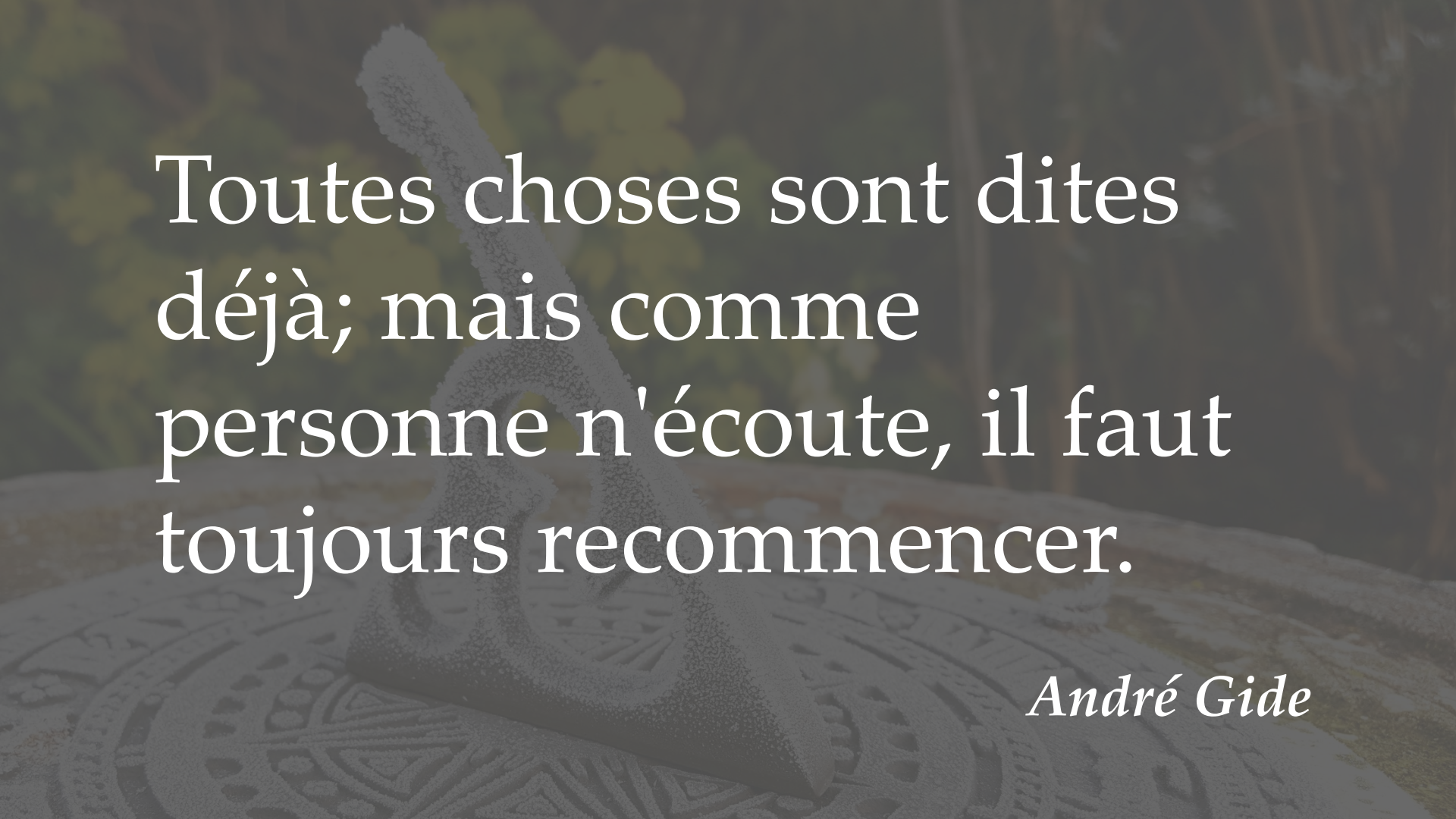




**Old Is the  
New New**

**@KevlinHenney**

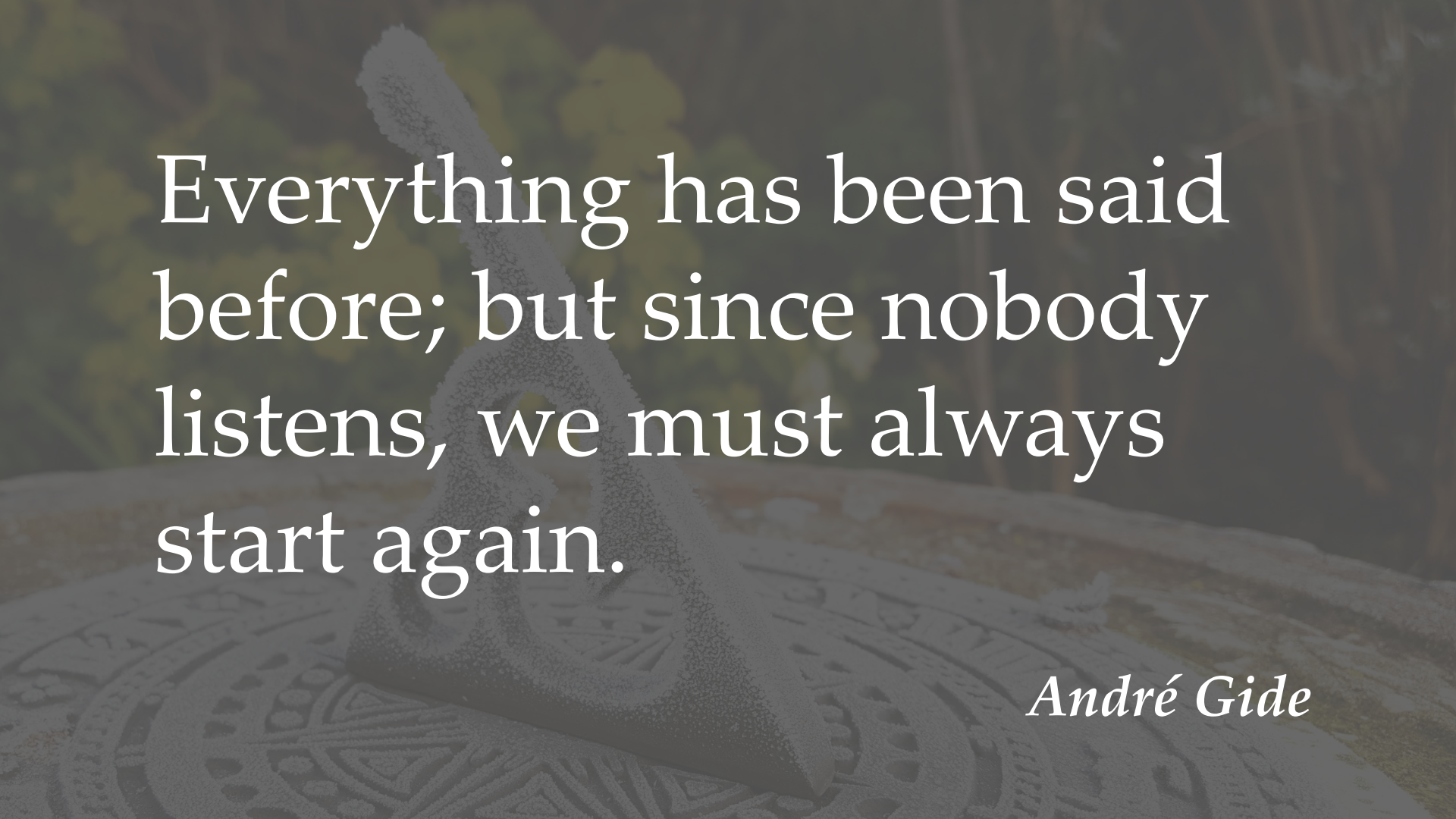




Toutes choses sont dites  
déjà; mais comme  
personne n'écoute, il faut  
toujours recommencer.

*André Gide*





Everything has been said  
before; but since nobody  
listens, we must always  
start again.

