Then a miracle occurs:
The 2007 Stevens Lecture on Software Development Methods

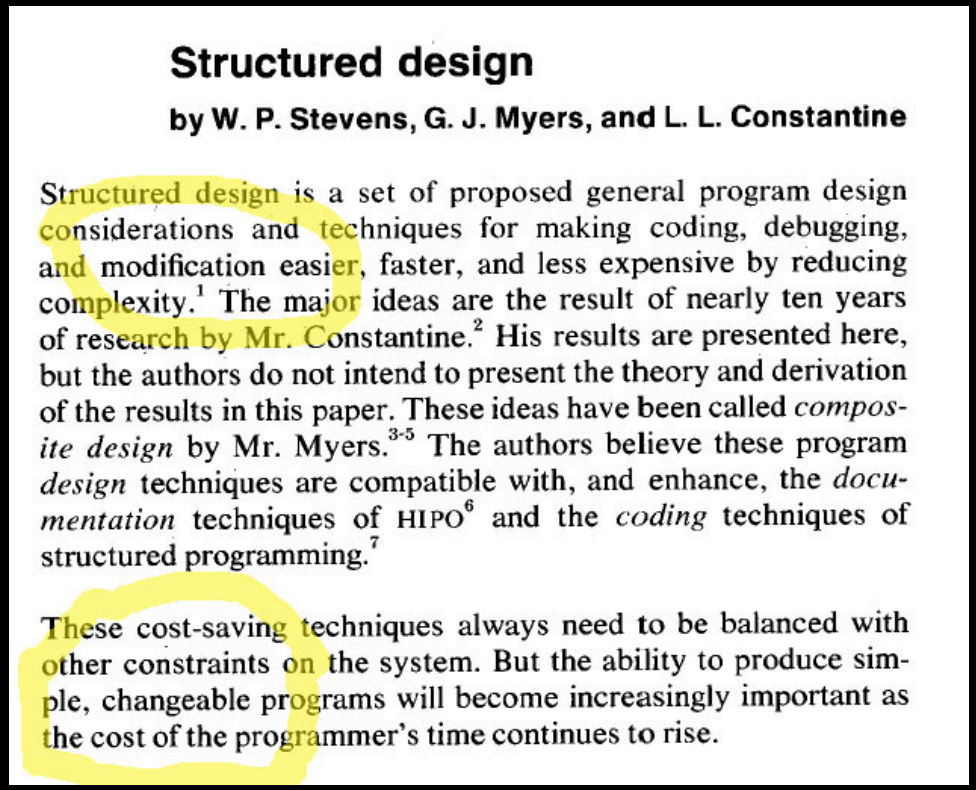
Nicholas Zvegintzov

zvegint@hotmail.com

11th European Conference on Software Maintenance and Reengineering

Thursday, March 22 2007, Amsterdam

- **Stevens, Myers, and Constantine: *Structured design***
  - W. P. Stevens, G. J. Myers, L. L. Constantine. Structured design. IBM Systems J., 13, 2, 1974, 115-139.



Structured design

by W. P. Stevens, G. J. Myers, and L. L. Constantine

Structured design is a set of proposed general program design considerations and techniques for making coding, debugging, and modification easier, faster, and less expensive by reducing complexity.[1] The major ideas are the result of nearly ten years of research by Mr. Constantine.[2] His results are presented here, but the authors do not intend to present the theory and derivation of the results in this paper. These ideas have been called *composite design* by Mr. Myers.[3-5] The authors believe these program *design* techniques are compatible with, and enhance, the *documentation* techniques of HIPO[6] and the *coding* techniques of structured programming.[7]

These cost-saving techniques always need to be balanced with other constraints on the system. But the ability to produce simple, changeable programs will become increasingly important as the cost of the programmer's time continues to rise.

It is becoming increasingly important to the data-processing industry to be able to produce more programming systems and produce them with fewer errors, at a faster rate, and in a way that modifications can be accomplished easily and quickly. Structured design considerations can help achieve this goal.

