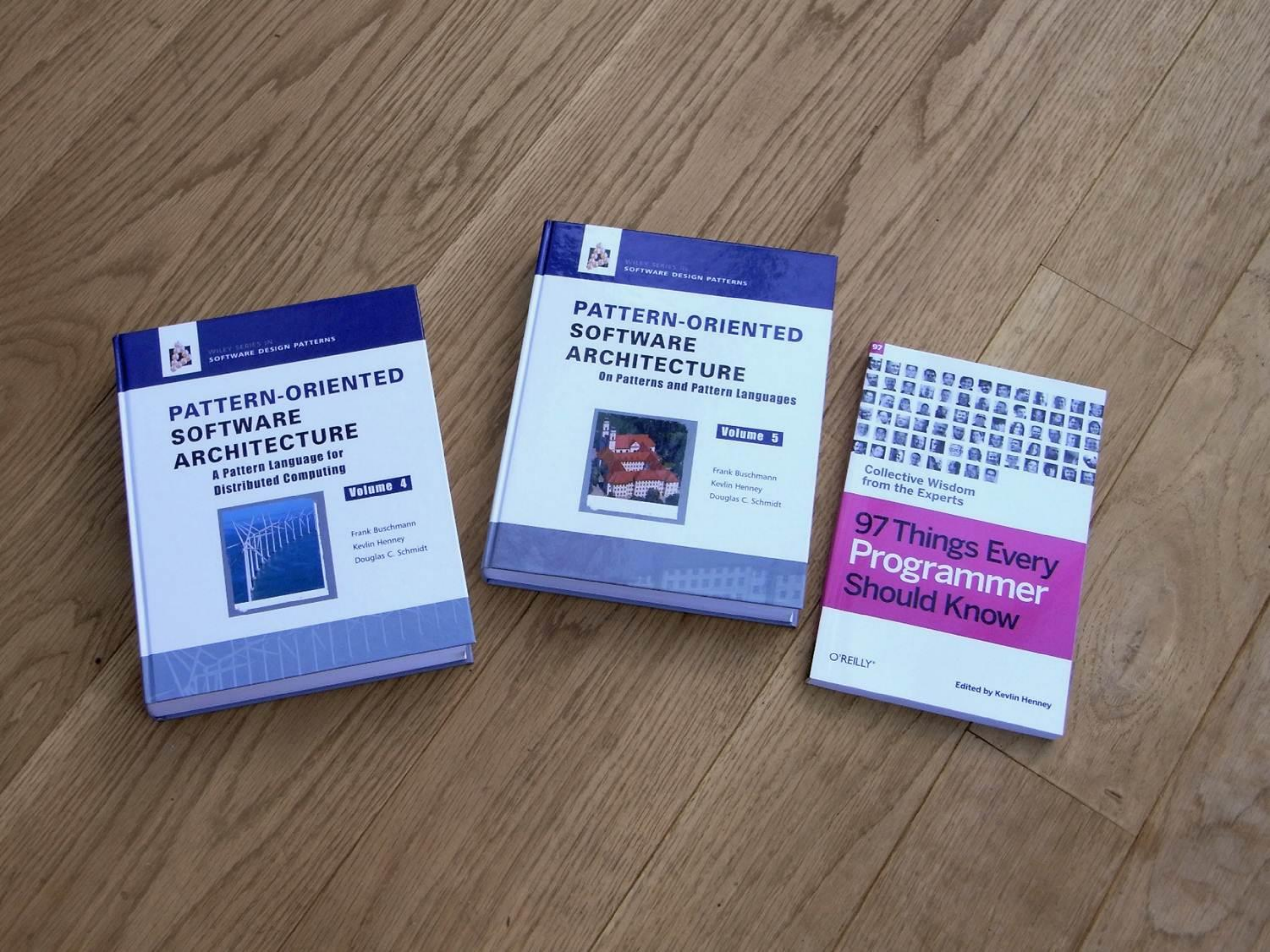
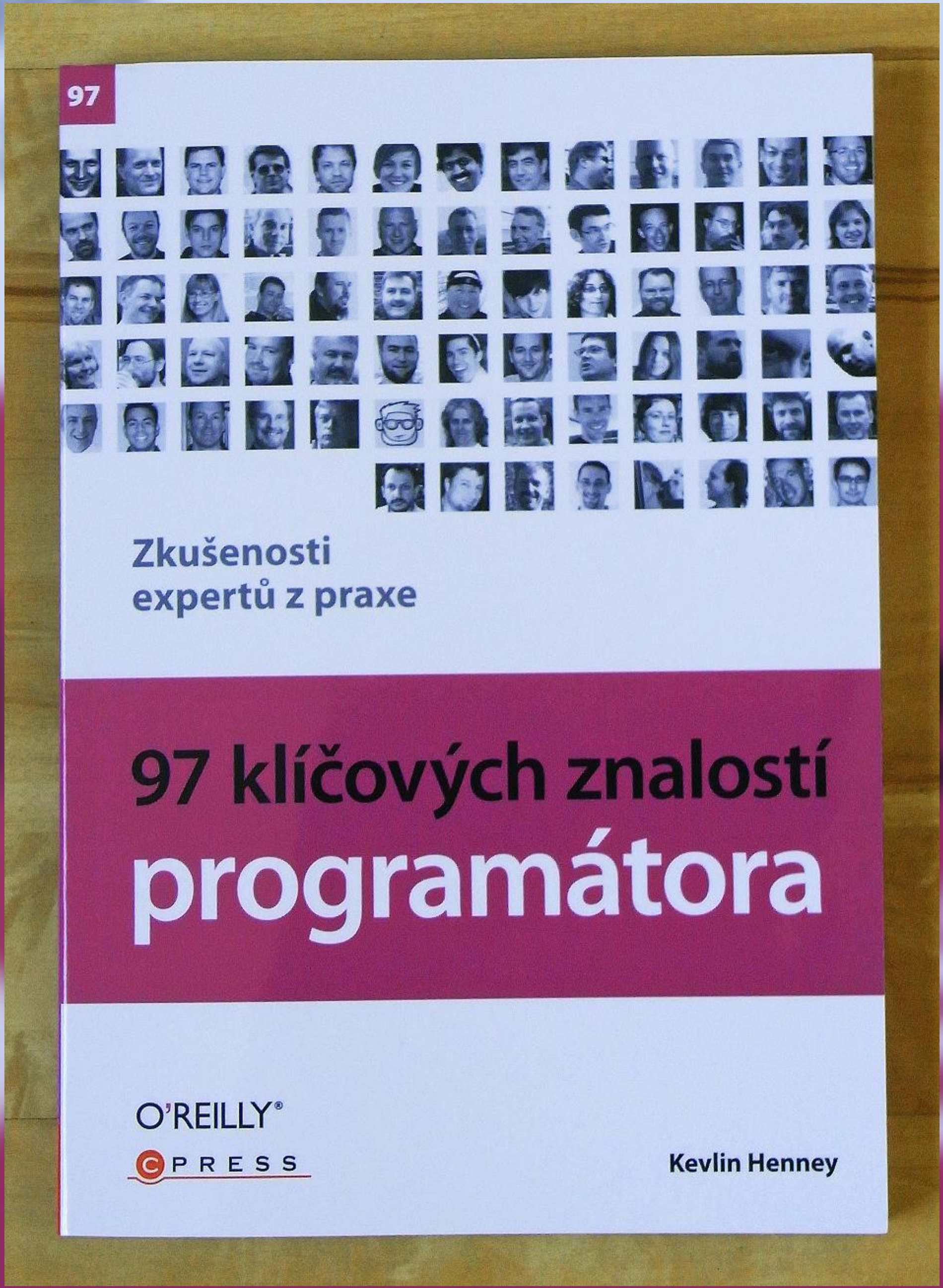
Worse Is Better, for Better or for Worse

@KevlinHenney


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
97
Collective Wisdom from the Experts
97 Things Every Programmer Should Know
O'REILLY
Edited by Kevlin Henney