*André Gide*

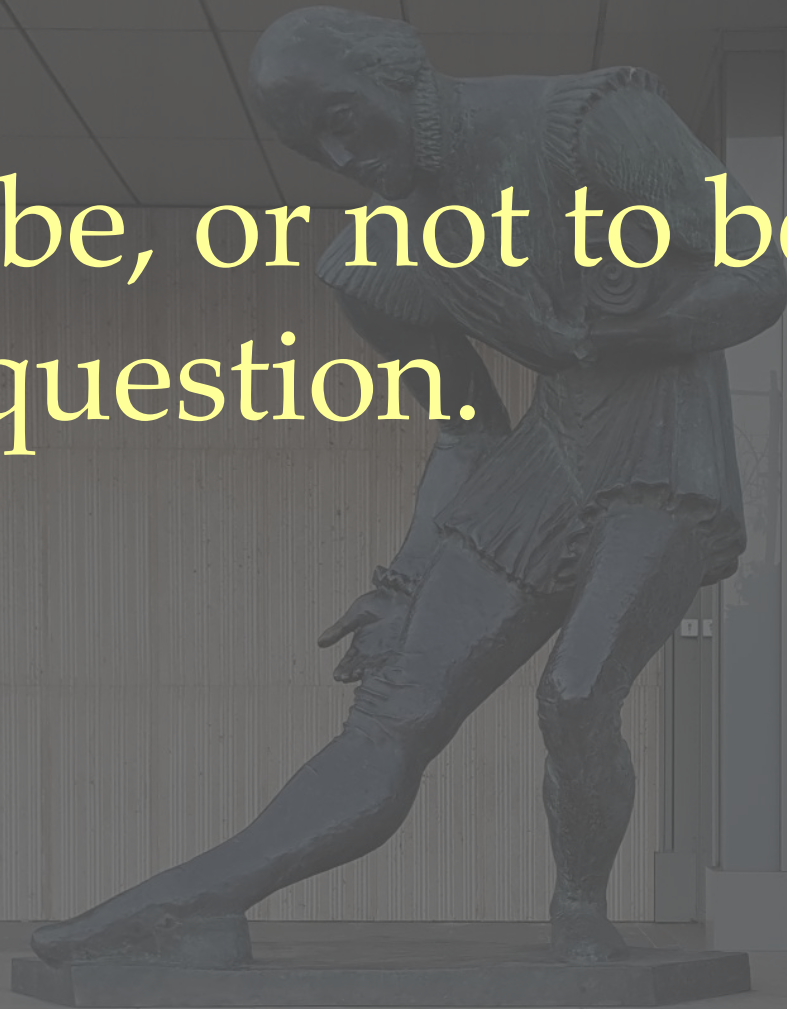




SHAKESPEARE



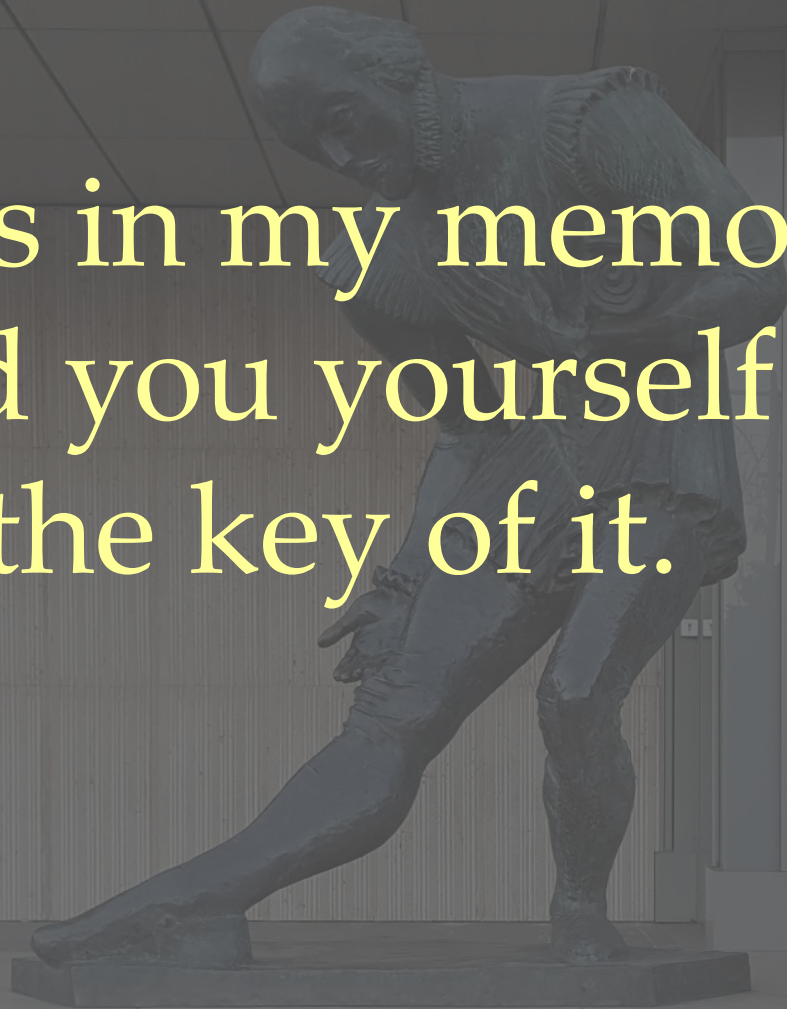
*Hamlet: To be, or not to be,  
that is the question.*



SHAKESPEARE



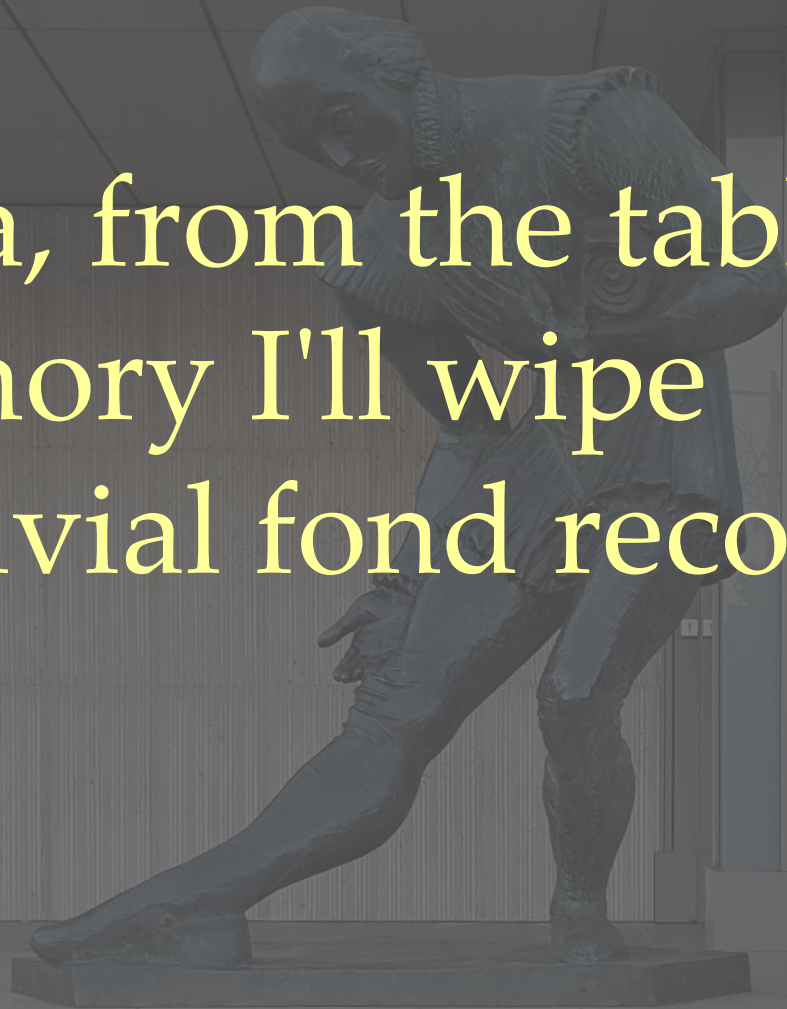
*Ophelia:* 'Tis in my memory  
locked, and you yourself  
shall keep the key of it.



SHAKESPEARE



*Hamlet: Yea, from the table  
of my memory I'll wipe  
away all trivial fond records.*





# **LISP 1.5 Programmer's Manual**

**The Computation Center  
and Research Laboratory of Electronics  
Massachusetts Institute of Technology**



# Revised Report on the Algorithmic Language **Algol 68**

Edited by

A. van Wijngaarden, B. J. Mailloux,

J. E. J. Back, C. H. A. Koster, M. Sintzoff





WILEY SERIES IN  
SOFTWARE DESIGN PATTERNS

# PATTERN-ORIENTED SOFTWARE ARCHITECTURE

**A Pattern Language for  
Distributed Computing**



**Volume 4**

Frank Buschmann  
Kevlin Henney  
Douglas C. Schmidt



WILEY SERIES IN  
SOFTWARE DESIGN PATTERNS

# PATTERN-ORIENTED SOFTWARE ARCHITECTURE

**On Patterns and Pattern Languages**



**Volume 5**

Frank Buschmann  
Kevlin Henney  
Douglas C. Schmidt



Share a Coke with

**Kevlin**

FROME VALLEY  
HEREFORDSHIRE

◆ HENNEY'S ◆  
DRY CIDER

MADE FROM 100% FRESH PRESSED JUICE

THE AGED 12 YEARS  
OF DUFFTOWN



SINGLETON

*Single Malt Scotch Whisky*  
*of Dufftown*

DUFFTOWN  
ESTD 1896

MATURED FOR  
12 YEARS

PRODUCT OF  
SCOTLAND

12



# Patterns Manifesto

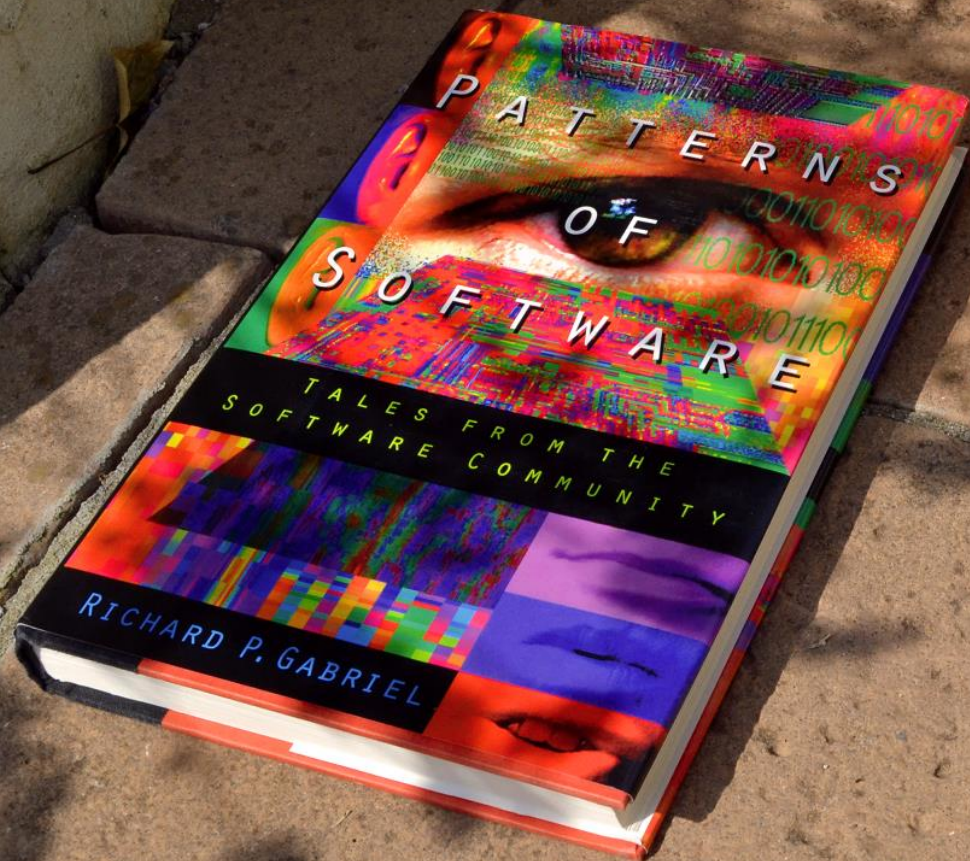
We are uncovering better ways of  
developing software by seeing  
how others have already done it.



