Real Logic
ACCELERATING SOFTWARE

Engineering You

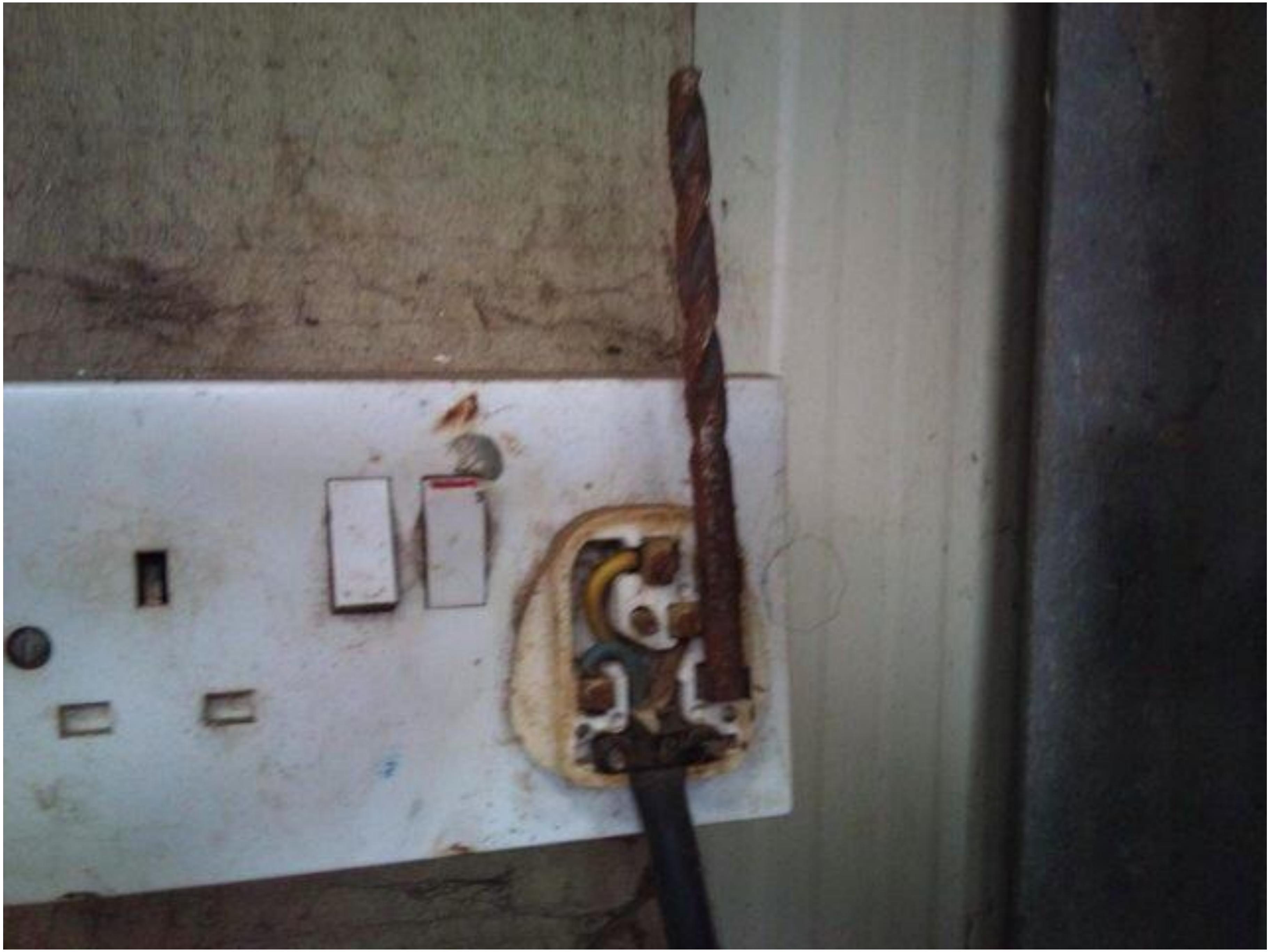
Martin Thompson - @mjpt777

"A software system can best be designed if the testing is interlaced with the design instead of being used after the design"

