Speaker – John R. Mashey

- Pennsylvania State University, 1964-1973, BS Math, MS/PhD Computer Science

New Jersey

- Bell Labs 1973-1983, MTS → Supervisor, early UNIX
  - Programmer's Workbench, shell programming, text processing, workload measurement/tuning in first UNIX computer center, UNIX+mainframe data mining apps, capacity planning/tuning

Silicon Valley

- Convergent Technologies 1983-1984, MTS → Director Software
  - Compiler & OS tuning, uniprocessor/multiprocessor servers
- MIPS Computer Systems 1985-1992, Mgr. OS → VP Systems Technology
  - System coprocessor, TLB, interrupt-handling; byte addressing(!), halfword instructions; ISA evolution, multiprocessor features, multi-page-size TLB, 64-bit
  - MIPS Performance Brief editor; a SPEC benchmarking group founder 1988 (GM)
  - Hot Chips Conference (Stanford) committee ... continuing
- Silicon Graphics 1992-2000, Dir. Systems Tech → VP & Chief Scientist
  - MIPS R10000 & later architecture, including performance counters & software
  - ccNUMA system architecture (NUMAflex in Origin3000, Altix)
  - Performance issues in HPC, DBMS; technology forecasting
  - Evangelist, much work with sales and marketing, business development, strategy
- Advise/consult for Venture Capitalists & high-tech companies

  Technical advisory boards (Dust Networks, Streetline Networks, Transitive, …); Computer History Museum (www.computerhistory.org) Trustee; VCTaskForce
  Travel; ski; hike; occasionally write articles & do talks for fun … i.e., ~ half-retired ☺

Small is Beautiful

And Other Thoughts on Programming Strategies (1977-)

For the 2002 BSDcon, I grabbed talks from 30 years ago, and used (images of) the original foils for authenticity, to help see what's changed and what's the same.*
The first part, "Small is Beautiful and Other Thoughts on Programming Strategies," was first used in 1977, and was later given many times as Association for Computing Machinery (ACM) National Lectures.

I was working on the Programmer's Workbench flavor of UNIX, and we'd had great success in making UNIX available to much wider audiences of software engineers targeting both UNIX-related and non-UNIX environments.
We were strong believers in UNIX philosophies of tool-building and -using, and keeping software teams small during an era when there was strong emphasis on methodologies and large teams that were anything but lightweight. This talk was the result, and was considered somewhat radical at the time.

➔ Scripting languages, development environments, “agile development”

* I still have the original foils, but they’re starting to wear out, and actually, old overhead projectors have started to disappear in favor of computers….
Originals were UNIX *troff* + hand-drawn graphics … not PowerPoint!
From: http://www.usenix.org/events/bsdcon/mashey_small, Thanks USENIX!

Dr John R. Mashey

Title

SEQUENCE NO. 1

TOP

DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

NOTES:

WHAT I'D LIKE TO TALK ABOUT:
- APPROACH THAT:
  APPLICABLE
  SOLVE PROBLEMS
  EASY TO APPLY

WHAT I WILL
PROGRAMMING PHILOSOPHY
APPLIES TO SOME
HELP SOLVE
WILL BE HARD
DO SAME WAY AS
S.P. VS. GOTOS
NO WORKED

NOT PANACEA
NOT RAGIL METHODOLOGY
WITH LOTS OF ACRONYMS

GENERALLY INDIVIDUAL,
NOT SPECIFIC:
SOME EXAMPLES
MOST DO BIBLIOGRAPHY
WILL HAVE VU-GRAPHS
questions, etc. feedback + a –
EXPERIENCED
Bell Laboratories
↓ unsimplicity of things, or easy
– not expert, but I've done them all

Small is Beautiful

and other thoughts on

Programming Strategies

J. R. Mashey

Iteration 5 - IH - 11/15/77

Code + design

00

Introduction

SEQUENCE NO.

TOP

DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

NOTES:

NOW I'LL TELL WHAT I'M LOOKING TO TELL.

INTRODUCTION

APPROACHES

- "Do it right"
- "Do it over"
- "Do it small, with tools"

OBSERVATIONS

["Why things are bad; how they get that way"]

- Success vs. Failure ["Nothing works"]
- Size of projects ["Small is beautiful"]
- Evolution and Entropy ["Everything falls apart"]

STRATEGIES AND TACTICS

["Keeping good; stopping bad from getting worse"]

RAYS OF HOPE

["A few candles in the tunnel"]

Bell Laboratories

D 7351 (5-77)

Approaches

NOTES:

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

APPROACHES

KINDS OF PROJECTS

- Development and support
- NOT research, although often should be

CONTINUUM OF APPROACHES

- Analyze and plan
- Design
- Implement
- Test
- Maintain [$$]

APPROACH CLASSIFICATION

- Emphasis and priority
- Timing [serial, parallel, duration]
- Risk

Bell Laboratories

E-7351 (5-77)

“Do It Right”

NOTES:

SYSTEMS ANALYSIS

SERIAL
LOW RISK

SUB-TEAM

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

“DO IT RIGHT”

- Emphasis on requirements analysis
- More effort at front end
- Lessen costs by fixing errors early
- Optimism — we know what we are doing

“The most deadly thing in software is the concept, which almost universally seems to be followed, that you are going to specify what you are going to do, and then do it. And that is where most of our troubles come from.”
— D. T. Ross, in [NAU69A]

“Hence plan to throw one away; you will, anyhow.”
F. P. Brooks, Jr. (architect of OS/360) in [BRO75A]

Bell Laboratories

VG. NO. 11

E-7351 (5-77)

"Do It Over"

NOTES:

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

"There are basically five approaches to validating requirements. Listing them in the order of increasing fidelity and cost to implement, they are: estimation, off-the-shelf simulation, new simulation, prototype development, and do-it-twice."
— W. W. Royce, in [HOR75A]

HARDWARE — BREADBOARD

"DO IT OVER"

- Less emphasis on early analysis
- More emphasis on early implementation
- Pessimism
- Assumptions on cost
- Still starts from scratch

Bell Laboratories

VG. NO. 12

E-7351 (5-77)

Do It Small, with Tools

NOTES:

WHO HAS A CATCHY ACRONYM?