Patterns are  
an aggressive  
disregard of  
originality.

Brian Foote





PATTERNS  
OF  
SOFTWARE

TALES FROM THE  
SOFTWARE COMMUNITY

RICHARD P. GABRIEL



**In 1990 I proposed a theory, called *Worse Is Better*, of why software would be more likely to succeed if it was developed with minimal invention. It is far better to have an underfeatured product that is rock solid, fast, and small than one that covers what an expert would consider the complete requirements.**



**Here are the characteristics of a worse-is-better software design:**

- ***Simplicity:*** The design is simple in implementation. The interface should be simple, but anything adequate will do.
- ***Completeness:*** The design covers only necessary situations. Completeness can be sacrificed in favor of any other quality.





```

#!/usr/bin/perl
# ----- PerlInterpreter
# PerlInterpreter must be the first line of the file.
#
# Copyright (c) 1995, Cunningham & Cunningham, Inc.
#
# This program has been generated by the HyperPerl
# generator. The source hypertext can be found
# at http://c2.com/cgi/wikibase. This program belongs
# to Cunningham & Cunningham, Inc., is to be used
# only by agreement with the owner, and then only
# with the understanding that the owner cannot be
# responsible for any behaviour of the program or
# any damages that it may cause.
# ----- InitialComments

```

```

# InitialComments
print "Content-type: text/html\n\n";
$DBM = "/usr/ward/$ScriptName";
dbmopen(%db, $DBM, 0666) || &AbortScript("can't open $DBM");
$CookedInput{browse} && &HandleBrowse;
$CookedInput{edit} && &HandleEdit;
$CookedInput{copy} && &HandleEdit;
$CookedInput{links} && &HandleLinks;
$CookedInput{search} && &HandleSearch;
dbmclose (%db);
if ($ENV{REQUEST_METHOD} eq POST) {
    $CookedInput{post} && &HandlePost;
}
# &DumpBinding(*CookedInput);
# &DumpBinding(*old);
# &DumpBinding(*ENV);
# ----- WikiInHyperPerl

```



Here are the characteristics of a worse-is-better software design:

- ***Simplicity:*** The design is simple in implementation. The interface should be simple, but anything adequate will do.
- ***Completeness:*** The design covers only necessary situations. Completeness can be sacrificed in favor of any other quality.
- ***Correctness:*** The design is correct in all observable aspects.
- ***Consistency:*** The design is consistent as far as it goes. Consistency is less of a problem because you always choose the smallest scope for the first implementation.



# SOFTWARE ENGINEERING

The design process  
is an iterative one.

Report on a conference sponsored by the

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7th to 11th October 1965

Andy Kinslow



# SOFTWARE ENGINEERING

Report on a conference sponsored by the

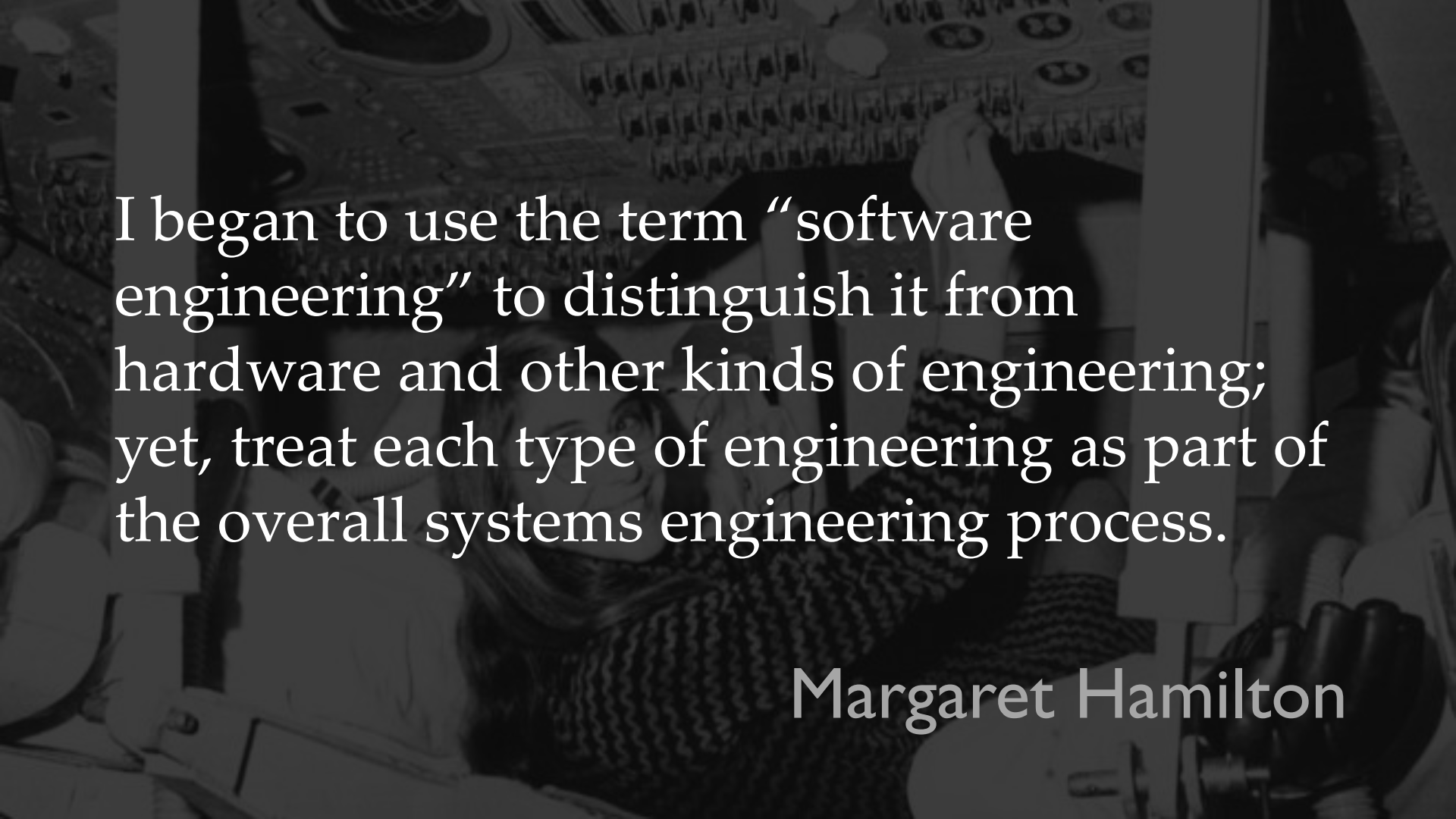
NATO SCIENCE COMMITTEE

Garmisch, Germany, 7th to 11th October 1968









I began to use the term “software engineering” to distinguish it from hardware and other kinds of engineering; yet, treat each type of engineering as part of the overall systems engineering process.

Margaret Hamilton



# SOFTWARE ENGINEERING

Report on a conference sponsored by the

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7th to 11th October 1968



# SOFTWARE ENGINEERING

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.

Report on a conference sponsored by the

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7th to 11th October 1968

Douglas Ross



# SOFTWARE ENGINEERING

And that is where most of  
our troubles come from.

