearly identifiable as feedback control! In fact we can identify so many feedback loops that the >system becomes untractable if we want to treat them all at the same time.

Agreed in part - that is as to intractability. But many of them are "controllers" eg may management messages re resources, changes in objectives, financial constraints, client reactions and so on. We need some definitions or at least the beginnings of a model that identifies the separate "agencies" (role clusters - such as project managers, corporate management, marketeers, clients, developers, QA etc). Dewayne and I have begun definition of such a model and are looking for some colaboration with Frank McCabe and Keith Clark in relation to their work and simulation system for representing and studying multiple agent (hence agencies above) systems.

>Apparently, apart from general "philosophy", the only way to exploit the feedback paradigm >would be to concentrate on simple, mutually dependent pairs of process-constituents and - >preferably! - on simple, well-identified control loops. To do so, we need to identify the >controllers in the process, determine the process-output components that feed the controllers, >identify the nature (and laws) of controllersaction on the process.

That is certainly a sensible strategy for getting off the ground and I believe that Dewayne and I are preparing the ground and will continue, hopefully with participation, in 3 weeks time.

>I do not think we can get simple closed formulae very easily (or at all!), but I hope that if we >succeed in identifying (at least some) of the controller elements and information paths through >which they are fed and through which they exert control on the

That's what I hope and expect. But even if not, the intellectual exploration and stimulation should be fun.

>Let me repeat: it appears that the research agenda should concentrate on identification of >controllers in the software process and on (at least) qualitative description of feedback control >loops: process<==>selected_controller.

>Cheers

Wlad

Received: Wed, 10 Aug 94 09:03:04 BST

Date: Wed, 10 Aug 94 10:02:23 +0200

From: Wladyslaw "M." Turski <wmt@mimuw.edu.pl>

To: mml@doc.ic.ac.uk

Cc: dep@research.att.com, fgm@doc.ic.ac.uk, klc@doc.ic.ac.uk

In-Reply-To: Manny Lehman's message of Tue, 9 Aug 1994 12:59:27 +0000 <14008.9408091157@gummo.doc.ic.ac.uk>

Subject: your message

Message-Id: <"heron.doc..596:10.07.94.08.03.02"@doc.ic.ac.uk>

Dear Manny,

First, let me thank you for the thoughtful arrangements for my visit. By 18th I'll let you know whether I can accept or not. I received the hard copy of assembled comments and, most importantly, the "minutes" FEAST-3 (dated 25/7/94, 6:01 p.m.); most of the comments that follow is provoked by these minutes.

But first - an immediate reaction to your message referred to in the header.

No, the controller is not just a role assigned to a part: it is a part which has a well-defined (known and measurable) influence on the other part (which may be - tentatively - named "producer"). Of course, there is a degree of arbitrariness. In the steam-engine if we wish to appear original we may consider the boiler a controller for the weights+plunger. This arbitrariness can be probably eliminated by requesting that controller has no output other than control exerted on producer, but I do not feel a particular urge to do so!

> Yes but that was some 40 years ago and there has been a lot of

>progress since.

Well, not really (Incidentally, when I first visited Imperial (1965) on Stan Gill's invitation, I was giving two courses: on Compiler Techniques and on Control Theory (and had, among others, David Maine for a student!)) We must distinguish real progress (where people can actually compute things) from noise progress (very large systems etc). I repeatedly quote the example of socio-economic systems: there is a very large number of books, papers etc. claiming a huge progress, yet in practice every so often there is a revolution or a stock-market crash or a money-market upheaval or a large company goes bust. Such events are a posteriori analyzed by the theorists who fit their graphs to the historical events, strangely enough - there are precious few (like zero) predictions in these socio-economic areas which would be verified by actual developments. There is no other test that sharp: if a science cannot make predictions or if its predictions are not verified in practice that science is not very powerful!

Now on some older stuff. I continue to hold the view that any iteration (as opposed to mere repetition) is a feedback phenomenon. Justification: Iteration is a process described by

WHILE condition DO move

(or, alternatively, by

DO move UNTIL condition)

"move" and "condition" are the two parts in Ashby's definition of feedback, whose performance (evaluation, elaboration) influence each other. It matters very little if the condition is as simple as the check for the last element of a sequence (as in matrix multiplication) or a more complex. Indeed, in the matrix multiplication the "move" must include index manipulation ("get next element"), which evidently influences the "condition". As to the influence of the "condition" on "move" part, it is obvious: either the next move is done or not!

(In many iterations this influence is more sophisticated than this, as when the condition determines the step size for the move; most better diff.eqns. algorithms do it.)

whose elaboration causes some (or all) components of the suite to be executed again with new/ varied initial conditions, both the actual selection of the forward moves to be repeated and their initial conditions are determined by the

test. QED.

I am afraid, we are not free to arbitrarily exclude some feedback phenomena from our consideration merely because we prefer to call them something else!

