SYSTEM INTEGRATION AND THE GLOBAL INFORMATION INFRASTRUCTURE

and

HOW RELIABLE CAN WE MAKE SOFTWARE

A Aho

Rapporteur: Francisco Vilar Brasileiro

Bellcore
Bell Communications Research

How Reliable Can We Make Software?

Alfred V. Aho
Morristown, NJ - USA
September 8, 1994

Outline

- Business Context
- Why is software hard?
- The multidimensional quality of software quality
- The software development process
- Techniques for improving software quality
- Challenges for the research community

Business Context

- New services and technologies need to be added to the global information infrastructure quickly, cost effectively, and reliably

AIN	-	Advanced Intelligent Network
ATM	-	Asynchronous Transfer Mode
CCS/SS7	-	Common Channel Signaling
ISDN	-	Integrated Services Digital Network
PCS	-	Personal Communication Services
SONET	-	Synchronous Optical Network
VDT	-	Video Dial Tone

ISDN Reliability

- Requirements for total system downtime (of a switch, all terminations down) = 3 min/per year (1 min scheduled, 2 minutes unscheduled
- Accumulated downtime for B channel circuit in an individual ISDN interface = 37 minutes per year
- Requirements for field performance (hardware + software), reliability, downtime ...
  - requirements stated in terms of outage reports and measurements required by the FCC
  - comparison with actual data determines feasibility of requirements

Why is Software Hard?

"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."

"The basic problem is that certain classes of systems are placing demands on us which are beyond our capabilities and our theories and methods of design and production at this time."

"The 'software gap' may not be immutable, but closing it will require metamorphosis in the practice of software production and its handmaiden, software design."

Why is Software Hard?

- Immature scientific foundation for software engineering
- Underlying problems are mathematically intractable
- Software development involves people, technology and process
- Software requires thinking

Big Changes in Last Twenty-Five Year

Reusable Subsystems

- operating systems
- database systems
- report generators

Fewer Hardware Constraints

- order of magnitude improvements in processors/memory/communication
- graphical user interfaces
- portability

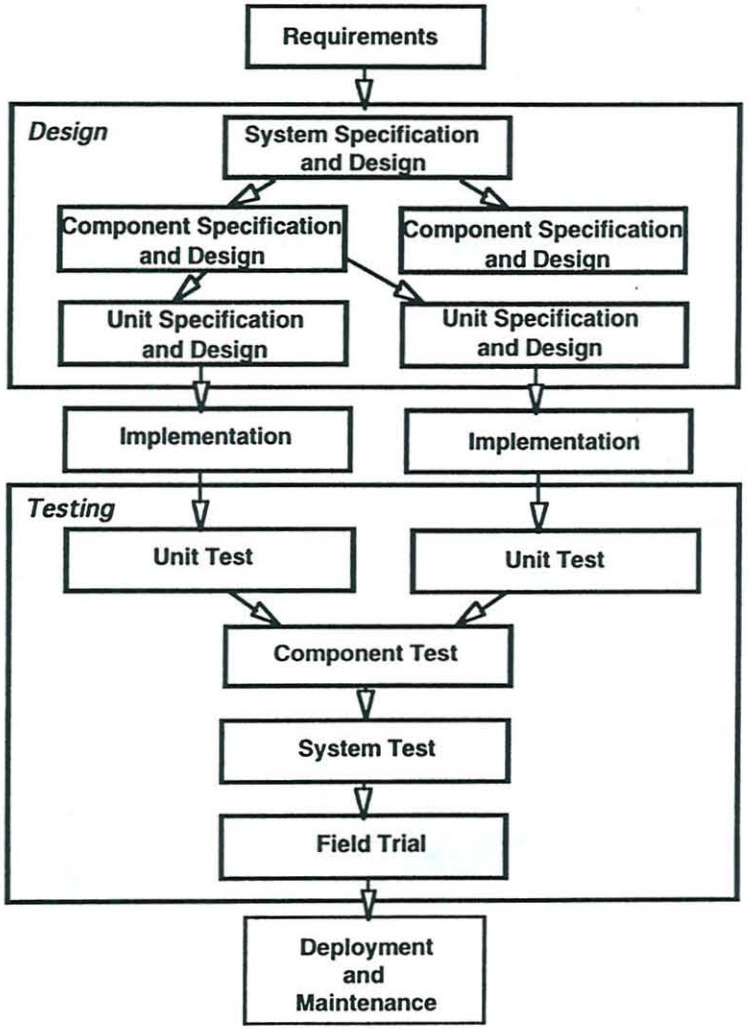
Integrated Software Development Environments

- high-level languages
- application generators
- configuration management tools

Better Development Processes

- structured design
- formal inspections
- more experienced managers
- observation and control of the software process