TURING 图灵程序设计丛书
WILEY
Pattern-Oriented Software Architecture Volume 4
A Pattern Language for Distributed Computing
面向模式的软件架构
分布式计算的模式语言
卷4
[德] Frank Buschmann
[英] Kevlin Henney 著
[美] Douglas C.Schmidt
肖鹏 陈立 译
人民邮电出版社
POSTS & TELECOM PRESS
TURING 图灵程序设计丛书
WILEY
Pattern-Oriented Software Architecture Volume 5
On Patterns and Pattern Languages
面向模式的软件架构
模式与模式语言
卷5
[德] Frank Buschmann
[英] Kevlin Henney 著
[美] Douglas C. Schmidt
肖鹏 等译
人民邮电出版社
POSTS & TELECOM PRESS
97
Collective Wisdom
from the Experts
程序员
应该知道的
97件事
O'REILLY
Kevlin Henney 编
李军 译 吕骏 审校
電子工業出版社



97

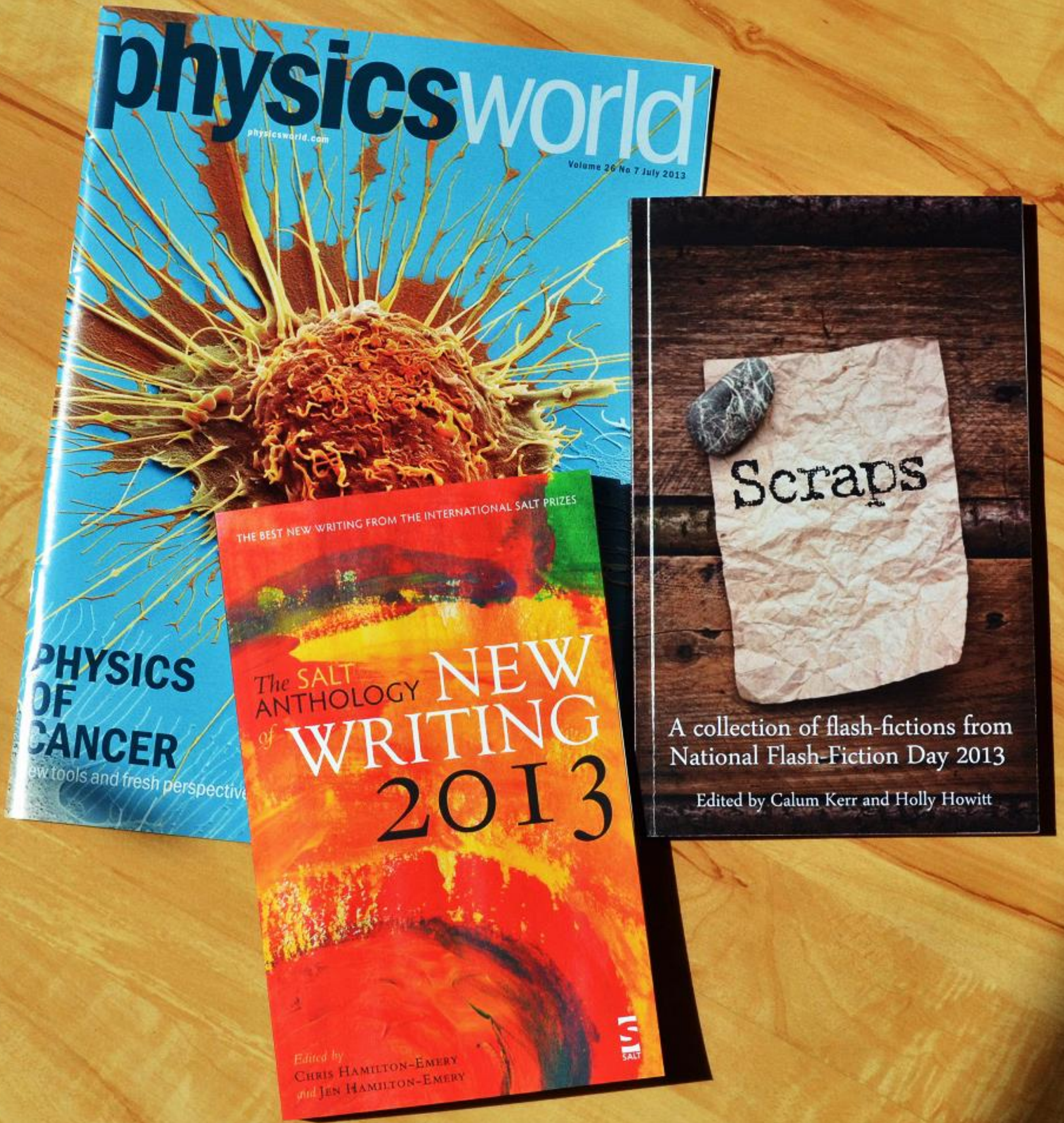
Zkušenosti expertů z praxe

97 klíčových znalostí programátora

O'REILLY®

CPRESS

Kevlin Henney


physicsworld
physicsworld.com
Volume 26 No 7 July 2013
PHYSICS OF CANCER
New tools and fresh perspective
THE BEST NEW WRITING FROM THE INTERNATIONAL SALT PRIZES
The SALT ANTHOLOGY of NEW WRITING 2013
Edited by Chris Hamilton-Emery and Jen Hamilton-Emery
SALT
Scraps
A collection of flash-fictions from National Flash-Fiction Day 2013
Edited by Calum Kerr and Holly Howitt

There are a thousand thoughts lying within a man that he does not know till he takes up the pen to write.

William Thackeray

I am irritated by my own writing. I am like a violinist whose ear is true, but whose fingers refuse to reproduce precisely the sound he hears within.

Gustave Flaubert

When I write, I feel like an armless, legless man with a crayon in his mouth.