A THEOSIS IN ZO USE IF YOU CAN

TOP

DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

DO IT SMALL, WITH TOOLS

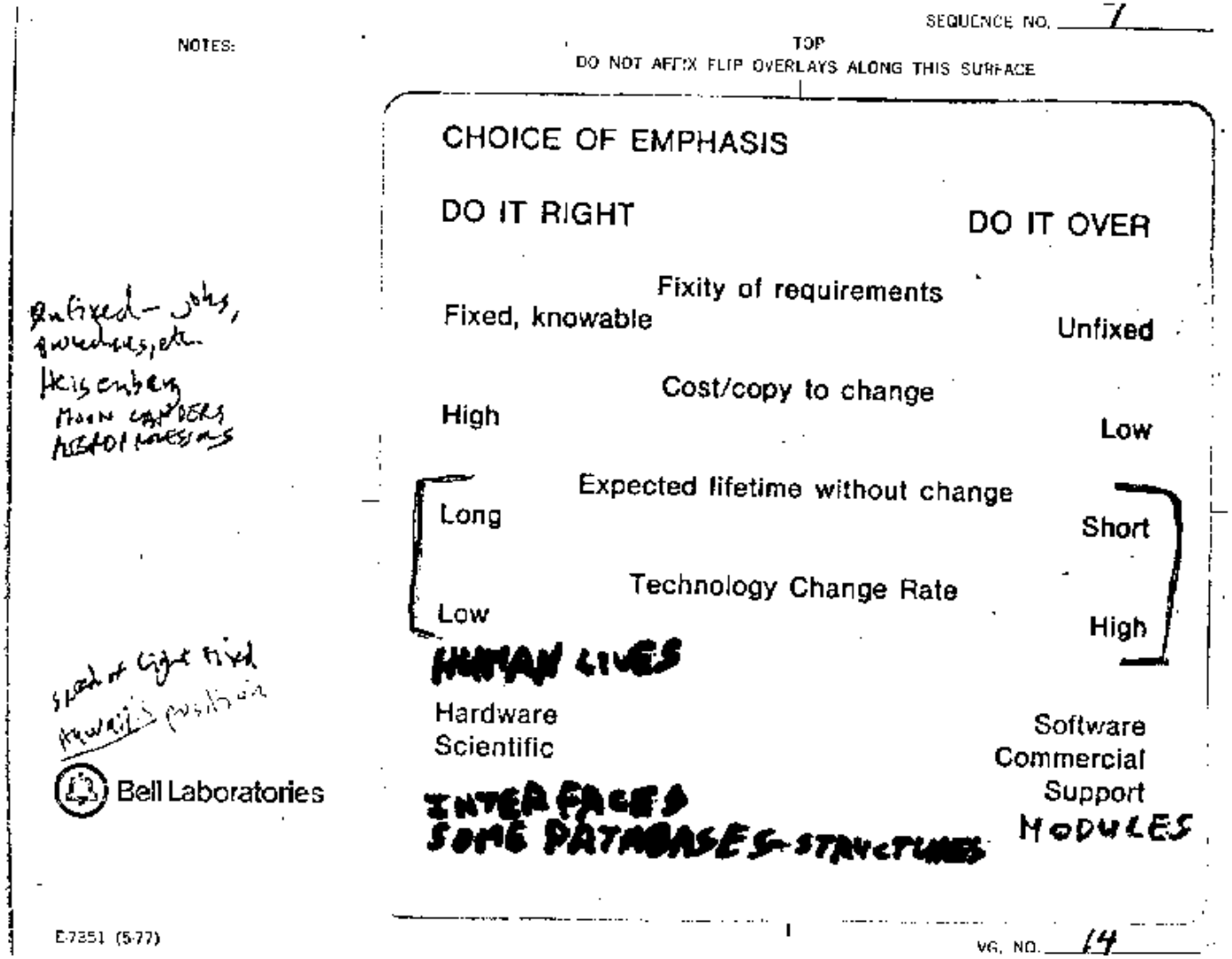
- Pick appropriate emphasis, timing, risk assessment
- Focus pre-planning
  - Most decisions are not very important
  - A few decisions are very important
  - Avoid unrecoverable states
- Build something fast, fail quickly if necessary
- Think TOOLS instead of SYSTEMS
- Think small rather than GRANDIOSE

Bell Laboratories

E-7351 (5-77)

VG. NO.

Overview


SEQUENCE NO. 7
NOTES:
TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE
CHOICE OF EMPHASIS
DO IT RIGHT
DO IT OVER
Fixity of requirements
Fixed, knowable
Unfixed
Cost/copy to change
High
Low
Expected lifetime without change
Long
Short
Technology Change Rate
Low
High
HUMAN LIVES
Hardware
Scientific
Software
Commercial
Support
INTERFACES
SOME DATABASES-STRUCTURES
MODULES
Bell Laboratories
E-7351 (5-77)
VG. NO. 14

Success vs Failure

NOTES:

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

SUCCESS vs FAILURE

OUTRIGHT SUCCESS

"Alive and healthy"

OUTRIGHT FAILURE

"Dead and buried"

PARTIAL FAILURE

"Sick, may recover"

"Living dead (the dreaded zombie)"

"It's awful, but it's very powerful once you learn everything about it."

"We realize that it's huge, expensive, and obsolete, and we'd clean it up, but we're scared to death of touching it, and besides, we're not sure where the source code is."

"We wish it would go away and leave us alone, but our whole operation depends on it."

Bell Laboratories

VG. NO. 20

Qualitative Metrics, a later addition

CUSTOMER NETWORK OPERATIONS

QUALITATIVE METRICS

- SUCCESS -

  WE, THE INVENTORS:

  - TELL YOU HOW GREAT IT WILL BE
  - HAVE BUILT ONE
  - ACTUALLY USE IT
  - AND SO DO OUR CLOSE FRIENDS
  - PLUS OTHERS, NOT SO CLOSE
  - ARE OVERRUN BY STRANGERS

Bell Laboratories

Data Processing

SEQUENCE NO. ____

NOTES:

TOP

DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

DATA PROCESSING

"Advanced Projects in Data Processing"
[EDP71A]

- Survey of projects
  - 18 projects total
  - 5 infeasible
  - 5 feasible, but no acceptance
  - 2 good fall-outs, no return
  - 3 partial success, return (?)
  - 3 success
- Look at 4 very successful projects
  - All < 1 year elapsed time, < 5 staff-years
  - None were accepted immediately

"We took as "successful" a project that met its requirements on schedule within the budgeted dollars and satisfied the customer. On this basis, out of 10 or 12 projects that we examined, we had one success and a whole lot of failures."
— J. Aron (IBM)

Bell Laboratories

BUX70A
[SUA. Eng]

VG. NO. 21

More D. P. (GOVT)

NOTES:

TOP
DO NOT AFFIX OVERLAYS ALONG THIS SURFACE

MORE D.P. (GOVT)

- $ OVERRUNS (>50%)
- CALENDAR OVERRUNS (>60%)
- 9 CONTRACTS ($6.8M)
  - $3.2M Delivered, never used
  - $1.9M Paid, never delivered
  - $1.3M Extensive rework
  - $.198M Used after changes
  - $.119M Used as delivered
- PRODUCTION -> PROTOTYPE
- COST TO FIX > = COST TO DEVELOP

GAO FGMSD-80-4, Nov. 9, 1979
COMPUTERWORLD Sept. 29, 1980

Bell Laboratories

J. R. MASHEY

VG. NO. 21A

E-7351 (5-77)

Other Areas of R&D and Outside R&D

NOTES:

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

William Joyce
BTL 20

OTHER AREAS OF R&D

"Organization of Unsuccessful R&D Projects" [JOY71A]

"In this sense [successful products] nearly every proposal considered is an eventual failure."

- Of 400-500 ideas, 1-2 work out
- 4/5 R&D hours spent on projects that do not reach commercial success

– AND OUTSIDE R&D...

Museum of Modern Art and designs of "lasting value"

1934	397	1 survived
1939	70	1 saw further development
1950	46	1 survived

"...a score of 3 successes and 510 misses is far from reassuring."
— V. Papanek, in [PAP73A]

FAILURE IS THE NORM

Bell Laboratories

E7351 (5-77)

VG. NO. 22

Ways in Which Projects Fail

SEQUENCE NO. _____

TOP

DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

NOTES:

WAYS IN WHICH PROJECTS FAIL

- Wrong problem
- Wrong approach
- Infeasible performance
- Cannot be finished at all
- Becomes irrelevant or inappropriate when done
- Lack of user acceptance
- Leapfrogged by somebody moving faster

"Don't look back. Something may be gaining on you."
— Satchel Paige

Bell Laboratories

F-7351 (5-77)