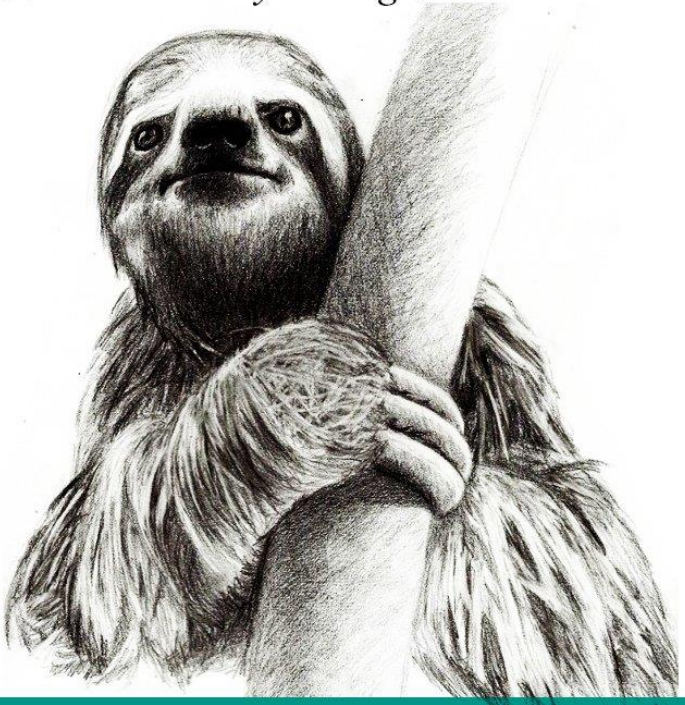
- Who and When???

"A software system can best be designed if the testing is interlaced with the design instead of being used after the design"

- *A.* J. Perlis (1968)



Cutting corners to meet arbitrary management deadlines



Essential

Copying and Pasting from Stack Overflow

O'REILLY®

The Practical Developer
@ThePracticalDev

How many generations of programmers have we?

ISO 27001

CMM

Welcome to the era of Software Alchemy

Engineering

The term ***Engineering*** is derived from the Latin *ingenium*, meaning "**cleverness**" and *ingeniare*, meaning "**to contrive, devise**".

Source: Wikipedia

Circa 1300

“***One who operates an engine***”,

where engine is a military machine such as a catapult.

Source: Wikipedia

Later the term “**Civil Engineering**” was introduced to cover those specialising in non-military projects

Source: Wikipedia

Engineers must work within constraints.

Constraints may include available resources, physical, imaginative or technical limitations, flexibility for future modifications and additions, and other factors, such as requirements for cost, safety, marketability, productivity, and serviceability.
By understanding the constraints, engineers derive specifications for the limits within which a viable object or system may be produced and operated.

Source: Wikipedia

Engineers must work within constraints. Constraints may include available resources, physical, imaginative or technical limitations, flexibility for future modifications and additions, and other factors, such as requirements for cost, safety, marketability, productivity, and serviceability.
By understanding the constraints, engineers derive specifications for the limits within which a viable object or system may be produced and operated.

Source: Wikipedia

Engineers must work within constraints. Constraints may include available resources, physical, imaginative or technical limitations, **flexibility for future modifications and additions, and other factors**, such as requirements for cost, safety, marketability, productivity, and serviceability.
By understanding the constraints, engineers derive specifications for the limits within which a viable object or system may be produced and operated.

Source: Wikipedia

Engineers must work within constraints. Constraints may include available resources, physical, imaginative or technical limitations, flexibility for future modifications and additions, and other factors, such as requirements for cost, **safety**, marketability, productivity, and serviceability. By understanding the constraints, engineers derive specifications for the limits within which a viable object or system may be produced and operated.

Source: Wikipedia

Engineers must work within constraints. Constraints may include available resources, physical, imaginative or technical limitations, flexibility for future modifications and additions, and other factors, such as requirements for cost, safety, marketability, productivity, and serviceability.

By understanding the constraints, engineers derive specifications for the limits within which a viable object or system may be produced and operated.