Kurt Vonnegut

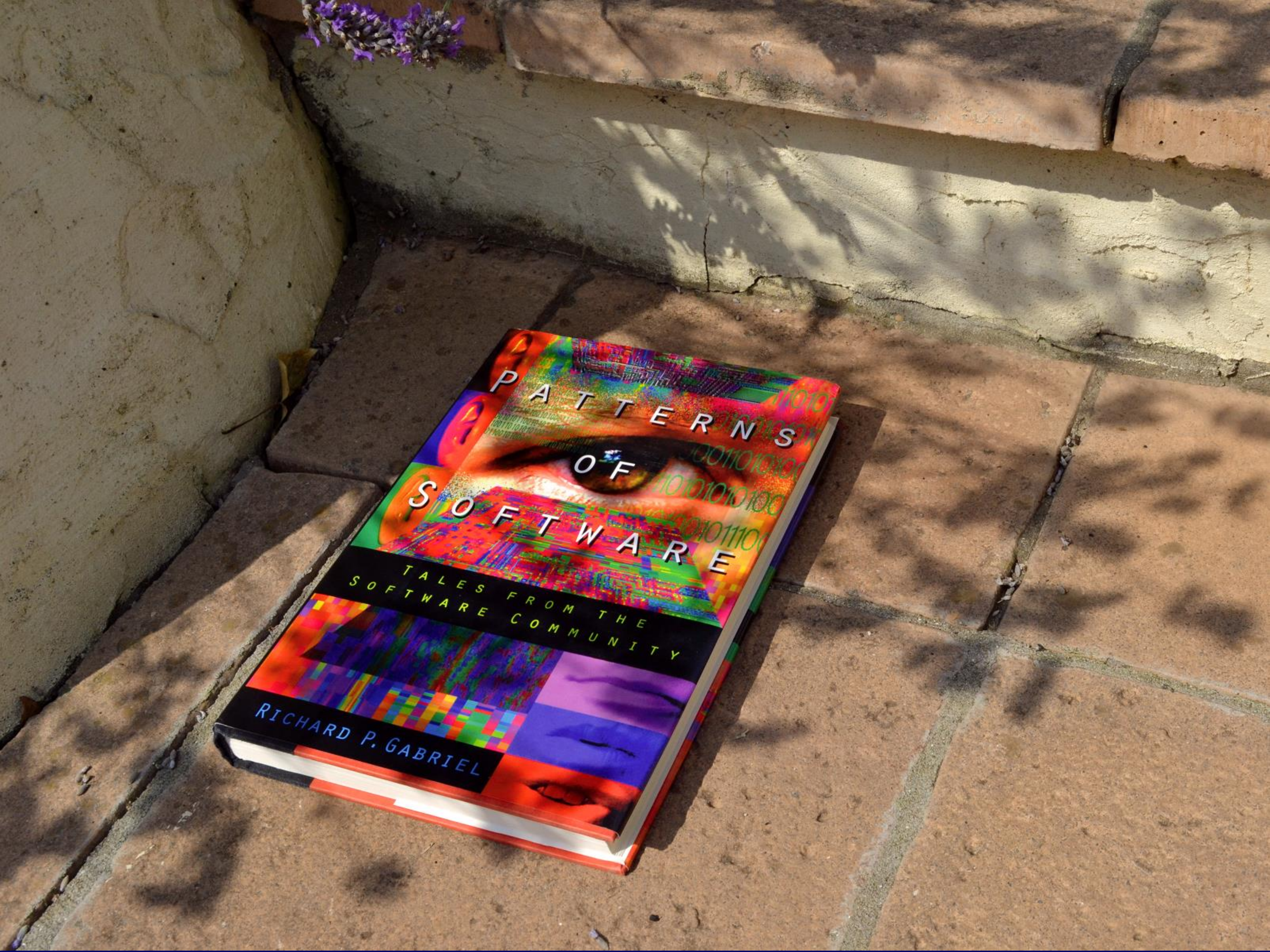
Stop Overpromising and Overshooting

We want to do everything all at once. Grand plans! Sweeping gestures! Epic 23-book fantasy cycles! Don’t overreach. Concentrate on what you can complete. Temper risk with reality.

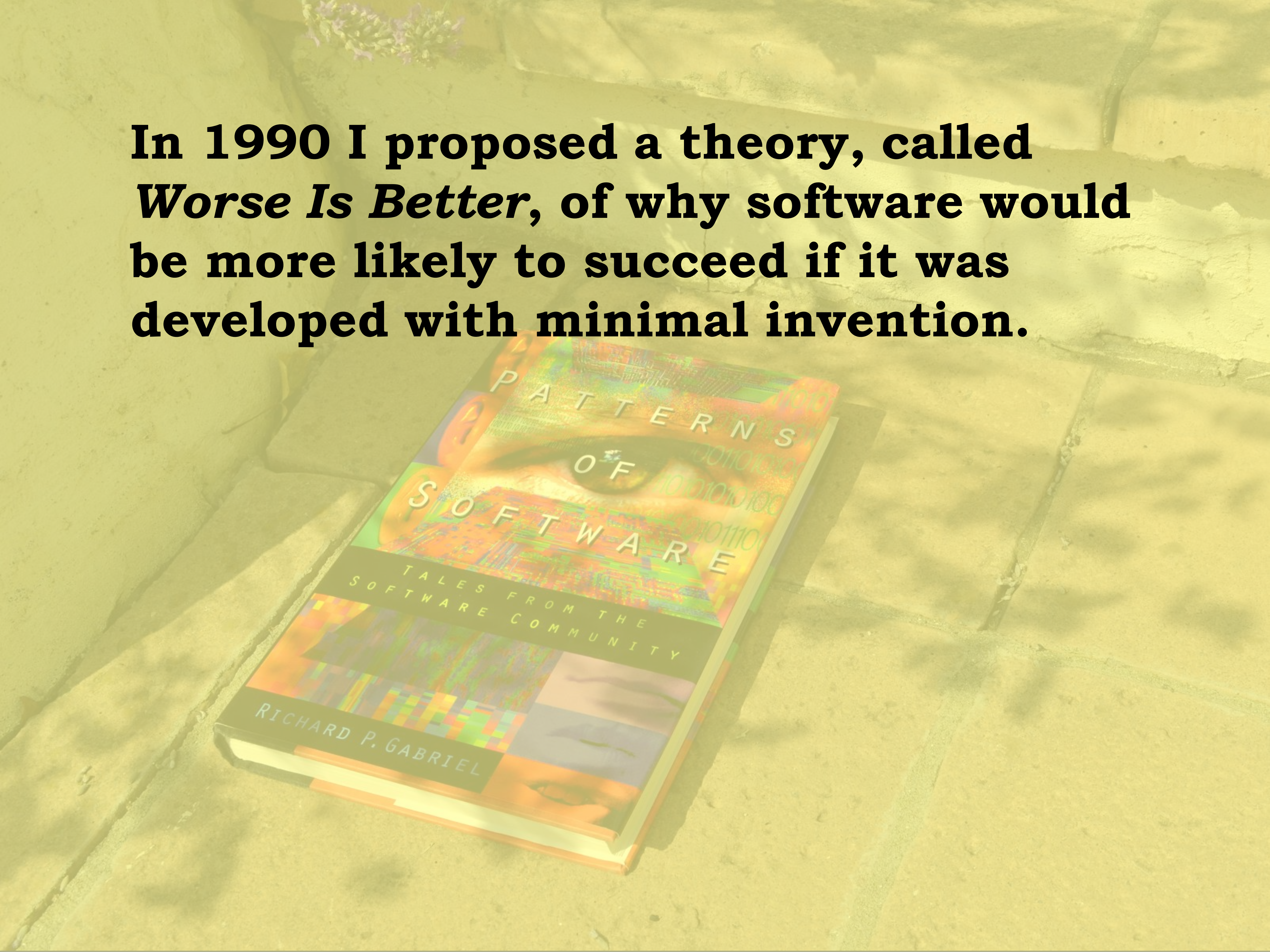
Chuck Wendig
"25 Things Writers Should Stop Doing"
http://terribleminds.com/ramble/2012/01/03/25-things-writers-should-stop-doing/

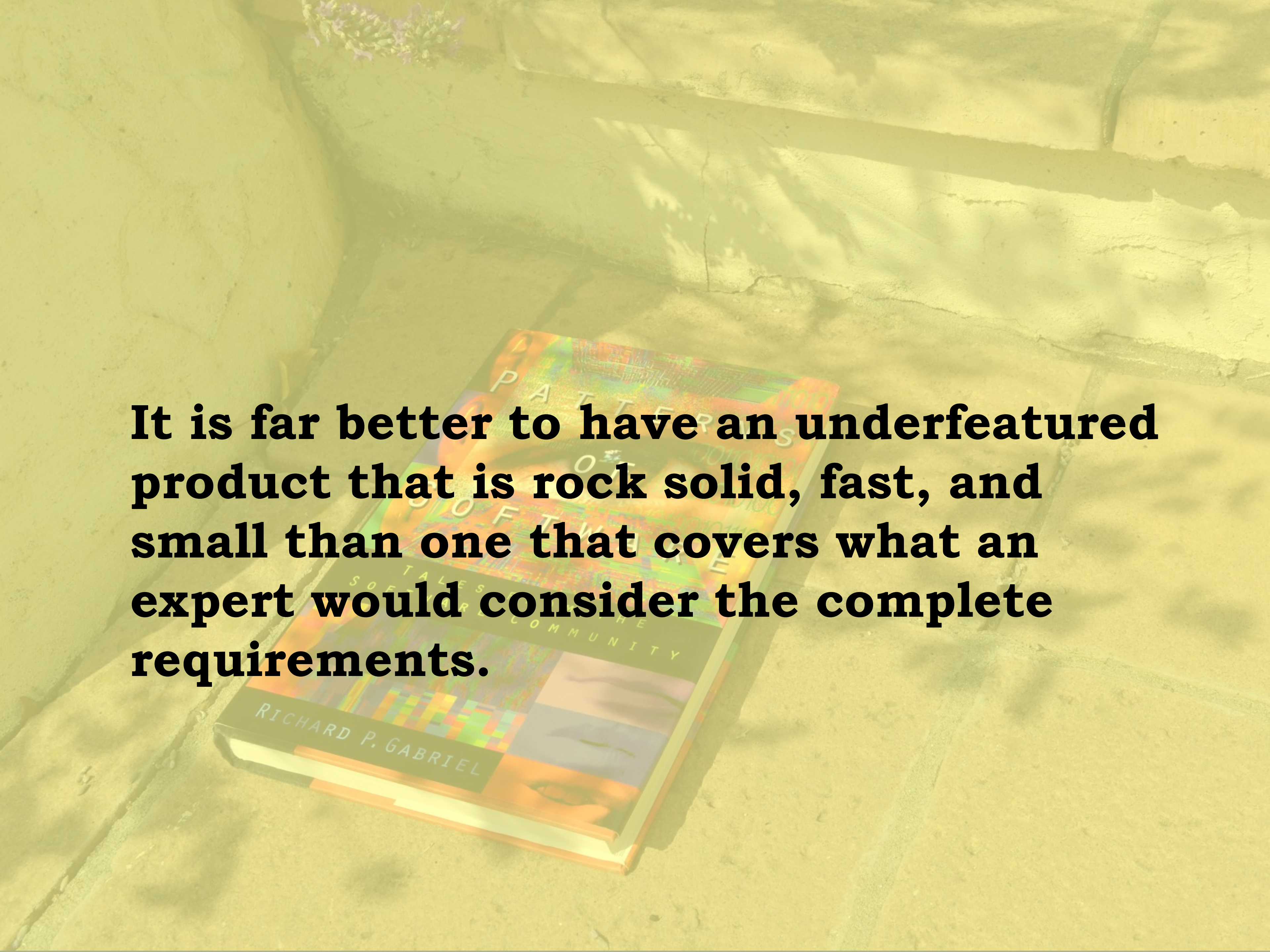
You have to finish things — that's what you learn from, you learn by finishing things.