Having said so, I must explain why I did mention that iteration and backtracking are instances of feedback. My sole purpose was to indicate how enormously broad is the scope of feedback phenomena in software making. Indeed, it is so broad that without some conscious narrowing we would be talking about nearly everything (which means we would not say much about anything).

This brings me to the main criticism of the "minutes". I simply fail to see what do we gain by abstracting ESWP, HSWP and VSWP to one process; go a step up and you get any process at all (software making, egg-cooking, oil-refining, airplane controlling ...) Then, of course, the only things to be said are such "helpful" observations as: "the process is characterized by a complex conglomerate of interconnected feedback loops". So it is, but what follows? Needless to say, I DO NOT CLAIM that the trio I proposed is perfect, far from it! But at least each of the three ESWP< HSWP and VSWP is sufficiently DIFFERENT from the remaining two to allow specific investigations!

Of course, in practice, any software shop is engaged in all three at the same time. The "global process" is a three-dimensional beast, but I fail to see the advantage from refusal to separate concerns. This is ever more important when we consider a realistic work programme!

I suspect that the holistic approach will lead nowhere, apart from general statements about how complex, multifaceted etc. the process is.

What appears much more promising is to identify individual feedback loops in ESWP, in HSWP and in VSWP (there must be dozens in each; I doubt that in a year or two we could identify more than a few). But let's do it properly: identify the controller, estimate the gain (and its sign!) yielded by control action, try to plot the characteristics (as for old-fashioned electronic tubes) etc. I know we will not get a complete picture from it; from there onwards we will have to use a simulation of a sort, but before we even attempt to simulate we need some dependable calibration measurements so that simulation constants and parameters are chosen realistically!

Thus - while recognizing that all influences all - I advocate to use the, dare I say, scientific approach: separate concerns, isolate phenomena that can be measured, measure them and then recombine into a whole!

Cheers,

Wlad

o: wlad

From: mml@doc.ic.ac.uk (Manny Lehman)

Subject: A first reaction to your message

Cc: Dewayne

Bcc:

X-Attachments:

Dear Wlad,

Hope that your progres continues/has been masintained and that we shall welcome you here in two weeks time. Sorry I did not get this off on Thursday Friday or yesterday bu hope to do so today. To make things a little more readable I shall erase those parts of your message to which I am not responding today.

>No, the controller is not just a role assigned to a part: it is a part which has a well-defined >(known and measurable) influence on the other part (which may be - tentatively - named >"producer"). Of course, there is a degree of arbitrariness. In the steam-engin if we wish to >appear original we may consider the boiler a controller for the weights+plunger.

I am confused. Who determines whether a mechanism is or is not a controller? Do you wish to distinguish between a mechanism in the feedback path some (at least) of whose outputs are transmitted (avoiding the undefined term "fed back") to a mechanism/process (the "producer") in the evolution path (from hereonwards "forward path") in such a way that the information being manipulated or the process by which it is manipulated is modified. Contrasted with this would be information that is fed back in the same way but which the recipient is free to ignore. Would you then regard the feedback source in the latter case as an "informant" rather than a "controller"?. That would seem to me a legitimate distinction although the determination is still, as you say, somewhat arbitrary. Am I still missing something? As we have said so often some very clear definitions are

>arbitrariness can be probably eliminated by requesting that controller has no output other than >control exerted on producer, but I do not feel a particular urge to do so!

On the contrary, in electrical circuits, for example, in audio amplifiersto gain independent of time (changes in component properties) or frequency by providing feedback that is generally simply a fraction of the forward output from an amplifier stage. So we should certainly not adopt that view although I suppose one could consider the attenuator as the (passive) control element..

> > Yes but that was some 40 years ago and there has been a lot of progress since.

>>Well, not really

>We must distinguish real progress (where people can actually compute things) from noise >progress (very large systems etc). I repeatedly quote the example of socio-economic systems: >there is a very large number of books, papers etc.claiming a huge progress, yet in practice >every so often there is a revolution or a stock-market crash or a money-market upheaval or a >large company goes bust. Such events are a posteriori analyzed by the theorists who fit their >graphs to the historical events, strangely enough - there are precious few (like zero)>redictions in these socio-economic areas which would be verified by actual developments. >There is no other test that sharp: if a science cannot make predictions or if its predictions are not verified in practice that science is not very powerful!

In this matter I bow to your more informed judgment. as to whether alleged progress in control theory is real or phoney. We know of the claims that are made but, as you point out, there are some real and painful facts that suggest otherwise. The question in my mind is therefore if the apparent lack of uniform success in exploiting progress in control theory and system dynamics is absolute so that we must admit there has been no reel progress or whether the failures visible to all eyes are those of extreme situations where the theory is inadequate. The latter case would analogous to someone before 1905 (or unaware of Einstein) observing the behaviour of some very fast moving objects, observing deviations from behaviour predicted by Newtonian mechanics (admittedly not a deviation of the magnitude of a stock market crash or of the other instances you cite-but that might be because in feedback systems you have the "boundary of instability" problem and related chaotic behaviour) that act turn minor deviations into major disasters.. SShould such a person be tempted into abandoning Newtonian mechanics. We know that all Theories of the real world are approximations.