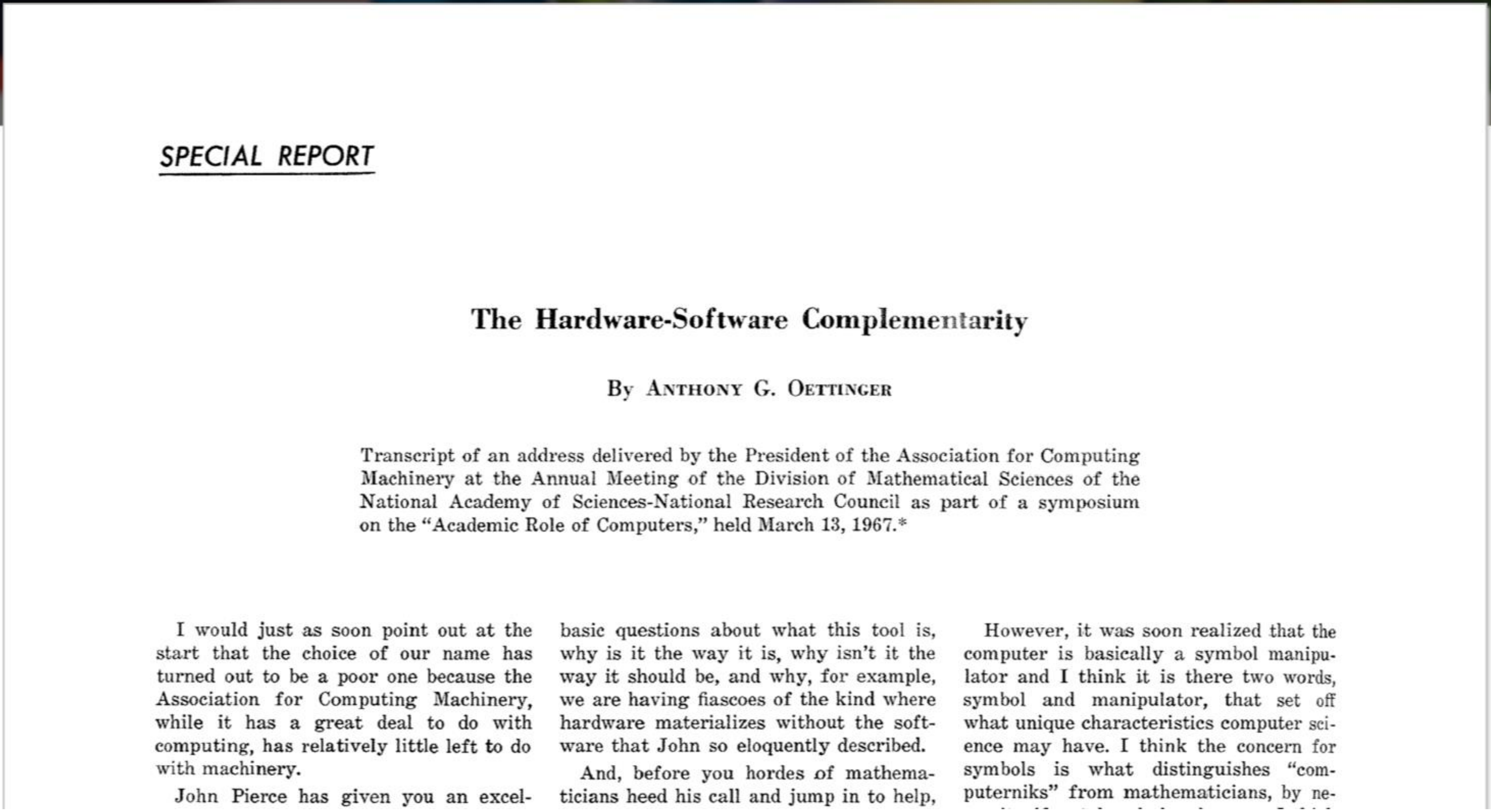
Source: Wikipedia

"Scientists investigate that which already is;
Engineers create that which has never been."

- Albert Einstein

“Software Engineering”?

http://dl.acm.org/citation.cfm?doid=363717.363722



SPECIAL REPORT

The Hardware-Software Complementarity

By Anthony G. Oettinger

Transcript of an address delivered by the President of the Association for Computing Machinery at the Annual Meeting of the Division of Mathematical Sciences of the National Academy of Sciences-National Research Council as part of a symposium on the "Academic Role of Computers," held March 13, 1967.*