Neil Gaiman

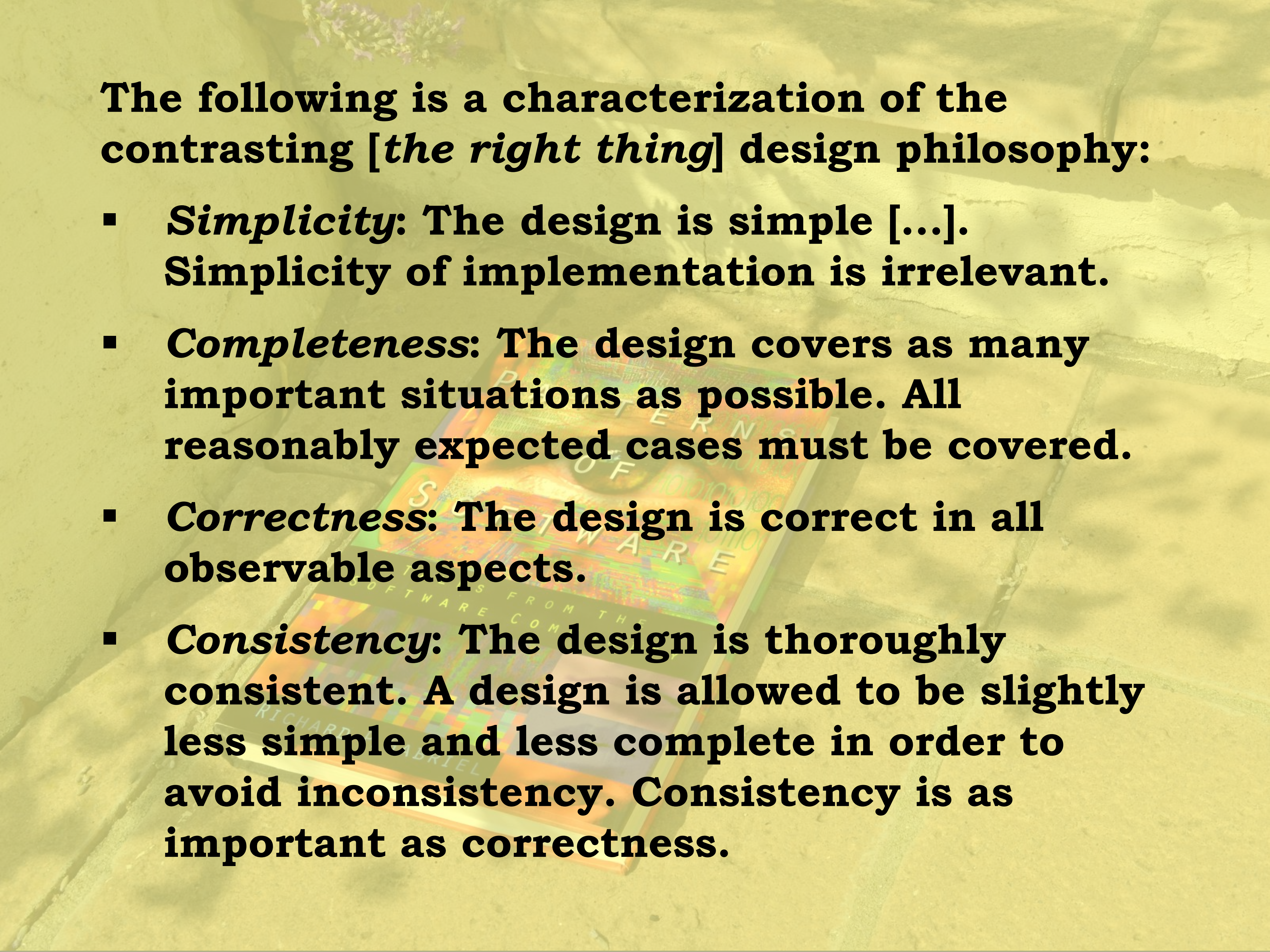

PATTERNS OF SOFTWARE
TALES FROM THE SOFTWARE COMMUNITY
RICHARD P. GABRIEL

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.