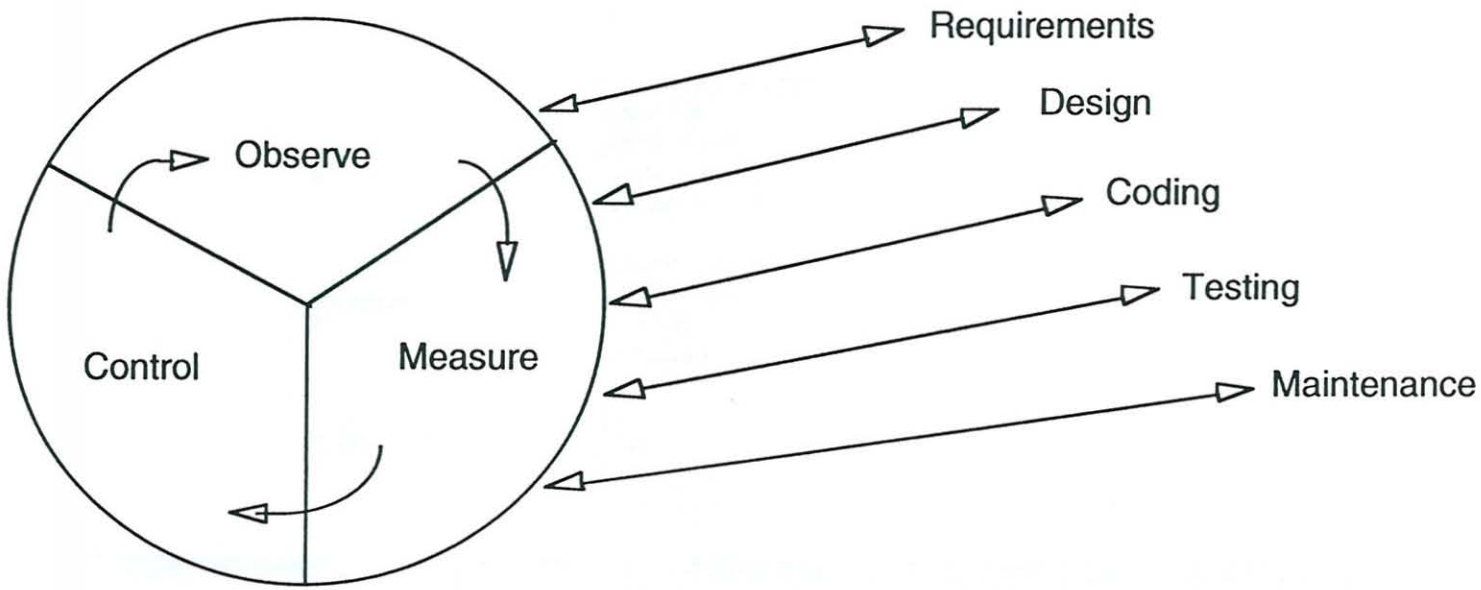
Software Development Process



AHO9.8.94 — 8

Bellcore
Bell Communications Research

Observation and Control of the Software Process



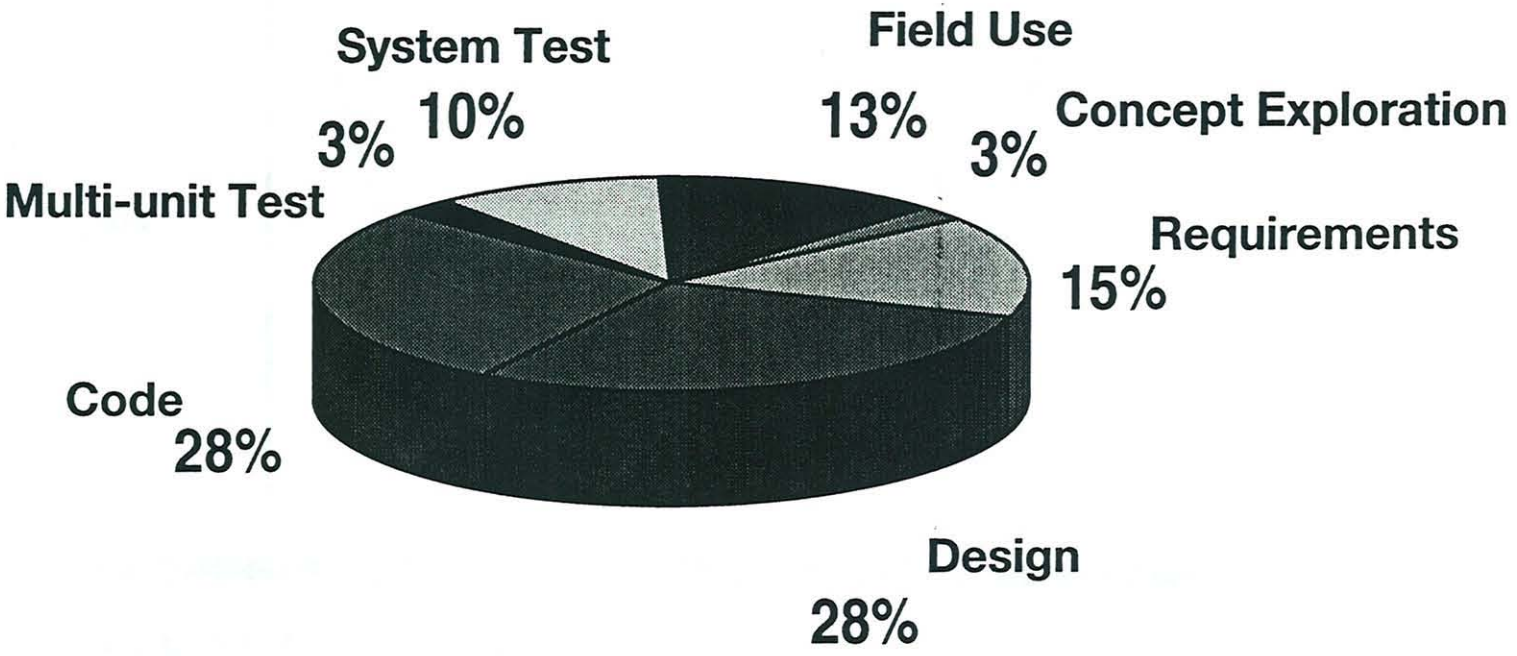
Bellcore
Bell Communications Research

Software Quality is Multidimensional

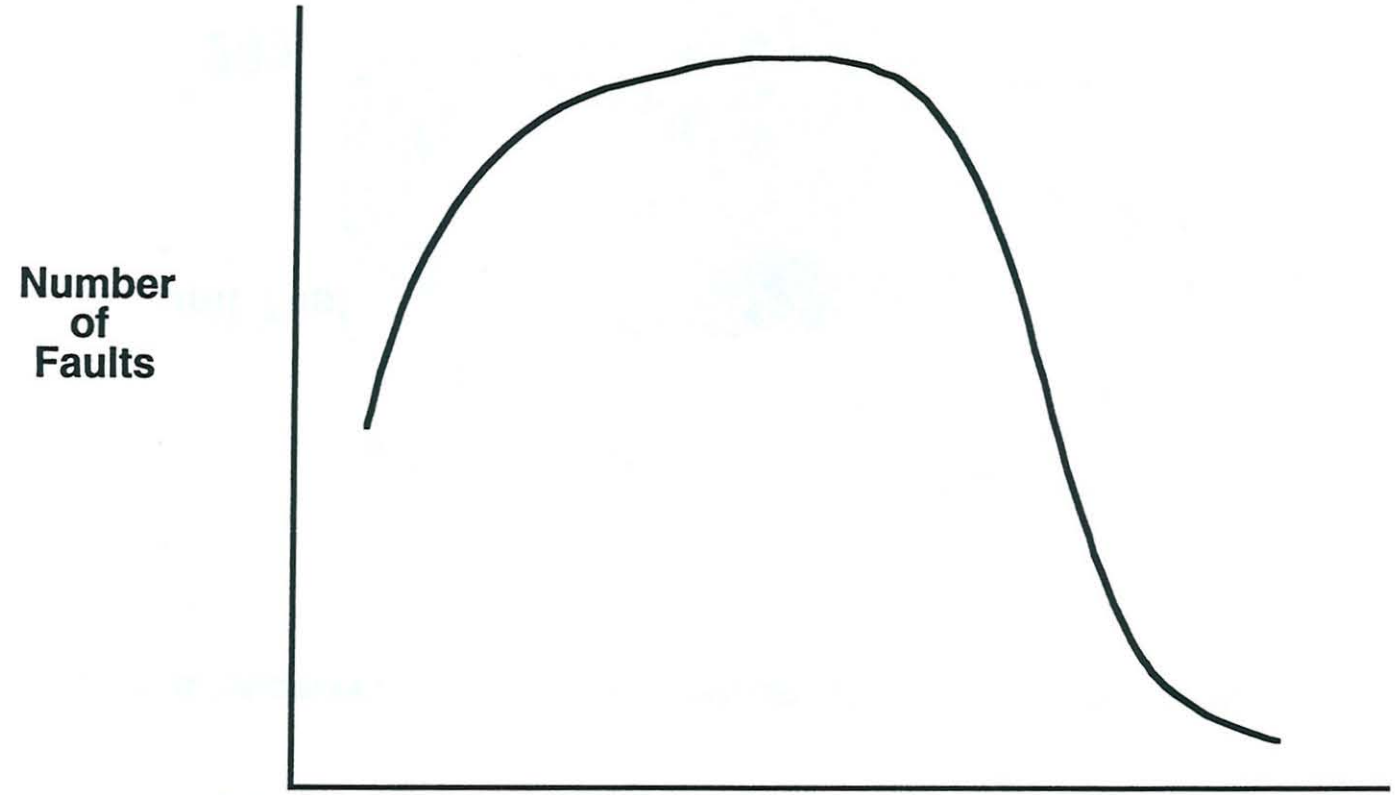
Functionality	Feature Set Capabilities Generality Security
Usability	Human Factors Aesthetics Consistency Documentation
Reliability	Frequency/Severity of Failure Recoverability Predictability Accuracy Mean Time to Failure
Performance	Speed Efficiency Resource Consumption Thruput Response Time
Supportability	Testability Extensibility Adaptability Maintainability Compatability Configurability Serviceability Installability Localizability