VG. NO. 23

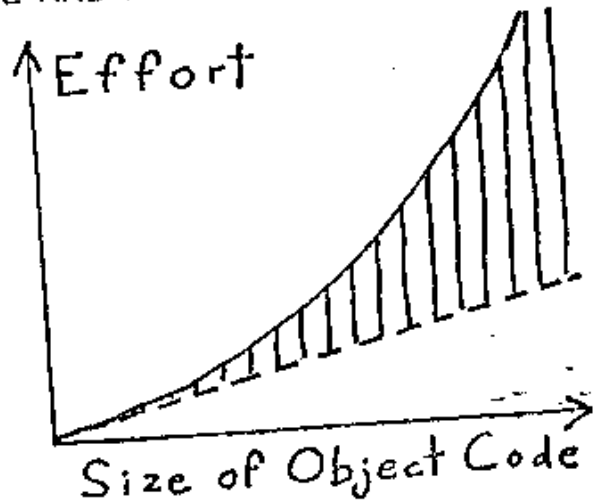
Aspects of Size

NOTES:

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

ASPECTS OF SIZE

SIZE AND PRODUCTIVITY



effort = c x (object size)$^{1.5}$ [BRO75A]

Bell Laboratories

VG. NO. 24

E7351 (5 77)

End-Product vs “Overhead”

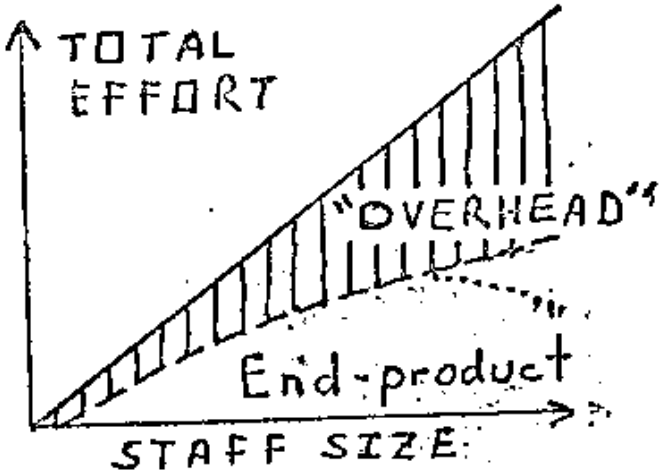
SEQUENCE NO.

NOTES

TOP

DO NOT ATTAX FLIP OVERLAYS ALONG THIS SURFACE

END-PRODUCT vs “OVERHEAD”



- Brooks’s Law —
  “Adding manpower to a late project makes it later.”
- Corollary —
  “Long project + staff turnover = steady-state zombie”

Bell Laboratories

E-7351 (5-77)

VG. NO. 25

Size of Support Methodology

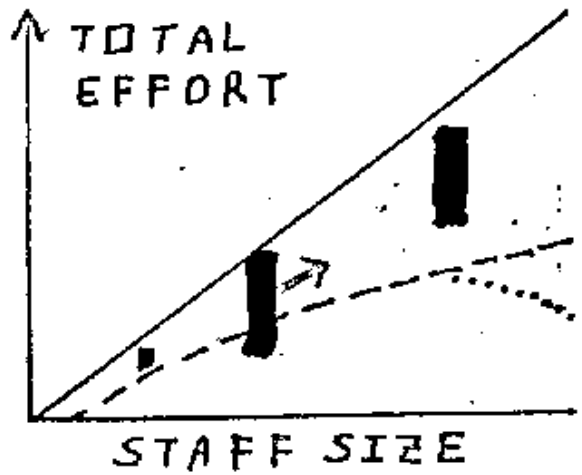
NOTES:

GOOD TEST:
CAN A PERSON USE IT.
NOT ALWAYS READY; BUT HOPE TO THINK ABOUT

DON'T GET BOGED UP WITH TOOLS
- Giant operating systems or languages no one understands

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

SIZE OF SUPPORT METHODOLOGY



- Cost to build, maintain, learn, and use
- Big support => big project
- Product ≃ organization ≃ support
- "Dive into support" phenomenon

Bell Laboratories
PROGRAM ADMINISTRATION SYSTEMS (PAS)
CELL DIVISION

E7551 (5-77)

VG. NO. 26

Evolution and Entropy

NOTES:

2nd Law of Thermo

LATER – NOT AVOIDING ENTROPY [IMPOSSIBLE] BUT KEEPING IT NO WORSE THAN NEEDED

TOP

DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

EVOLUTION AND ENTROPY

"In all cases observed in nature there is a tendency for processes to proceed toward a state of greater disorder."

PROGRAM EVOLUTION — Belady and Lehman

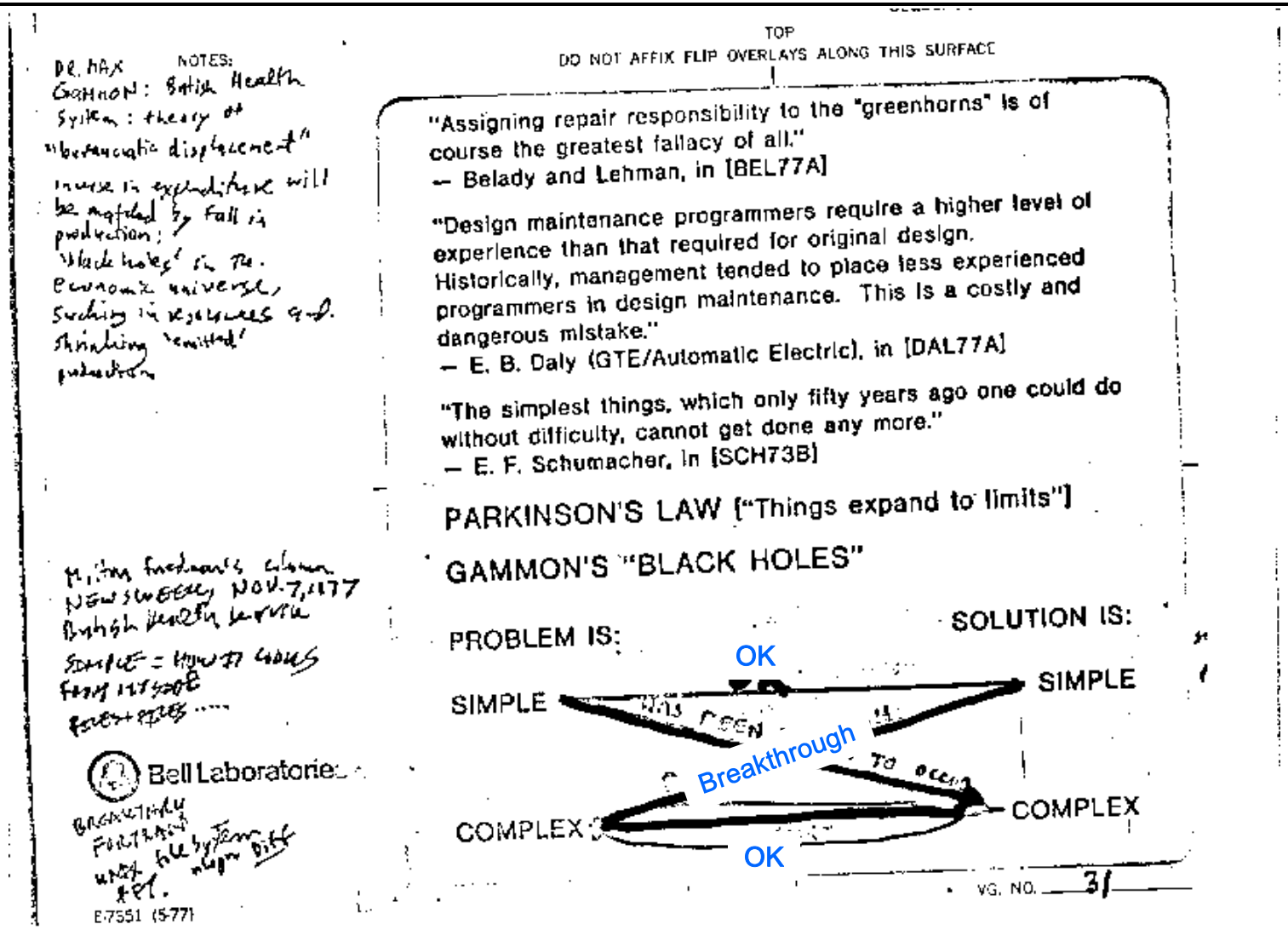
- Law of continuing change — a system that is used undergoes continuing change until it is judged more cost effective to freeze and recreate it.
  - Variety generated by the desire to perfect — continuing enhancement
  - Variety generated by imperfection — continuing maintenance
  - The result of continued evolution — structural complexity
- Law of increasing unstructuredness (entropy) — the entropy of a system increases with time, unless specific work is executed to maintain or reduce it.