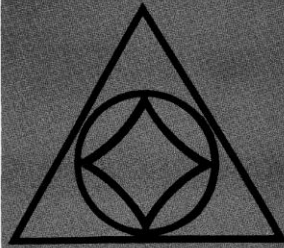
Report on a conference sponsored by the

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7th to 11th October 1968

Douglas Ross





THE THOMPSON BOOK COMPANY  
National Press Building  
Washington, D.C. 20004

# **AFIPS**

**CONFERENCE  
PROCEEDINGS**

**VOLUME 33  
PART ONE**

# **1968**

**FALL JOINT  
COMPUTER  
CONFERENCE**

December 9-11, 1968  
San Francisco, California

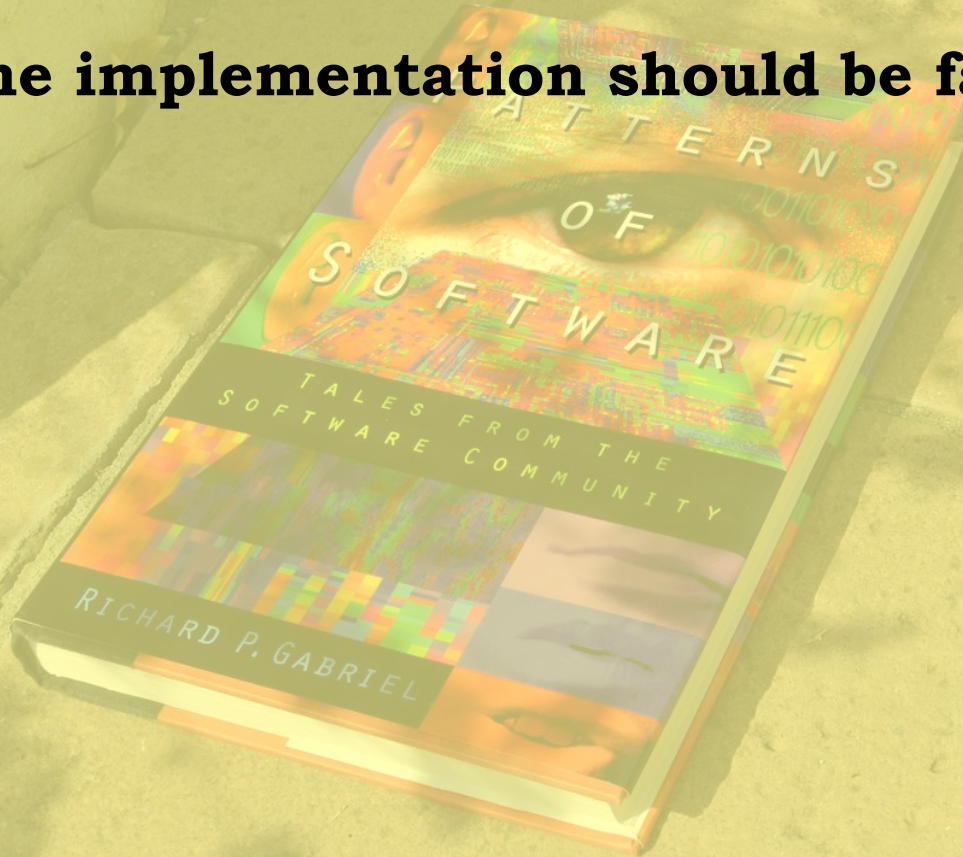


The key to a nationwide capability for computer analysis of medical signals.....	381	<i>C. H. Caceras, G. Kishner, D. E. Winer, A. L. Wehrer</i>
A legal structure for a national medical data center.....	387	<i>R. N. Freed</i>
<b>A RESEARCH CENTER FOR AUGMENTING HUMAN INTELLECT</b>		
A research center for augmenting human intellect.....	395	<i>D. C. Engelbart, W. K. English</i>
<b>PLANNING MODELS FOR MANAGEMENT</b>		
An approach to simulation model development for improved planning..	411	<i>J. McKenney</i>
Operations research in a conversational environment.....	417	<i>M. Conners</i>
MEDIAC—On-line media planning system.....	425	<i>L. Lodish, J. Little</i>
Development and use of computer models for the Southern Pacific Company.....	431	<i>A. Seelenfreund, E. P. Anderson, R. K. McAfee</i>
<b>PLAIN TALK: MACHINES THAT SPEAK YOUR LANGUAGE</b>		
A computational model of verbal understanding.....	441	<i>R. Simmons, J. Burger, R. Schwartz</i>
Procedural semantics for a question-answering machine.....	457	<i>W. A. Woods</i>
Natural language compiler for on-line data management.....	473	<i>C. Kellogg</i>
<b>PRICING COMPUTER SERVICES—WHAT? WHY? HOW?</b>		
Prices and the allocation of computer time.....	493	<i>N. Singer, H. Kantor, A. Moore</i>
The use of hard and soft money budgets and prices.....	499	<i>S. Smidt</i>
Priority pricing with application to time-shared computers.....	511	<i>M. Marchand</i>
Flexible pricing: An approach to the allocation of computer resources...	521	<i>N. Nielsen</i>
<b>DATA STRUCTURES FOR COMPUTER GRAPHICS</b>		
Data structures and techniques for remote computer graphics.....	533	<i>I. Cotton, F. S. Grotorez, Jr.</i>
Graphical systems communications: An associative memory approach.....	545	<i>E. H. Sibley, R. W. Taylor, D. G. Gordon</i>
Description of a set-theoretic data structure.....	557	<i>D. L. Childs</i>
<b>HYBRID SYSTEMS FOR PARTIAL DIFFERENTIAL EQUATIONS</b>		
Applications of functional optimization techniques for the serial hybrid computer solution of partial differential equations.....	565	<i>H. H. Hara, W. J. Karplus</i>
Hybrid assumed mode solution of non-linear partial differential equations.....	575	<i>J. C. Strauss, D. J. Newman</i>
Hybrid computer integration of partial differential equations by use of an assumed sum separation of variables.....	585	<i>J. R. Ashley, T. E. Bullock</i>



**Implementation characteristics are foremost:**

- **The implementation should be fast.**





Remember that  
there is no code  
faster than no code.

*Taligent's Guide to Designing Programs*



# More Programming Pearls

Confessions of a Coder

Jon Bentley





# More Programming Pearls

Confessions of a Coder

Jon Bentley

# The fastest I/O is no I/O.

Nils-Peter Nelson



Command-line tools  
can be 235x faster than  
your Hadoop cluster

Adam Drake

*<http://aadrake.com/command-line-tools-can-be-235x-faster-than-your-hadoop-cluster.html>*

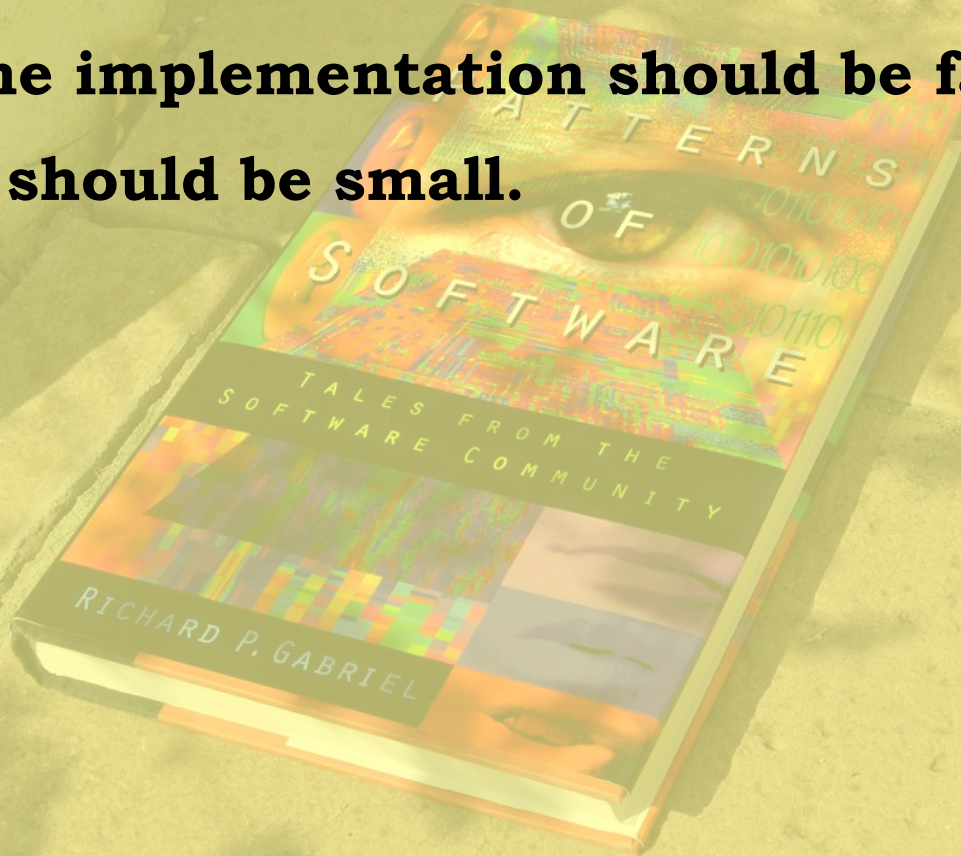


299792458



**Implementation characteristics are foremost:**

- **The implementation should be fast.**
- **It should be small.**





My point today is that, if we wish to count lines of code, we should not regard them as "lines produced" but as "lines spent".