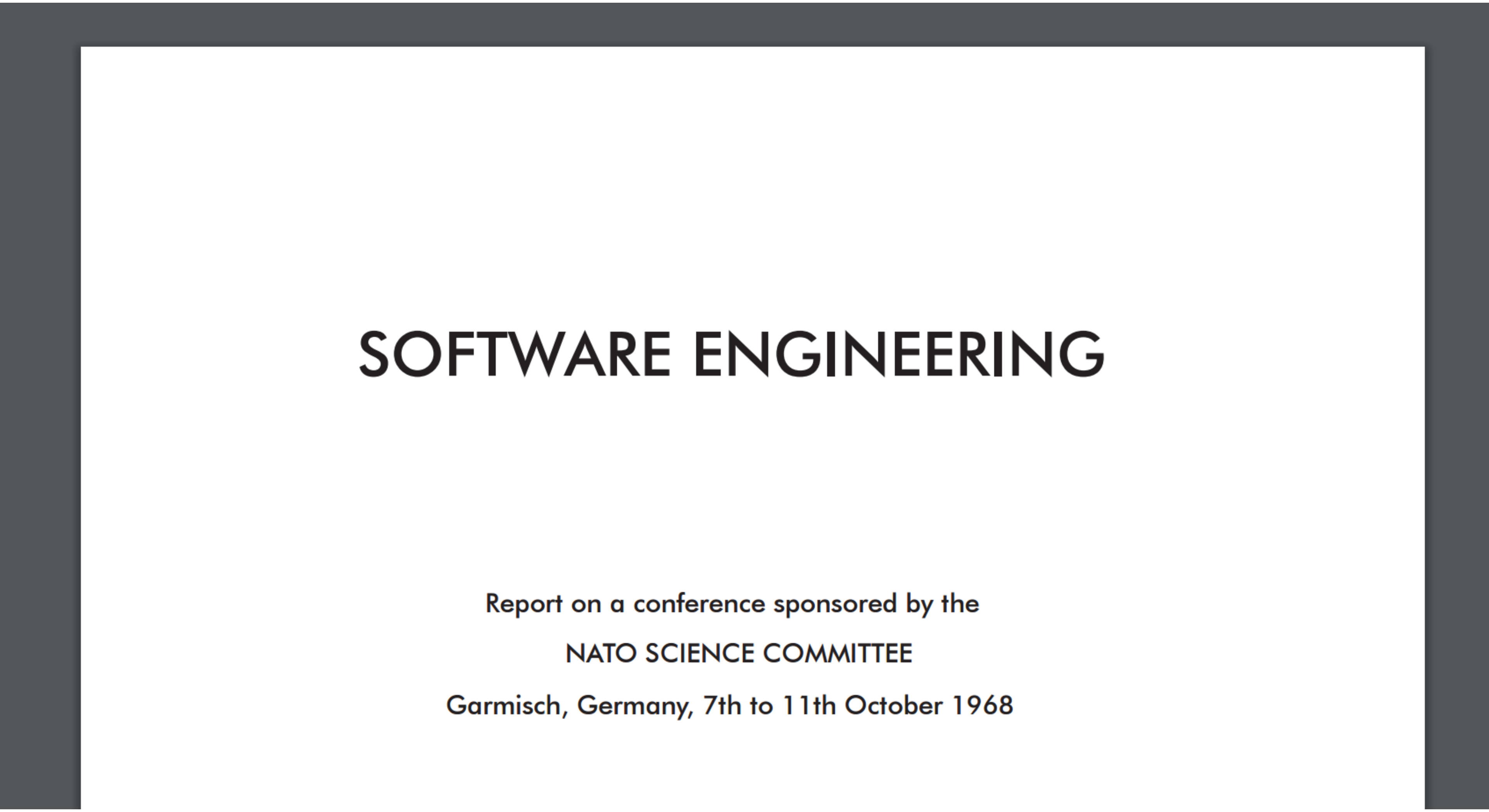
I would just as soon point out at the start that the choice of our name has turned out to be a poor one because the Association for Computing Machinery, while it has a great deal to do with computing, has relatively little left to do with machinery.

John Pierce has given you an excel- basic questions about what this tool is, why is it the way it is, why isn't it the way it should be, and why, for example, we are having fiascoes of the kind where hardware materializes without the software that John so eloquently described.

And, before you hordes of mathematicians heed his call and jump in to help,

However, it was soon realized that the computer is basically a symbol manipulator and I think it is there two words, symbol and manipulator, that set off what unique characteristics computer science may have. I think the concern for symbols is what distinguishes "computerniks" from mathematicians, by ne-

http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF



SOFTWARE ENGINEERING

Report on a conference sponsored by the

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7th to 11th October 1968

The design process is an iterative one:

1. Flowchart until you think you understand the problem.
2. Write code until you realize that you don’t.
3. Go back and re-do the flowchart.
4. Write some more code and iterate to what you feel is the correct solution.