[Grady '92]

Sources of Defects



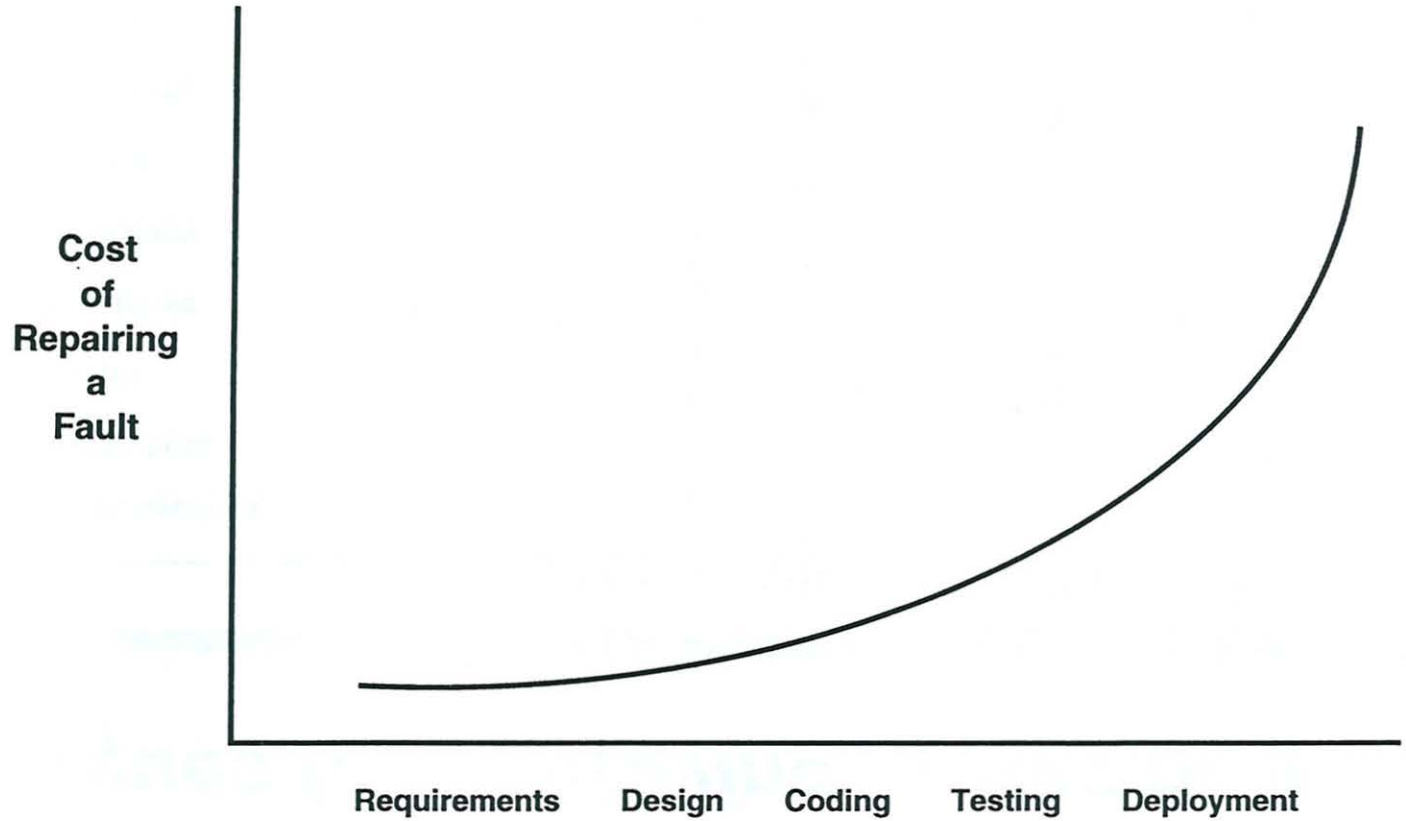
Introduction and Removal of Faults During Software Life Cycle



AHO9.8.94 — 13

Bellcore
Bell Communications Research

Cost of Repairing a Fault During Software Life Cycle



AHO9.8.94 — 14

Bellcore
Bell Communications Research

Techniques for Improving Software Quality

	Requirements	Design	Coding	Testing
Formal inspections	X	X	X	
Formal methods	X	X	X	X
CASE tools	X	X	X	X
Software reuse	X	X	X	X
OO technology	X	X	X	X
Prototyping	X	X		
Fault tolerance	X	X	X	X
Groupware	X	X	X	X
Design for testability		X		X
Fault/failure analysis		X		X
Reliability modeling		X	X	X
Smart testing				X
Programming methodology		X	X	

Bellcore
Bell Communications Research

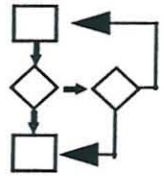
Comparison of Defect-Removal Techniques

TECHNIQUES	EFFICIENCY (Defects Found/Hour)
Regular Use	.21
Black Box	.282
White Box	.322
Reading/Inspection	1.057

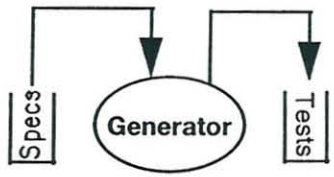
[Grady '92]

What is Smart Testing?