CITED REFERENCES AND FOOTNOTES

1. This method has not been submitted to any formal IBM test. Potential users should evaluate its usefulness in their own environment prior to implementation.

are 0, and its use in this paper was initiated by R. Ballow of the IBM Programming Productivity Techniques Department.

9. L. A. Belady and M. M. Lehman, *Programming System Dynamics or the Metadynamics of Systems in Maintenance and Growth*", RC 3546, IBM Thomas J. Watson Research Center, Yorktown Heights, New York (1971).

- What are good software maintenance methods?

- We already found good software development methods!

- We already found good software development methods

  - Well, we will by the end of the year for sure!

- We found good software development methods

Flowcharts!

- Well, we will by the end of the year for sure

- We found good software development methods

Flowcharts!

Structured Design!

- Well, we will by the end of the year for sure

- We found good software development methods

Flowcharts!
Structured Design!
Structured Programming!

  - Well, we will by the end of the year for sure

- We found good software development methods

Flowcharts!

Structured Design!

Structured Programming!

4GLs!

  - Well, we will by the end of the year for sure

- We found good software development methods

Client-server!

Prototyping!

UML !

JAD! R A D !

Spiral development!

Well, we will by the end of the year for sure

- We found good software development methods

Object oriented!

patterns!

Agile methods! Formal methods!

ERP! ARCHITECTURES! OPEN SOURCE!

CMMI®!

- Well, we will by the end of the year for sure

- We found good software development methods

- Outsourcing!

Well, we will by the end of the year for sure

- **Wovon man nicht sprechen kann, darüber muss man babbeln**
- Waar men niet over kan spreken, daar moet men over babbelen
- **What one cannot speak about one must babble over**
- (What one cannot speak about one must keep silent over)

- **Are they in software maintenance activities?**
  - Functional enhancement
  - Correction
  - Adaptation to hardware and software changes
  - Software improvement
  - User support