I just want to make the point that **reliability really is a design issue**, in the sense that unless you are conscious of the need for reliability throughout the design, you might as well give up.

The good systems that are presently working were written by **small groups**. More than twenty programmers working on a project is usually disastrous.

Begin with skeletal coding:

Rather than aiming at finished code, **the 46 first coding steps should be aimed at exploring** interfaces, sizes of critical modules, complexity, and adequacy of the modules […].

Some critical items should be checked out.

Another interesting concept we might apply is that used in the Air Force, **to fly a number of hours each month, in order to retain one's 'wings'**.[…] In a situation where code actually has to be produced, nobody should be allowed in the system who doesn't write some given number of **lines of code per month**.

Many of the people who design software refer to users as ‘they’, ‘them’. They are some odd breed of cats living there in the outer world, knowing nothing, to whom nothing is owed. **Most of the designers of manufacturers’ software are designing**, **I think, for their own benefit** — they are literally playing games. They have no conception of validating their design before sending it out, or even evaluating the design in the light of potential use.

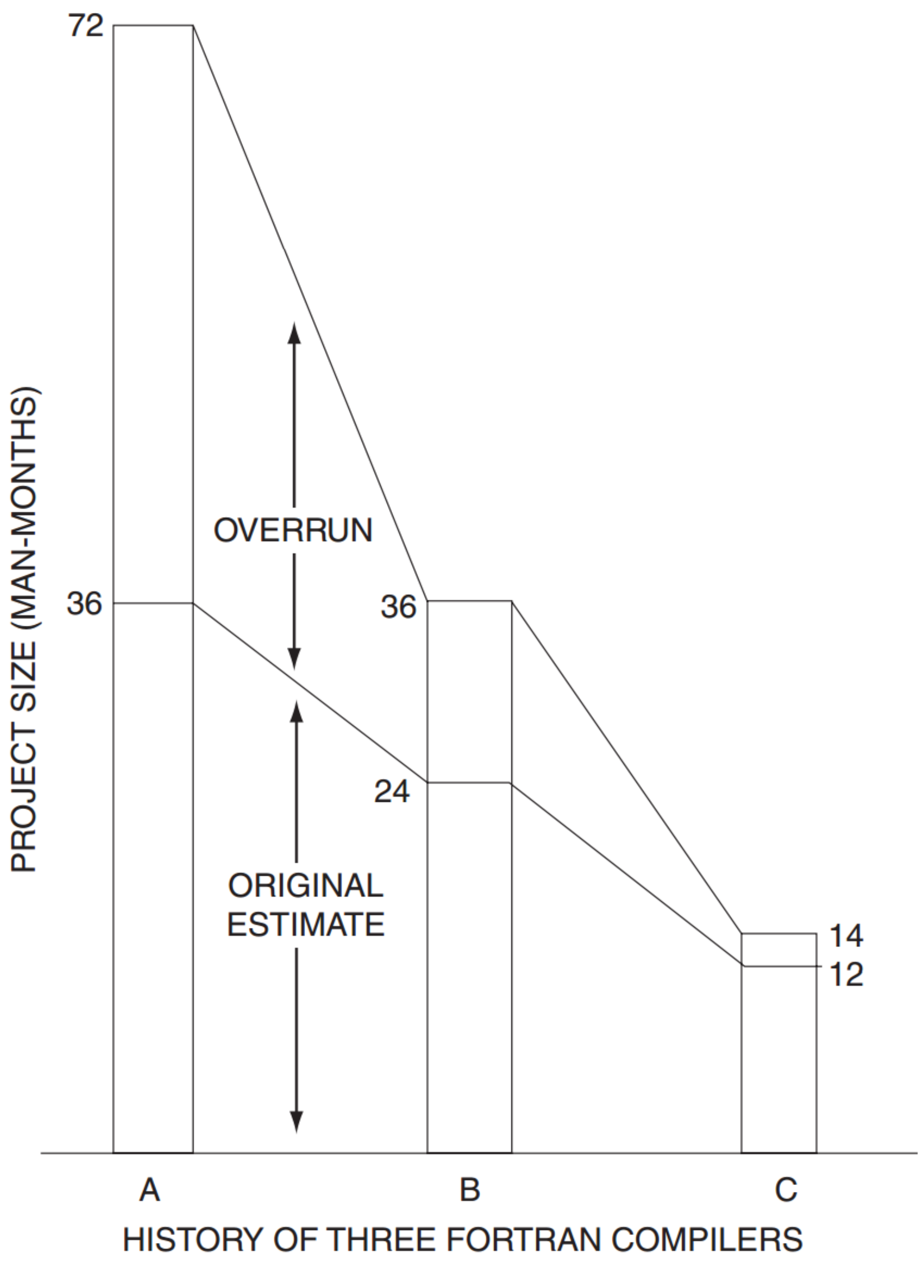


Figure 7. Provided by McClure

http://www.cs.utexas.edu/~EWD/transcriptions/EWD10xx/EWD1036.html

On the cruelty of really teaching computing science

- Edsger W. Dijkstra

Radical Novelty

One has to approach the radical novelty with a blank mind, consciously refusing to try to link it with what is already familiar, because the familiar is hopelessly inadequate.

Any one who has learned quantum mechanics knows what I am talking about.

Earlier scientific examples are the theory of relativity and quantum mechanics; later technological examples are the atom bomb and the pill.

Divide and Rule

Decomposition on an unprecedented scale

Amplification of Changes

Changing a single bit can have the most drastic consequences

We are all a product of our own experiences

Uncomfortable Truth

What? Where? How?

What should you learn?

What should you learn?

Algorithms & Data Structures

What should you learn?

Algorithms & Data Structures

Design Fundamentals

What should you learn?

Algorithms & Data Structures

Design Fundamentals

Programming Paradigms

What should you learn?

Algorithms & Data Structures

Design Fundamentals

Programming Paradigms

Decomposition & Abstraction

