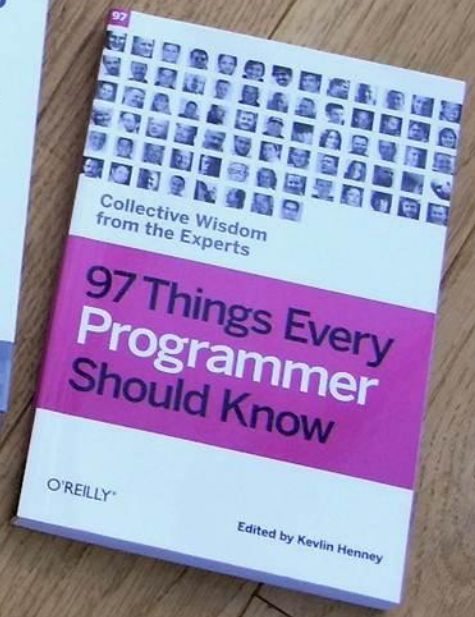
Patterns for the People

Kevlin Henney
kevlin@curbralan.com
@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

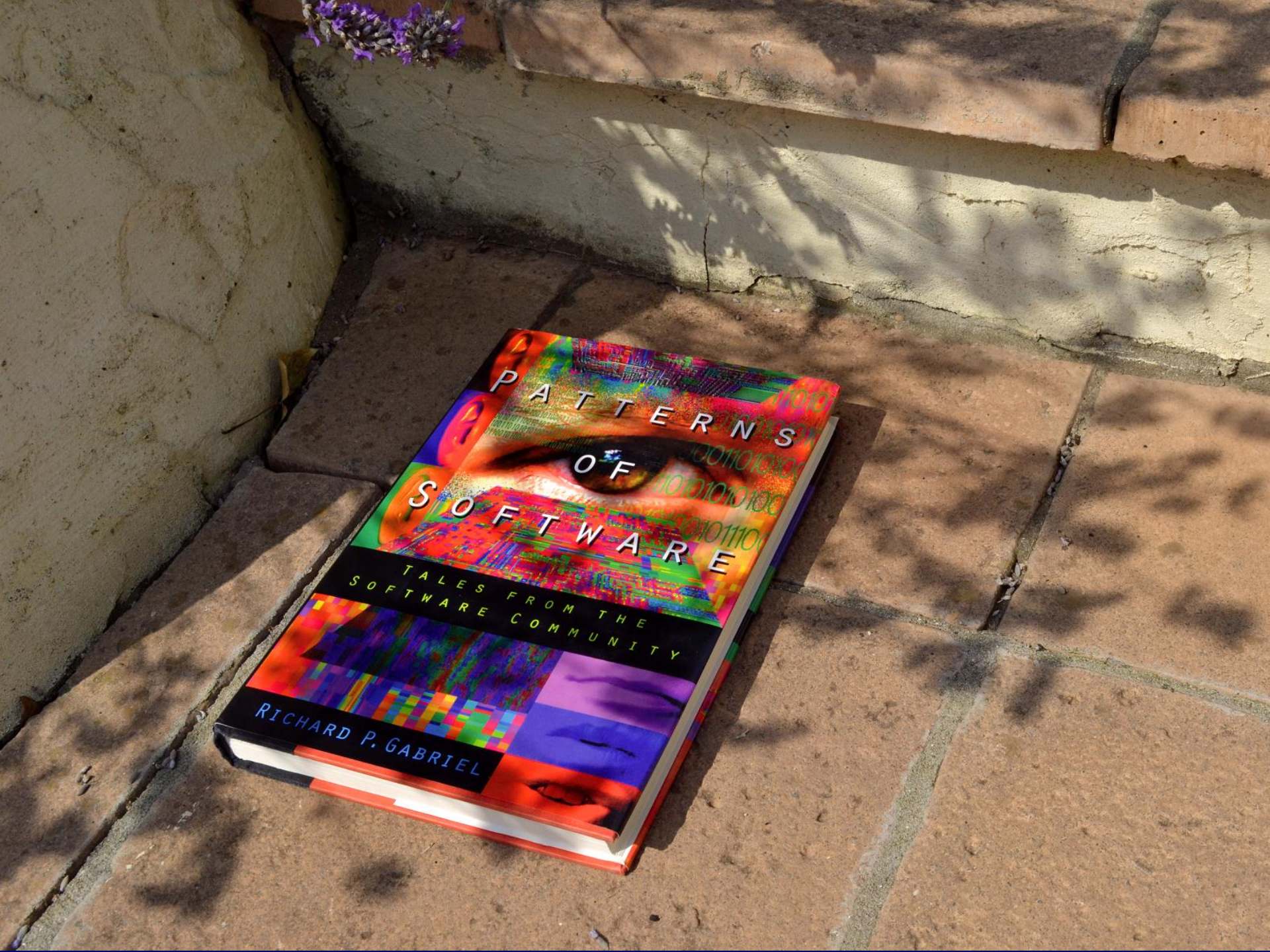

97
Collective Wisdom from the Experts
97 Things Every Programmer Should Know
O'REILLY
Edited by Kevlin Henney

FIRMITAS

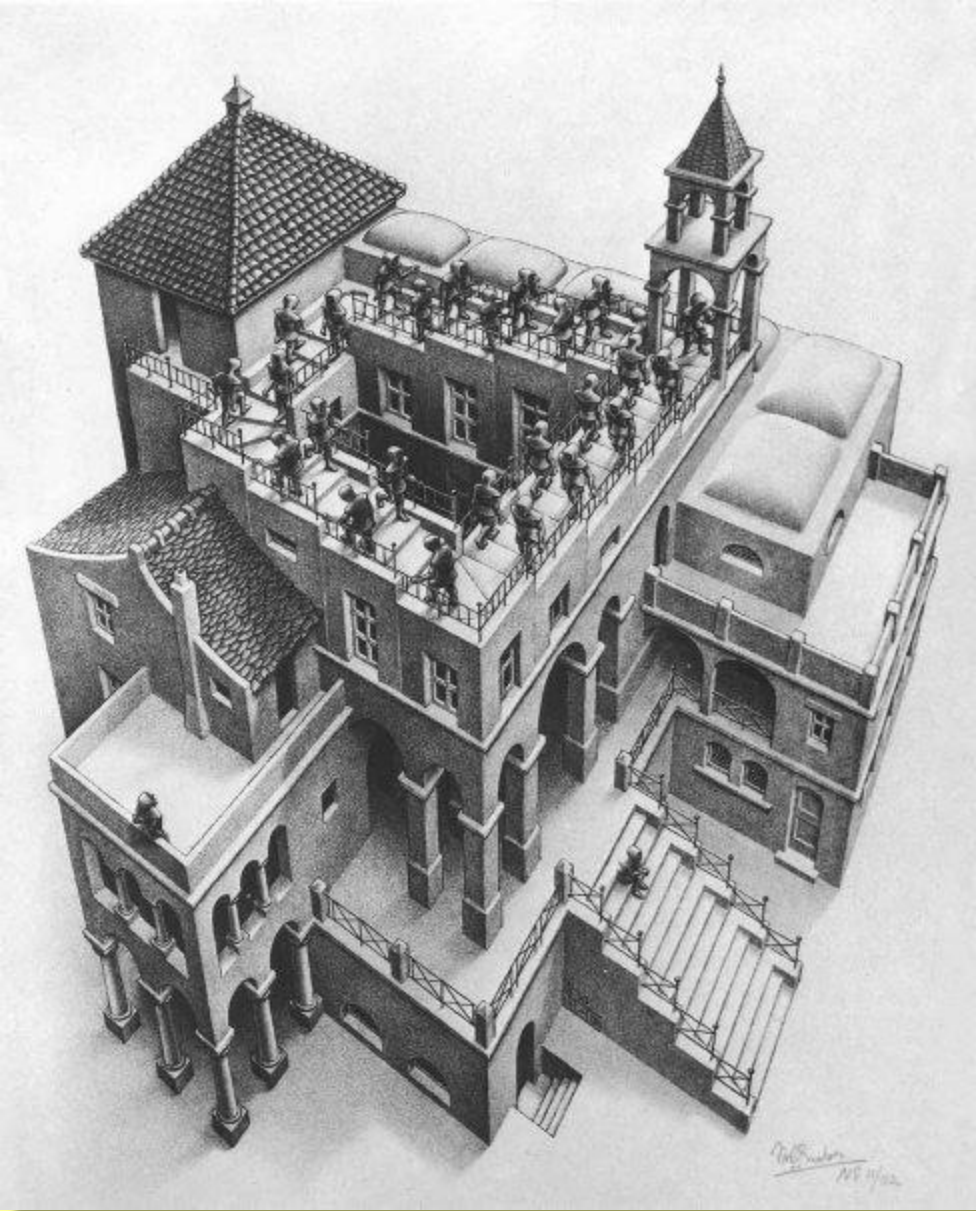