Bell Laboratories

SPECIFIC WORK = SKILL + EXPERIENCE

VG. NO. 70

How Things Get Complex



NOTES:

TOP

DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

"Assigning repair responsibility to the "greenhorns" is of course the greatest fallacy of all."
— Belady and Lehman, in [BEL77A]

"Design maintenance programmers require a higher level of experience than that required for original design. Historically, management tended to place less experienced programmers in design maintenance. This is a costly and dangerous mistake."
— E. B. Daly (GTE/Automatic Electric), in [DAL77A]

"The simplest things, which only fifty years ago one could do without difficulty, cannot get done any more."
— E. F. Schumacher, in [SCH73B]

PARKINSON'S LAW ["Things expand to limits"]

GAMMON'S "BLACK HOLES"

PROBLEM IS: SOLUTION IS:

SIMPLE → SIMPLE

COMPLEX → COMPLEX

Bell Laboratories

E-7551 (5-77)

VG. NO. 31

Local Scenarios

SEQUENCE NO. k

NOTES:

TOP

DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

LOCAL SCENARIOS

Producer

"Here's a wonderful new feature!"

"Somebody might want this someday."

Consumer

"I want/need this feature yesterday."

"Don't tell me it's better; I need 100% upward compatibility forever."

"Fix this MR now." (MR myopia vs MR as symptom)

Producer/Consumer Assymmetry

"Why do you make me use this complex software with a zillion options and weird syntax? I always write simple things with reasonable features [like that nifty one I added yesterday.]"

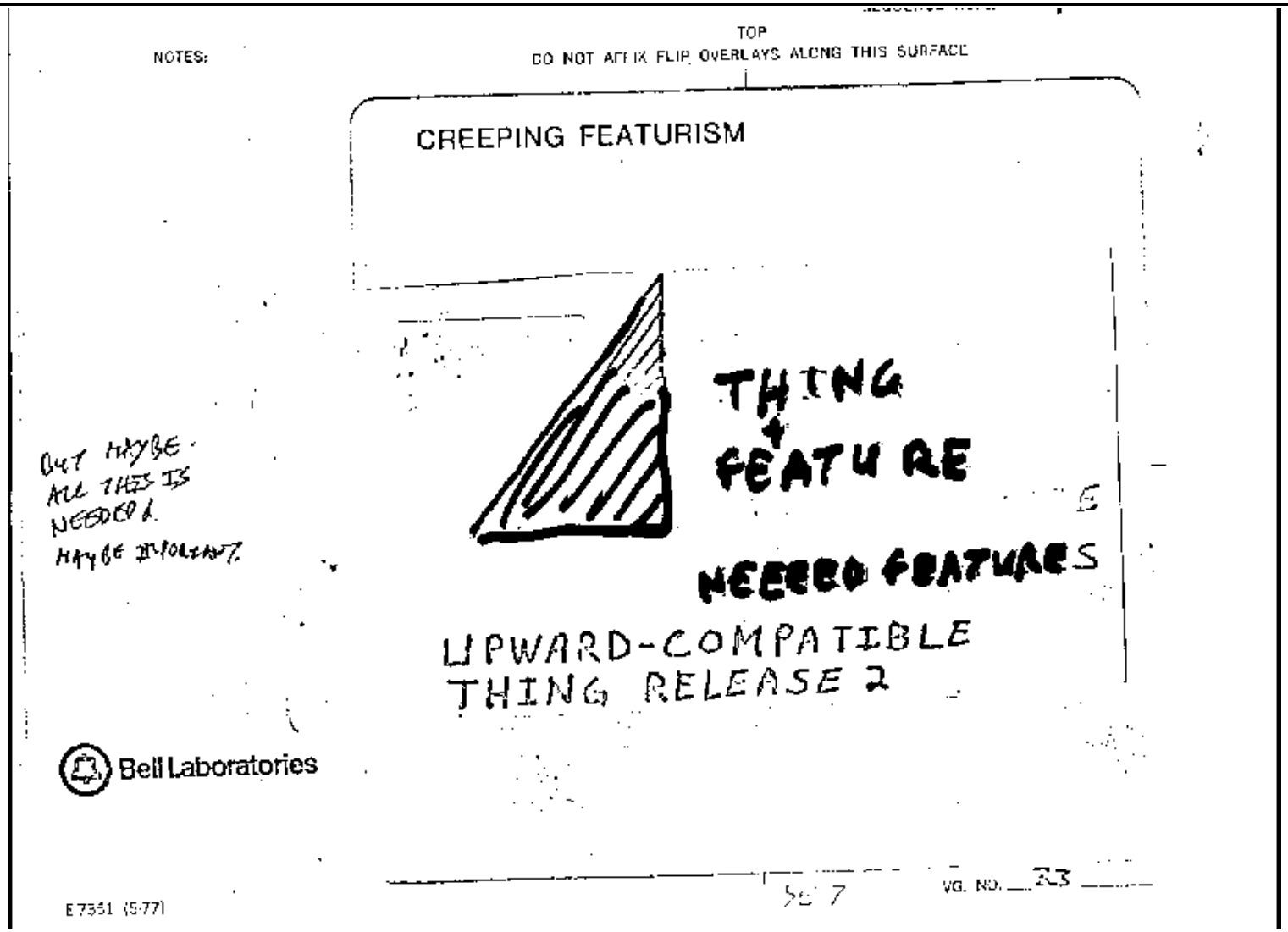
"A second complexity problem is offering the user too many options."
— G. J. Myers [MYE76A]

well, what about the case where..."

Bell Laboratories

VG. NO. 32

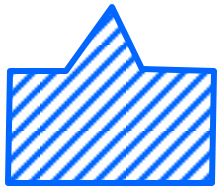
Creeping Featurism (* coined in 1976 paper, I think)