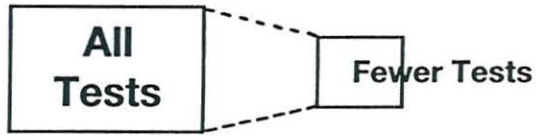
- Use Program Structure to Aid Testing
  (e.g., coverage testing)



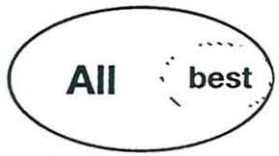
- Use Automated Techniques Where Possible
  (e.g., test generation)



- Use Optimized Tests Where Possible
  (e.g., test set minimization)



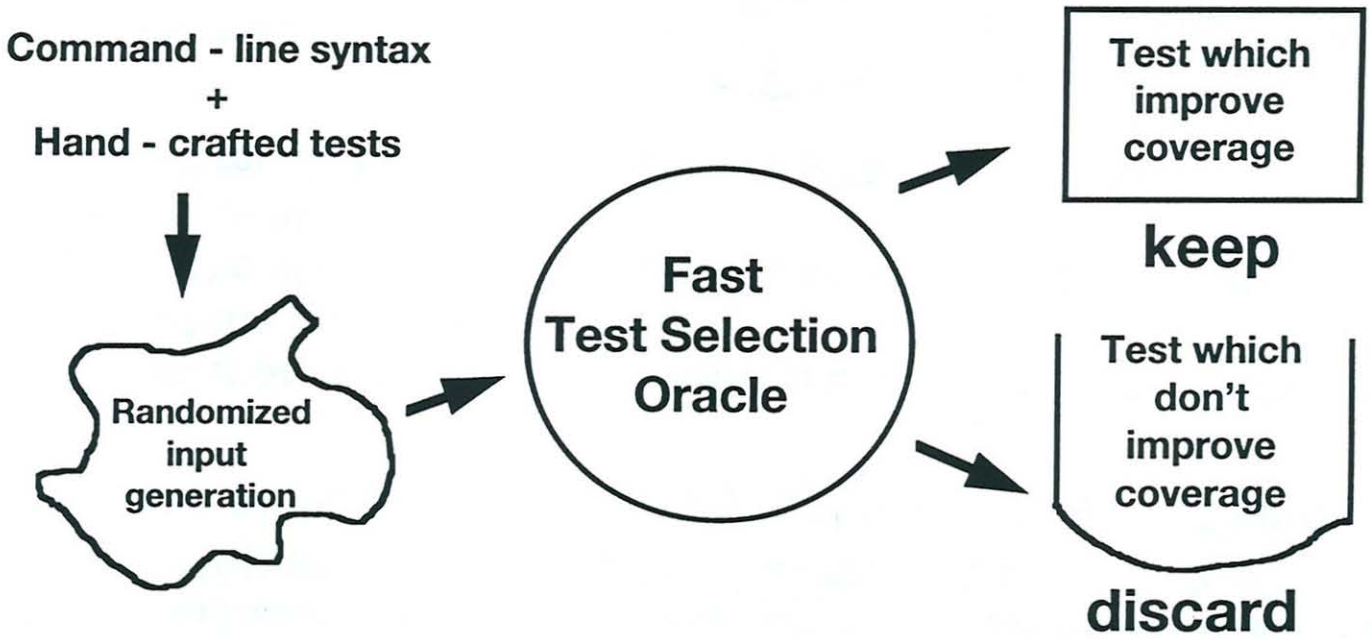
- Use Most cost-effective Tests When Necessary
  (e.g., test set prioritization)



Bellcore
Bell Communications Research

Smart Testing and Analysis

Use of Automated Test Generation



Example result for the Unix sort function. Coverage with automatically generated tests:

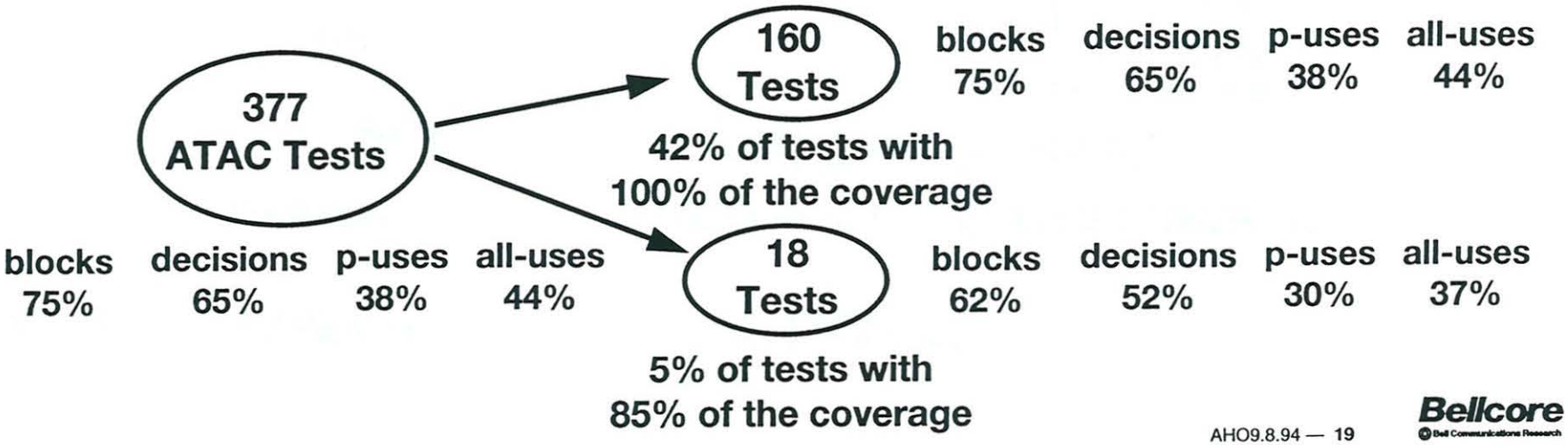
basic blocks	decisions	p-uses	all-uses
95%	83%	74%	78%