>Now on some older stuff.

>I continue to hold the view that any iteration (as opposed to mere repetition) is a feedback >phenomenon. Justification: Iteration is a process described by WHILE condition DO move >2244(or, alternatively, by DO move UNTIL condition)

>"move" and "condition" are the two parts in Ashby's definition of feedback, whose >performance (evaluation, elaboration) influence each other. It matters very little if the >condition is as simple as the check for the last element of a sequence (as in matrix >multiplication) or a more complex. Indeed, in the matrix multiplication the "move" must

>include index manipulation ("get next element"), which evidently influences the "condition". >As to the influence of the "condition" on "move" part, it is obvious: either the next move is >done or not! (In many iterations this influence is more sophisticated than this, as when

>the condition determines the step size for the move; most better diff.eqns. algorithms do it.)

I accept that iteration is feedback in the sense that the information fed back to the input of the forward path is "some test or other indicates that you have not finished yet though you are closer to finishing than you were on last reaching the test point so do this procedure again. This is an example of homing towards a goal, a step in the right direction. This iterative process can be linearised or unwound, the same process is repeated with a modified input.

I note by the way that I could imagine an iterative approach to program generation - in fact this is precisely what appears to be being proposed in some areas of AI. I generate a code sequence based on some degree of understanding of what is required and how the satisfactory result might be obtained (shut my eyes and take a plunge). I note the extent to which my result generates the solution I want and then iterate around making changes to the code until, some (considerable) time later and following several miracles I have what I believe to be a satisfactory - even correct?- solution to the problem. Now we clearly totally and absolutely reject this process not only because it is inelegant but because we do not (cannot) rely on miracles and we are certain that in all but the most trivial situations the process cannot produce a satisfactory solutions with known properties.

=>As to BACKTRACKING, the two parts are: a suite of forward moves followed by a >success/failure (or beauty contest) test whose elaboration causes some (or all) components of >the suite to be executed again with new/varied initial conditions; both the actual selection of >the forward moves to be repeated and their initial conditions are determined by the test. QED

No, its more complex than that though admittedly once again a matter of definition. In backtracking we are not following a homing procedure, a step by step approach, but a "we goofed" or "this is unsatisfactory", "do it again" approach. We may change the input to the procedure possibly by using a measure of the rejected output to modify the input as in the Newtonian solution to polynomial equations (though I would be inclined to call that iteration). But equally we may have to change the process by which we derive the output (or both of course). We may for example have detected an omission in the specification and given a modified spec. find it necessary to solve a different class of equations using quite different techniques

>I am afraid, we are not free to arbitrarily exclude some feedback phenomena from our >consideration merely because we prefer to call them something else!

I could not agree more and that is the last thing in the world that I wish to do. But we must be rather careful to understand and define the different characteristics that feedback may have and the consequences of those differences.

>

>Having said so, I must explain why I did mention that iteration and backtracking are instances >of feedback. My sole purpose was to indicate how enormously broad is the scope of feedback >phenomena in software making. Indeed, it is so broad that without some conscious narrowing >we would be talking about nearly everything (which means we would not say much about >anything).

>This brings me to the main criticism of the "minutes". I simply fail to see what do we gain by >abstracting ESWP, HSWP and VSWP to one process, go a step up and you get any process at >all (software making, egg-cooking, oil-refining, airplane controlling ...) Then, of course, the >only things to be said are such "helpful" observations as: "the process is characterized by a >complex conglomerate of interconnected feedback loops". So it is, but what follows? Needless >to say, I DO NOT CLAIM that the trio I proposed is perfect, far from it! But at least each of >the three ESWP, HSWP and VSWP is sufficiently DIFFERENT from the remaining two to >allow specific investigations!

>Of course, in practice, any software shop is engaged in all three at the same time. The "global >process" is a three-dimensional beast, but I fail to see the advantage from refusal to separate >concerns. This is ever more important when we consider a realistic work programme!

Of course, no question about that and your separation may well be valid. But to separate concerns you must be sure you understand the space being separated and be sure that you have covered the space under consideration. What our model does is to provide a single model that represents the space, the process being considered. Once we have that then it may well be that your partitioning is correct and justified. I, for one will have to do some more thinking even understanding more clearly what precisely are the differences between the three and what are the consequences of those differences. At the end of a long day and just before my Israel trip (22 - 29 August) I can't undertake that so this will probably be an item on our agenda two weeks from now.

>I suspect that the holistic approach will lead nowhere, apart from general statements about >how complex, multifaceted etc. the process is.

Agreed completely - have always recognised this and this certainly guided us in the generation of the "objectives"

>What appears much more promising is to identify individual feedback loops in ESWP, in >HSWP and in VSWP (there must be dozens in each; I doubt that in a year or two we could >identify more than a few). But let's do it properly: identify the controller, estimate the gain >(and its sign!) yielded by control action, try to plot the characteristics (as for old-fashioned >electronic tubes) etc. I know we will not get a complete picture from it; from there onwards >we will have to use a simulation of a sort, but before we even attempt to simulate we need

>>some dependable calibration measurements so that simulation constants and parameters are >chosen realistically!