Edsger Dijkstra



*"After 20 years, this is still the best exposition of the workings of a 'real' operating system."*  
Ken Thompson

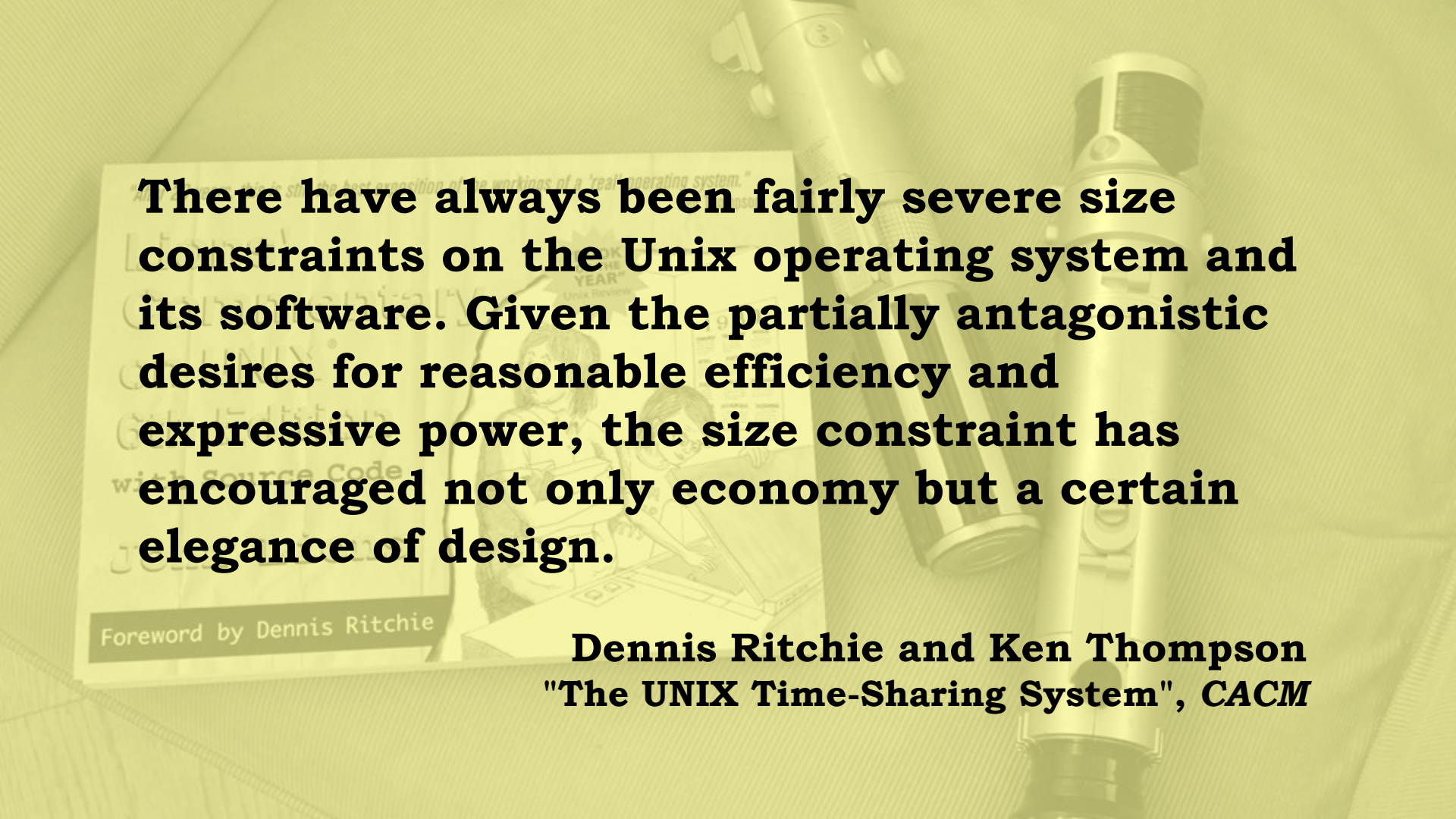
# Lions' Commentary on UNIX® 6th Edition with Source Code John Lions

Foreword by Dennis Ritchie

"BOOK  
OF THE  
YEAR"  
Unix Review





The background features a light green overlay with a faint image of a microscope on the right and a book cover on the left. The book cover has text including "The UNIX Time-Sharing System", "with Source Code", and "Foreword by Dennis Ritchie".

**There have always been fairly severe size constraints on the Unix operating system and its software. Given the partially antagonistic desires for reasonable efficiency and expressive power, the size constraint has encouraged not only economy but a certain elegance of design.**

Foreword by Dennis Ritchie

**Dennis Ritchie and Ken Thompson  
"The UNIX Time-Sharing System", *CACM***







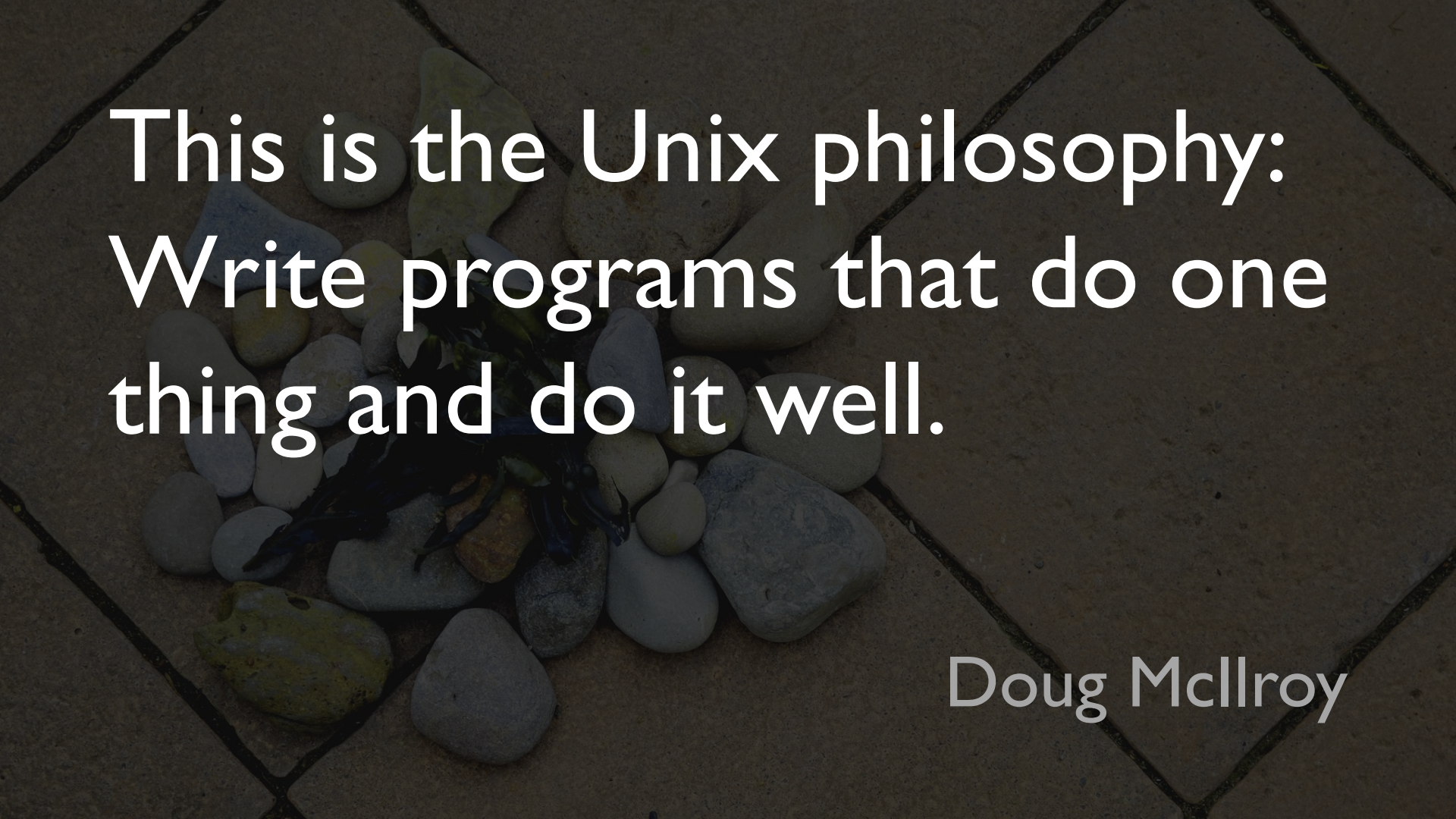
μονόλιθος

A dark, atmospheric photograph of the Stonehenge monument in England. The large, grey stone structures are silhouetted against a very dark, overcast sky. The foreground is a flat, dark field. The overall mood is mysterious and ancient.









This is the Unix philosophy:  
Write programs that do one  
thing and do it well.

Doug McIlroy





userservices



# SOFTWARE ENGINEERING

Define a subset of the system which is small enough to bring to an operational state [...] then build on that subsystem.

Report on a conference sponsored by the

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7th to 11th October 1968

E E David



# SOFTWARE ENGINEERING

This strategy requires that the system be designed in modules which can be realized, tested, and modified independently, apart from conventions for intermodule communication.

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7th to 11th October 1968

E E David



# Component Software

*Beyond  
Object-Oriented  
Programming*



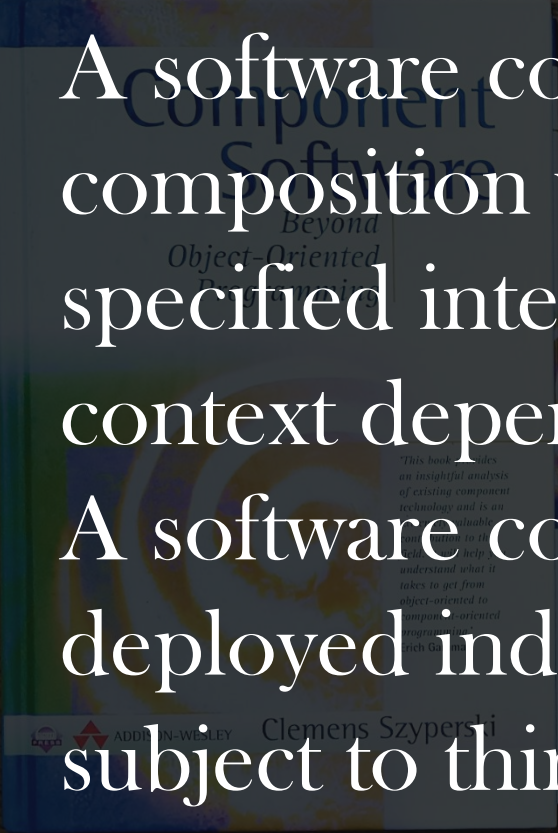
*"This book provides  
an insightful analysis  
of existing component  
technology and is an  
extremely valuable  
contribution to the  
field. It will help you  
understand what it  
takes to get from  
object-oriented to  
component-oriented  
programming."*  
Erich Gamma



ADDISON-WESLEY

Clemens Szyperski



The background of the slide features a dark, textured surface with a faint, semi-transparent image of a book cover. The book cover is for 'Component Software: Beyond Object-Oriented Programming' by Clemens Szyperski, published by Addison-Wesley. The cover has a blue and green color scheme with a circular graphic element. The text on the cover is partially legible, showing the title and author's name.