UTILITAS

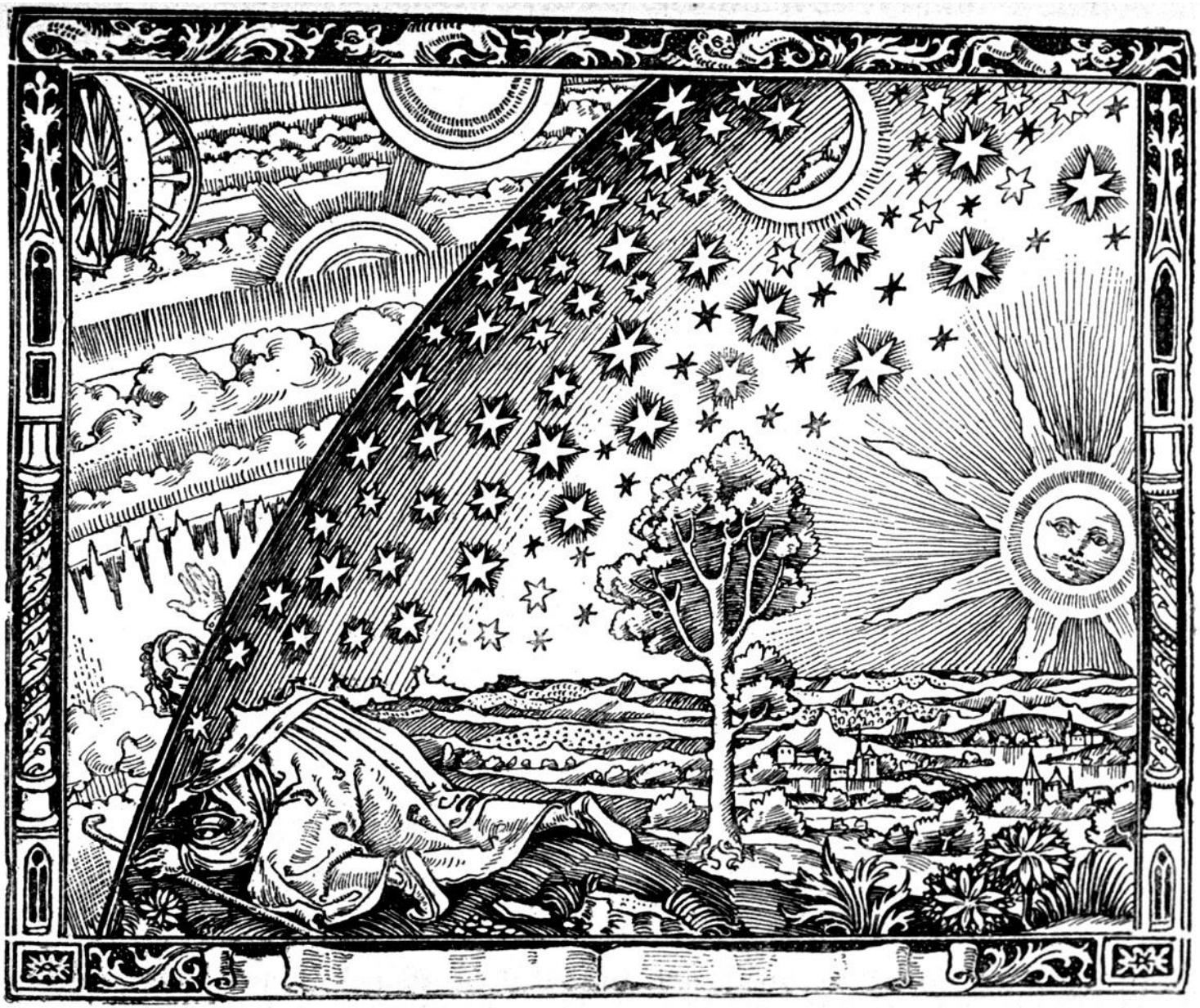
VENUSTAS

VITRUVIUS

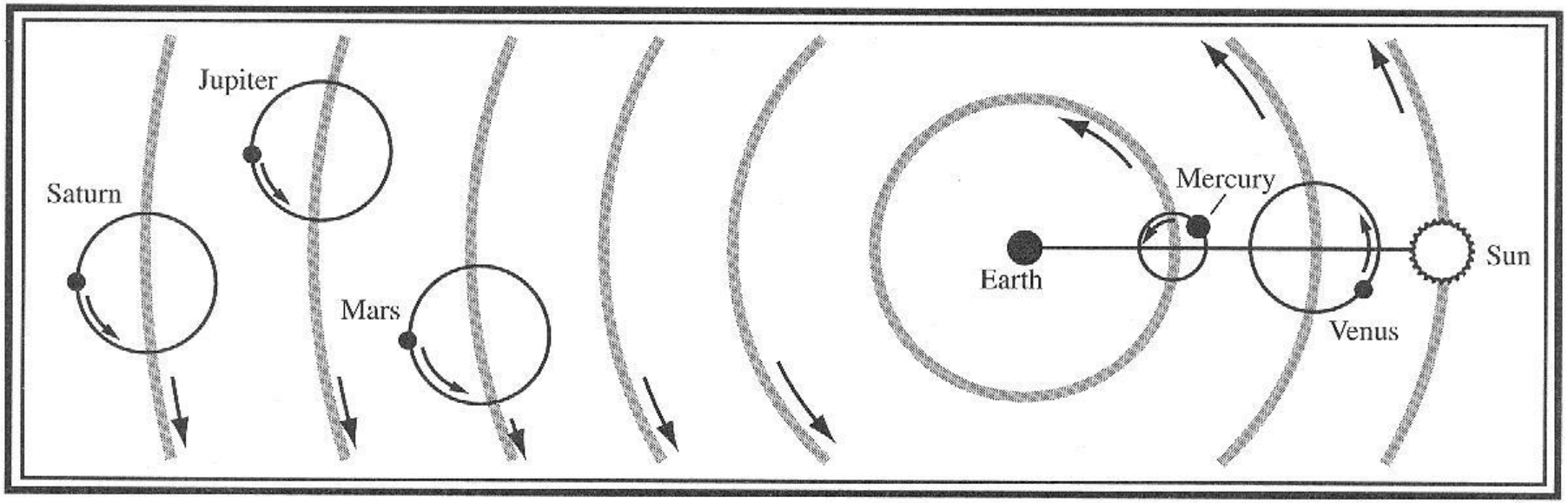

PATTERNS
OF
SOFTWARE
TALES FROM THE
SOFTWARE COMMUNITY
RICHARD P. GABRIEL

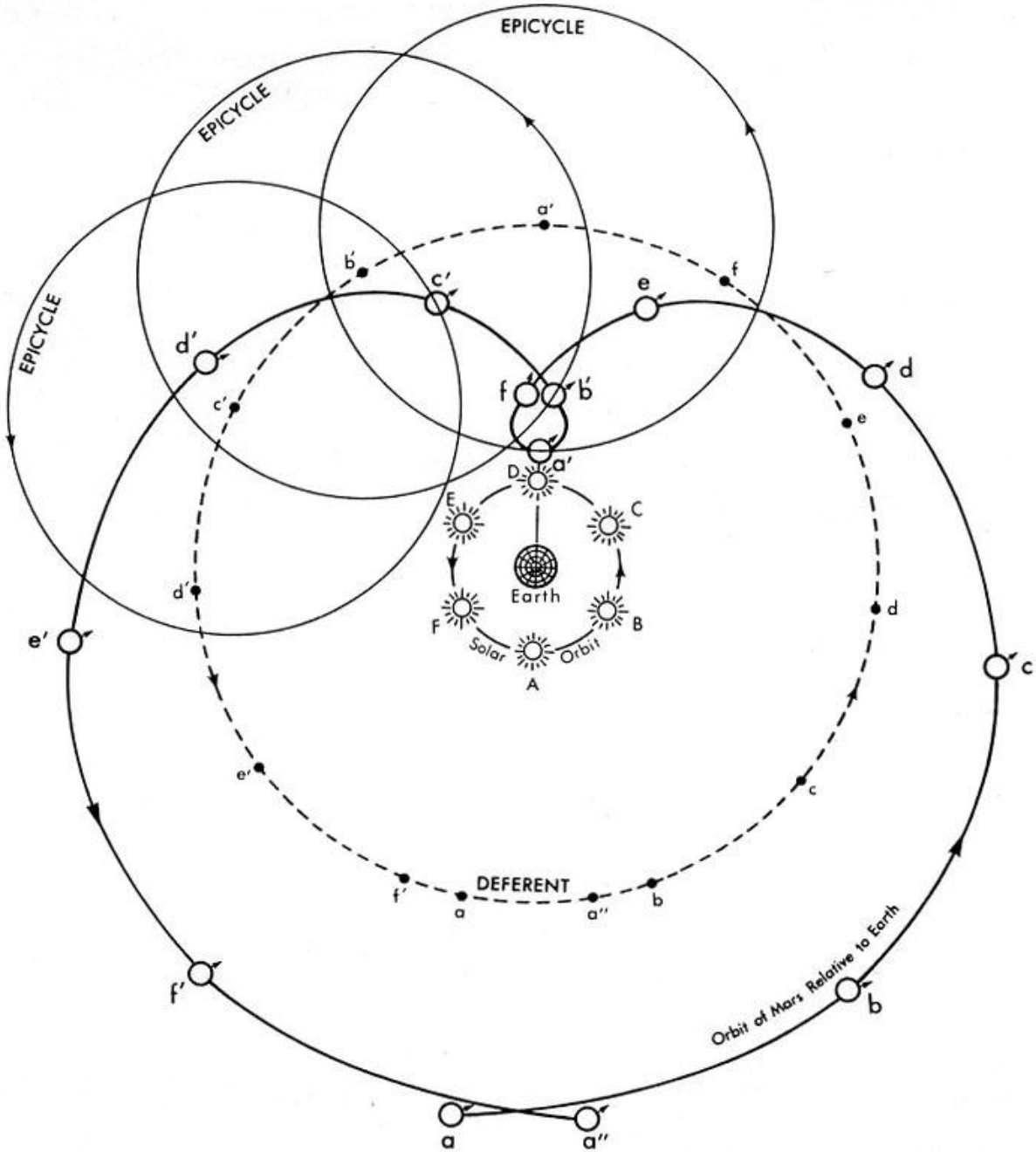
Habitability is the characteristic of source code that enables programmers, coders, bug-fixers, and people coming to the code later in its life to understand its construction and intentions and to change it comfortably and confidently. [...] Habitability makes a place livable, like home. And this is what we want in software — that developers feel at home, can place their hands on any item without having to think deeply about where it is.