Yes you are putting into very clear words what I have visualised all along so there is really no difference between us except that I am not (yet) 100% happy with your partitioning. I plan to rectify that over the next two weeks (in between all the other things I have to do).

>Thus - while recognizing that all influences all - I advocate to use the, dare I say, scientific >approach: separate concerns, isolate phenomena that can be measured, measure them and then >recombine into a whole!

I simply must stop at this point but hope to be well prepared for further exploration, discussion and refinement leading to a plan of investigation in two weeks time and I do hope you will be able to make it.

Regards and the best

Manny

Date: Wed, 17 Aug 94 08:56:54 +0200

From: Wladyslaw "M." Turski <wmt@mimuw.edu.pl>

To: mml@doc.ic.ac.uk

Cc: dep@research.att.com

In-Reply-To: Manny Lehman's message of Mon, 15 Aug 1994 16:23:55

Subject: A first reaction to your message

Dear Manny,

First, a decision: I am coming to London at the end of this month. (The recovery is far from complete and I still get quite

accommodation, despite its advantages. the walk across the College grounds is certainly more than I can hope to traverse. So it is the Embassy Hotel, if FEAST funds can stomach it.

Incidentally, when am I supposed to arrive: there is some confusion in your messages. The first night of the hotel reservation being 31st, I can report to work the soonest on the 1st, and other messages were mentioning the 30th as Day One.

Please advise on: exact and firm dates (I have to book the tickets) and hotel reservation (name of the hotel, its address).

Now on the substance. Hoping to have long conversations later in the month, just two brief observations:

I would strongly advise to avoid "antropomorphization" of control models we consider (even if in the reality they are "implemented" by people).

So a controller cannot "choose to ignore" any part of information it receives. It is made (defined) to accept a well-defined part (may be the whole) of the process-output, transform it according to well-defined algorithm into control-signal. The latter is a part of process-inputs and modifies the process transformation (process-input --> process-otput) in a well-defined manner.

Second, there is no specific feedback source apart from the "process", the whole idea of feedback is that output is fed back as input.

But we would argue this whe we meet.

Best regards,

Wlad

To: Wladyslaw "M." Turski <wmt@mimuw.edu.pl>

From: mml@doc.ic.ac.uk (Manny Lehman)

Subject: Re: A first reaction to your message

Cc: Dewayne

>Dear Manny,

>First, a decision: I am coming to London at the end of this month.

Hurrah - delighted

> (The recovery is far from complete and I still get quite unpleasant days, but on the average I >can walk up to 150 m at a time.) Unfortunately, I dare not risking the College accomodation, >despite its advantages: the walk across the College grounds is certainly more than I can hope >to traverse. So it is the >Embassy Hotel, if FEAST funds can stomach it.

OK Will cancel College and confirm Embassy which is at 31 Queen's Gate directly opposite 170 (where we had dinner last time)

>Incidentally, when am I supposed to arrive: there is some confusion inyour messages. The >first night of the hotel reservation being 31st, I can report to work the soonest on the 1st, and >other messages were mentioning the 30th as Day One.

Monday 29th is Bank Holiday and also day of my return from Israel. So we plan to start work on Tuesday morning the 30th. Not sure whether Dewayne is travelling overnight on the 28th or the 29th - or even by day on the 29th. Anyway 10AM on the 30th is start time

>Now on the substance. Hoping to have long conversations later in the month, just two brief >observations:

>I would strongly advise to avoid "anthropomorphization" of control models we consider (even >if in the reality they are "implemented" by people).

I accept that as entirely reasonable, necessary in fact. Clearly we have to be precise in all we say

>So a controller cannot "choose to ignore"any part of information it receives.

Of course. No I was being sloppy and imprecise. What I should have written was "if a mechanical controller has been 'programmed' or 'designed' to block or modify feedback subject to stated conditions" (no wish to rule out a decision based on random info/data) . However where the controller is a human, as will often be the case, the situation is somewhat different but I totally agree that a statement like "choose to ignore" is unacceptable

>It is made (defined) to accept a well-defined part (may be the whole) of the process-output, >transform it according to well-defined algorithm into control-signal. The latter is a part of >process-inputs and modifies the process transformation

But is not what we are likely to encounter in the software process. Do we simply throw up our arms in despair, do we seek statistical representations that are precise in the way you describe, eg. 40% of change requests are rejected and sent back to the requestor, 30% are put on a wait list and 30% are executed "as soon as possible - in the next release?", or what else?

>Second, there is no specific feedback source apart from the "process", the whole idea of >feedback is that output is fed back as input.