Bellcore
Bell Communications Research

Smart Testing and Analysis

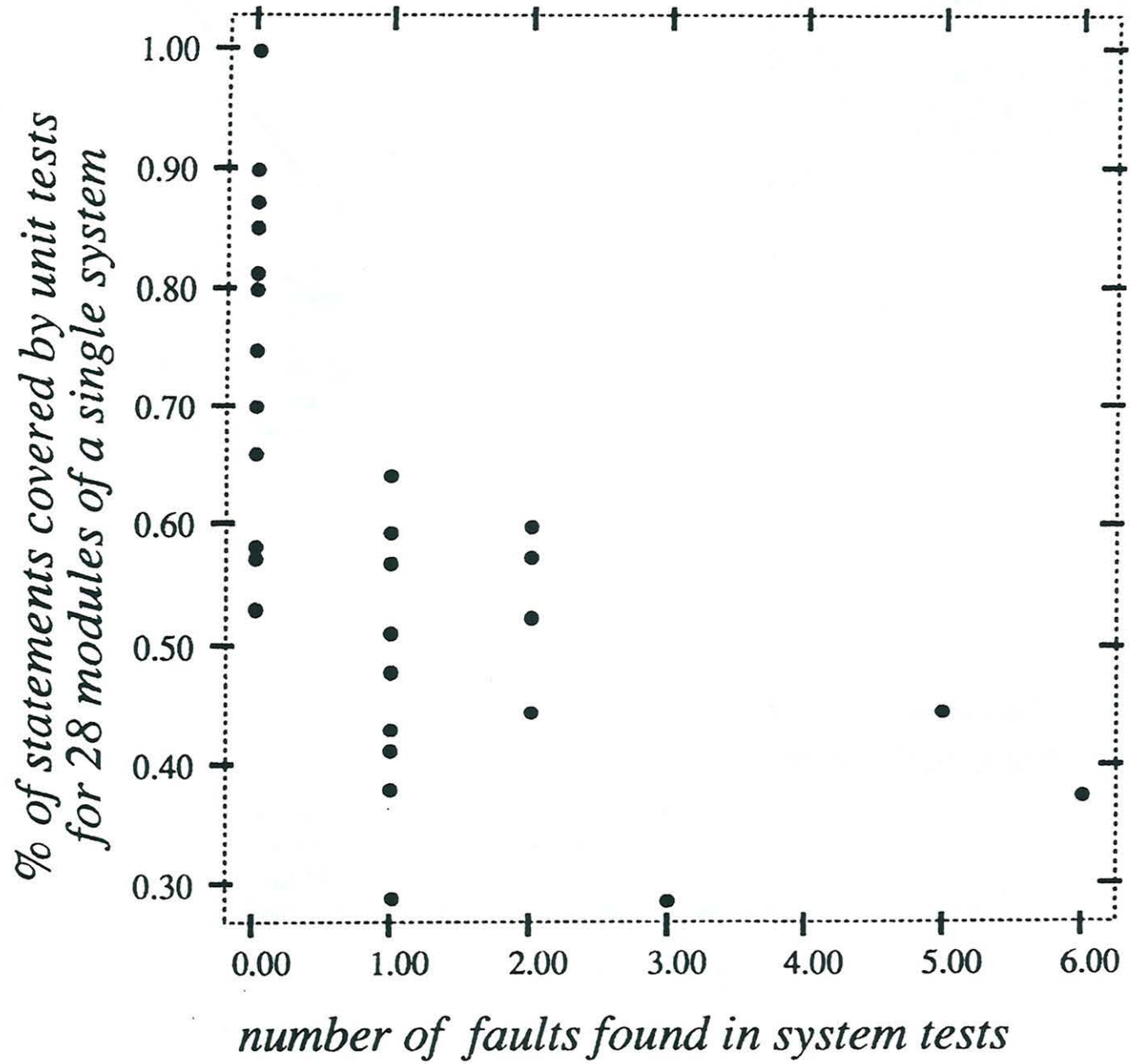
Replace all tests with a minimized set of tests.
Replace all tests with a small, highly cost - effective subset.

- The cost of running and maintaining a test suite is proportional to the size of the test suite.
- Given the 80/20 rule, a very small subset of tests will have a very high cost benefit.
- For example, a 12 KLOC module of ATAC has 377 tests:



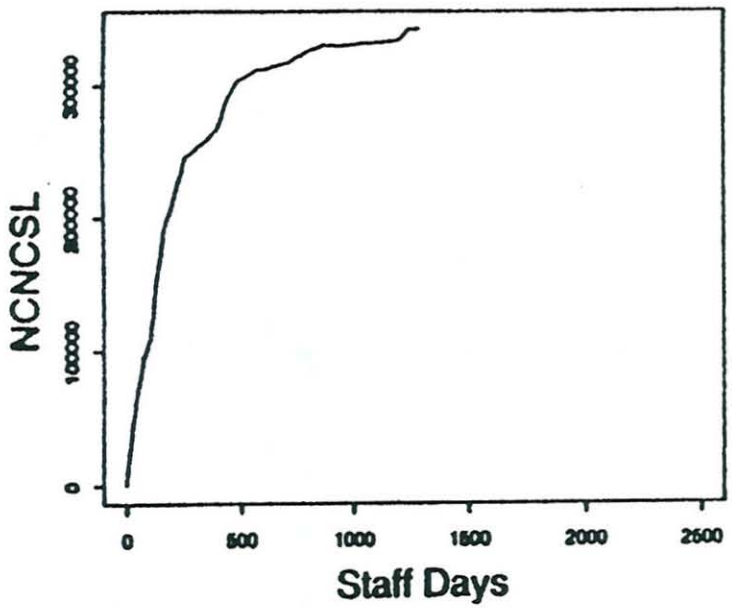
Bellcore
Bell Communications Research

Smart Unit Testing Reduces System Test Faults



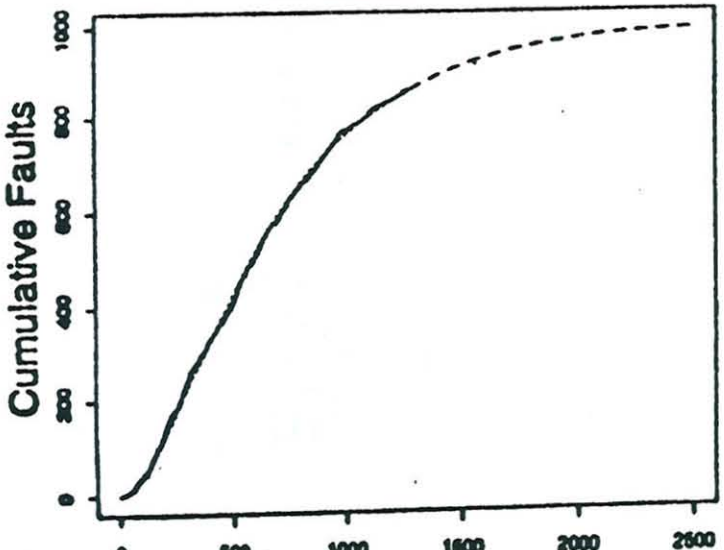
Fault Prediction

1. Cumulative Code Plot



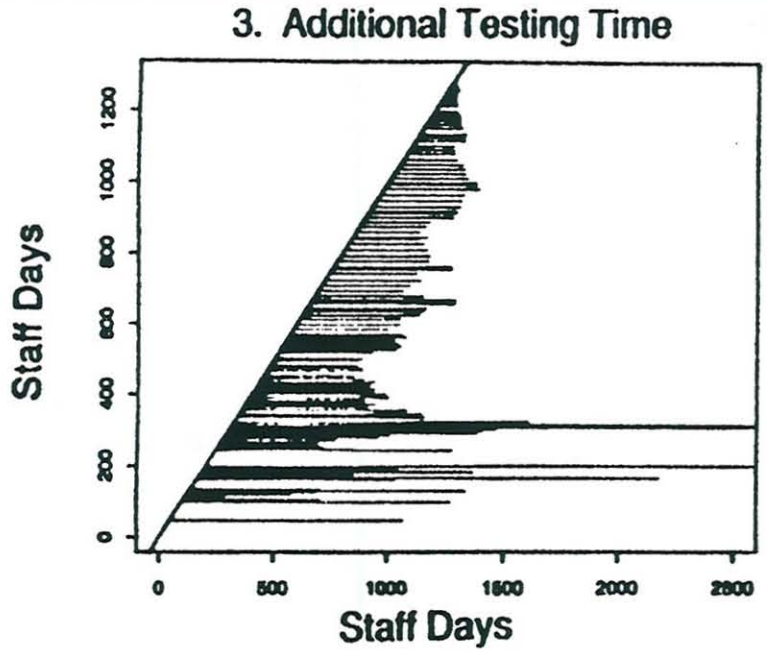
Software modules come to system test in batches.