- **No...**

- **Are they in software maintenance processes?**

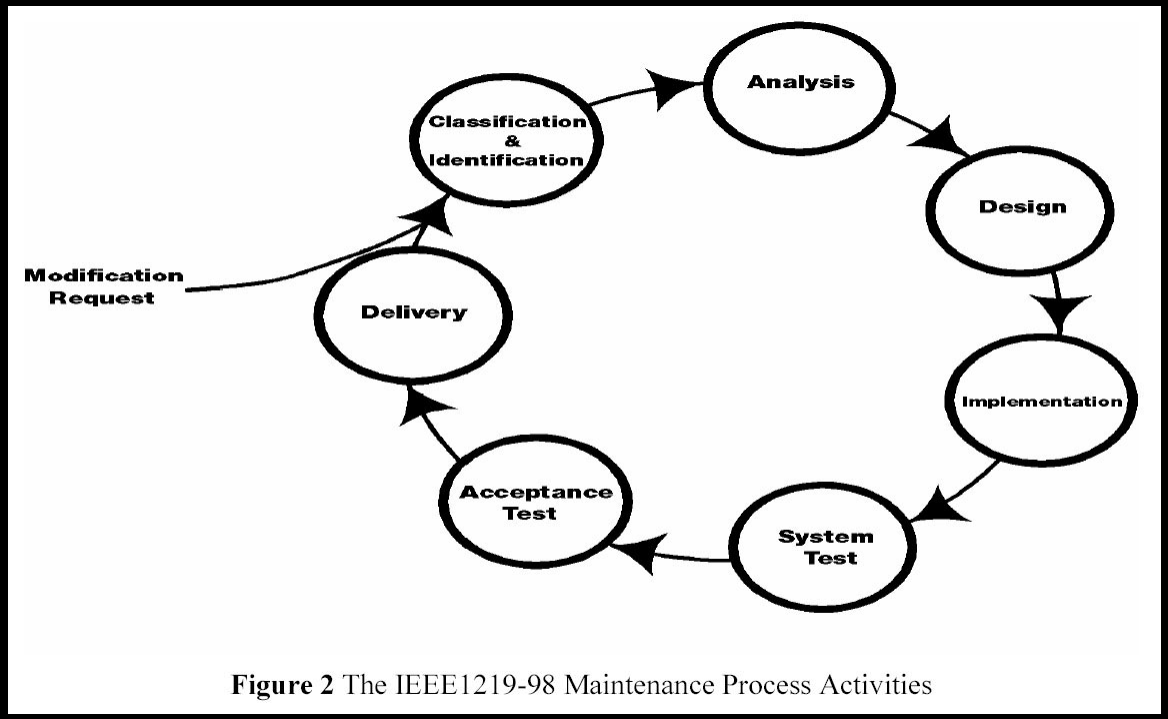


Figure 2 The IEEE1219-98 Maintenance Process Activities

No...

- **Are they in software maintenance controls?**
  - Service desk
  - Problem management
  - Change management
  - Configuration management
  - Verification and Validation
  - Release management

- No...

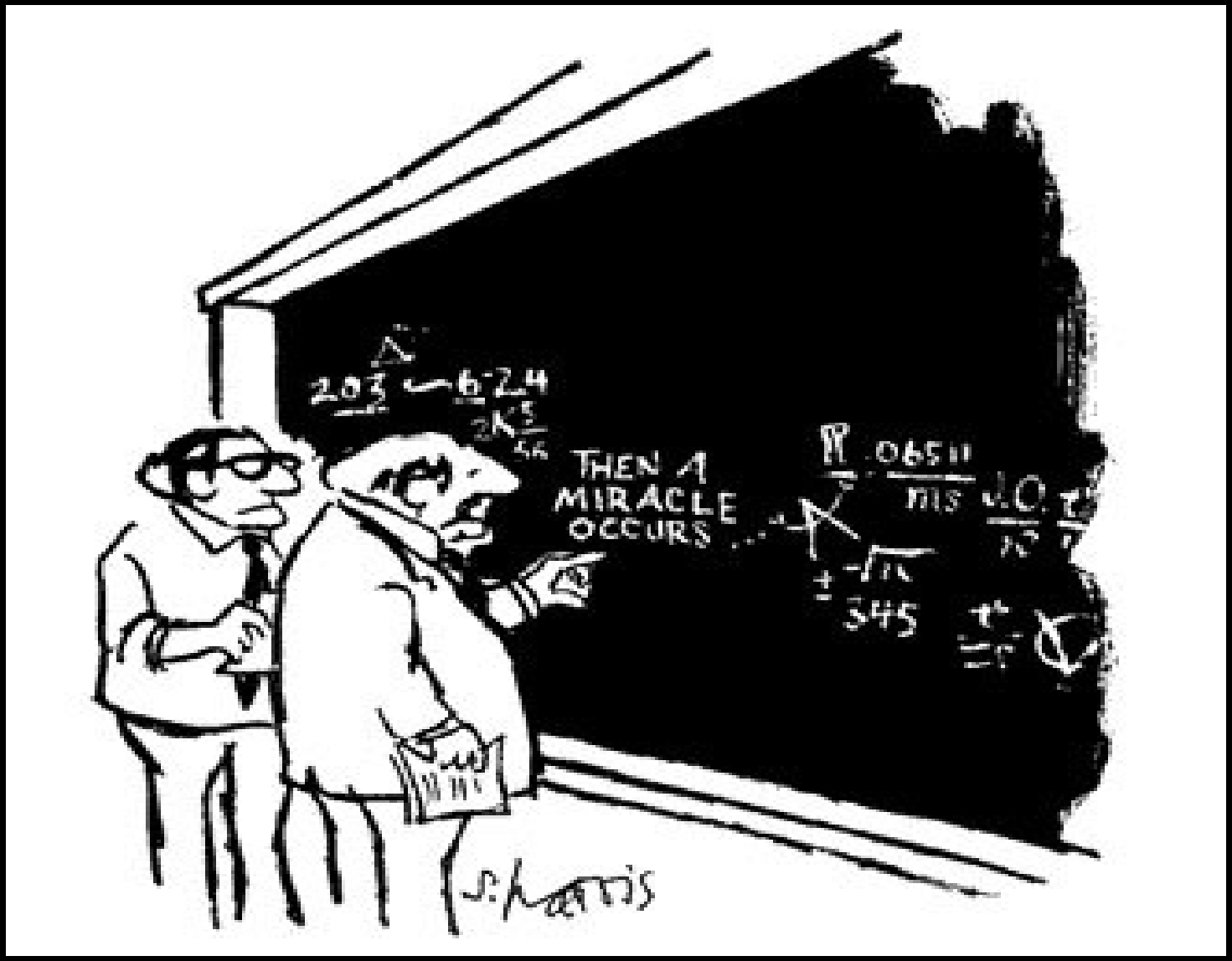
- Activity: Functional enhancement
- Given: A specification of a desired changed behavior