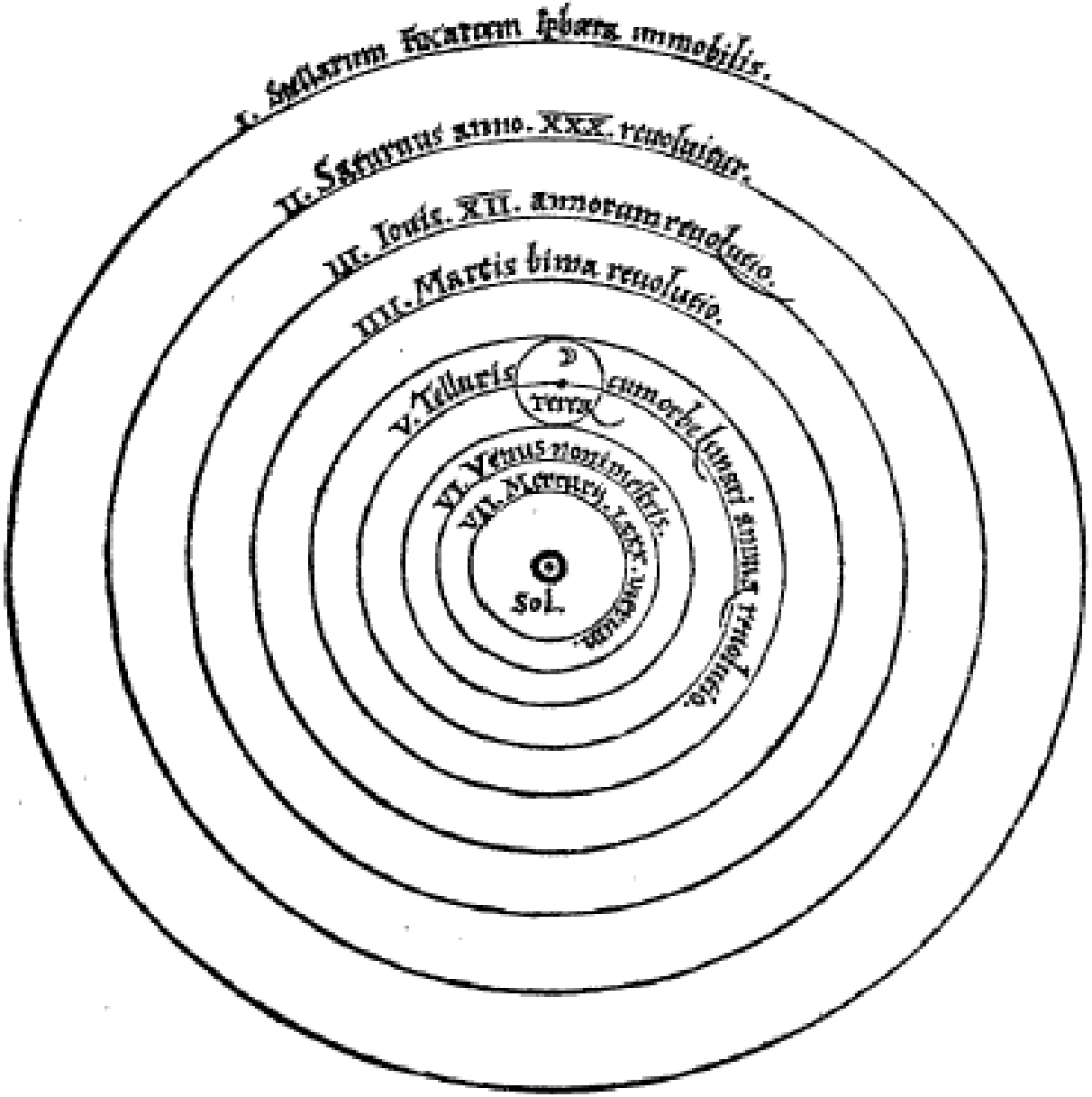


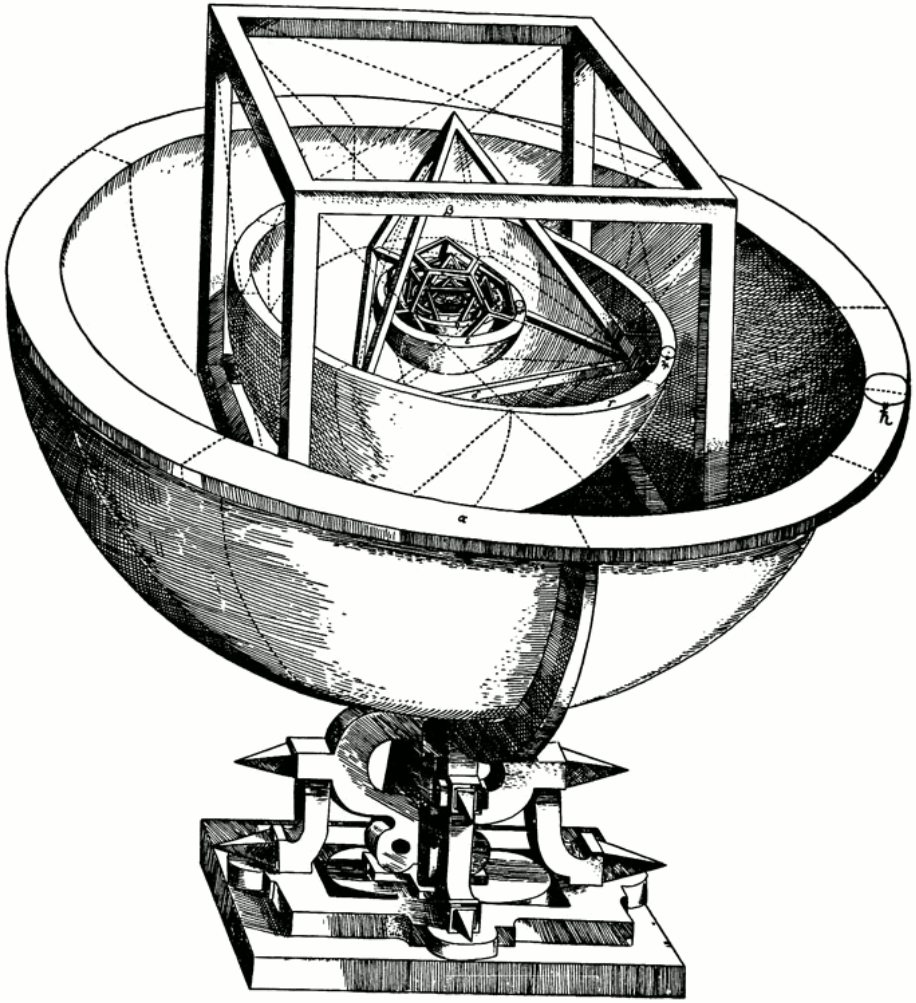


Un missionnaire du moyen âge raconte qu'il avait trouvé le point où le ciel et la Terre se touchent...


Jupiter
Saturn
Mars
Mercury
Earth
Sun
Venus


EPICYCLE
EPICYCLE
EPICYCLE
a'
b'
c'
d'
e'
f'
a
a''
b
c
d
e
f
Earth
A
B
C
D
E
F
Solar Orbit
DEFERENT
Orbit of Mars Relative to Earth


I. Stellarum fixarum sphæra immobilis.
II. Saturnus anno. XXX. revolvitur.
III. Iovis. XII. annorum revolutio.
IIII. Martis bima revolutio.
V. Telluris
Terra
cum orbe lunari annua revolutio.
VI. Veneris mensis revolutio.
VII. Mercurii. LXXX. dierum revolutio.
Sol.



That which is overdesigned, too highly specific, anticipates outcome; the anticipation of outcome guarantees, if not failure, the absence of grace.

William Gibson

Failure is a far better teacher than success.

Philip Delves Broughton

http://www.ft.com/cms/s/0/f33f5508-f010-11e0-bc9d-00144feab49a.html

THE AUTHOR OF THE WINDOWS FILE COPY DIALOG VISITS SOME FRIENDS.

http://xkcd.com/612/

If you want to learn how to build a house, build a house. Don't ask anybody, just build a house.

Christopher Walken

It has become commonplace to suggest that failure is good for entrepreneurs. In this view, failure that comes early in a founder's career can teach them important lessons about doing business and harden them up for the next start-up attempt.

David Storey, "Lessons that are wasted on entrepreneurs"

In the UK, the evidence is that novices are neither more nor less likely to have a business that either grows or survives than experienced founders. In Germany, where much more extensive statistical work has been undertaken, it is clear that those whose business had failed had worse-performing businesses if they restarted than did novices.

David Storey, "Lessons that are wasted on entrepreneurs"

In short, the assumption that entrepreneurs use the lessons of their own experience to improve their chances of creating a series of profitable businesses is not borne out by the evidence. Success in business remains, as in life, something of a lottery.

David Storey, "Lessons that are wasted on entrepreneurs"

The assertion that we can learn something from every failure is often heard. This study by Earl Miller and his colleagues Mark Histed and Anitha Pasupathy of the Massachusetts Institute of Technology's Picower Institute for Learning and Memory tests that notion by looking at the learning process at the level of neurons. The study shows how brains learn more effectively from success than from failure.

http://www.asfct.org/documents/journal/2009-11/Vol1-2-9.pdf

Brain cells keep track of whether recent behaviours were successful or not. When a certain behaviour was successful, cells became more finely tuned to what the animal was learning. After a failure, there was little or no change in the brain – nor was there any improvement in behaviour.