2. Model Fit

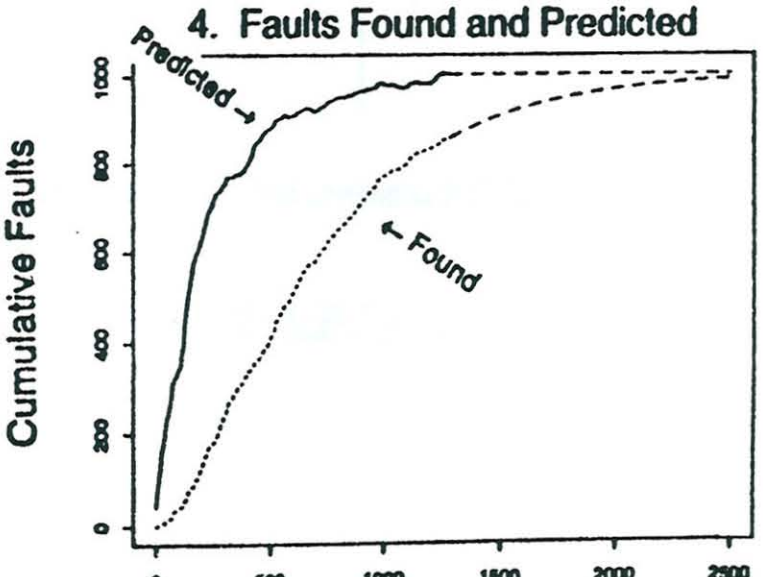


New statistical models accurately predict rate at which faults will be found during system test.

Fault Prediction

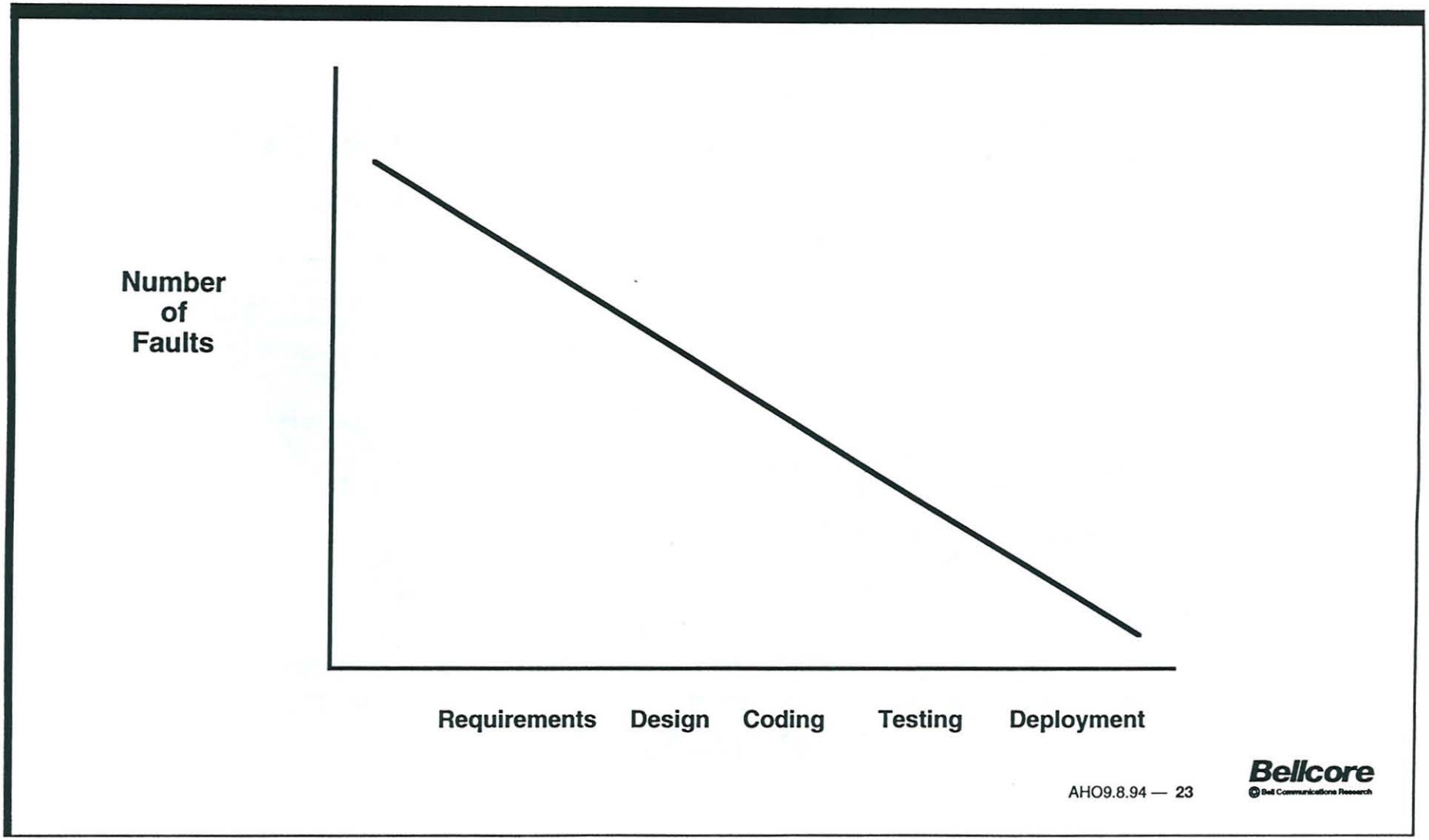


Model predicts additional amount of testing needed to attain quality goals.