- The maintainer must: Change the implemented system to perform the desired specified behavior

- Activity: Functional enhancement
- Given: A specification of a desired changed behavior
- **(Methods)**
- The maintainer must: Change the implemented system to perform the desired specified behavior

- So what happened here?
  - I spoke.
  - Sound entered your ears, vibrated your cochleae, stimulated your auditory nerves.
  - You understood what was requested.
  - You decided to act.
  - Impulses went to your muscles.
  - You raised your hand.

- Where could we help this process?
  - I spoke.
  - ↑ Sound entered your ears, vibrated your cochleae, stimulated your auditory nerves.
  - ↓ You understood what was requested.
  - ↓ You decided to act.
  - ↑ Impulses went to your muscles.
  - ↑ You raised your hand.



Then a miracle occurs

“Then a miracle occurs” © Sidney Harris, originally appeared in American Scientist in 1977, used with permission.

- **What maintainers do**
  - Activity: Functional enhancement
  - Given: A specification of a desired changed behavior
  - **Miracle? Method?**
  - The maintainer must: Change the implemented system to perform the desired specified behavior

2.1.1. Limited understanding
[Dor02:v1c9s1.11.4; Pfl01:c11s11.3; Tak97:c3]

Limited understanding refers to how quickly a software engineer can understand where to make a change or a correction in software which this individual did not develop. Research indicates that some 40% to 60% of the maintenance effort is devoted to understanding the software to be modified. Thus, the topic of software comprehension is of great interest to software engineers.

Alain Abran and James W. Moore, Executive Editors.
Guide to the Software Engineering Body of Knowledge - SWEBOK®.
IEEE Computer Society, 2004. ISBN 0-7695-2330-7.

Limited Understanding. In addition to balancing user needs with software and hardware needs, the maintenance team deals with the limitations of human understanding. There is a limit to the rate at which a person can study documentation and extract material relevant to the problem being solved. Furthermore, we usually look for more clues than are really necessary for solving a problem. Adding the daily office distractions, we have a prescription for limited productivity.

Parikh and Zvegintzov (1983) report that 47% of software maintenance effort is devoted to understanding the software to be modified. This high figure is understandable when we consider the number of interfaces that need to be checked whenever a component is changed. For example, if a system has m components and we need to change k of them, there are

$$k * (m - k) + k * (k - 1)/2$$

Shari Lawrence Pfleeger: *Software engineering – Theory and practice*. Prentice-Hall, Inc., 1998. ISBN 0-13-147364-6.

Girish Parikh and Nicholas Zvegintzov. *Tutorial on software maintenance*. IEEE Computer Society Press, 1983. ISBN 0-8186-0002-0.

Programming Resources in the Maintenance Process

As we defined it, the maintenance process consisted of modification, enhancement, and correction of programs in the production library. In each of these activities of change to existing programs, understanding the intent and style of implementation of the original programmer was the major cause of time and difficulty in making the change.