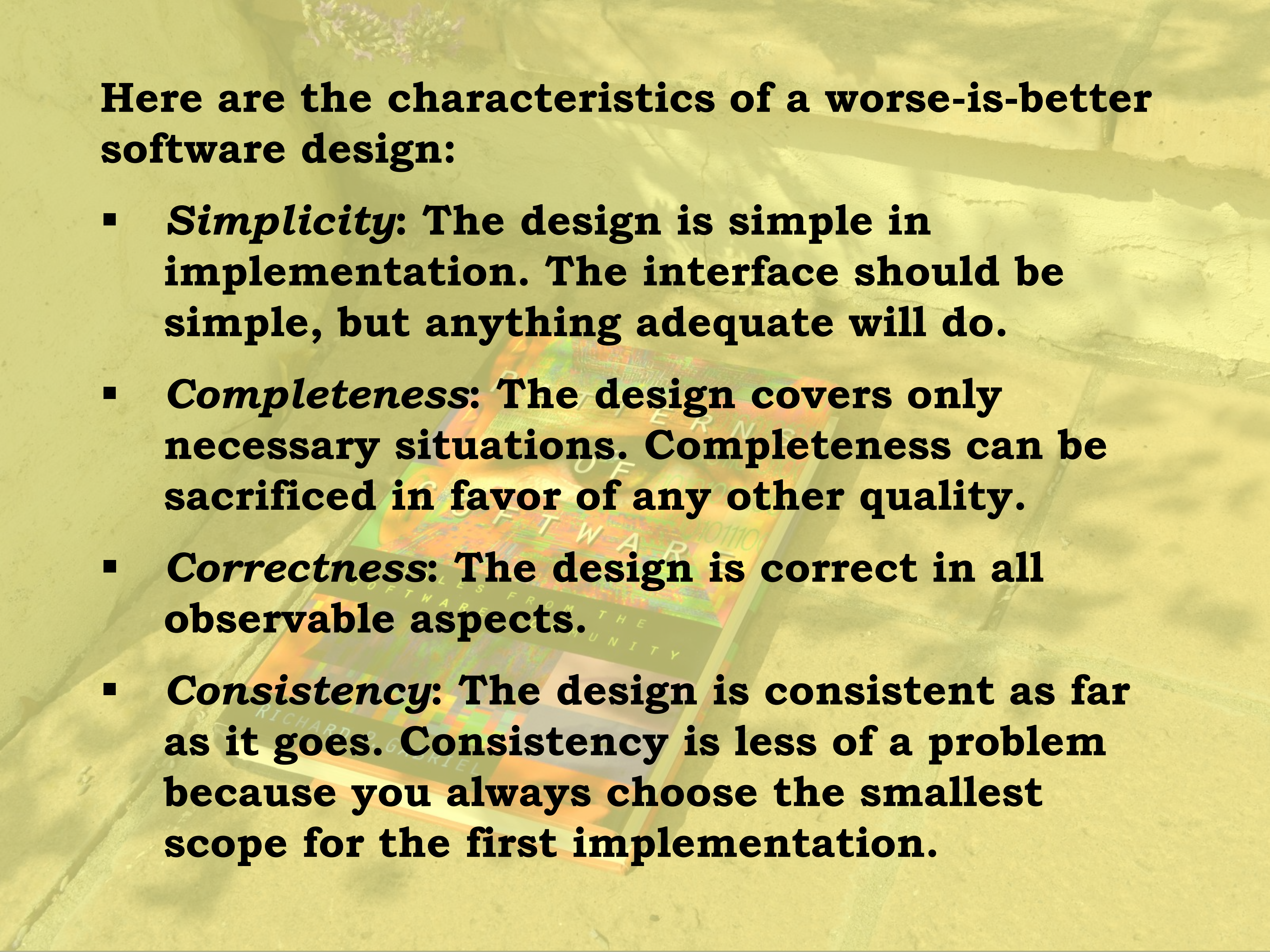


It is far better to have an underfeatured product that is rock solid, fast, and small than one that covers what an expert would consider the complete requirements.



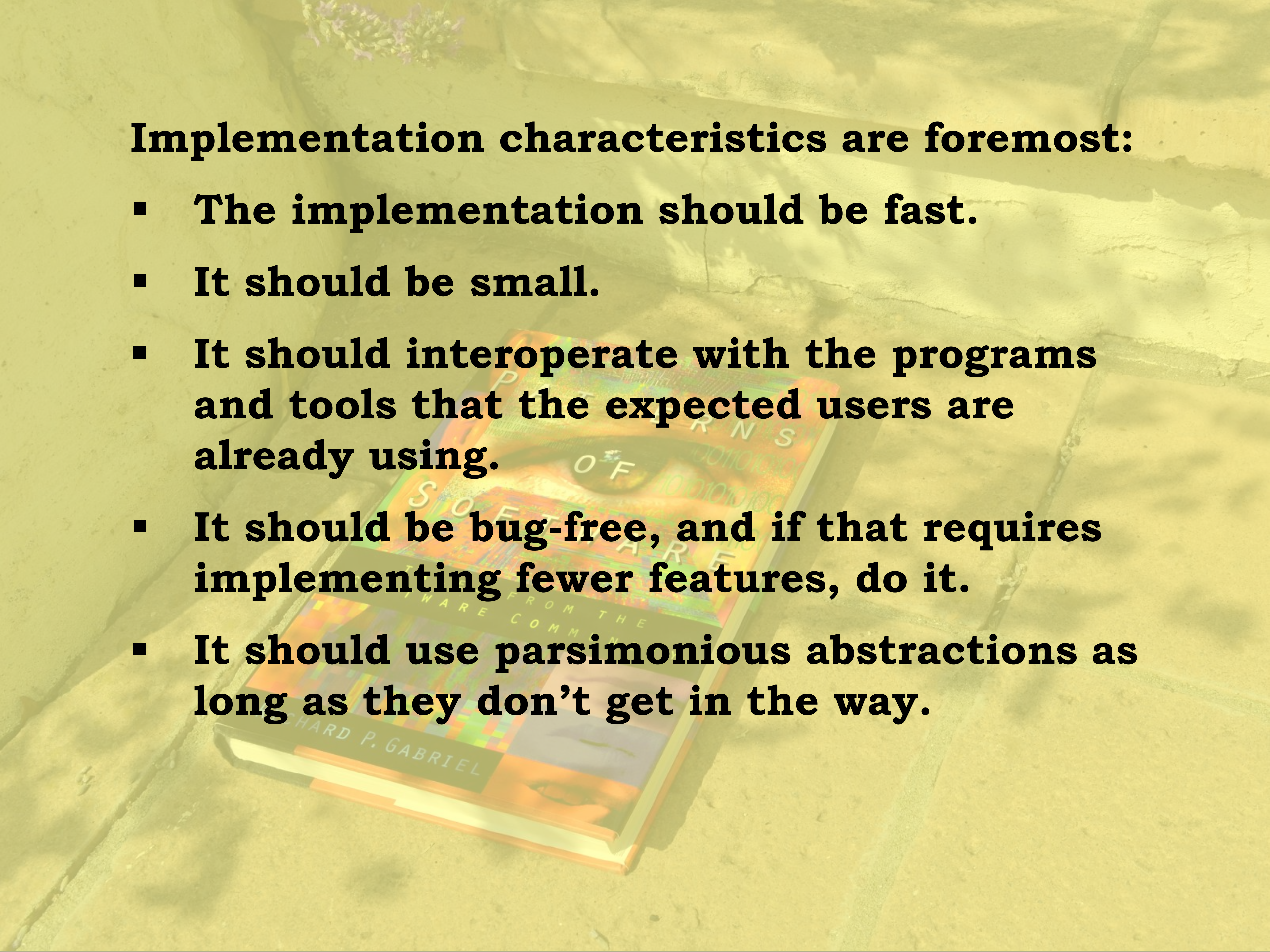
The following is a characterization of the contrasting [*the right thing*] design philosophy:

- *Simplicity*: The design is simple [...]. Simplicity of implementation is irrelevant.
- *Completeness*: The design covers as many important situations as possible. All reasonably expected cases must be covered.
- *Correctness*: The design is correct in all observable aspects.
- *Consistency*: The design is thoroughly consistent. A design is allowed to be slightly less simple and less complete in order to avoid inconsistency. Consistency is as important as correctness.



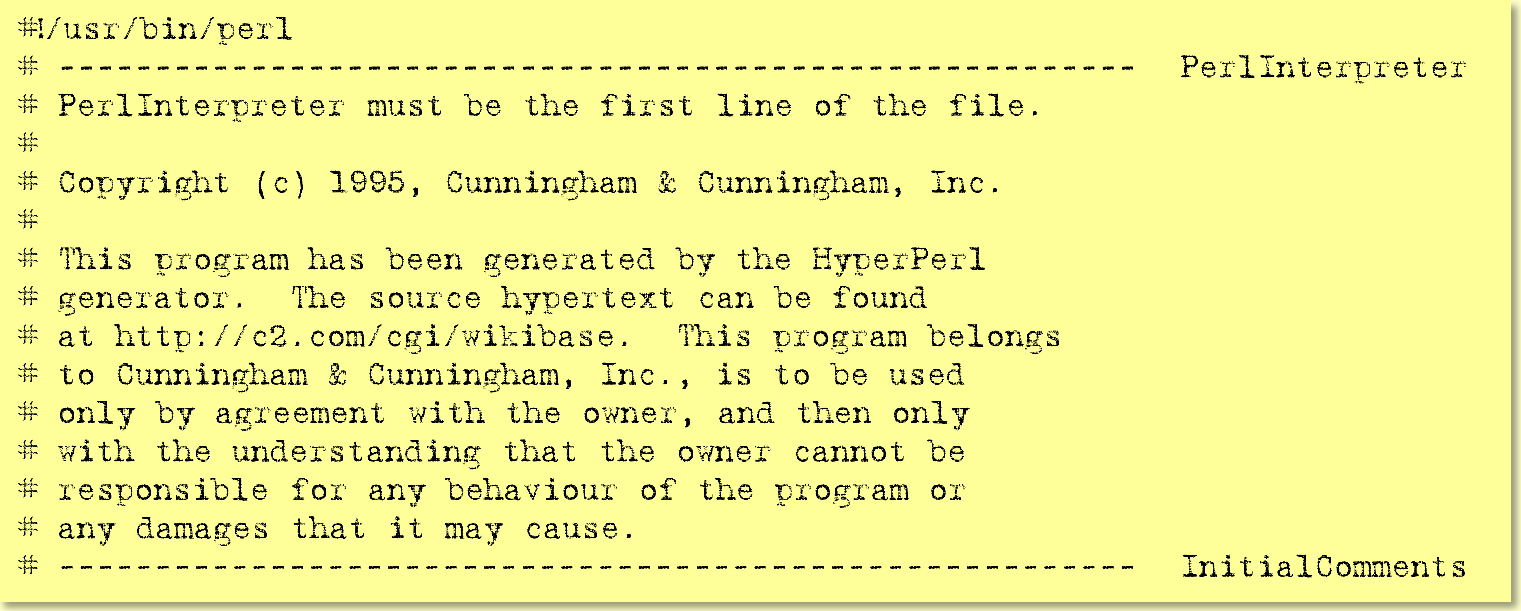
Here are the characteristics of a worse-is-better software design:

- ***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.**

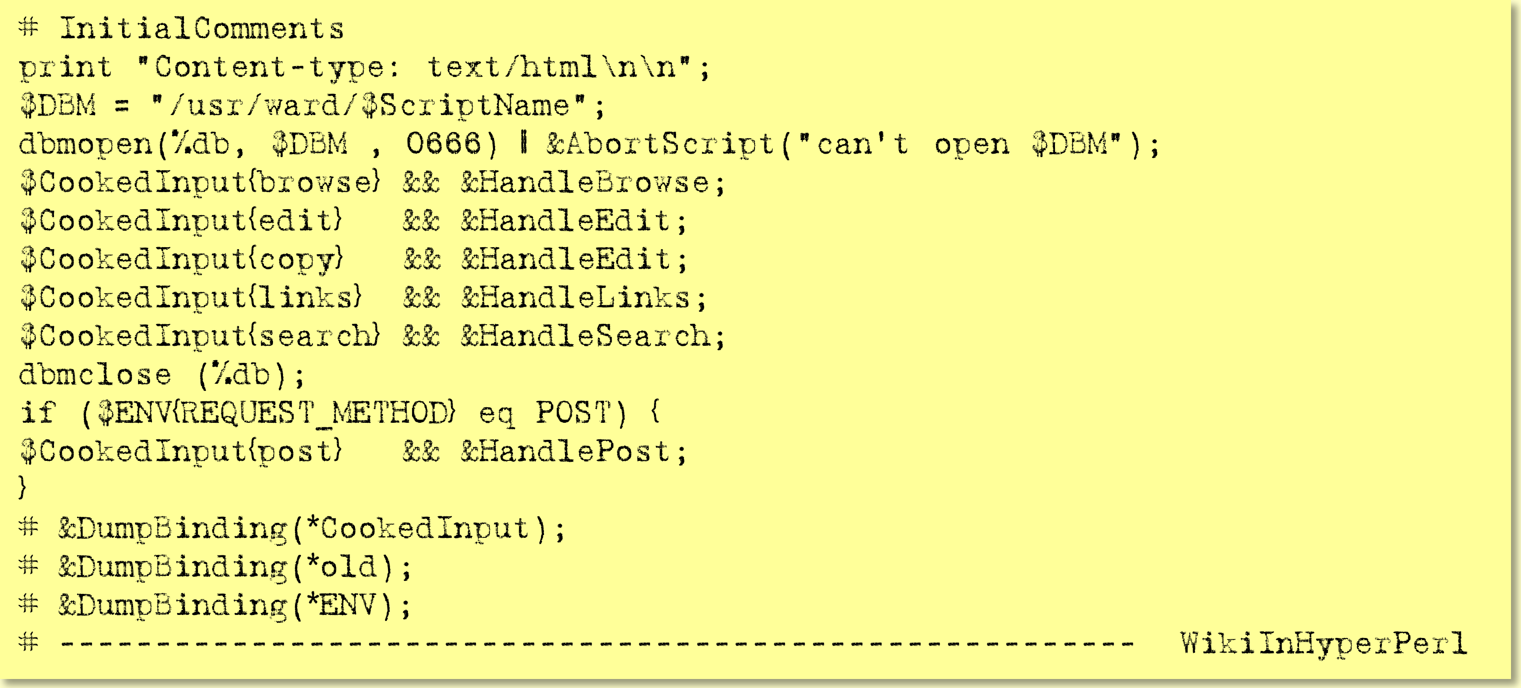


Implementation characteristics are foremost:

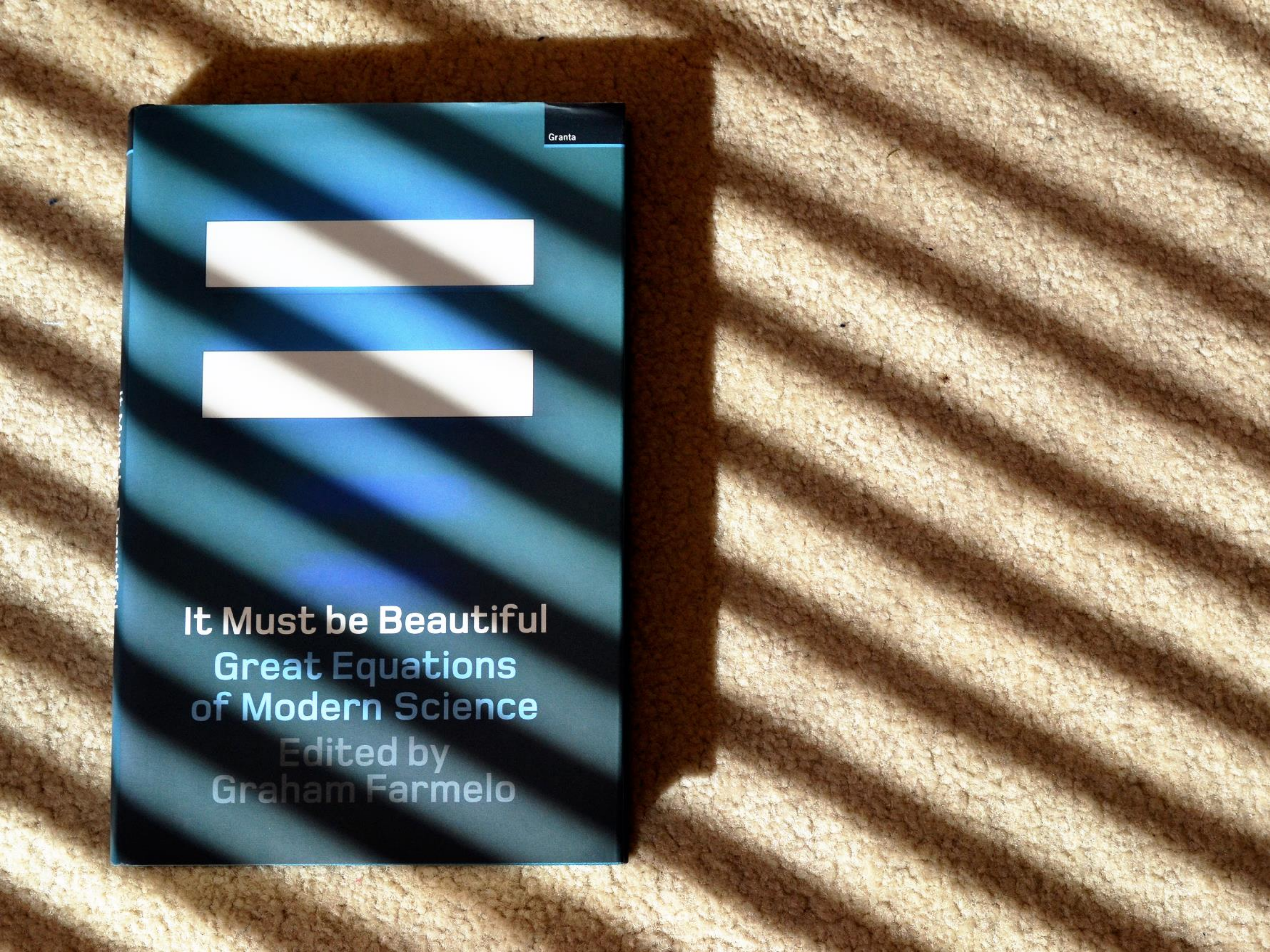
- **The implementation should be fast.**
- **It should be small.**
- **It should interoperate with the programs and tools that the expected users are already using.**
- **It should be bug-free, and if that requires implementing fewer features, do it.**
- **It should use parsimonious abstractions as long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