No I can't agree. In most real feedback systems there are intermediate as well as total feedback loops. For example audio amplifiers, the one I am most familiar with, one often flattens the frequency response of individual stages through negative feedback or even a combination of positive and negative feedback to boost the gain at the extremes of the desired range and ensure a sharp cut-off. Similarly the process may have many outputs, executable code, the code in execution, user documentation, technical documentation, sales reports, fault reports all of which may (indeed do) represent sources of feedback that have significant impact on the process (even on the technical process - for example many occurrences of qualitatively similar faults may indicate an inadequacy in the process - a method that is difficult to follow reliably and that we may wish to improve by partial automation or by changes to the procedural definition. In any event the source of feedback may be relevant

>But we would argue this when we meet.

Best regards,

>Wlad

Exactly and looking forward

Manny

Sun, 18 Sep 94 21:20 EDT
From: dep@research.att.com (Dewayne Perry)
To: mml@doc.ic.ac.uk, wmt@mimuw.edu.pl
Subject: repeat of manifesto

The Feast Project

This *Manifesto* lays the technical foundations for what we believe could be an interesting, worthwhile, multi-year funded project. Following from the results of the first workshop, it provides the basis from which, among other things, a project definition and workplan should be developed for a realistic and manageable startup project. The subsequent direction of the project is dependent on the initial results.

General Goals:

1. To produce specific recommendations, guidelines, methods and tools for software evolution process improvement.
2. To contribute to a science of software evolution.

Postulates:

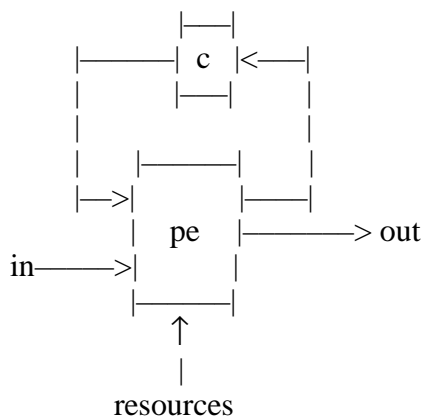
The Feast project is limited to consideration of process systems that govern the evolution of software systems and that satisfy the following postulates:

1. They have rich networks of feedback.
2. Some of the feedbacks stabilize characteristics in these systems.
3. Some feedbacks are controllable.

[**Note added by mml.** We believe that there exist practical software processes that satisfy these postulates. It will be an objective of the FEAST project to identify and model such processes or to demonstrate that they do not (cannot?) hold in practice. It will also be of interest to explore how they might be changed or supplemented to increase their applicability and/or value. **End of note**]

Process Unit Model:

Consider a process element (pe) which applies resources to transform inputs into outputs. If one of the destinations of the output is a controller (c), where output is fed back into the process element, we obtain a general controlled feedback loop (as in the diagram below). We term this total configuration a process unit (pu). Process elements can contain process units. As a first approach we assume that control units (Cs) are themselves not, and do not contain, p_{us}, ie are not, themselves, controlled by feedback



Hypothesis:

A process that satisfies the postulates above can be usefully decomposed into a manageable number of process units.

End of Manifesto

Date: Sun, 18 Sep 94 21:19 EDT
From: dep@research.att.com (Dewayne Perry)
To: mml@doc.ic.ac.uk, wmt@mimuw.edu.pl
Subject: summary of discussions

[**Note by mml.** This is the minute as recorded and written up by Dewayne. In what follows I have not

Summary of Discussions (31 August - 6 September -- Manny, Wlad, Dewayne)

A. Controlled feedback effects

In controlled feedback loops, control may have the following effects on the controlled process element (note that we have used physical systems feedback and control as metaphors for software systems):

1. change the processing (eg, increase the steam); and
2. change the process (eg, use alternate plumbing) -- we distinguish two cases in changing the process:
 - a. statically (eg, use alternative routes), and
 - b. dynamically (eg, create new routes). [**Note by mml.** I believe this includes changes to the characteristics of existing route. **End of note**]

In some sense, categorizing a control effect is more a matter of viewpoint. For example, one could view changes in the process (ie, using alternative routing) as forcing one of several alternatives in processing rather than changing the process. Similarly, one could view changing the process dynamically as though one had previously created what would now be one of the current set of static alternatives.

B. Balanced interchange rate at interface

There are interfaces between what we have termed agencies (large-scale processes such as users, marketing, project management, corporate management, etc). These systems tend to stabilize on an interval that balances change and the ability to learn or understand (that is, it is governed by the law of conservation of familiarity). In work such as evolving software, a new tool, language, process etc is not effective until it can be used without thinking about it. The greater the incremental changes, the longer it takes to stabilize -- that is, the greater the changes, the longer the rate of absorption. Either stability will be achieved or the the enterprise will "wind down".

When the flow across the interface is balanced, the system is balanced; imbalance causes instability in the system behavior.

Manny's study of OS360 is a case in point. He observed the rate of change and the resulting resistance, plotted it, and inferred the nature of this phenomenon.

Here we might have some practical advise for managers: balance the rate at the interface and determine the actual character of that balance; look at what feeds the interface to see if there is any safety margin to enable speed up or if you have to reduce the rate of interchange.