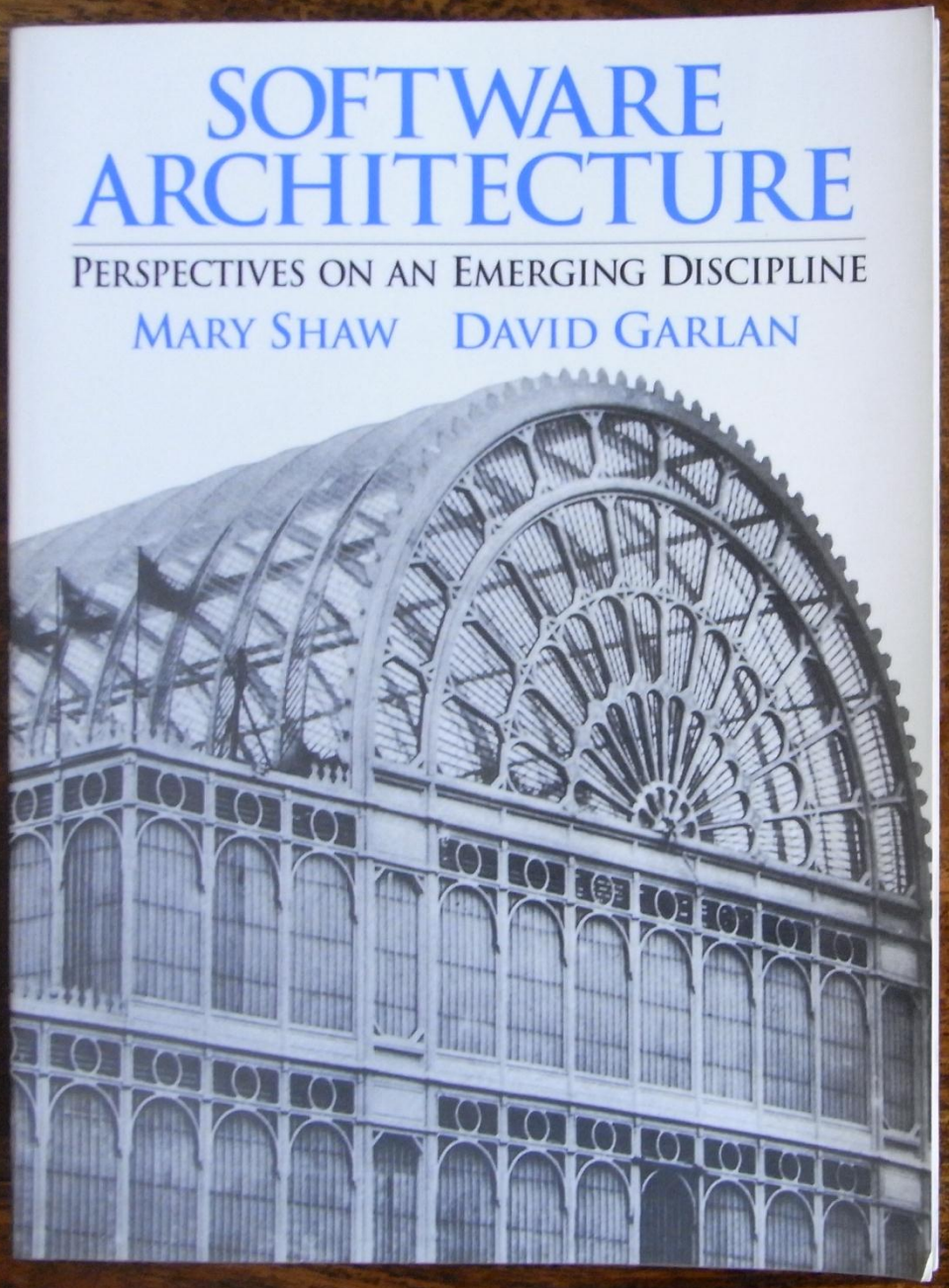
http://www.asfct.org/documents/journal/2009-11/Vol1-2-9.pdf

Anti-patterns don't provide a resolution of forces as patterns do, and they are dangerous as teaching tools: good pedagogy builds on positive examples that students can remember, rather than negative examples. Anti-patterns might be good diagnostic tools to understand system problems.

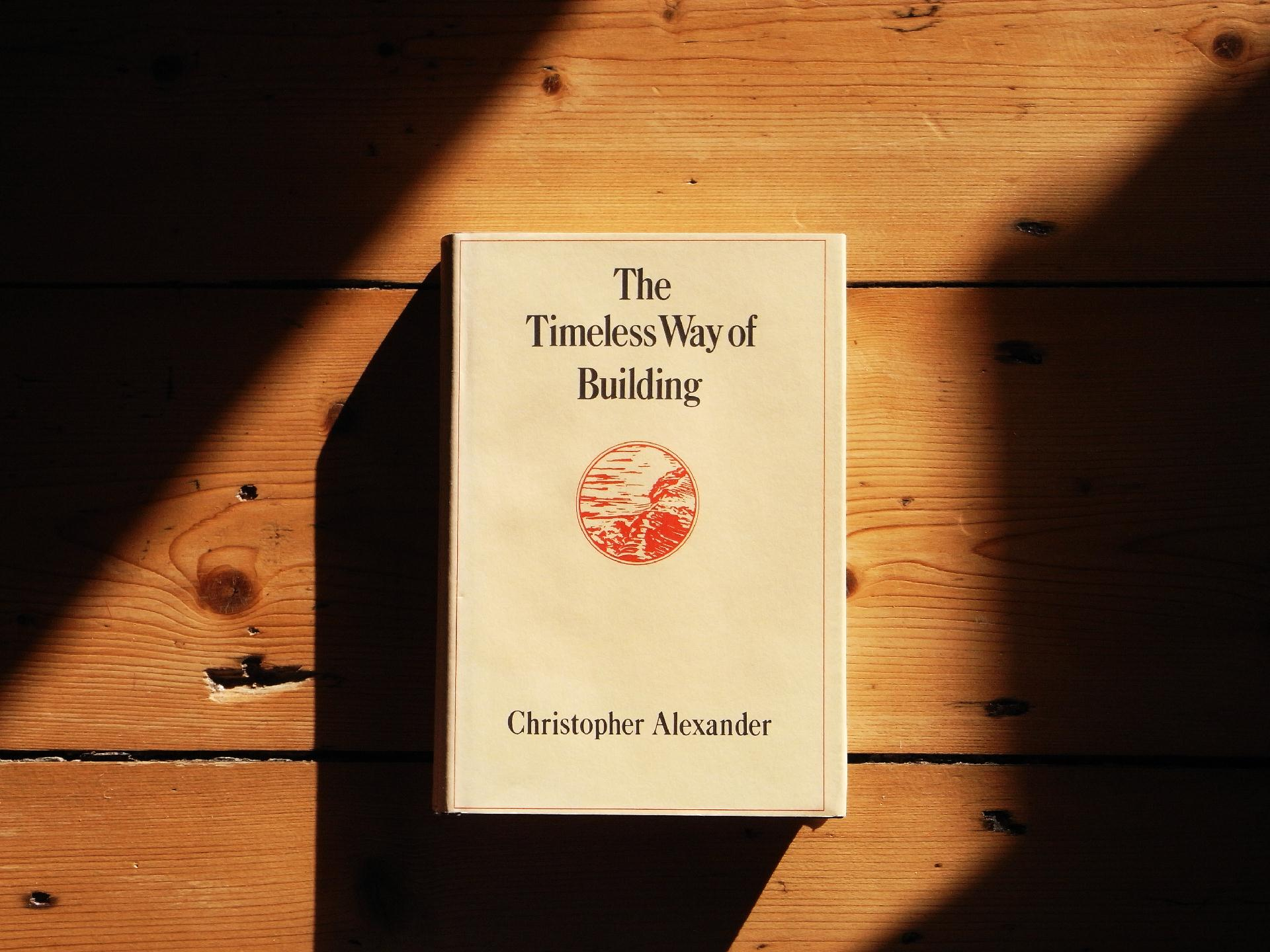
James Coplien, Software Patterns

Wise men profit more from fools than fools from wise men; for the wise men shun the mistakes of fools, but fools do not imitate the successes of the wise.