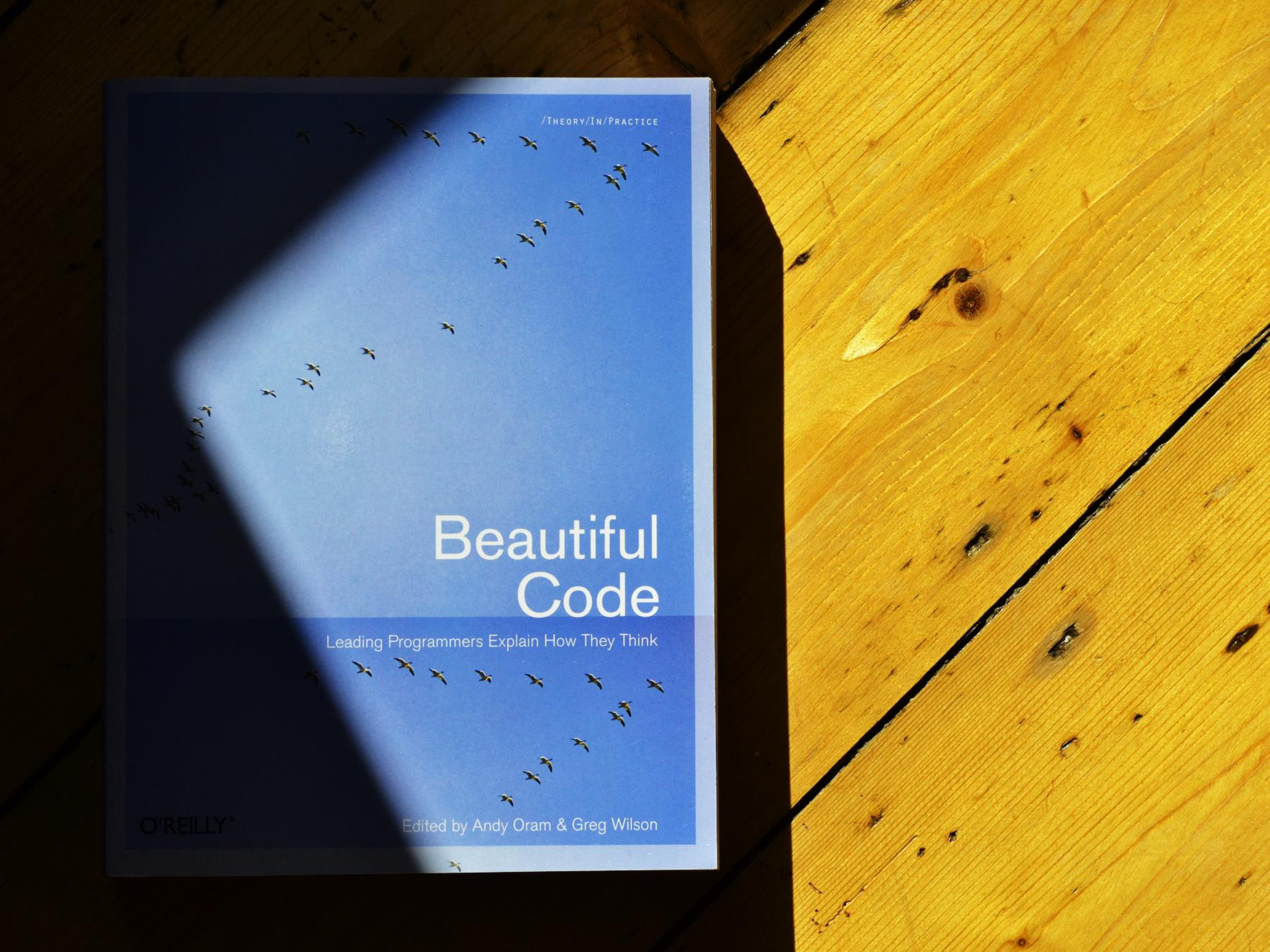


```
#!/usr/bin/perl
# ---------------------------------------------------------  PerlInterpreter
# PerlInterpreter must be the first line of the file.
#
# Copyright (c) 1995, Cunningham & Cunningham, Inc.
#
# This program has been generated by the HyperPerl
# generator.  The source hypertext can be found
# 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
```


Granta
It Must be Beautiful
Great Equations of Modern Science
Edited by Graham Farmelo


/Theory/In/Practice
Beautiful Code
Leading Programmers Explain How They Think
O'REILLY®
Edited by Andy Oram & Greg Wilson

