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
Then a miracle occurs

Stevens, Myers, and Constantine: *Structured design*


**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. The major ideas are the result of nearly ten years of research by Mr. Constantine. 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. The authors believe these program design techniques are compatible with, and enhance, the documentation techniques of HIPO and the coding techniques of structured programming.

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.
Then a miracle occurs

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!
Then a miracle occurs

- 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!

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!
- RAD!
- Spiral development!

Well, we will by the end of the year for sure
Then a miracle occurs

- We found good software development methods

**Object oriented!**

**patterns!**

- **A gentle 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
Then a miracle occurs

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?**

No...

Then a miracle occurs

So – what **are** software maintenance methods?
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” © 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.
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 Zveigintzov (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

\[
k \times (m - k) + k \times (k - 1)/2
\]
Then a miracle occurs

What maintainers do
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.

**Charts 6**

**Change Process**

<table>
<thead>
<tr>
<th></th>
<th>Modifications &amp; Enhancements (%)</th>
<th>Corrections (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Define &amp; Understand The Change</strong></td>
<td>18</td>
<td>25</td>
</tr>
<tr>
<td><strong>Review Documentation</strong></td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td><strong>Trace Logic</strong></td>
<td>23</td>
<td>33</td>
</tr>
<tr>
<td><strong>Implement Change</strong></td>
<td>19</td>
<td>15</td>
</tr>
<tr>
<td><strong>Test</strong></td>
<td>28</td>
<td>20</td>
</tr>
<tr>
<td><strong>Update Documentation</strong></td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>


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**

<table>
<thead>
<tr>
<th>Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>Know materials</td>
</tr>
<tr>
<td>Know the capabilities of materials</td>
</tr>
<tr>
<td>Define the goal</td>
</tr>
<tr>
<td>Visualize actions, sequence of steps</td>
</tr>
<tr>
<td>Carry out steps</td>
</tr>
<tr>
<td>Assess</td>
</tr>
</tbody>
</table>

Then a miracle occurs
<table>
<thead>
<tr>
<th>Problem-solving</th>
<th>Cooking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Know materials</td>
<td>Vegetables, meats, seasonings</td>
</tr>
<tr>
<td>Know the capabilities of materials</td>
<td>Taste, peeling, chopping, cooking, decoration, etc.</td>
</tr>
<tr>
<td>Define the goal</td>
<td>The dish</td>
</tr>
<tr>
<td>Visualize actions, sequence of steps</td>
<td>The recipe</td>
</tr>
<tr>
<td>Carry out steps</td>
<td>Prep, mix, cook, garnish</td>
</tr>
<tr>
<td>Assess</td>
<td>Eat</td>
</tr>
<tr>
<td>Problem-solving</td>
<td>Modify software</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Know materials</td>
<td>Programming and data language(s)</td>
</tr>
<tr>
<td>Know the capabilities of materials</td>
<td>Syntax, semantics</td>
</tr>
<tr>
<td>Define the goal</td>
<td>Change request</td>
</tr>
<tr>
<td>Visualize actions, sequence of steps</td>
<td>Design</td>
</tr>
<tr>
<td>Carry out steps</td>
<td>Programming</td>
</tr>
<tr>
<td>Assess</td>
<td>Exercise and test</td>
</tr>
</tbody>
</table>

Then a miracle occurs
Then a miracle occurs


How to create and modify software
<table>
<thead>
<tr>
<th>Problem-solving</th>
<th>Learn problem-solving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Know materials</td>
<td>Study, use</td>
</tr>
<tr>
<td>Know the capabilities of materials</td>
<td>Study, use, observation</td>
</tr>
<tr>
<td>Define the goal</td>
<td>Observation, practice, critique</td>
</tr>
<tr>
<td>Visualize actions, sequence of steps</td>
<td>Observation, practice, critique</td>
</tr>
<tr>
<td>Carry out steps</td>
<td>Observation, practice, critique</td>
</tr>
<tr>
<td>Assess</td>
<td>Experience, discipline</td>
</tr>
</tbody>
</table>
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.
Then a miracle occurs

A software maintenance wrang-wrang
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
Then a miracle occurs

What are good software maintenance methods?

Analysis...
Problem-solving...
Applied reasoning...
What are good software maintenance methods?

- We should not remain silent
- We should not babble
Then a miracle occurs

“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?
What are good software maintenance methods?

- We should not remain silent
- We should not babble
- We should be more explicit here in step 2
“Then a miracle occurs”
“Dan gebeurt er een wonder”

Thank you
Dank u
Then a miracle occurs