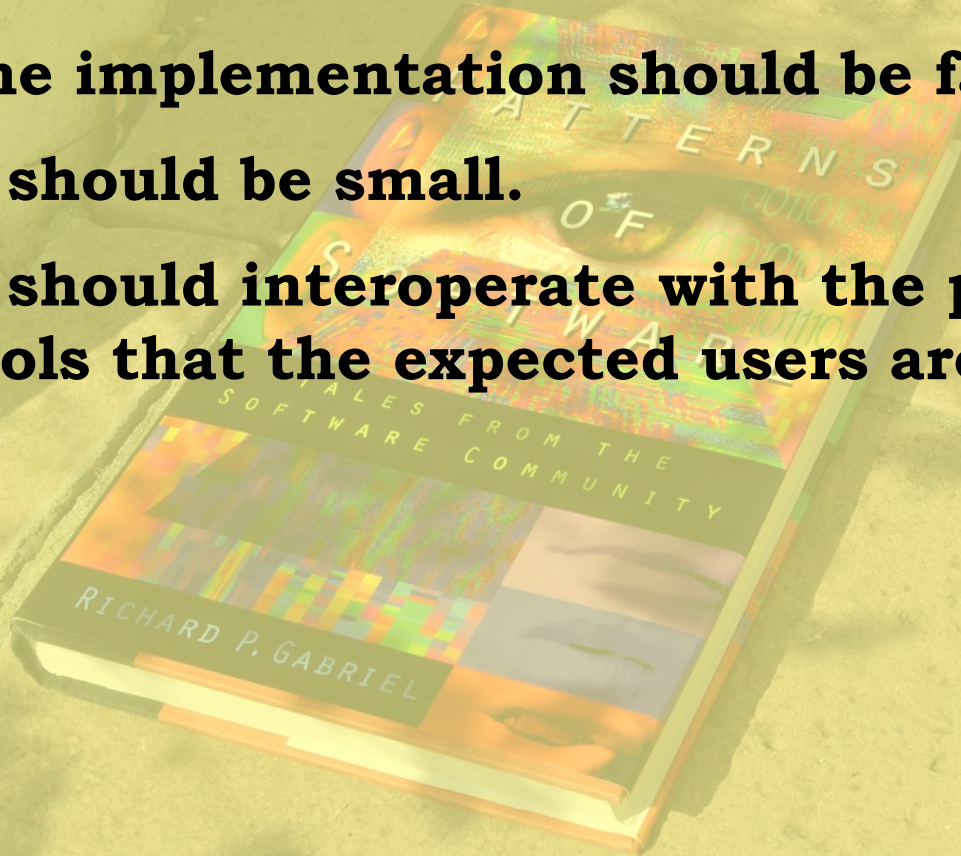
A software component is a unit of composition with contractually specified interfaces and explicit context dependencies only.

A software component can be deployed independently and is subject to third-party composition.

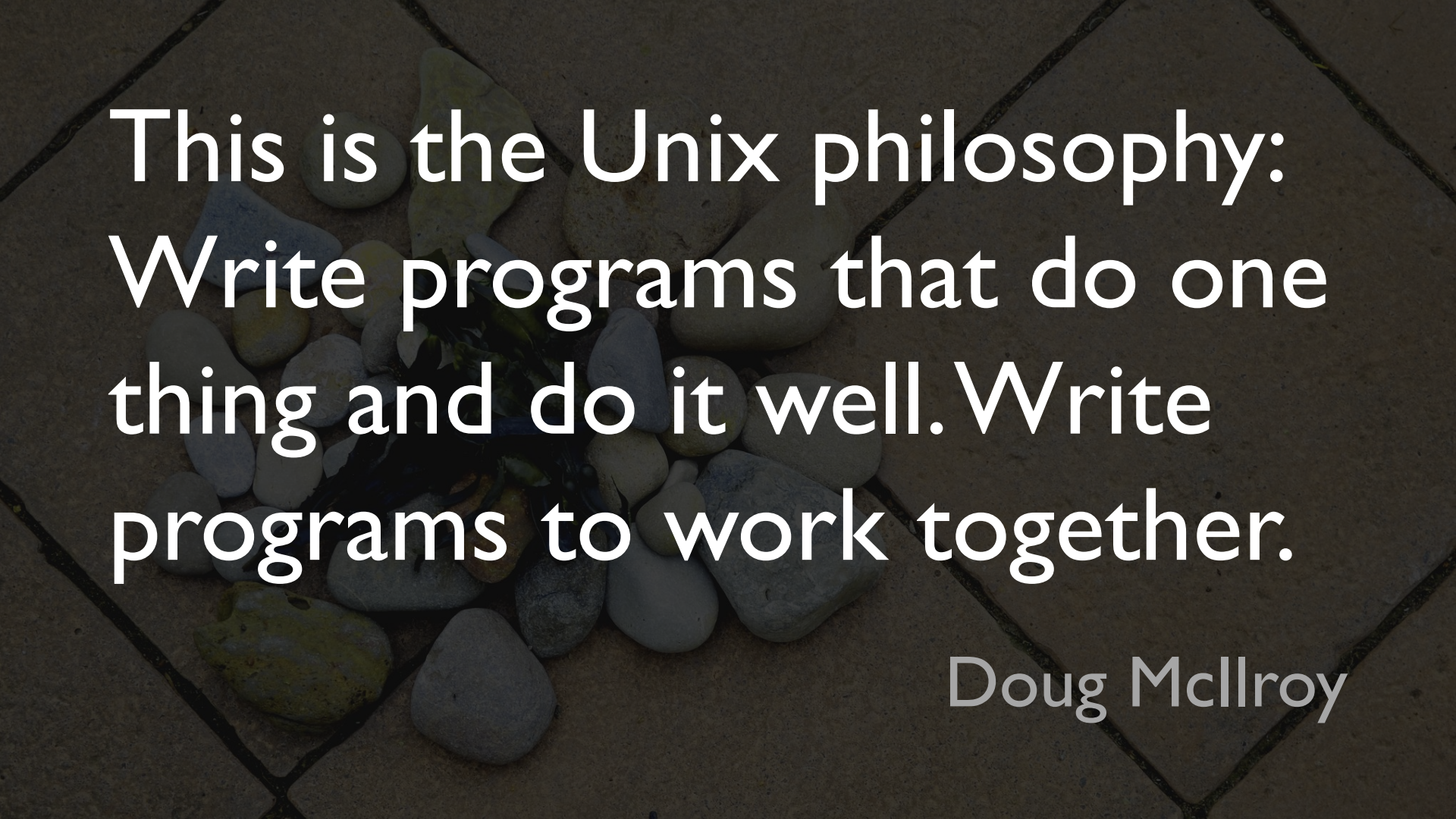


**Implementation characteristics are foremost:**

- **The implementation should be fast.**
- **It should be small.**
- **It should interoperate with the programs and tools that the expected users are already using.**