CHART 6

CHANGE PROCESS

	MODIFICATIONS & ENHANCEMENTS (%)	CORRECTIONS (%)
DEFINE & UNDERSTAND THE CHANGE	18	25
REVIEW DOCUMENTATION	6	4
TRACE LOGIC	23	33
IMPLEMENT CHANGE	19	15
TEST	28	20
UPDATE DOCUMENTATION	6	3

- 8 -

R. K. Fjeldstad and W. T. Hamlen.

Application program maintenance study - report to our respondents. In *GUIDE 48 Proceedings*, May 1979.

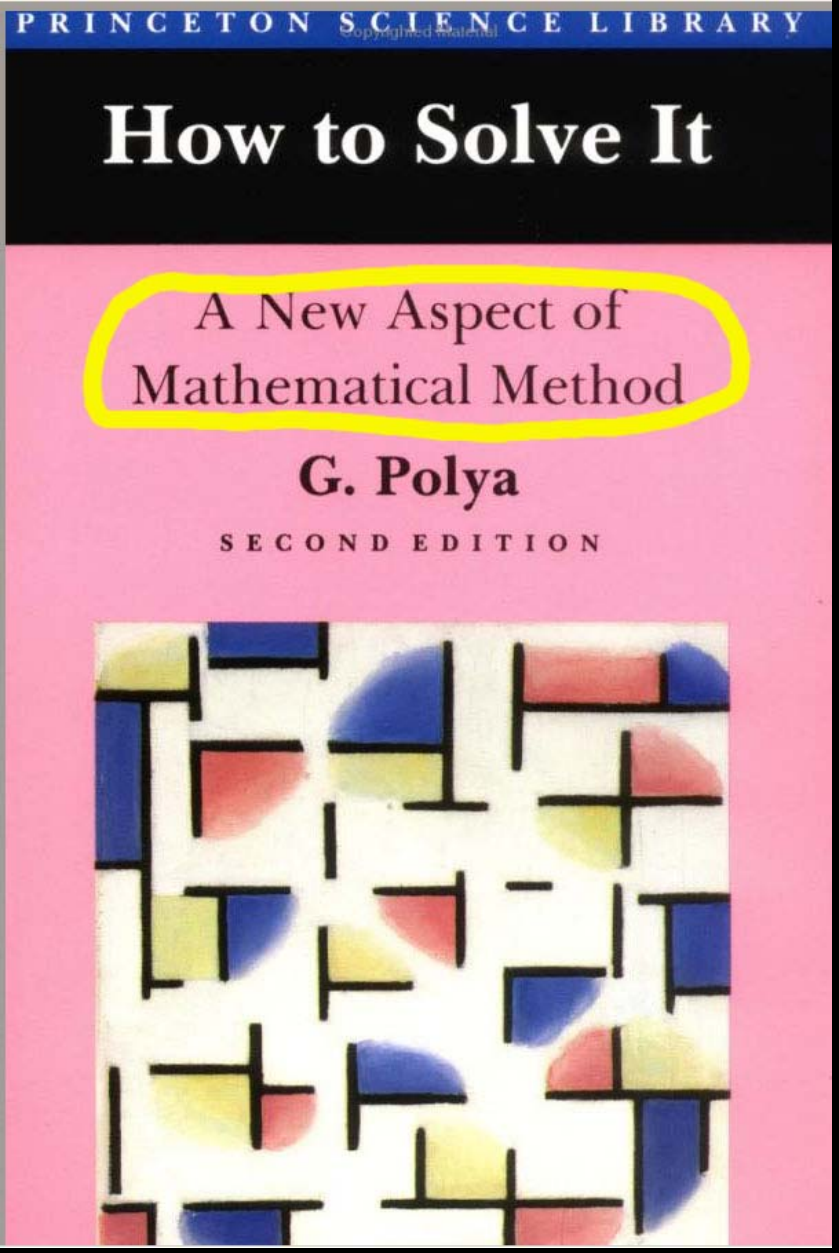
- **What maintainers do**
  - Activity: Functional enhancement
  - Given: A specification of a desired changed behavior
  - **Miracle? Method?**
  - The maintainer must: Change the implemented system to perform the desired specified behavior