Cato the Elder



One of the hallmarks of architectural design is the use of idiomatic patterns of system organization. Many of these patterns — or architectural styles — have been developed over the years as system designers recognized the value of specific organizational principles and structures for certain classes of software.


The
Timeless Way of
Building
Christopher Alexander

We know that every pattern is an instruction of the general form:

context → conflicting forces → configuration

So we say that a pattern is good, whenever we can show that it meets the following two empirical conditions:

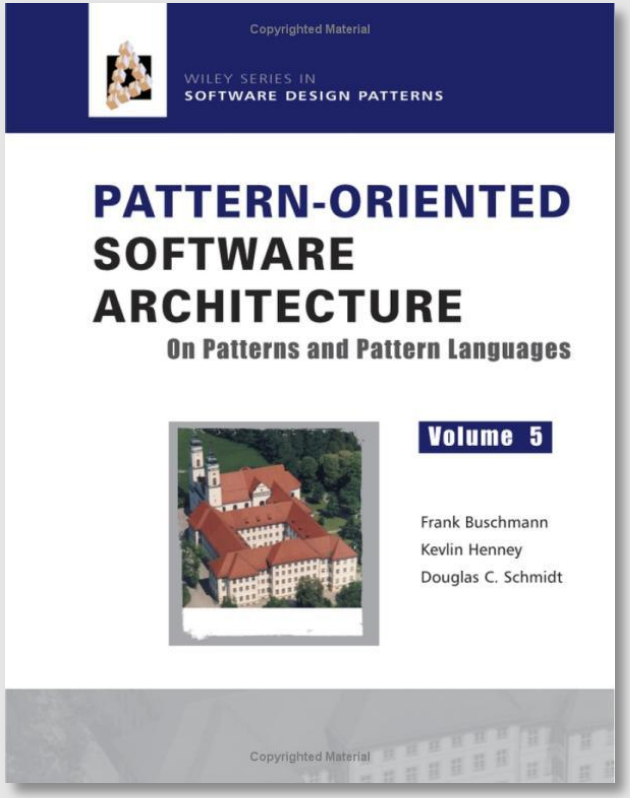
1. ***The problem is real.*** **This means that we can express the problem as a conflict among forces which really do occur within the stated context, and cannot normally be resolved within that context. This is an empirical question.**
2. ***The configuration solves the problem.*** **This means that when the stated arrangement of parts is present in the stated context, the conflict can be resolved, without any side effects. This is an empirical question.**


PATTERN
SHOP
Caution
Uneven Floor
Patterns

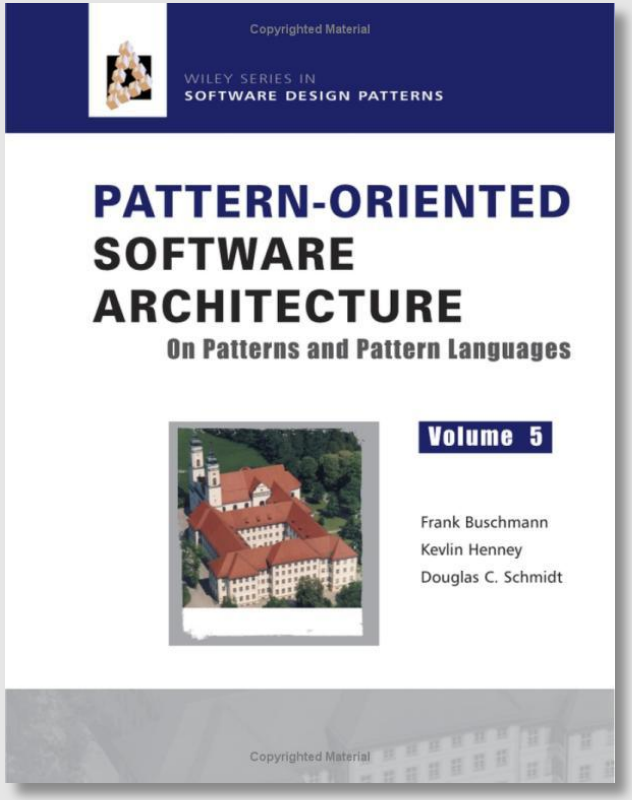

THE SINGLETON
Single Malt Scotch Whisky
of Dufftown
DUFFTOWN
ESTD 1896
MATURED FOR
12 YEARS
12
PRODUCT OF
SCOTLAND
DISTILLED MATURED AND BOTTLED IN SCOTLAND
PERFECTLY BALANCED, NATURALLY RICH & SMOOTH
40%vol
1litre
DUFFTOWN DISTILLERY, DUFFTOWN, KEITH
BANFFSHIRE AB55 4DB

Style is the art of getting yourself out of the way, not putting yourself in it.

David Hare



The form used for a pattern description matters a great deal to both pattern authors and readers. It defines a vehicle for presentation, along with the perspective and bias that can bring. Thus, although in one sense the choice of form can be considered arbitrary, in another sense it is anything but: the essence of a good pattern can be considered independent of any description of it, but the description frames how the pattern will be perceived.