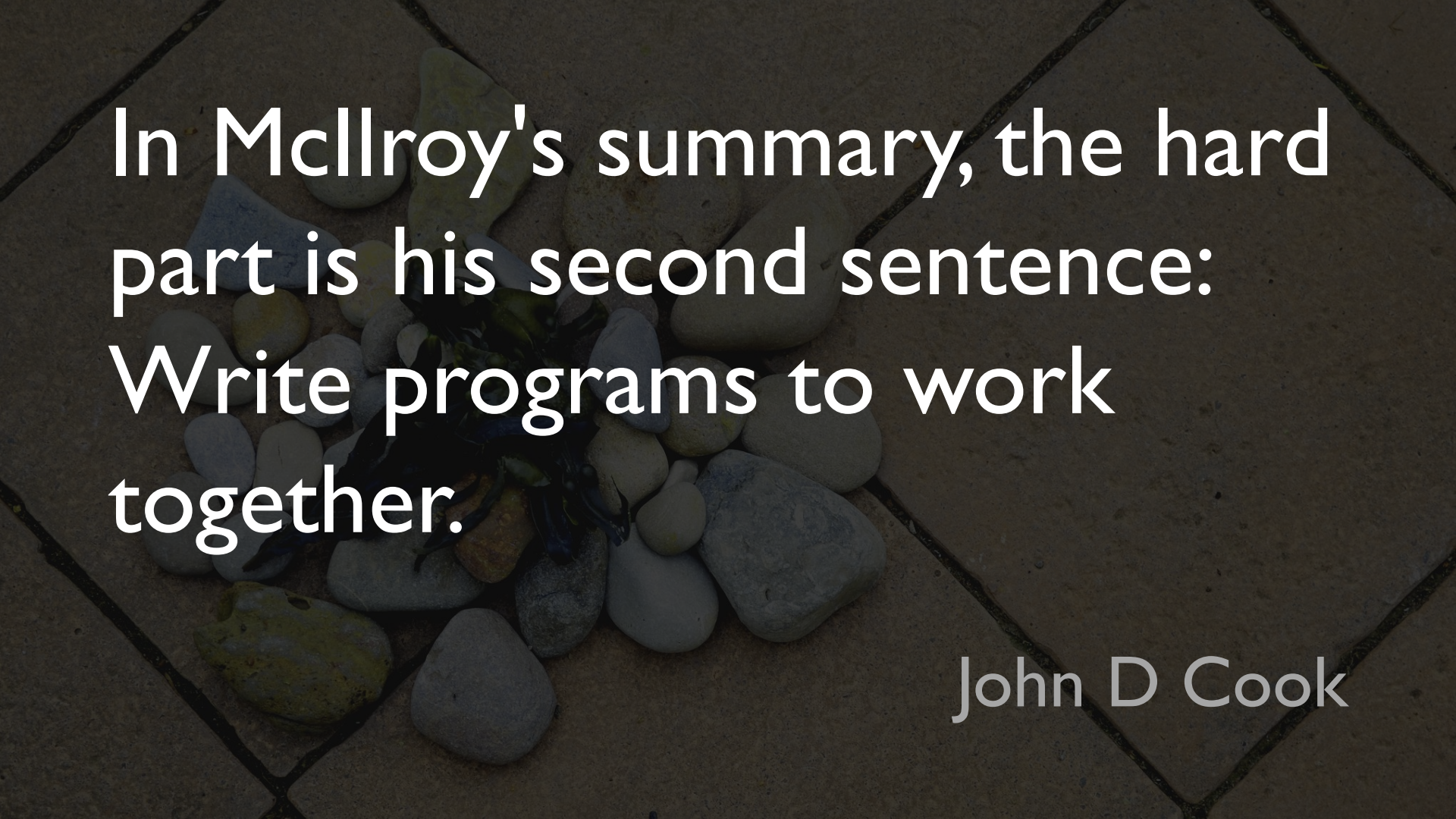




This is the Unix philosophy:  
Write programs that do one  
thing and do it well. Write  
programs to work together.

Doug McIlroy





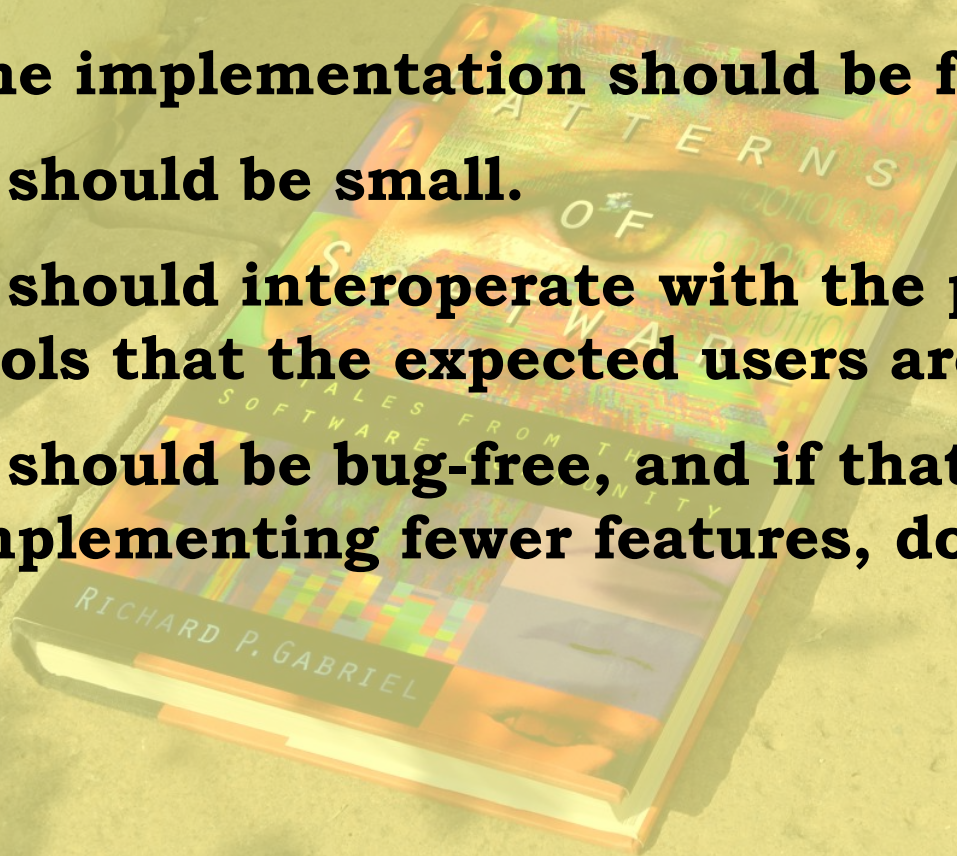
In McIlroy's summary, the hard  
part is his second sentence:  
Write programs to work  
together.

John D Cook



## **Implementation characteristics are foremost:**

- **The implementation should be fast.**
- **It should be small.**
- **It should interoperate with the programs and tools that the expected users are already using.**
- **It should be bug-free, and if that requires implementing fewer features, do it.**





# SOFTWARE ENGINEERING

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

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7th to 11th October 1968

Alan Perlis



We instituted a rigorous regression test for all of the features of AWK. Any of the three of us who put in a new feature into the language [...], first had to write a test for the new feature.

Alfred Aho

[http://www.computerworld.com.au/article/216844/a-z\\_programming\\_languages\\_awk/](http://www.computerworld.com.au/article/216844/a-z_programming_languages_awk/)





**jasongorman**

@jasongorman

Fun Fact: "Given... when... then..." is what we call a Hoare Triple [en.wikipedia.org/wiki/Hoare\\_log...](http://en.wikipedia.org/wiki/Hoare_logic)

7:42 PM - 3 Mar 2015



17



16



$P \{Q\} R$



## **Implementation characteristics are foremost:**

- **The implementation should be fast.**
- **It should be small.**
- **It should interoperate with the programs and tools that the expected users are already using.**
- **It should be bug-free, and if that requires implementing fewer features, do it.**
- **It should use parsimonious abstractions as long as they don't get in the way.**



Shipping first time code is like going into debt.  
A little debt speeds development so long as it  
is paid back promptly with a rewrite.

The danger occurs when the debt is not  
repaid. Every minute spent on not-quite-right  
code counts as interest on that debt.

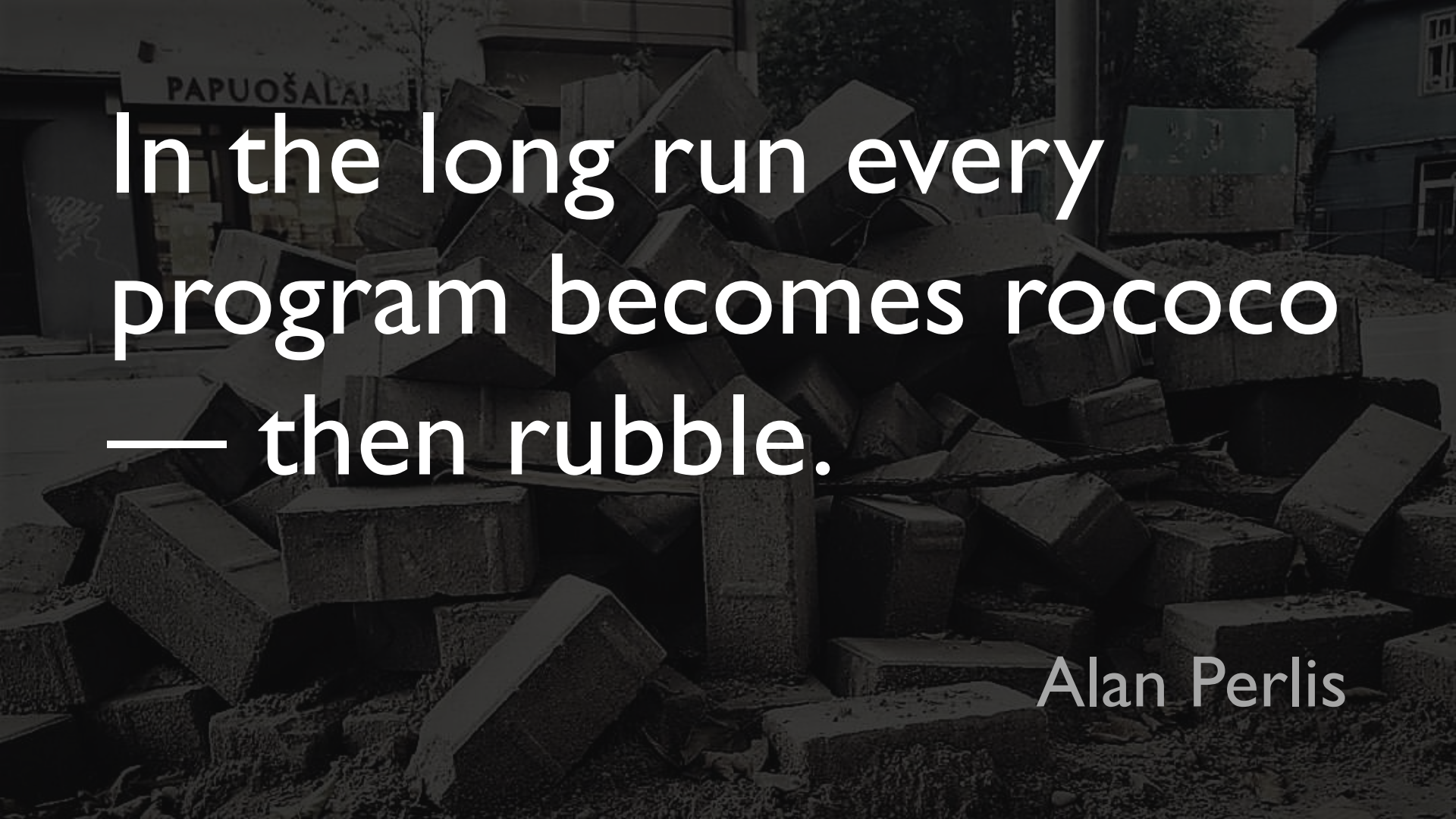
Ward Cunningham

<http://c2.com/doc/oopsla92.html>







A dark, grainy photograph of a large pile of rubble, including concrete blocks and debris, with a building in the background. The text is overlaid in white.