Thus, one approach to process improvement could be to look at the interface assessing the flow of information in both directions and then change the characteristic of the process by changing that flow in either or both directions,

Another practical approach is to explore how much we can change the characteristics and still remain balanced. Or, in an unstable system, how much change can be made, or must be made, to achieve balance. [**Note by mml.** I suspect that this section may be meaningless to those not at our discussion, it is to me at this point. Needs clarification. **End of note**].

C. Feedback and Control

Stabilization comes from control. Disintegration eventually follows from unresolved instability.

We are interested in those feedbacks that are both stabilizing and controllable, their kind, nature, etc., and how they can be used to improve processes. Thus we want 1) to identify and characterize them and 2) to determine how to fiddle with them to gain desired effects.

What are the *ad hoc* characteristics that we have stabilized on? What are the desired characteristics that we ought to stabilize on?

They may well be organizational rather than technical. For example, it might be something like the preservation of the 'health of an **organization**' that motivates provision of system resources

How do we distinguish 'just what happens' (ie, random forces) from controlled feedback loops in which self-developed characteristics can be improved by conscious application of control? Are feedbacks controllable independently? Are there interdependent feedbacks or interdependent feedback controls?

Note that fiddling the control when the system is not active has no effect. Note further that there is no feedback

D. Unsuccessful hypotheses

Hypothesis:

There exists a level of description of the network feedbacks at which the coupling among the feedback mechanisms is sufficiently weak that one may consider the affect of changes to a single feedback mechanism independently of the others.

Assumption:

The higher the level of control, the more the effect. If we fiddle the local control, we expect to get a local effect - but it is difficult to determine the global effect. Global control is more closely related to global characteristics.

Discussion:

Do the higher level control mechanisms propagate to the lower level control mechanisms? No matter how you decompose the feedback and control the strength and feedback is identical independent of the level of description. It may be decomposed inside the process element, but the strength of the feedback remains the same entering the element independent of how many levels it is decomposed into.

The basic problem with the conjecture is that abstraction does not remove the strength of the connection.

Revised hypothesis:

At some level of abstraction, if they appear to be independent, then as a first order approximation, we can consider them independent.

Alternative hypothesis:

With experimentation on the entire system and subsequent knowledge about the system, we can approximate independence of feedback control as predictors of system behavior.

The problem with this formulation is that while it is obviously true it does not help us much.

At this point in the discussion, we asked where are the feedback loops and what are the feedback controllers, the transformers. We can arrive at our answers by experimentation (expensive, but must be done first), analysis and simulation (which needs calibration to be believable). But consider a single transformer: vary all the inputs and measure the outputs; this provides us with an empirical validation of relations, an approximation of behavior. At this point we arrived at the model we defined in the manifesto.

E. Applying the Process Unit model to the general process picture

We decided to apply our model to our global process picture as a check on both our model and our understanding of it. The basic problem in this picture is to distinguish feedback control from input/output. Much of what we have in the picture is just information being passed around.

There seems to be confusion about 'control' and 'the right to modify'. The latter is just part of production; the former is modification of the operation. The 'right to fiddle' is not control; the 'impulse to control' is control.

E.1. The first pass

As an example consider approval of a workplan:

Collecting signatures is part of work/production

Control is looking at the predicate governing approval

Comments and rework are production, not control

Production: produce workplan and collect signatures

Control: signatures sufficient, yes or no?

E.2. The second pass

The problem with this first pass is that in our model one of the destinations of the output is the controller. In the case of the workplan, it will not be output until it is approved. Thus, approval must be part of work and determining whether the signatures are sufficient is also part of work.

Thus in the revised example:

do workplan until approved' is a production loop

Control observes various qualities in the output (ie in the workplans) or the length of time it takes to produce it and controls the stream of workplans.

[note by MML in editing, which Dewayne and Wlad have not yet commented on] In re-reading this I question

separate output and not part of the main output. Our earlier observation as recorded above simply may be demonstrating that our 'picture' is correct in suggesting that the feedback output is not to be regarded as a processing unit output and can be (is) made available to the controller to receive the intended output and so to determine whether that output may be made generally available. This comment is compatible with Wlad's email comment after the meeting:

>As I told Dewayne on the way to the airport, further thinking about the "approval loop" led me to >mildly interesting observations in relation to the "approval" example:

>1.The loop

> DO assignment UNTIL approved-by-whomever

>is a typical feedback loop in the production_element. The "whomever" may think himself a >controller, but he is very much in the production line!

And mml response: Yes of course and more than mildly interesting since it is surely a prototype of many situation in which what appears a feedback control is (can be **represented** as being) in the forward path. On the other hand such approval control can be a major holder up on progress such that an improvement in the forward path that, for the sake of argument removes or reduces the need for approval but that is not reviewed when the improvement is introduced could explain the less then expected benefit. So viewing it as a feedback control would draw attention to the need to review. So what you have drawn attention to is the ambiguity that exists in the identification of feedback paths and that the mode of representation and "total process system" review refers to just that, ie total system, without regard to the representation model. End of mml comment]

F. Implications of the Process Unit Model

There is still a matter of delay and amplification, but here control becomes a predicate on some subset of the output. We now have something that is more like classical feedback control: complex predicates that indicate stop or go, continuous functions (eg, rate of work), and discrete alternatives.