- **What maintainers do**
  - Activity: Functional enhancement
  - Given: A specification of a desired changed behavior
  - **Problem-solving, applied reasoning**
  - The maintainer must: Change the implemented system to perform the desired specified behavior

Problem-solving	
Know materials	
Know the capabilities of materials	
Define the goal	
Visualize actions, sequence of steps	
Carry out steps	
Assess	

Problem-solving	Cooking
Know materials	Vegetables, meats, seasonings
Know the capabilities of materials	Taste, peeling, chopping, cooking, decoration, etc.
Define the goal	The dish
Visualize actions, sequence of steps	The recipe
Carry out steps	Prep, mix, cook, garnish
Assess	Eat

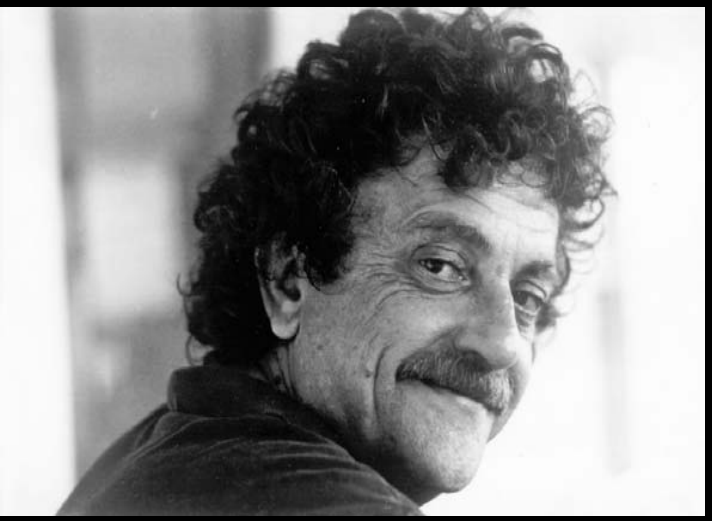
Problem-solving	Modify software
Know materials	Programming and data language(s)
Know the capabilities of materials	Syntax, semantics
Define the goal	Change request
Visualize actions, sequence of steps	Design
Carry out steps	Programming
Assess	Exercise and test



George Polya. *How to solve it: A new aspect of mathematical method*. Princeton University Press, 1945. ISBN 0-691-08097-6.

Problem-solving	Learn problem-solving
Know materials	Study, use
Know the capabilities of materials	Study, use, observation
Define the goal	Observation, practice, critique
Visualize actions, sequence of steps	Observation, practice, critique
Carry out steps	Observation, practice, critique
Assess	Experience, discipline

- A wrang-wrang (definition):
  - “A person who steers people away from a line of speculation by reducing that line, with the example of the wrang-wrang's own life, to an absurdity”



Kurt Vonnegut.

Cat’s cradle.


WEEKLY WORLD
NEWS®
September 15, 1998 $1.39 U.S. $1.59 CANADA/70p U.K.
THE COMPUTER CRASH OF THE MILLENNIUM!
JANUARY 1, 2000
THE DAY THE EARTH WILL STAND STILL!
ALL BANKS WILL FAIL!
FOOD SUPPLIES WILL BE DEPLETED!
ELECTRICITY WILL BE CUT OFF!
THE STOCK MARKET WILL CRASH!
VEHICLES USING COMPUTER CHIPS WILL STOP DEAD!
TELEPHONES WILL CEASE TO FUNCTION!
DOMINO EFFECT WILL CAUSE A WORLDWIDE DEPRESSION!
37>
0 74820 08101 3

Y2K!