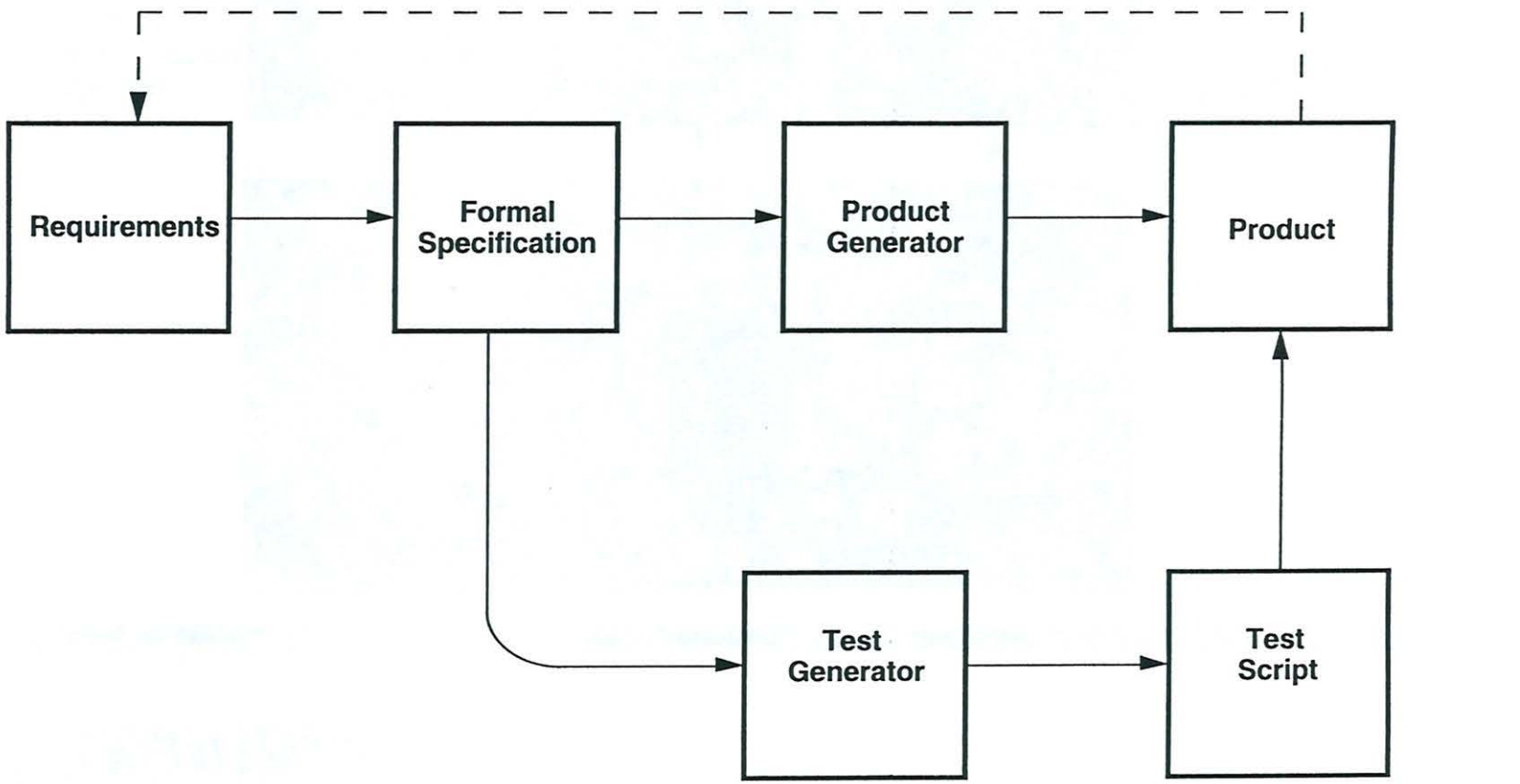
Ship code when quality goals are met.

Ideal Curve?