Now we don't need to distinguish between forwards and backward feed because it is just information flow around the network. that is, work is separated from control. We still have complex interdependent information flows, but not interdependent feedback control loops.

Controllers now appear to be independent of agencies. However, this is not necessarily the case. People in these agencies may act as controllers of both the feedback as well as of sources and resources. While we intuitively regard project management as having control of the project - as providing feedback -- we must define the control as part of the process unit, regardless of which agency the controller comes.

This seems to make some of the lines of organizational or agency control implicit rather than explicit.

At this point we have cleaned things up from a theory point of view. We have a clean model, formalizable and we may even be able to find mathematics for it. The problem now is to relate the model to real organizations and processes. Is such a relation possible? Does it give a realistic representation of such processes and organizations?

G. Possible extensions to the PU model

We may find in the end that our model is too simple. the model might be extended in at least the following ways:

1. allow controllers to be decomposed into process units;
2. allow controllers to be changed as part of feedback control; and
3. allow input to controllers (where currently only the process elements have input).

mml[FEAST-F5]

23/11/94

Date: Mon, 26 Sep 94 17:52:21 BST

Re: Manifesto and minutes

I have had an initial read through the manifesto document, particularly the first page (though the write up on the discussions was interesting).

I have a few comments now as follows.

I guess my main concern is that I gain the impression that the subject being studied is the software process. That's ok, (after all what we are trying to do is to make it more effective in the light of some better understanding), BUT I don't believe that this can be done in anything like isolation. My view is that the problem space of end systems (by which I mean the larger system of a business, an aircraft, a government department) which will change in unknown ways (unknown to the software engineer) must be taken together with the space of evolving, embedded software systems, whose presence is of course a factor for change in the systems in which it is embedded and therefore a factor in its own need for evolution. These relationships say to me that it will be necessary to understand the nature of these various systems and their inter-workings rather than try to study the evolutionary nature of the software process itself. Not to do so may well be rather putting the cart before the horse. My hope (!) is that we may well discover something of quite some significant value to the software SYSTEMS industry. So I feel that the subject matter must be drawn a bit wider.

My more detailed observations are derived from this point of view and are:

on General Goals:

How do I parse goal #1? Presumably as "... (software evolution) process improvement. What is the intended meaning? Possibly I guess "the improvement of the process of software evolution". Not sure I really understand what that means either.

It seems to me that goal #1 ought to be something like :

1. To produce specific recommendations ... to allow the effective evolutionary development of software systems in volatile application environments.

That's not right, but I would want to capture both the fact that software systems evolve AND that the problem space into which they fit does so also. The former systems are the subject of our constructive endeavours, the latter we must understand but cannot control.

on Postulates

I think that there are significant concerns with the first sentence. Any limitation must be such that the set of problems within its scope has to be interesting and representative of a wide range of problems in the "real world". I suspect that it might be difficult to formulate such a limitations except by a proper formulation of the nature of the "E-type problem" - which is what my opening paragraph was all about.

I like the use of the word "some" in 2 and 3! Does this definition include 0? or is it strictly >0? on Process Unit Model -----

I haven't thought much about this, but with such an element how are feedbacks/controls themselves controlled? Can a "c" is one element provide a control signal to another "pe"? Don't think so. I suspect therefore that using this model you will end up with lots of "control messaging" through the "pe"s (has to be because that's the only inter element communication). So, although the hypothesis may be true, I suggest that something rather richer will be necessary. (And I guess

you have already started to raise such thought by the last point in the discussion.)

I'd like to feel that I could model the systems and inter-relationships I noted in my introductory paragraph using such a building block (whatever it is), and thence be able to derive useful constructive properties. This, of course, could well be the essence of the FEAST project, so putting up a single candidate may be premature!

Anyway, these are just initial thoughts/observations, probably completely off target!

Bob.

To: ras@kid01pml.icl.co.uk

From: mml@doc.ic.ac.uk (Manny Lehman)

Re: Manifesto and minutes

Cc: wlad,dewayne

Thanks for your response which was awaiting me when I returned to day. Herewith a quick reaction though from my initial reading I believe that we share a common viewpoint.

>I guess my main concern is that I gain the impression that the subject being studied is the software process. That's ok, (after all what we are trying to do is to make it more effective in the light of some better understanding), BUT I don't believe that this can be done in anything like isolation.