Creeping Featurism – Overlay Build

THING

Creeping Featurism – Overlay Build

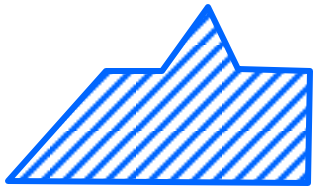


THING

+

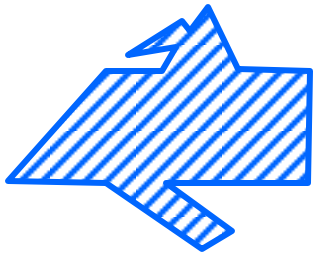
FEATURE

Creeping Featurism – Overlay Build



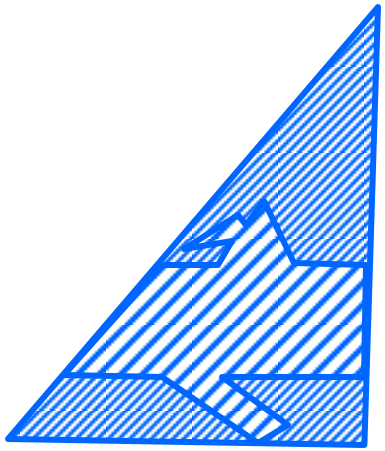
THING
+ FEATURE
+ FUTURE FEATURE

Creeping Featurism – Overlay Build



THING
+ FEATURE
+ FUTURE FEATURE
+ NEEDED FEATURES

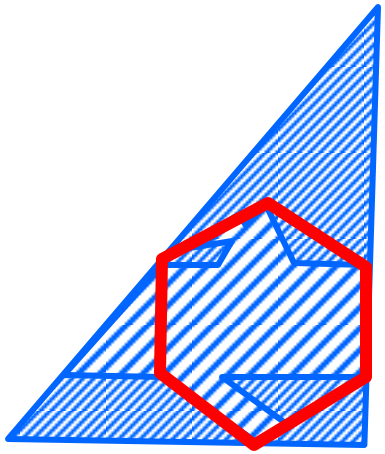
Creeping Featurism – Overlay Build



THING
+ FEATURE
+ FUTURE FEATURE
+ NEEDED FEATURES

UPWARD-COMPATIBLE
THING RELEASE 2

Creeping Featurism – Overlay Build

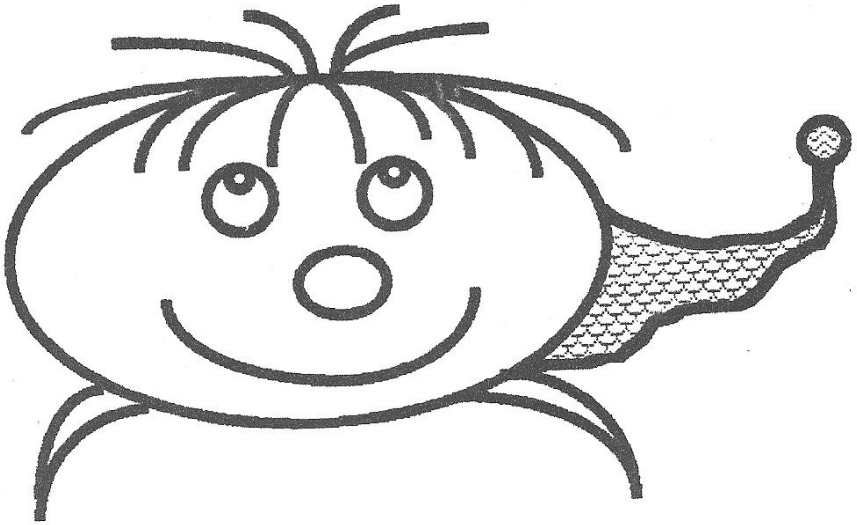


THING
+ FEATURE
+ FUTURE FEATURE
+ NEEDED FEATURES

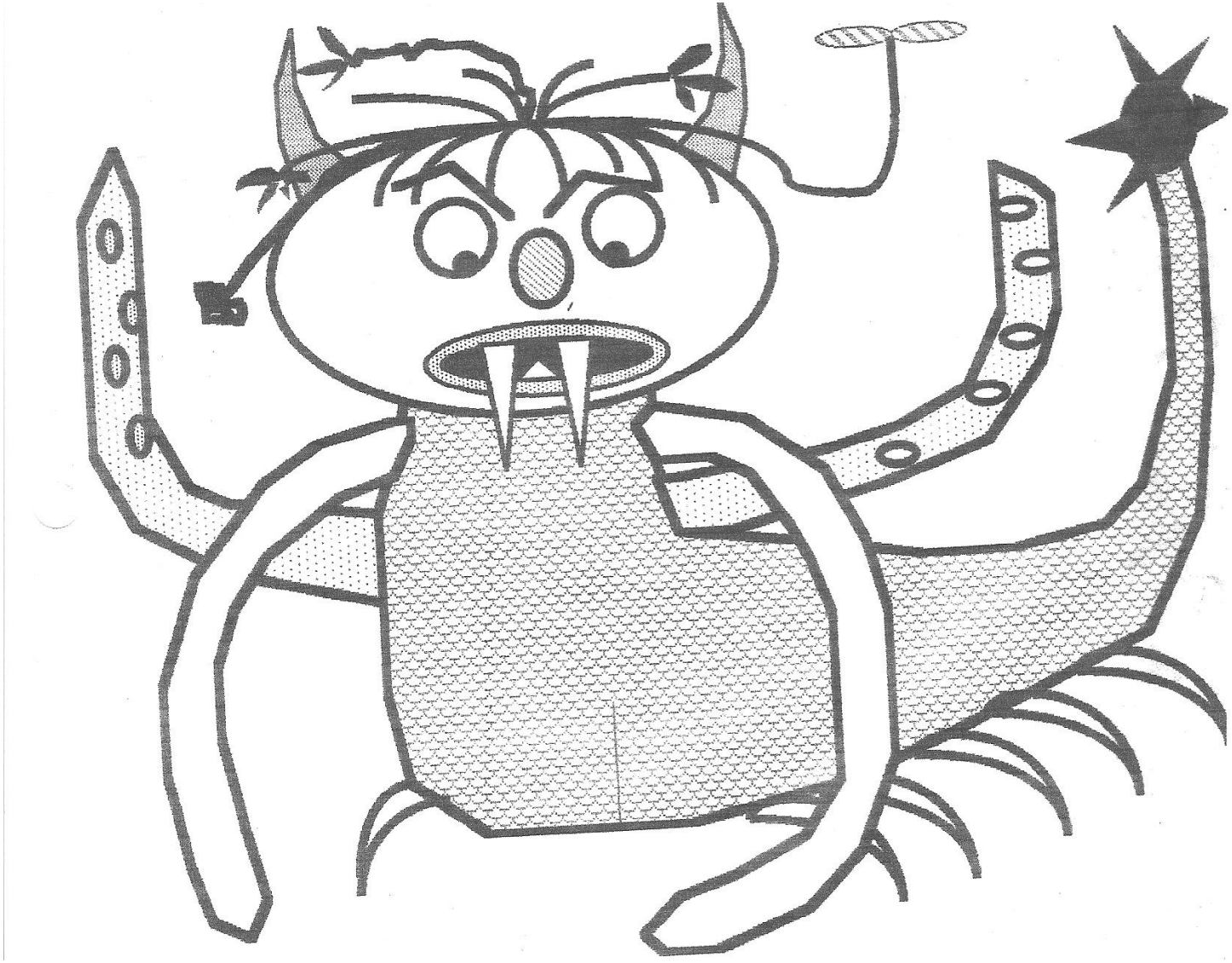
UPWARD-COMPATIBLE
THING RELEASE 2

BUT GOOD ENOUGH
COVERS THE REAL NEEDS

Featuris Creepis (Baby) later addition



Featuris Creepis (Adult) later addition



Usage Concentration and Intuition

NOTES:

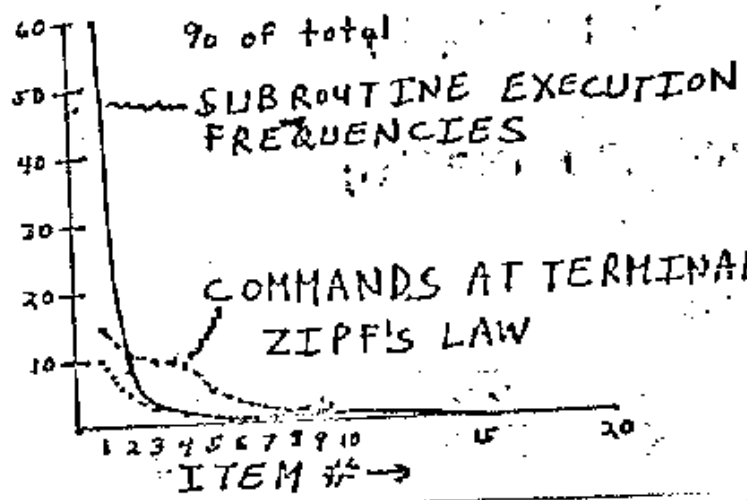
BUT MAYBE NOT

TOP

DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

USAGE CONCENTRATION AND INTUITION

- Zipf's Law (natural languages)
- 90/10 rules for computer usage
- Word frequency, command execution frequency, time spent in parts of programs, "feature" usage ...
- Intuition is bad [ELS76A, ELS77A, KNU71A]



NOW:
THINGS FAIL
BIG THINGS ARE EXPENSIVE
PROGRAMS GET BAD
HARD TO TELL WHAT'S IMPORTANT
SO, WHAT DO WE DO?