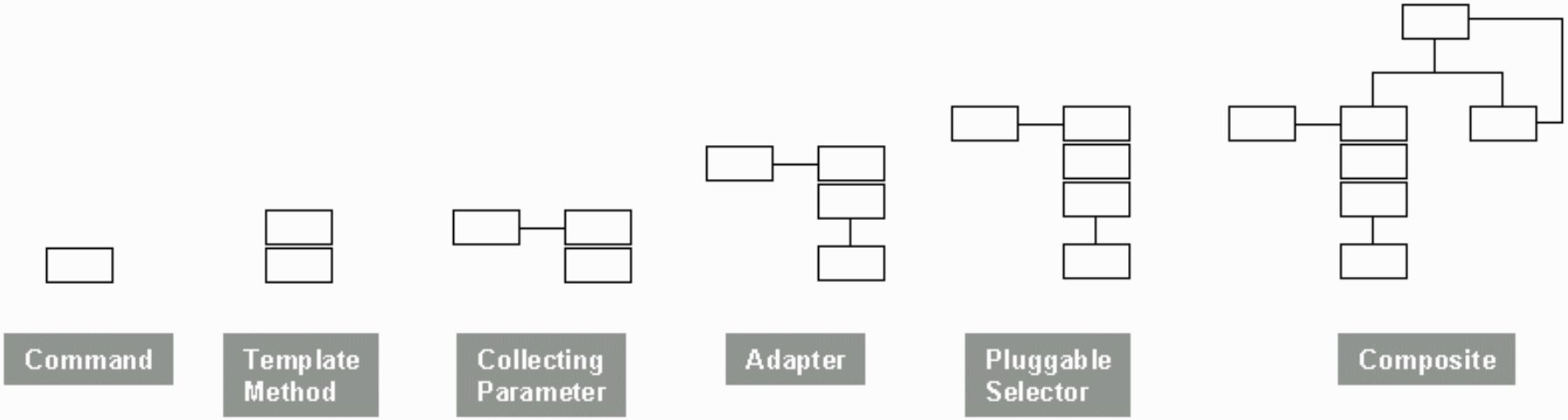


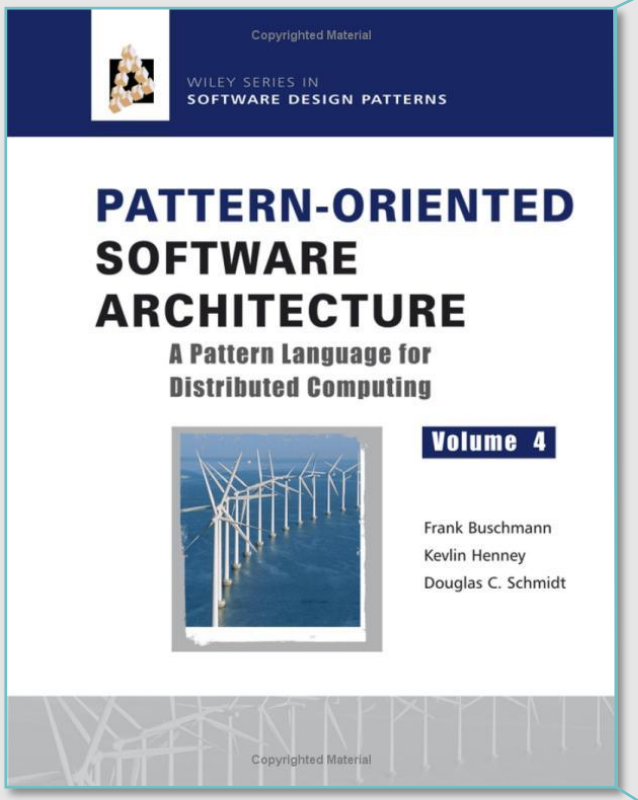
A pattern's audience is ultimately always human. Although a developer may support application of a software pattern solution through libraries and generators, it is the developer and not the technology that is aware of the pattern. A pattern is more than just a solution structure, so its audience must also have a sense of the context, the forces, and the consequences that are associated with a solution.


COMMAND PATTERN
NEED TO KNOW WHEN THE COMMAND
BEEN EXECUTED
EXAMPLE
SYSTEM
ACTOR


Pepe's Pizza
2222 5555
www.peppes.no

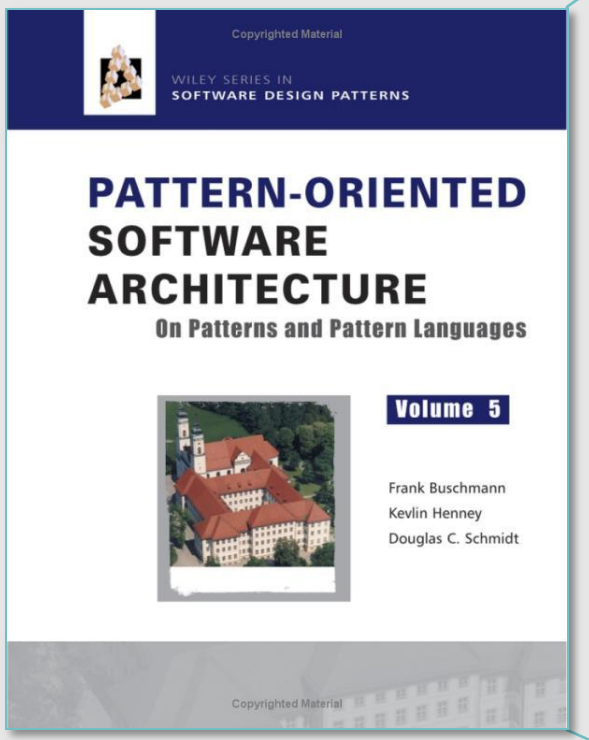

0
5
10
15


Command
Template Method
Collecting Parameter
Adapter
Pluggable Selector
Composite



History rarely happens in the right order or at the right time, but the job of a historian is to make it appear as if it did.

James Burke



portion from the grammar of our example pattern language for request handling, derived from the pattern sequences presented above:

$\varnothing \rightarrow^{\circ}$ (COMMAND $\rightarrow$ EXPLICIT INTERFACE $\rightarrow^{\circ}$
(MEMENTO $\rightarrow^{\circ}$ COMPOSITE $\rightarrow^{\circ}$ COMMAND PROCESSOR $\rightarrow$
COLLECTIONS FOR STATES $\rightarrow$ STRATEGY $\rightarrow$ NULL OBJECT)
| (COMPOSITE $\rightarrow^{\circ}$ MEMENTO))

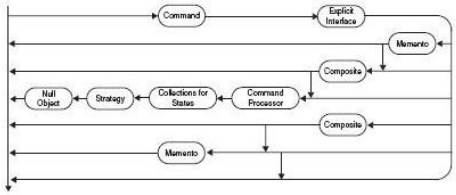
A BNF-derived notation [EBNF96], as used for specifying the syntax of programm
grammar of a
alternative:

COMMAND
lowed by
followed
POSITE, w
which if
which m
NULL OB
followed
MENTO.

Graphical no
pattern langu
which visual
design [CzEi0
language for
root *concept*
combinations
as whether t
tional. In bo
confused wit
concepts via

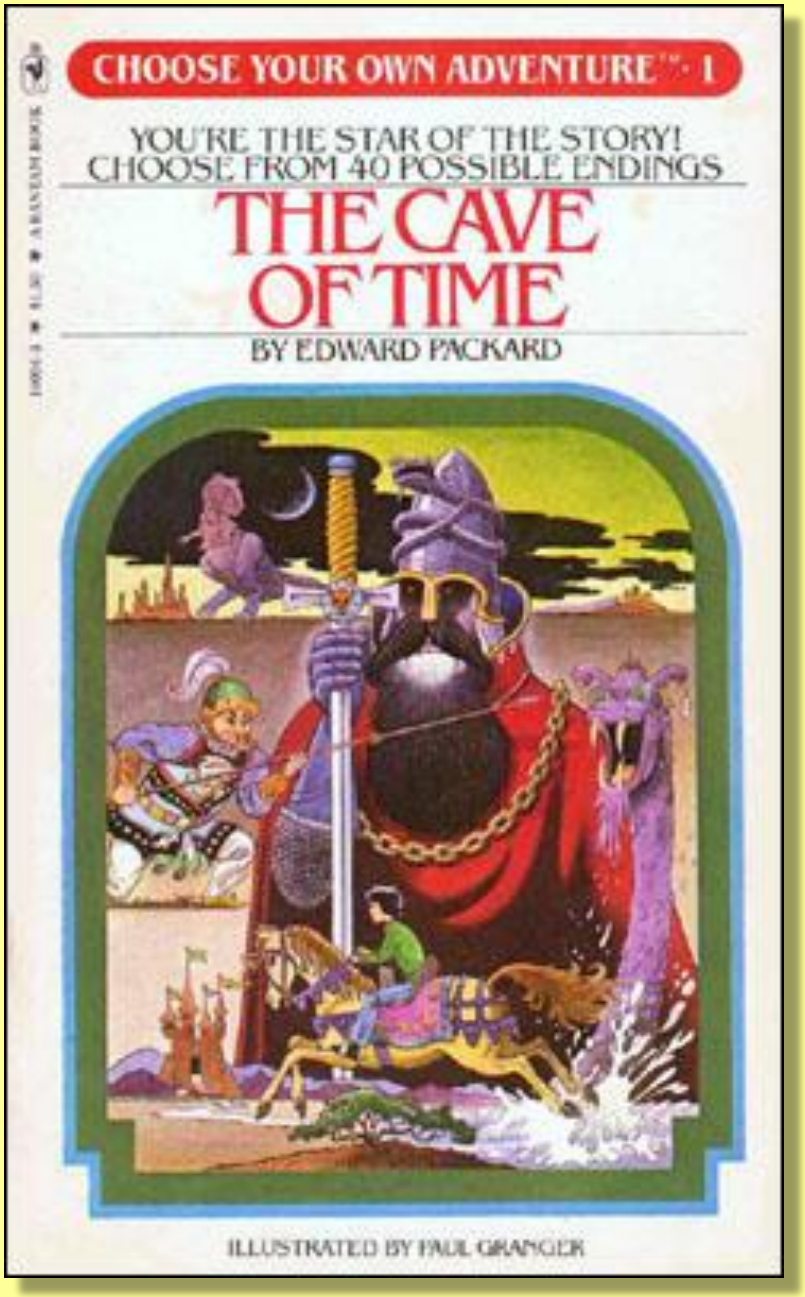
Another grap
'railroad' not
since the 197
in particular