SMALLTALK–80

THE LANGUAGE

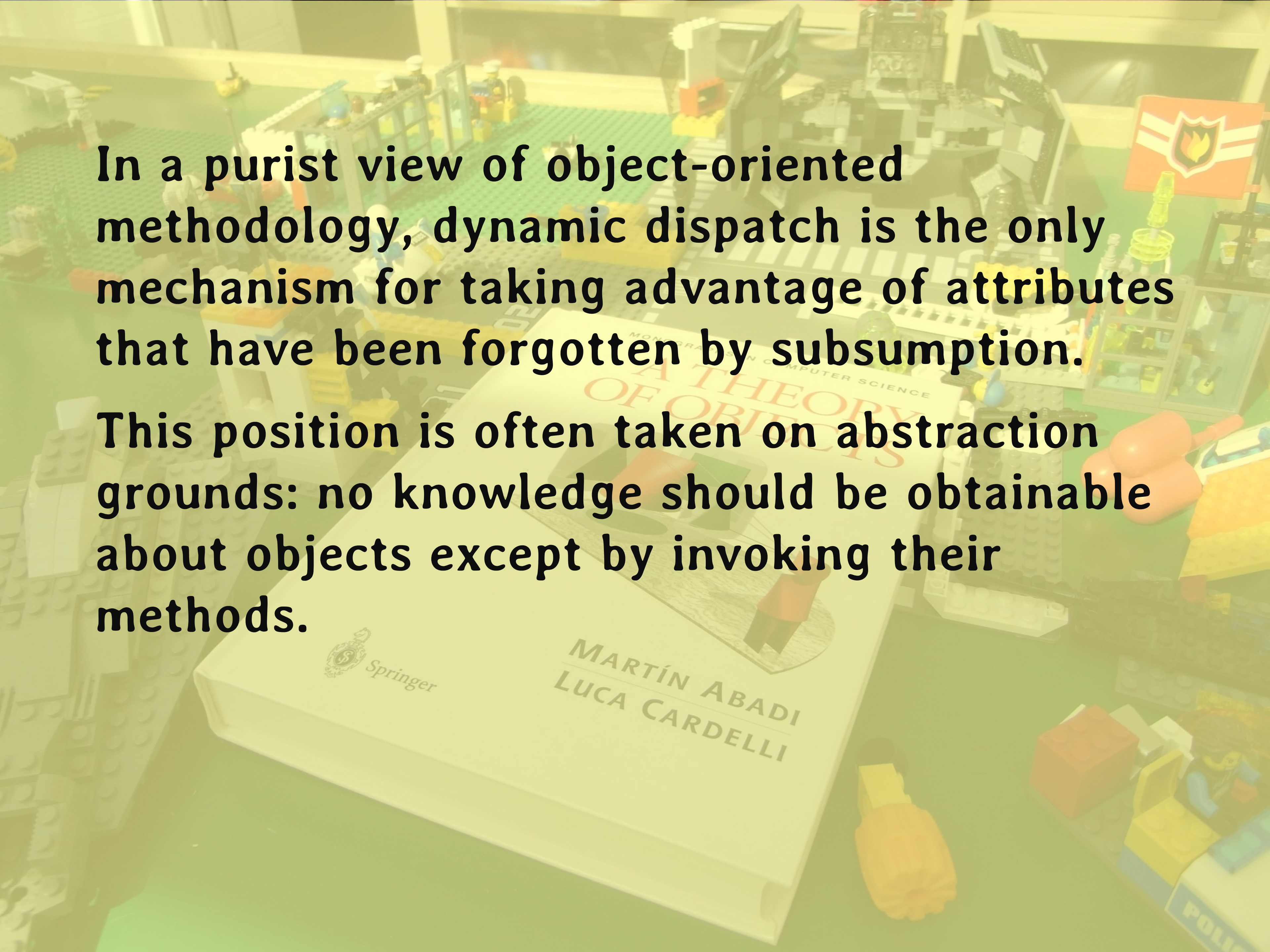
Adele Goldberg and David Robson

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

It can be done in Smalltalk and in LISP. There are possibly other systems in which this is possible, but I'm not aware of them.

Alan Kay

MONOGRAPHS IN COMPUTER SCIENCE
A THEORY OF OBJECTS
Springer
MARTÍN ABADI
LUCA CARDELLI
POLICE



In a purist view of object-oriented methodology, dynamic dispatch is the only mechanism for taking advantage of attributes that have been forgotten by subsumption.

This position is often taken on abstraction grounds: no knowledge should be obtainable about objects except by invoking their methods.

One of the most pure object-oriented programming models yet defined is the Component Object Model (COM).

It enforces all of these principles rigorously.

William Cook
"On Understanding Data Abstraction, Revisited"

There are only two kinds of languages: the ones people complain about and the ones nobody uses.

Bjarne Stroustrup

I always have it in the back of my head that I want to make a slightly better C.

But getting everything to fit, top to bottom, syntax, semantics, tooling, etc., might not be possible or even worth the effort.

As it stands today, C is unreasonably effective, and I don't see that changing any time soon.

Damien Katz

http://damienkatz.net/2013/01/the_unreasonable_effectiveness_of_c.html

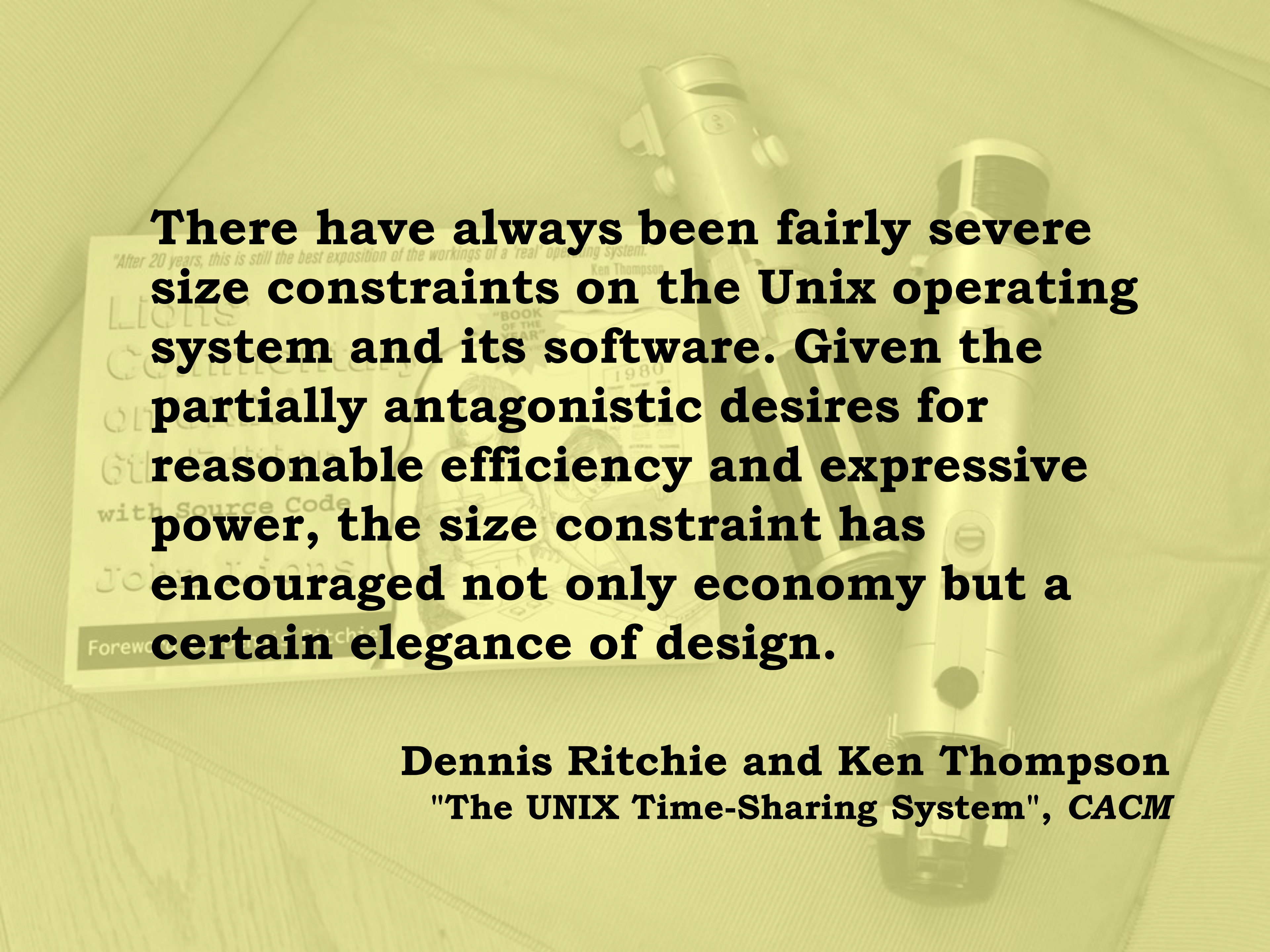
"After 20 years, this is still the best exposition of the workings of a 'real' operating system."
Ken Thompson

Lions' Commentary on UNIX® 6th Edition

with Source Code

John Lions

Foreword by Dennis Ritchie


"After 20 years, this is still the best exposition of the workings of a 'real' operating system."
Ken Thompson
"BOOK OF THE YEAR"
1980
with Source Code
Forew

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.

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

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

Doug McIlroy

The hard part isn’t writing little programs that do one thing well. The hard part is combining little programs to solve bigger problems. In McIlroy’s summary, the hard part is his second sentence: Write programs to work together.

John D Cook

http://www.johndcook.com/blog/2010/06/30/where-the-unix-philosophy-breaks-down/

Software applications do things they're not good at for the same reason companies do things they're not good at: to avoid transaction costs.

John D Cook

http://www.johndcook.com/blog/2010/06/30/where-the-unix-philosophy-breaks-down/

THOS. SMITH & SONS
OF SALTLEY LD
1901

Architecture is the decisions that you wish you could get right early in a project, but that you are not necessarily more likely to get them right than any other.

Ralph Johnson

The "defined" process control model requires that every piece of work be completely understood. Given a well-defined set of inputs, the same outputs are generated every time.

Ken Schwaber
Agile Software Development with Scrum

The empirical process control model, on the other hand, expects the unexpected. It provides and exercises control through frequent inspection and adaptation for processes that are imperfectly defined and generate unpredictable and unrepeatable results.

Ken Schwaber
Agile Software Development with Scrum

101 Things I Learned
in Architecture School
Matthew Frederick

Properly gaining control of the design process tends to feel like one is *losing* control of the design process.

The classic essay on "worse is better" is either misunderstood or wrong.

Jim Waldo

Decide for yourselves.

Richard P Gabriel