Bell Laboratories

VG. NO. 34

E-735L (5-77)

Strategies and Tactics

NOTES:

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

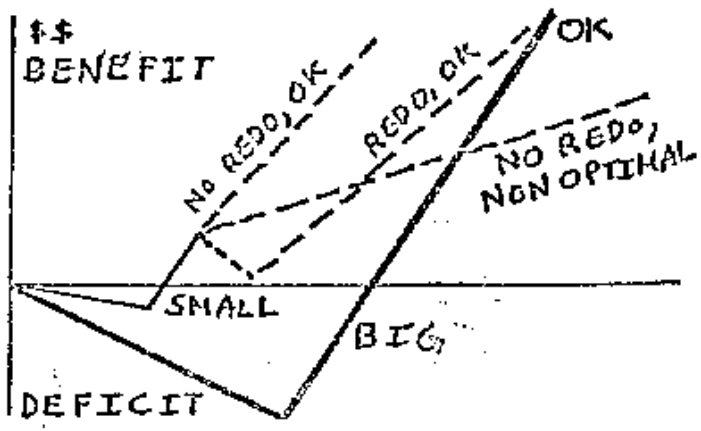
STRATEGIES AND TACTICS

WHAT ORDER TO DO THINGS? [JOY71A]

- Independent projects (pick one)

p = pr(success)
r = expected return if successful
c = expected cost

Pick projects with high p x r / c



ALL IS HELP →

ISK - what happens when cutoff
HALL - MAY HAVE
SE.
ISK - EXTERNAL FACTORS

Bell Laboratories

VG. 60 40

E-7351 (6-77)

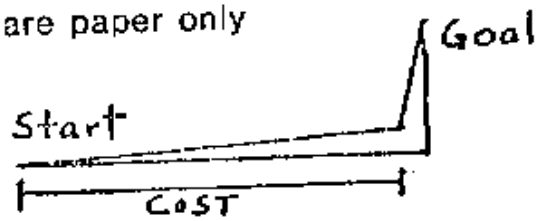
Dependent Projects

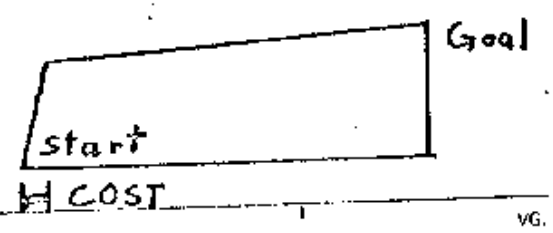
NOTES:

TOP

DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

- Dependent projects (do all)
  - High-risk/low-cost BEFORE low-risk/high-cost
  - Fail cheaply
  - Difficulties with human nature, merit review ...
  - Watch out for top-down design as religion, especially when early results are paper only





Bell Laboratories

VG. NO. 41

Risk Assessment – 1 Alligator, later addition

NOTES:

TOP
DO NOT AFFIX OVERLAYS ALONG THIS SURFACE

RISK ASSESSMENT - 1 ALLIGATOR

$$\text{Pr (success)} = \frac{1}{2^{\text{number of risks}}}$$

• NEW HARDWARE	(1)
• NEW TYPE OF HARDWARE	(2)
• NEW SOFTWARE	(1)
• NEW TYPE OF SOFTWARE	(2)
• NEW TEAM	(1-2)
• NEW DEVELOPMENT ENV.	(1)
• EXTERNALS (VENDORS, ETC.)	(1)

DON'T FAIL STUPID, FAIL SMART

Bell Laboratories

McClure, INAU69A, pp. 73-75

J. R. MASHEY

VG. NO. 4/A

E-7351 (5-77)

Build It Quick

NOTES:

DOES NOT MEAN SLOPPY CODING - BUT DON'T BE TOO ATTACHED TO IT

ACCEPTANCE IS OFTEN WORST PROBLEM -

WHEN YOU ASK WHAT SOMEBODY WANTS - ANS. ALWAYS EVERYTHING
a) misses some.
b) Has more.
Give something, get killed
Viewpoint

HAVE YOU EVER USED SOFTWARE THAT OUGHT TO HAVE BEEN THROWN AWAY

Bell Laboratories

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

BUILD IT QUICK

- Be ready to throw away
- Purpose is INSIGHT
- Partnership rather than adversary relationship
- High cost of throwing away (?)

"Many of the people who design software refer to users as "they." They are some odd breed of cats living there in the outer world, knowing nothing, to whom nothing is owed."
— J. W. Smith, in [NAU69A]

"The users are the people who do our design, once we get started."
— J. D. Babcok, in [NAU69A]

VG. NO. 42.

E-2353 (5-77)

Use Existing Tools

NOTES:

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

USE EXISTING TOOLS ["Stop rebuilding oscilloscopes"]

- Earlier start, chance to avoid leapfrog
- Help keep product source small
- Work at highest level possible
- Tools available
  - Parser generators, compiler-compilers
  - Utility programs
  - Command languages used as programming languages
  - Existing code
- Uniform support facilities exist already, support and target can be separated

Bell Laboratories

VG. NO. 43

F 2551 (5-77)

Build Tools (Instead of Systems)

NOTES:

TOP

DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

PRODUCER IS CONSULTANT WHO HELPS PEOPLE SOLVE PROBLEMS

WHO MAKES TOOLS TOGETHER IS NOT CONSULTANT.

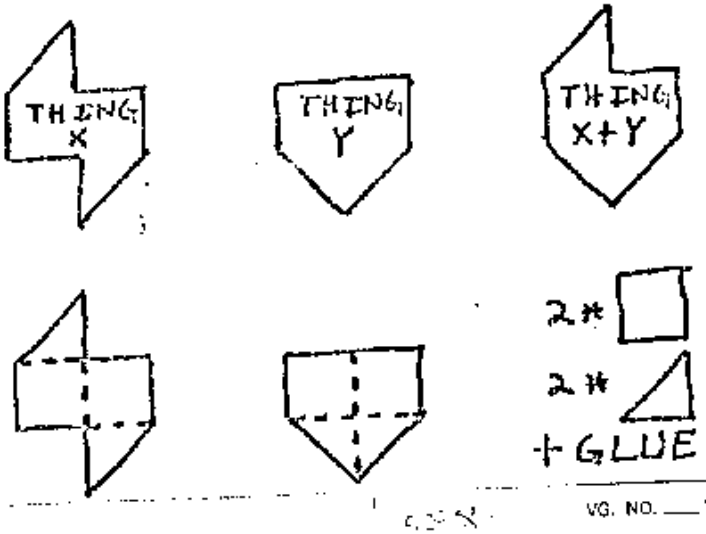
RECS/REMARKS (40)

WHO SUPPORT

GLUE IF FAIL – USEFUL PIECES

BUILD TOOLS (INSTEAD OF SYSTEMS)

