97 Things Every Software Architect Should Know: Collective Wisdom from the Experts

Richard Monson-Haefel (Editor)

97 Things Every Software Architect Should Know: Collective Wisdom from the Experts

Richard Monson-Haefel (Editor)

97 Things Every Software Architect Should Know: Collective Wisdom from the Experts Richard Monson-Haefel (Editor)
In this truly unique technical book, today's leading software architects present valuable principles on key development issues that go way beyond technology. More than four dozen architects -- including Neal Ford, Michael Nygard, and Bill de hOra -- offer advice for communicating with stakeholders, eliminating complexity, empowering developers, and many more practical lessons they've learned from years of experience. Among the 97 principles in this book, you'll find useful advice such as:

Don't Put Your Resume Ahead of the Requirements (Nitin Borwankar) Chances Are, Your Biggest Problem Isn't Technical (Mark Ramm) Communication Is King; Clarity and Leadership, Its Humble Servants (Mark Richards) Simplicity Before Generality, Use Before Reuse (Kevlin Henney) For the End User, the Interface Is the System (Vinayak Hegde) It's Never Too Early to Think About Performance (Rebecca Parsons) To be successful as a software architect, you need to master both business and technology. This book tells you what top software architects think is important *and* how they approach a project. If you want to enhance your career, *97 Things Every Software Architect Should Know* is essential reading.

97 Things Every Software Architect Should Know: Collective Wisdom from the Experts Details

Date	:	Published February 12th 2009 by O'Reilly Media (first published January 1st 2009)
ISBN	:	9780596522698
Author	:	Richard Monson-Haefel (Editor)
Format	:	Paperback 222 pages
Genre	:	Computer Science, Programming, Software, Architecture, Nonfiction, Science, Technology

From Reader Review 97 Things Every Software Architect Should Know: Collective Wisdom from the Experts for online ebook

Steve Whiting says

This small and extremely slender book, consisting of more white space than content, contains a series of what are basically one-liners from a variety of authors, with about a page of exposition on each.

A few of the points are incisive, the vast majority are blindingly obvious, and a small number are ridiculous (I'm still trying to work out why a software architect "must" be able to cable a network. No reason why s/he shouldn't be able to, but why is this a necessity?).

The targetting of the advice is pretty wayward as well - about half of the points are relevant to a lead developer/designer, but the rest are scatter-gun advice more relevant to a project or product manager, or a team leader.

If you're taking your first steps from "developer" to "lead developer", you might find some of the advice pertinent. If you have much more experience, you're likely to be wasting both your time, and (even at the Amazon price rather than the ludicrous $36 cover price) your money.

Robert Richter says

Some tips were great. Some were just common sense. A lot of overlapping stuff.

Robson Castilho says

This book is a set of 97 short essays written by a lot of software architects. It brings us nice advises encompassing technical aspects and soft skills.
Easy and fast reading, it's a good book for everyone who wishes to understand better the skills needed to a software architect and a good start to think about the "big picture" of a software project.

Tess Huelskamp says

Why not 98, 99, or 100 things every architect should know?

There are a few good points in this book but each essay is less than 2 single-spaced pages. That's not nearly enough depth to go into a worthwhile conversation and this book would've been better written as a series of tweets.

Context: I had to read this book for class and wouldn't have picked it up myself.

Tomi Turtiainen says

The problem is that this is not really a book per se, but instead just a collection of short articles circling around the topic of software architects. It does give you some sort of an idea about the work of software architects – what they job consists of, what kinds of problems they're dealing with. But then again most of the advice given is really common sense. The most interesting part was the story how a database was selected for the M1 Abrams battle tank's software system. I'm sure there are better books if you're interested in being a software architect.

???? ????? says

If you are familiar with Agile software development and Domain Driven Design, you will find many of the tips in the book have relevance!
To sum up the most important things I got from the book:
- Go Agile, Go Agile, Go Agile
- Design patterns is important (they are not those of GoF patterns only), but you should know anti-patterns, and refactor to patterns not try to force applying them in the first place!
- An architect is a leader as well
- An architect should have hands on experience, and can write code

Steve says

Short, personal essays on software architecting as a practice. Some fell flat, yet enough hit the mark--and some quite pointedly--on a theme worth remembering, reflecting on against one's experience. The kind of book perhaps to skim again after a new phase of completed projects.

Sure, don't expect to make this your one book on software architecture. And Yet. In a realm full of large, highly organized, overly prescriptive books, this reads more like a series of conversations you might have with colleagues at a conference or q&a with a workshop speaker. Without the powerpoints, travel and crummy reception food.

Ushan says

A collection of 97 two-page essays by 97 software architects about things they think every software architect should know. Most of them are very reasonable (don't use clever tricks; don't let your software have too many layers; understand that the usual abstractions are broken in the error cases). Some, however, are strange (be a good manager for different kinds of people as described in Neal Stephenson's Cryptonomicon; realize that software architecture has ethical consequences? it does, but they are usually far removed from the decisions software architects make; it is all about the data? for a bank, maybe; for an airline flight scheduler, no).

Leo Mazzi says

Well why 97? That was my first question. Then after reading it it was - it was not even 97! Several advices are same thing said with other words, a lot of it are advices from same autors so in general was not much usefull or new. That is why I probbably needed almost a month to finish it. :)

Eabait says

Dos o tres consejos bien. El resto son generalidades, cosas que ya no sirven, o malos consejos

Saul Gonzalez says

Trae comentarios interesantes, algun que otro buen consejo.

Sebastian Gebski says

97 one-liners, chosen to be approachable & non-controversial. Fit more for an aspiring developer than actual architect. Not really revealing, bloated with repeated expert bios. Sadly this book quickly becomes annoying & it gets harder & harder to resist the temptation to skip subsequent "chapters".

Skip it.

Alex Ott says

Main problem with this book, is that isn't real book - it set of very short essays, that almost not linked to each other excluding mega-topic called "software architecture".
So it very inconsistent - after essay on performance, goes essay on communication skills, then essay on code organization, etc. It's hard to read such things...
It's better to read normal books, like "Release It!", "12 essential skills for software architects", etc. if you want to read something about software architecture processes and related things

Mark Seemann says

97 two-page pieces of advice about software architecture. Since each entry is only two pages, there's a limit to how in-depth they can be, so they tend to stay fairly general.

There's a few good points here and there, but I think I've already forgotten most of them again.

Ankita Mogha says

After a long time, I have read a technical book and reading this book has been a completely different experience as compared to the other technical books I have read before. This book gives insights of the things a software architect should know from the perspective of the technical experts. I wrote an article as my learnings from this book while I was in middle of it: https://www.linkedin.com/pulse/things...
I have an urge of re-reading the book and write another refined article of my other major learnings from this book.