In the long run every  
program becomes rococo  
— then rubble.

Alan Perlis



Walking on water and  
developing software  
from a specification  
are easy if both are  
frozen.

Edward V Berard

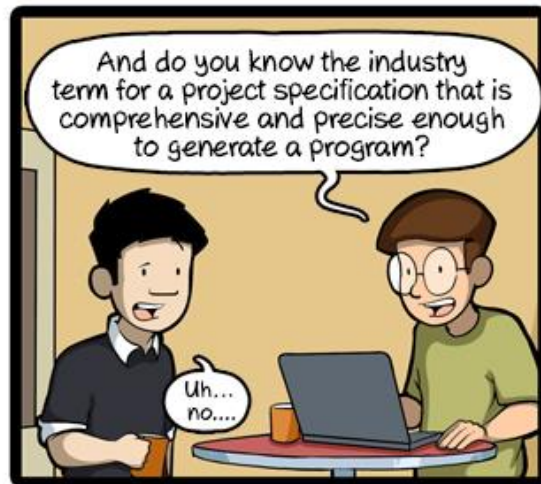




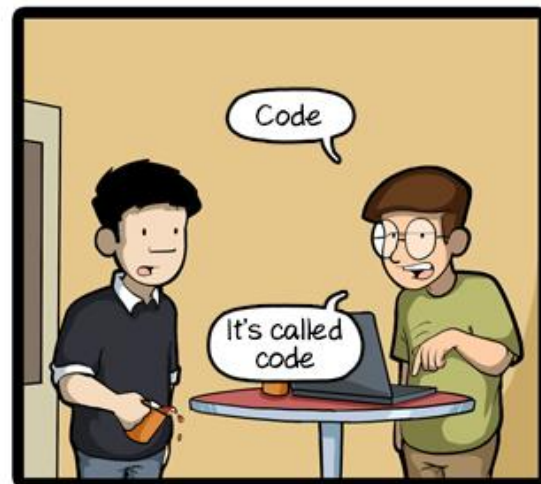
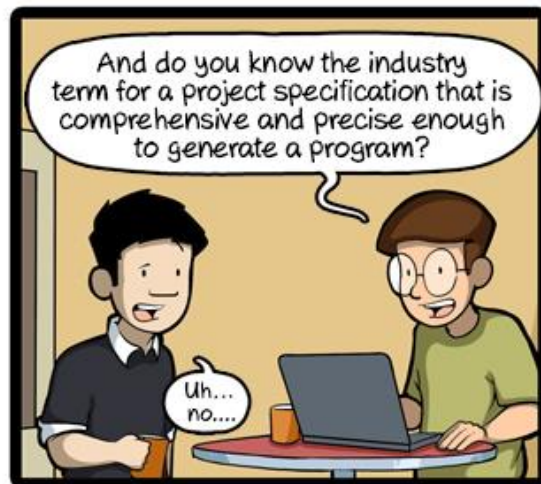














Programming is  
a design activity.

Jack W Reeves

*What Is Software Design?*



# SOFTWARE ENGINEERING

The replication of multiple copies of a software system is the phase of software manufacture which corresponds to the production phase in other areas of engineering.

Report on a conference sponsored by the

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7-11 October 1972  
**Peter Naur and Brian Randell**



# SOFTWARE ENGINEERING

It is accomplished by simple copying operations, and constitutes only a minute fraction of the cost of software manufacture.

Report on a conference sponsored by the

NATO SCIENCE COMMITTEE

Garmisch, Germany, 7-11 October 1968  
**Peter Naur and Brian Randell**

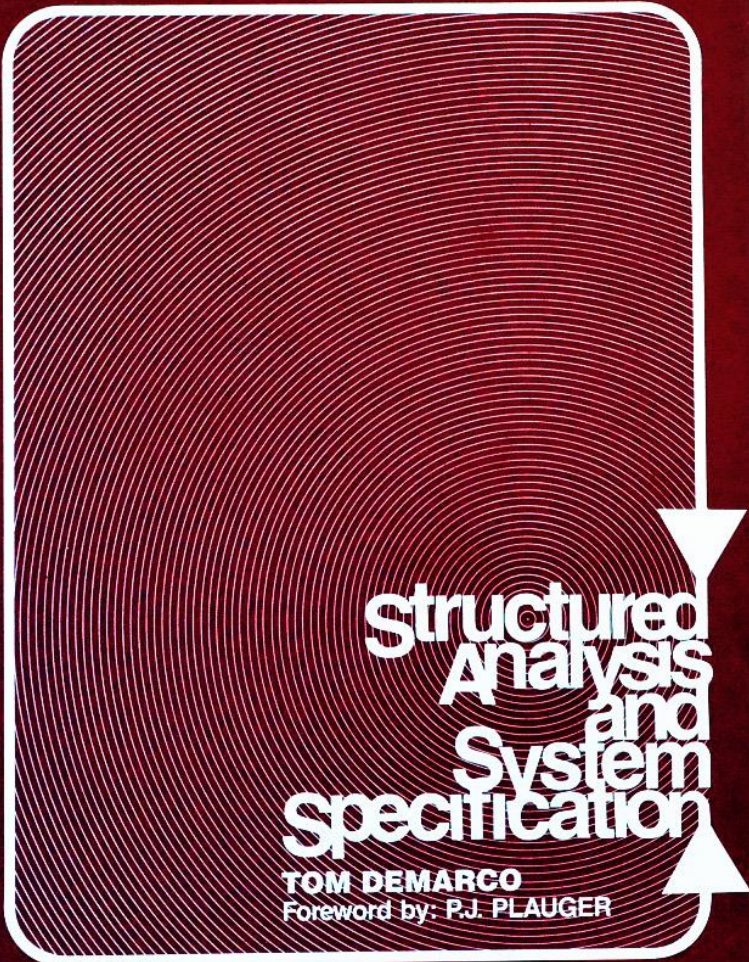


We propose [...] that one begins with a list of difficult design decisions or design decisions which are likely to change. Each module is then designed to hide such a decision from the others.

David L Parnas

"On the Criteria to Be Used in Decomposing Systems into Modules"





# Structured Analysis and System Specification

**TOM DEMARCO**  
Foreword by: P.J. PLAUGER

YOURDON PRESS COMPUTING SERIES



Cohesion is a measure of the strength of association of the elements inside a module.

A highly cohesive module is a collection of statements and data items that should be treated as a whole because they are so closely related.

Any attempt to divide them up would only result in increased coupling and decreased readability.

**TOM DEMARCO**  
Foreword by: P.J. PLAUGER

YOURDON PRESS COMPUTING SERIES



OOP to me means only  
messaging, local retention  
and protection and hiding of  
state-process, and extreme  
late-binding of all things.

*Alan Kay*





知るべき  
97 Things Every Prog

Kevlin Henney 編  
李军译 吕骏审校  
電子工業出版社  
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY  
http://www.pri.com.cn

O'REILLY®  
オライリー・ジャパン



Collective Wisdom  
from the Experts

# 97 Things Every Programmer Should Know

O'REILLY®

Edited by Kevlin Henney

97件事

97件事

JOB  
MMICT



**Instead of using threads and shared memory as our programming model, we can use processes and message passing. Process here just means a protected independent state with executing code, not necessarily an operating system process.**

***Russel Winder***

**"Message Passing Leads to Better Scalability in Parallel Systems"**



**Languages such as Erlang (and occam before it) have shown that processes are a very successful mechanism for programming concurrent and parallel systems. Such systems do not have all the synchronization stresses that shared-memory, multithreaded systems have.**

***Russel Winder***

**"Message Passing Leads to Better Scalability in Parallel Systems"**



C.A.R. Hoare  
**Communicating  
Sequential  
Processes**

PRENTICE-HALL  
INTERNATIONAL  
SERIES IN  
COMPUTING  
AND TECHNOLOGY

C.A.R. HOARE SERIES EDITOR





---

**ABCL**

*An Object-Oriented Concurrent  
System*

---

*edited by Akinori Yonezawa*

*The MIT Press*



# **LISP 1.5 Programmer's Manual**

**The Computation Center  
and Research Laboratory of Electronics  
Massachusetts Institute of Technology**



Lambda-calculus  
was the first  
object-oriented  
language (1952)



**Excel is the world's  
most popular  
functional language.  
Simon Peyton-Jones**



Separation of tasks is a good thing, on the other hand we have to tie the loose ends together again.

Edsger Dijkstra



We can build a complete programming model out of two separate pieces—the computation model and the coordination model.

David Gelernter + Nicholas Carriero

"Coordination Languages and their Significance"



## Summary--what's most important.

To put my strongest concerns in a nutshell:

1. We should have some ways of coupling programs like garden hose--screw in another segment when it becomes when it becomes necessary to massage data in another way. This is the way of IO also.
2. Our loader should be able to do link-loading and controlled establishment.
3. Our library filing scheme should allow for rather general indexing, responsibility, generations, data path switching.
4. It should be possible to get private system components (all routines are system components) for fiddling around with.

K. D. McIlroy  
Oct. 11, 1964





**Kevlin Henney**

@KevlinHenney

Software development can only be considered immature because of how we use our experience, not because we lack experience.

10:50 AM - Nov 1, 2009

<https://twitter.com/KevlinHenney/status/5334796004>