- Tools are easier to change than monoliths
- Morale seems better for producer and consumer, e.g. in maintenance
- User sees tools plus connectors
- Multi-user problem moved outboard
- Left-overs



Bell Laboratories

VG. NO. 44

Connecting Tools

NOTES:

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

CONNECTING TOOLS

- "Bad" interface = unrecoverable disaster
  - Hardware — interconnection structure
  - Compilers/assemblers — subroutine linkage
  - Operating systems — program invocation, inter-process communication
  - File systems — # and type of file formats
- Connectors are central, must be small and simple
- Implicit cooperation must be normal, not just possible

"In fact, this recognition that interfaces not only are the most humble starting point, but undoubtedly the most crucial, can provide the primary guideline for standardization efforts."
— D. T. Ross, in [ROS76A]

Bell Laboratories

E-7351 (5-77)

VG. NO. 46

Progress in Tool-Oriented Approach

NOTES:

TOP

DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

PROGRESS IN TOOL-ORIENTED APPROACH

- New tools
- Improvements to old tools
- Improved connections
- Raise level of work — hardware vs software
  - Machine instructions
  - Macros and subroutines
  - Commands (load modules)
  - Command language procedures
  - Specialized processors, front-ends, back-ends, network tools

Bell Laboratories

E-7351 (5-77)

VG. NO. 47

Tools Are Good, later addition

CUSTOMER NETWORK OPERATIONS

TOOLS ARE GOOD!

⇒ EVERYBODY SAYS SO

⇒ CURRENT CURE-ALL

BUT...

Problems, later addition

CUSTOMER NETWORK OPERATIONS

PROBLEMS

- MORE TOOLS ⇒ HARDER TO FIND THE RIGHT ONE
- SUBSET GUIDES, ROADMAPS, SYNTHESIS METHODOLOGIES
- HUMAN LIMITS
- ⇒ TOOLKIT ⇒

Bell Laboratories SEQUENCE NO. ____ VG. NO. ____

Small tactics (External)

SEQUENCE NO. _____

NOTES:

TOP
DO NOT AFFIX OVERLAYS ALONG THIS SURFACE

SMALL TACTICS (EXTERNAL)

- "LIFEBOAT" THEORY
  - ADDING FEATURE N+1
  - THROW 1 BACK
- "SINKING LIFEBOAT" THEORY
  - ANTICIPATION OF FUTURE REQUESTS

Bell Laboratories

J. R. MASHEY

NO. NO. 4 1 B

Small Tactics (Internal)

SEQUENCE NO. _____

NOTES:

TOP
DO NOT AFFIX OVERLAYS ALONG THIS SURFACE

SMALL TACTICS (INTERNAL)

- PEOPLE
  - GENERATORS
  - ANALYZERS
  - CONSOLIDATORS
- CONSOLIDATION WALKTHRUS ("CRUNCH MEETINGS")
  - FIGHT TOP DOWN REDUNDANCY
  - SPECIFIC ANTI-ENTROPY EFFORT
  - PROGRAMMABLE WORK
  - WHEN MODULE NEXT TOUCHED

Bell Laboratories

J. R. MASHEY

Performance

NOTES:

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

PERFORMANCE

"Premature optimization is the root of all evil."
— D. E. Knuth

- Modest early analysis, to avoid obvious impossibility
- Measure, understand, change
  - New Algorithm
  - Minor code rearrangements
  - Small rewrite from high to low level
- Binding times
  - Defer binding as long as possible
  - Use higher level constructs [late binding], convert if needed
  - Control structure late binding, processing early — SNOBOL, APL, shell procedures

Bell Laboratories

VG. NO. 49

E-7351 (5-77)

Rays of Hope

NOTES:

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

RAYS OF HOPE

Increasing awareness of simplicity

- PASCAL vs PL/I, mini vs maxi

UNIX*

- Philosophy
- Support facility for micro to maxi targets
- UNIX-based projects

UNIX and C portability efforts

[BUT NO PANACEAS]

*UNIX is a Trademark/Service mark of the Bell System

Bell Laboratories

VG. NO. 50

E-7351 (5-77)

Conclusion

NOTES:

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

CONCLUSION

APPROACHES FORM A CONTINUUM

FAILURE IS THE NORM

STRATEGIES

- Build it fast
- Keep it small and simple
- Build to be changed

MODERATE OPTIMISM, NO PANACEAS

PEOPLE

Bell Laboratories

VG. NO. 51

E-7351 (5-77)

We Have Met the Enemy

NOTES:

TOP
DO NOT AFFIX FLIP OVERLAYS ALONG THIS SURFACE

"WE HAVE MET
THE ENEMY
AND THEY IS US."

— Pogo

Bell Laboratories

VG. NO. 52

C7351 (5-77)

Bibliography - 1 of 4

``Small is Beautiful and Other Thoughts on Programming Strategies''
Bibliography
John R. Mashey -- Bell Laboratories, Whippany, N. J. 07981

BEL77A Belady, L. A., and Lehman, M. M. The Characteristics of Large Systems. Proc. Conf. on Research Directions in Software Technology, P. Wegner, Ed., Oct. 10-12, 1977, Brown University.
Change, growth, entropy. See also earlier works by same authors.

BOE73A Boehm, B. A. Software and its impact: a quantitative assessment. Datamation 19, 3 (May 1973), 48-59.
Programming costs a lot; it will get worse before it gets better.

BRO75A Brooks, F. P., Jr. The Mythical Man-Month. Addison-Wesley, Reading, Mass., 1975.
Readable classic. But get the 1995 version.

CHA74A Chase, W. P. Management of System Engineering. Wiley-Interscience, New York, 1974.
See ``Fallibility'' and ``Evil of Complexity'', pp. 16-20.

DAL77A Daly, E. B. Management of Software Development. IEEE Trans. on Software Eng. SE-3, 3(May 1977), 230-242.
Data on ESS programming at GTE/Automatic Electric.

DIJ72A Dijkstra, E. W. The Humble Programmer. Comm. ACM 15, 10(Oct. 1972), 859-866.
Evils of complexity; PL/I as the fatal disease.

DOL76A Dolotta, T. A., and Mashey, J. R. An Introduction to the Programmer's Workbench. Proc. Second Int. Conf. on Software Engineering, Oct. 13-15, 1976, 164-168.
Design philosophy {7.2}: we argued a lot about this but it worked.

Bibliography - 2 of 4

EDP71A Advanced Projects in Data Processing. EDP Analyzer 9, 11(Nov. 1971), 1-14.
Hardly anything works.

EDP77A Getting the Requirements Right. EDP Analyzer 15, 7(July 1977).
...is not easy to do.

ELS76A Elshoff, J. L. An Analysis of Some Commercial PL/I Programs. IEEE Trans. on Software Eng. SE-2, 2(June 1976), 113-120.
PL/I: 100K lines of code: most ``features'' are hardly ever used.

ELS77A Elshoff, J. L. The Influence of Structured Programming on PL/I Program Profiles. IEEE Trans. on Software Eng. SE-3, 5(Sept. 1977), 364-368.
Real data in place of faith in how wonderful it will be.

GAL75A Gall, J. SYSTEMANTICS How Systems Work and Especially How They Fail. Quadrangle/New York Times Book Company, New York, 1975.
Systems work poorly or not at all: a light treatment.

HOR75A Horowitz, E., Ed. Practical Strategies for Developing Large Software Systems. Addison-Wesley, Reading, Mass., 1975.
Articles by many; see esp. Royce and Schwartz.

Bibliography - 3 of 4

JOY71A Joyce, W. B. Organization of Unsuccessful R&D Projects. IEEE Trans. on Engineering Management Vol. EM-18, 2(May 1971), 57-65.
Most R&D projects fail; planning for failure saves money.