Bellcore
Bell Communications Research

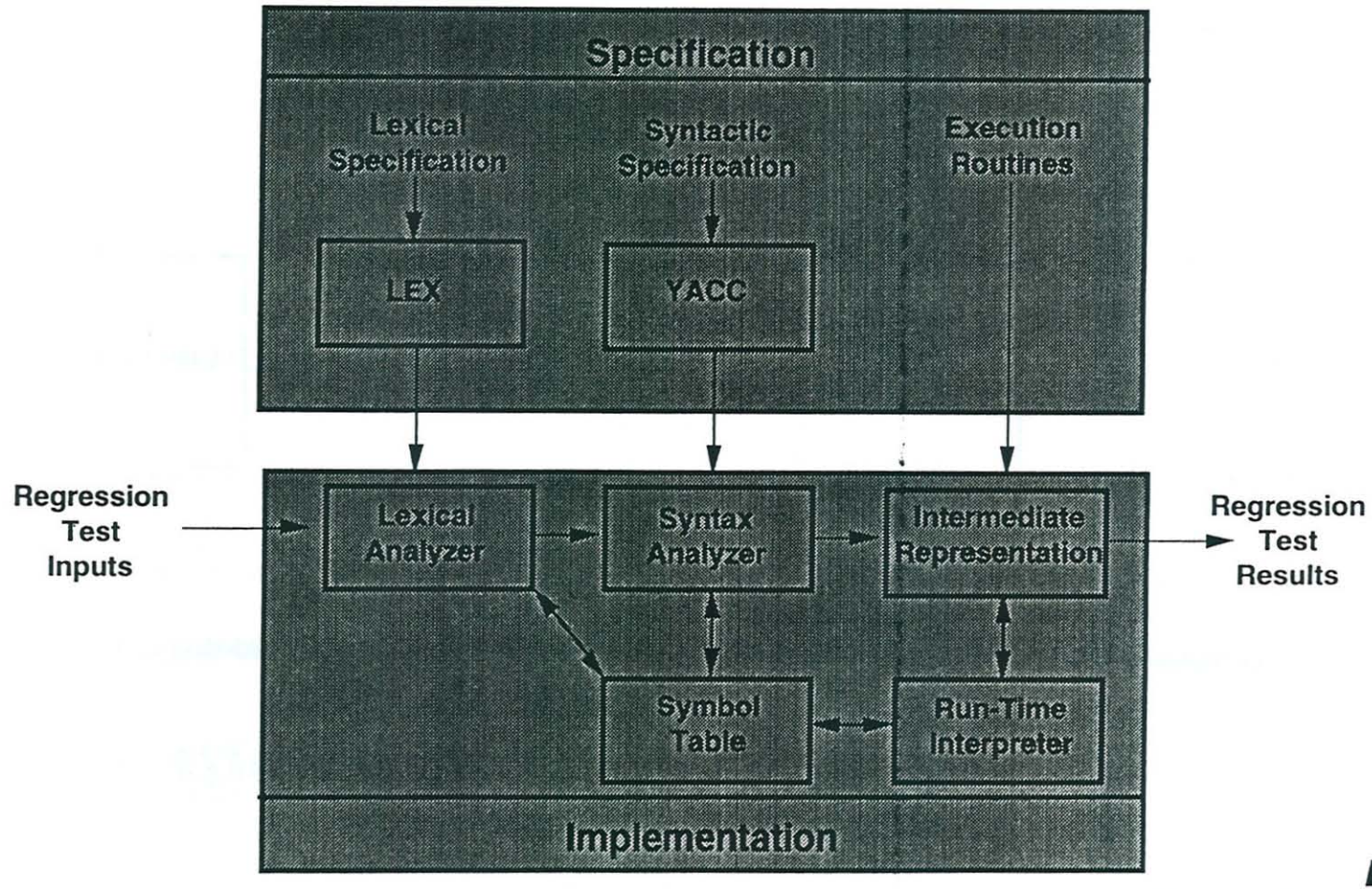
Closing the Loop



AHO9.8.94 — 24

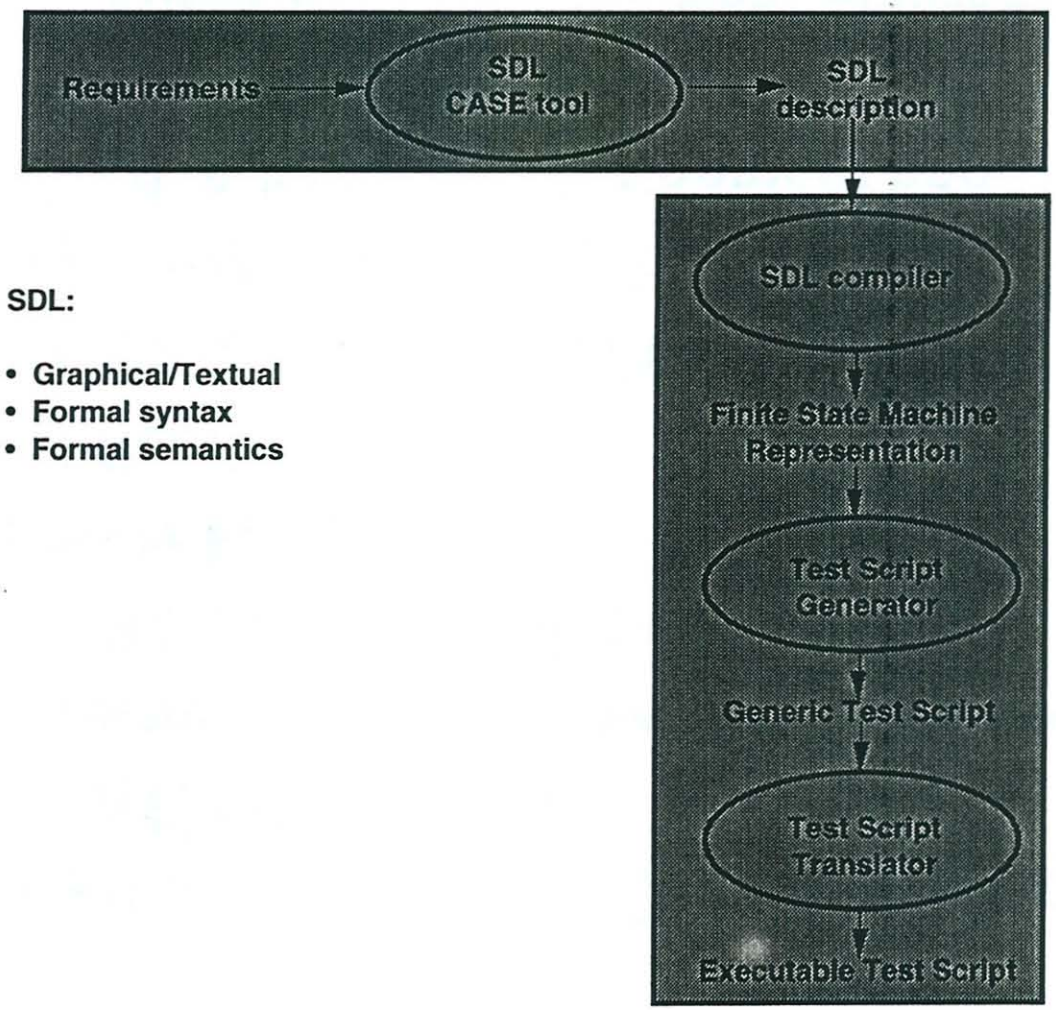
Bellcore
Bell Communications Research

AWK Example



Bellcore
Bell Communications Research

Formalizing and Automating Test Script Generation



SDL:

- Graphical/Textual
- Formal syntax
- Formal semantics

Bellcore
Bell Communications Research

Qualities of a World-Class Software Organization

1. Clear Vision, Goals and Roles
2. Customer Focus
3. Cost Effectiveness
4. Pervasive Quality Program
5. Agility in Achieving Time to Value
6. Standard, Advanced Development and Delivery Processes
7. Responsive Customer Service
8. Highly Automated Development Environment
9. Talented people Skilled in Application Domain and Software
10. Aggressive Technology Development and Application

Parting Challenge

How to improve software quality by a factor of:

- **1 to 2**
- **2 to 10**
- **10 or better**

Bellcore
Bell Communications Research

How Reliable Can We Make Software?

Alfred V. Aho
Morristown, NJ - USA
September 8, 1994

DISCUSSION

Rapporteur: Francisco Vilar Brasileiro

Lecture One

Dr Sventek referred to the R&D research agenda with which Dr Aho finished his talk, and asked which points would be more likely to provide the solutions to the problems of the global information infrastructure. Dr Aho replied that probably multimedia, and personal communications will be the main drivers of such solutions.

Professor Brailsford pointed out that the traffic in the Internet is steadily increasing, and raised concerns on how the service would be charged. Dr Aho reminded that this is the same kind of problem that the providers of telephone services experienced in the past, and suggested that an analogous solution could be applied for the case of network service providers. Mr. Ainsworth asked if this problem was not going to become more serious, since Internet looked cheap mainly because of its inter-elasticity, which may be much lighter in the future. Dr Aho replied that new technologies that are coming about will help to reduce traffic. He mentioned a project that tries to avoid unwanted information reaching the user, so that instead of having the user looking for the information, the pertinent information would go to the user.

Lecture Two

Dr Sventek remarked that the development of both hardware, and software systems required people, process, and technology, nevertheless hardware development had been much more successful. Professor Randell raised the issue of scale, which largely differentiates the aims of the two development processes.

Professor Morrison asked if it was proper to expect that the requirements of software systems would not change during the development process, specially when governments are the clients. Dr Aho replied that the problem is not in having the requirements changed while the developing process is in progress, but in failing to re-negotiate the effects of such changes, in the cost, and delivery expectation of the project. He remarked that the re-negotiation process is easier to be realised when the client is not the government. Mr. Ainsworth pointed out that normally if the development process takes more than one year, there will be fundamental changes in the requirements which could not be foreseen when the initial contract was firmed. So, the real issue is how to live with this situation.

Professor Randell remarked that there are several techniques for developing redundant software, which can be cost-effective.

Professor Marneffe asked if the inspection of the software is performed by reading the source code. Dr Aho replied that inspection involves reading the code and seeing if it matches the specification, but it is normally restricted to critical sections of the code. He also remarked that inspection can achieve very good results if performed by experienced people.

Professor Randell remarked that it is not code quality that must be measured, but the implied system reliability.

Professor Ercoli remarked that there is a big semantic gap between what one means by, and what one understands by, and therefore what is needed is to teach people to express themselves better.