The portion of our pattern language for request handling outlined above could be represented as follows using the 'railroad notation':



The preferences of pattern language authors or the demands of their target audience determine the specific expression of a grammar that works best—whether a list of pattern sequences, formal or semi-formal prose, or a graphical form of describing grammar rules, and whether interwoven with the pattern descriptions or separate. For example, the pattern language for distributed computing from POSA4 expresses grammar rules in prose, interwoven with the pattern descriptions [POSA4]. This option has been chosen by most pattern languages in the software area, from design-centric pattern languages [VSW02] [Fow02] [HoWo03] [VKZ04] to pattern languages for development process and organization, and project and people management [Ker95] [Cun96] [CoHa04].

Regardless of which grammar form is chosen, however, it is important that documented pattern languages actually offer guidance on the issue of meaningful paths through a language. Otherwise, it is hard to avoid the selection of ill-formed pattern sequences that create fundamentally broken software. The set of sensible sequences through a language is part of the language and not something accidental or separate. Thus making the grammars of pattern languages more explicit is one method for supporting their appropriate use. However, we must also recognize some practical limitations in this endeavor: the grammar for

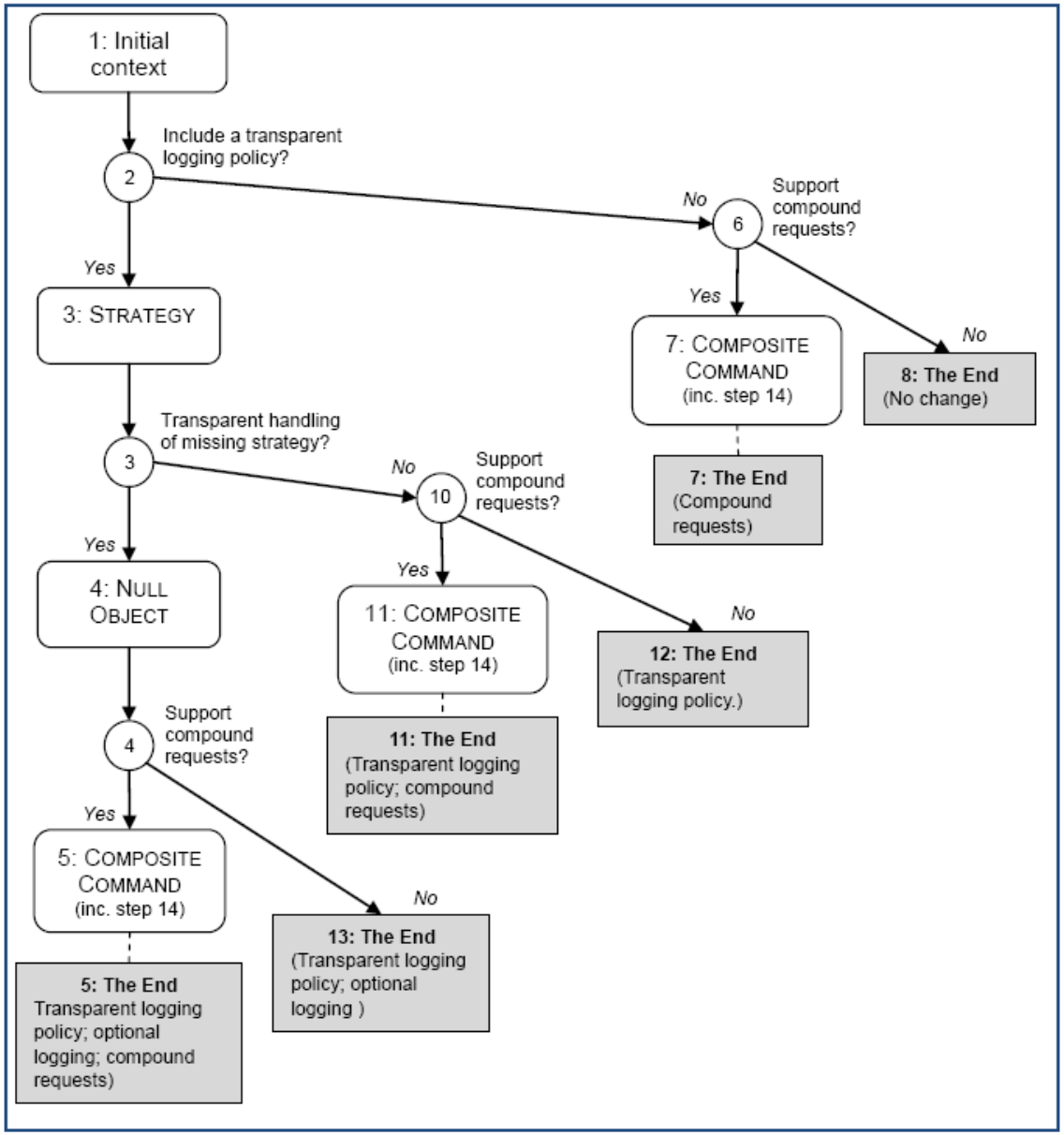

CHOOSE YOUR OWN ADVENTURE™ · 1
YOU'RE THE STAR OF THE STORY!
CHOOSE FROM 40 POSSIBLE ENDINGS
THE CAVE OF TIME
BY EDWARD PACKARD
ILLUSTRATED BY PAUL GRANGER

2

You now realise that the framework needs a logging facility for requests, and wonder how logging functionality can be parameterized so that users of the framework can choose how they wish to handle logging, rather than the logging facility being hard-wired.

If you wish to use inheritance to support variations in housekeeping functionality, turn to 7.

Otherwise if you prefer the use of delegation, turn to 3.


1: Initial context
Include a transparent logging policy?
2
No
6
Support compound requests?
Yes
3: STRATEGY
Yes
7: COMPOSITE COMMAND (inc. step 14)
No
8: The End (No change)
Transparent handling of missing strategy?
3
No
10
Support compound requests?
7: The End (Compound requests)
Yes
4: NULL OBJECT
Yes
11: COMPOSITE COMMAND (inc. step 14)
No
12: The End (Transparent logging policy.)
Support compound requests?
4
11: The End (Transparent logging policy; compound requests)
Yes
5: COMPOSITE COMMAND (inc. step 14)
No
13: The End (Transparent logging policy; optional logging)
5: The End Transparent logging policy; optional logging; compound requests)

Patterns Manifesto

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