KER76A Kernighan, B. W., and Plauger, P. J. Software Tools. Addison-Wesley, Reading, Mass., 1976.
Good tools built before your eyes; déja vu for UNIX folks.

KER79A Kernighan, B. W., and Mashey, J. R. The UNIX™ Programming environment. Software-Practice and Experience 9, (1979), 1-15.
See especially "UNIX AND MODERN PROGRAMMING METHODOLOGIES".

KNU71A Knuth, D. E. An empirical study of FORTRAN programs. Software-Practice and Experience 1, (1971), 105-133.
Human intuition about programs and languages is terrible.

MYE76A Myers, G. J. Software Reliability. Wiley-Interscience, New York, 1976.
good thoughts on programming in general; balanced (if conventional) view.

MYE77A Myers, S., Sweezy, E. E. Why Innovations Falter and Fail: A Study of 200 Cases. Cunningham, D. E., Craig, J. R., Schlie, T. W. Eds. Technological Innovation, Westview Press, Boulder, Col. 1977, 83-99.
Nothing works; 58 ideas to yield one successful product.

NAU69A Naur, P., Randell, B. (eds.) Software Engineering. Scientific Affairs Division, NATO, Brussels 39, Belgium, Jan. 1969.
Required reading, if you can find a copy. See McIlroy's comments.

PAP73A Papanek, V. Design for the Real World. Bantam Books, Toronto, 1973.
Friends of Bucky Fuller will like this one.

PAP77A Papanek, V., and Hennessey, J. How Things Don't Work. Pantheon Books, New York, 1977.
Good thoughts for designers of anything; see Ch. 7 especially.

Bibliography - 4 of 4

RIT77A Ritchie, D. M. The UNIX Time-Sharing System -- A Retrospective. Proc. of Tenth Hawaii International Conf. on Systems Science, Honolulu, Hawaii, Jan. 1977.
Many good thoughts and thought-provoking history.*

ROS76A Ross, D. T. Homilies for Humble Standards. Comm. ACM 19, 11(Nov. 1976), 595-600.
A rational view on the role of standards; read before imposing any.

SCH73B Schumacher, E. F. Small is Beautiful. Harper & Row, New York, 1973.
No mention of programming, but the philosophy is applicable.

STR72A Strunk, W., Jr., and White, E. B. The Elements of Style, 2nd ed.. MacMillan, New York, 1972.
Everyone should read this once a year and write code that reads like this.

THO75A Thompson, K. In Structured Programming, The UNIX Command Language. Infotech State of the Art Report, London, Mar. 10-12, 1975. 375-384.
Good minimalist philosophy; features that fill much-needed gaps.*

* For newer versions, see Bell System Technical Journal 57, No. 6, Part 2 (July-August 1978), an entire issue on UNIX and its offspring.

CUSTOMER NETWORK OPERATIONS

DARWINIAN SELECTION +
THE ECOLOGICAL NICHE

- MUTATION ⇄ SELECTION
- FILL NICHE & DEFEND IT
  - LARGE – GENERALITY (HUMANS & RATS)
  - SMALL – SPECIALIZATION FUNCTION OR SPEED (ANTEATERS & PENGUINS)
  - SURVIVE ICE AGES
- WORLD ONLY SO BIG.

Bell Laboratories SEQUENCE NO. _____ VG. NO. 1

CUSTOMER NETWORK OPERATIONS

SOCIAL · STRUCTURE

- DOES IT ADAPT TO DIFFERENT ORGANIZATIONS & TYPES OF TEAMS?
- DOES IT FIT REALITY?
  - "GULP FACTOR"
  - OLD VS NEW
  - $$
- TECHNOLOGY SPREAD
  - TOP-DOWN
  - BOTTOM-UP (LOCAL GURU)

Bell Laboratories SEQUENCE NO VG NO

1977-now Retrospective

- Somewhat non-mainstream at the time
  - Language Design for Reliable Software (rejected) ➔
  - B. W. Kernighan, J. R. Mashey, "The UNIX Programming Environment", Computer 14, 4 (April 1981), 12-24.
- Tools, components,
- Shell programming, awk ➔ scripting languages
- Automation – source control, build tools
- "Agile programming" somewhat of a descendant
  - Same things get rediscovered again and again, with different names
- Still hard to get requirements right
  - Many implicit decisions
  - Some addressed in later "Software Army on the March" talk
  - Best tool I've seen so far: Ravenflow, www.ravenflow.com . SEE THIS.

Retrospective – Open Source

- "Open Source" is most recent term for "ancient" practice
  - ~1948 – David Wheeler invests subroutines for EDSAC@ Cambridge
  - ~1952 – John von Neumann donates designs for Princeton IAS
  - 1955 – IBM SHARE User's Group founded
  - 1961 – DECUS (Digital Equipment Corporation) user group founded
  - 1960s – IBM HASP (mainframe OS code, user-modified)
    - » "Should old Chuck Forney be forgot, and HASP songs sung no more."
  - User-contributed libraries; trading amongst users
    - » Penn State ASSIST (Mashey & others), 1970- … still running 38 years later.
  - 1970s – UNIX "open source" within Bell Labs
  - 1970s – UNIX licensed to universities, government, "as is, don't call us"
  - 1970s – John Lions "Commentary on UNIX, with Source Code"
  - 1970s – Berkeley UNIX, Ken @ Berkeley, DARPA $, Internet
  - 1976 – B. W. Kernighan, P. J. Plauger, "Software Tools", RATFOR.
    - » ➔ Software Tools User's Group (STUG)
  - 1979 – UNIX V7 released – (reasonably) portable OS
  - 1985 – Free Software Foundation (UNIX commands, especially GNU C)
  - 1991 – Linux (kernel)

- Local libraries ➔ magnetic tapes ➔ UUCP ➔ Internet ➔ Web
- Local groups ➔ vendor-based groups ➔ large expansion

Retrospective … Future

- John R. Mashey, "Languages, Levels, Libraries, and Longevity"
  - ACM Queue, Vol. 2, No. 9 - Dec/Jan 2004-2005
  - http://www.acmqueue.org/modules.php?name=Content&pa=printer_friendly&pid=245&page=1

'In 50 years, we've already seen numerous programming systems come and (mostly) go, although some have remained a long time and will probably do so for: decades? centuries? millennia? The questions about language designs, levels of abstraction, libraries, and resulting longevity are numerous. Why do new languages arise? Why is it sometimes easier to write new software than to adapt old software that works? How many different levels of languages make sense? Why do some languages last in the face of "better" ones?

We can gather insights from the last 50 years of programming systems to the current time. For the far future, Vernor Vinge's fine science-fiction novel, A Deepness in the Sky, rings all too true. The young protagonist, Pham, has joined a starship crew and is learning the high-value vocation of "programmer archaeologist," as the crew's safety depends on the ability to find needed code, use it, and modify it without breaking something. He is initially appalled at the code he finds:

The programs were crap…Programming went back to the beginning of time…There were programs here that had been written five thousand years ago, before Humankind ever left Earth. The wonder of it—the horror of it…these programs still worked…down at the very bottom of it was a little program that ran a counter. Second by second, the Qeng Ho counted from the instant that a human had first set foot on Old Earth's moon. But if you looked at it still more closely… the starting instant was actually about fifteen million seconds later, the 0-second of one of Humankind's first computer operating systems…

"We should rewrite it all," said Pham.

"It's been done," said Sura.

"It's been tried," corrected Bret…"You and a thousand friends would have to work for a century or so to reproduce it… And guess what—even if you did, by the time you finished, you'd have your own set of inconsistencies. And you still wouldn't be consistent with all the applications that might be needed now and then…"

"The word for all this is 'mature programming environment.''