OF COURSE NOT. That's my view too and that surely is, as you say, what the project is about in the long run. I would express it as "focussing on the software process, the evolution (improvement) of it and its products IN THE CONTEXT OF all the organisations (in the plural) and organisational objects and processes that impact (are involved or interact with) the software process.". Now clearly to study such a very complex system we must, IN THE FIRST INSTANCE, engage in some limitation exercise, abstracting, structuring, simplifying, and so on to achieve a manageable object of study which we enlarge, extend and detail as insight and understanding increases or as interactions are perceived and the need to extend the model is understood. This is precisely the point which I was trying to make (apparently badly, unclearly) in saying that past efforts at process improvement have concentrated on the forward software technology path and have tended to ignore (or not recognise) the interactions, involvements and feedback from other (what I now term) AGENCIES that must surely have a profound influence on the changes and direction of changes in the process. It is in the spirit of that view that I generated the "sausage" model which Wlad, Dewayne and I discussed at some length but which has not yet been totally reconciled with the Manifesto postulates. In view of your comments I will be adding a couple of "boxes" to make explicit what was implicit in the original model. I hope to prepare a picture of this model before I leave for the States on Monday and if I succeed will send you a copy, though without too much explanation at this time.

>My view is that the problem space of end systems (by which I mean the larger system of a business, an aircraft, a government department) which will change in unknown ways (unknown to the software engineer) must be taken together with the space of evolving, embedded software systems, whose presence is of course a factor for change in the systems in which it is embedded and therefore a factor in its own need for evolution. Yes that is correct and implicit in my comment that every E-type program contains a model of itself, the root source of evolution, installing the program changes the operational domain, that is its processes and hence its objects and these changes drive one to recognise the need and eventually change the software which being part of the operational domain changes that domain etc. These relationships say to me that it will be necessary to understand the nature of these various systems and their inter-workings rather than try to study the evolutionary nature of the software process itself. Not to do so may well be rather putting the cart before the horse. My hope (!) is that we may well discover something of quite some significant value to the software SYSTEMS industry. So I feel that the subject matter must be drawn a bit wider.

No question about that but controlling the growth in size and complexity of our model and what it encompasses is going to be a critical issue. Given the feedback model suggests that the coupling too and dependence on external objects and processes is via a somewhat longer delay loop (whether forward or feedback) and hence, while such influences must be borne in mind from the start and regularly reviewed, their full incorporation in the overall model can be at a later stage of the project.

>My more detailed observations are derived from this point of view and are:

>on General Goals: -----

>How do I parse goal #1? Presumably as " ... (software evolution) process improvement. What is the intended meaning? Possibly I guess "improvement of the process of software evolution".

I think that is precisely what we meant and provided you understand that evolution includes both development and maintenance in their conventional senses I think that's clear enough.

1. To produce specific recommendations ... to allow the effective evolutionary development of software systems in volatile application environments. That's not right, but I would want to capture both the fact that software systems evolve AND that the problem space into which they fit does so also. The former systems are the subject of our constructive endeavours, the latter we must understand but cannot control.

It will be clear from my remarks above that I sympathise with that while postulating a need for care in expanding the boundaries of the project too fast. Needs more thought. These issues will presumably come up in the Workshop session that discusses the Manifesto.

>on Postulates -----

>I think that there are significant concerns with the first sentence. Any limitation must be such that the set of problems within its scope has to be interesting and representative of a wide range of problems in the "real world". I suspect that it might be difficult to formulate such a limitations except by a proper formulation of the nature of the "E-type problem" - which is what my opening paragraph was all about.

Yes. and I will be making a first stab at this by presnting a discussion on the characteristics of S and E type systems at the opening session which will, I trust, lead to an illuminative and fruitful discussion.

>I like the use of the word "some" in 2 and 3! Does definition include 0? or is it strictly >0?

Clearly it must include the zero case though for systems for which both 2 and 3 are zero my "feedback hunch" is, presumably not valid.

on Process Unit Model -----

>I haven't thought much about this, but with such an element how are feedbacks/controls themselves controlled? Can a "c" is one element provide a control signal to another "pe"? Don't think so.

At the moment NO. But I believe we will have to find a way of relaxing this condition. Once again we start of with the simplest situation and introduce variation/complexity as this is shown to be necessary.

>I suspect therefore that using this model you will end up with lots of "control messaging" through the "pe"s (has to be because that's the only inter element communication). So, although the hypothesis may be true, I suggest that something rather richer will be necessary. (And I guess you have already started to raise such thought by the last point in the discussion.)

I agree and so, probably, will Wlad and Dewayne but this has to be done in controlled fashion.

>I'd like to feel that I could model the systems and inter-relationships I noted in my >introductory paragraph using such a building block (whatever it is), and thence be able to >derive useful constructive properties. This, of course, could well be the essence of the FEAST project, so putting up a single candidate may be premature!

That's the nub of the problem. But the sensible (scientific?) approach is to start with the simplest and most elegant solution and to introduce variations/complexities as these are shown to be necessary or rather as it becomes clear prevents one from finding an acceptable solution

>Anyway, these are just initial thoughts/observations, probably completely off target!

Not at all, simply opening shots in what should prove a productive and fruitful interchange which I hope and trust will get us places. My response too is off the cuff since I leave for the States on Monday (to day is Thursday) and I still have a lot to get ready.

So thanks.

Manny