- What are good software maintenance methods?
  - We keep silent
  - We babble

- What are good software maintenance methods?
- What are good software development methods?
  - We keep silent
  - We babble

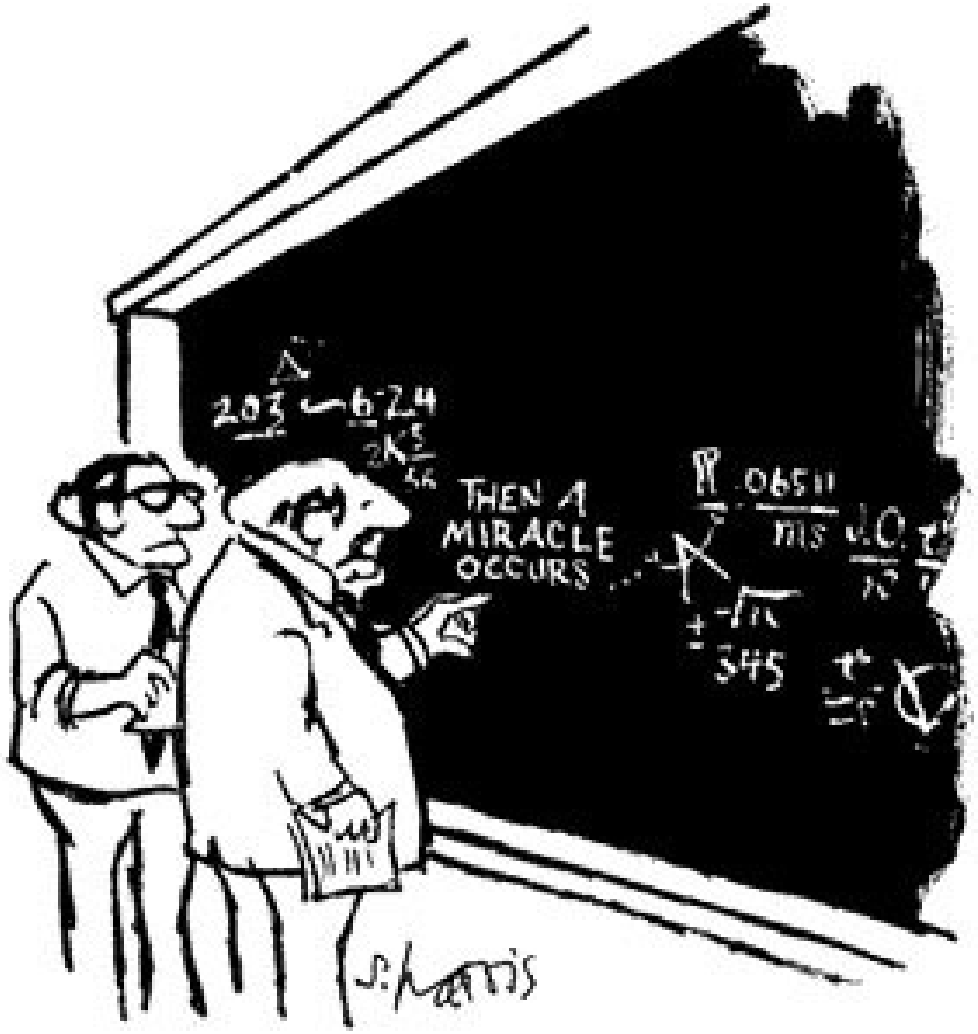
- We found good software development methods ;-)
  - New languages
  - Bragging over new languages
  - More development will solve the problems of existing development
  - “Design” means decomposition and diagramming

Analysis...

Problem-solving...

Applied reasoning...

- What are good software maintenance methods?
  - We should not remain silent
  - We should not babble



"I THINK YOU SHOULD BE MORE EXPLICIT HERE IN STEP TWO."

"Then a miracle occurs" © Sidney Harris, originally appeared in American Scientist in 1977, used with permission.

- What are good software maintenance methods?
  - We should not remain silent
  - We should not babble
  - We should be more explicit here in step 2

The 2007 Stevens Lecture on Software Development Methods

“Then a miracle occurs”

“Dan gebeurt er een wonder”

Thank you

Dank u