What should you learn?

Algorithms & Data Structures

Design Fundamentals

Programming Paradigms

Decomposition & Abstraction

Mathematics

What should you learn?

Algorithms & Data Structures

Design Fundamentals

Programming Paradigms

Decomposition & Abstraction

Mathematics

Business Domains

What should you learn?

Algorithms & Data Structures

Design Fundamentals

Programming Paradigms

Decomposition & Abstraction

Mathematics

Business Domains

*Communications**

From where can we learn?

From where can we learn?

Personal Practice

From where can we learn?

Personal Practice

People & Teams

From where can we learn?

Personal Practice

People & Teams

Research Papers

From where can we learn?

Personal Practice

People & Teams

Research Papers

Reading Code

From where can we learn?

Personal Practice

People & Teams

Research Papers

Reading Code

Projects – Tackle Unknowns First

From where can we learn?

Personal Practice

People & Teams

Research Papers

Reading Code

Projects – Tackle Unknowns First

Online Resources

How can we learn?

How can we learn?

Automate Repetitive Tasks

How can we learn?

Automate Repetitive Tasks

Focus on Feedback Cycles

How can we learn?

Automate Repetitive Tasks

Focus on Feedback Cycles

Experimentation

How can we learn?

Automate Repetitive Tasks

Focus on Feedback Cycles

Experimentation

Measure

How can we learn?

Automate Repetitive Tasks

Focus on Feedback Cycles

Experimentation

Measure

Apply Scientific Honesty

How can we learn?

Automate Repetitive Tasks

Focus on Feedback Cycles

Experimentation

Measure

Apply Scientific Honesty

Revisit & Refine

How can we learn?

Automate Repetitive Tasks

Focus on Feedback Cycles

Experimentation

Measure

Apply Scientific Honesty

Revisit & Refine

“What can go wrong?”

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Simple Testing Can Prevent Most Critical Failures: An Analysis of Production Failures in Distributed Data-Intensive Systems

Ding Yuan, Yu Luo, Xin Zhuang, Guilherme Renna Rodrigues, Xu Zhao, Yongle Zhang, Pranay U. Jain, and Michael Stumm, *University of Toronto*

https://www.usenix.org/conference/osdi14/technical-sessions/presentation/yuan

25% Ignored Errors

The War On Complexity

“Lines of code spent”

In Closing…

Don't feel bad…
We are living in the era of
Software Alchemy

Do epic shit,
or die trying.

Questions?

http://mechanical-sympathy.blogspot.com/

Twitter: @mjpt777

"It does not matter how intelligent you are,
if you guess and that guess cannot
be backed up by experimental evidence,
then it is still a guess."

- Richard Feynman