The Best Software Engineering Books

karllhughes.com/posts/software-engineering-books

2021年4月7日

Karl Hughes

Buying and Scaling Digital Service Businesses
2021, Apr 07 — 11 minute read

Want to learn how to scale your digital service business?

Early in my career as an engineering manager, I wrote a long list of my favorite engineering management books. That list has reached thousands of new technical leaders, but I'm often asked for book recommendations by engineers who *aren't* interested in making the leap into management.

I've had this informal list brewing for years, and I've often shared sections of it with new bootcamp graduates, but this is the first time I'm sharing this list publicly. I hope this helps you sort through the thousands of software engineering books that are available and move your career forward - wherever you are today.

Sponsor

For Early-Career Engineers

Whether you just graduated from college, a developer bootcamp or you're a self-taught coder, these books are where I would tell a new software engineer to start. These books are mostly focused on practical, applied programming knowledge, but I have included some reads that will give you a taste of theory and interpersonal best practices.

The first three years of your software development career will likely be the rockiest, but I hope these books will help you get through it.

Cracking the Coding Interview: 189 Programming Questions and Solutions

Let me be honest, I hate the state of coding interviews right now. So, while I don't put engineers through the gauntlet like this, whiteboard and algorithmic interviews are still very common in our industry. In order to break in, it will help to get good at these. That's where a book like *Cracking the Coding Interview* comes in.

"I have some small quibbles with how some of the problems were worded, but the level of difficulty is very representative of what the big tech companies are using. You might even run into these questions in your own interviews since Gayle is choosing questions that are popular among interviewers today."

The Imposter's Handbook

Rob Conery's book was written explicitly for developers *without* a traditional computer science background. Like *Cracking the Coding Interview* above, it will help you get your foot in the door, but I'd argue *The Imposter's Handbook* is more broadly useful as well. It covers many of the fundamental concepts that will help you get a better idea of the lower-level pieces of computer programming.

"I am being schooled right now and it feels like good! I cannot recommend this too much for people like myself who never went to college for computer science but wish for proper understanding of the concepts."

Clean Code: A Handbook of Agile Software Craftsmanship

From testing to class and function design to variable naming, *Clean Code* covers applied software design fundamentals. I buy a copy of this for every engineer I hire and encourage them to read (or re-read) it as part of their onboarding. I love the examples and concrete logic behind each of Uncle Bob Martin's recommendations.

Note: I wrote a longer review of Clean Code a few years ago if you'd like to check that out.

Apprenticeship Patterns: Guidance for the Aspiring Software Craftsman

Dave Hoover helped create Dev Bootcamp, one of the first short-term coding schools in the United States, and wrote this great model for new developers. It covers many of the common issues you're likely to face as a new developer, mostly focusing on interpersonal and motivational challenges.

"Brilliant stuff! Reading this book was like being in a time machine that pulled me back to those key learning moments in my career as a professional software developer and, instead of having to learn best practices the hard way, I had a guru sitting on my shoulder guiding me every step towards master craftsmanship."

For Senior Engineers

As you progress into a senior or principal engineering role, much of your learning will start to get more specialized. That said, there are a few books that will help you continue to grow as an engineer and architect during the long, middle phase of your career.

The Architecture of Open Source Applications

As I gained real-world experience as a software engineer, I always wanted to read more case studies. It's exceedingly rare to find a deep dive into the architecture and evolution of most corporate software because the internals are a closely guarded secret, but that's where *The Architecture of Open Source Applications* really shines. This series of books includes dozens of internal accounts from well-known open-source companies and projects.

"Most software engineers do not spend enough time looking at existing software to learn from them. This book tries to fill that gap by making available descriptions of many interesting projects. You can't replace actually studying major software projects, but this comes a pretty close second."

Clean Architecture: A Craftsman's Guide to Software Structure and Design

Where *Clean Code* focuses on functions and classes, *Clean Architecture* zooms out one level to cover composition, service boundaries, and high-level application architecture. Even if you never take on the title of "Software Architect," this book will give you a lot of insight into how you can design better software that scales.

“If you’re looking for a primer for why you should take software/system architecture seriously and what the benefits of doing so would be, then I would highly recommend this book.”

Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems

With data eating the world, the baseline for building applications that can handle “big data” is rising. *Designing Data-Intensive Applications* gives you an overview of how they are built.

“Kleppman has coherently blended the relevant computer science theory with modern use cases and applications…Design concepts don’t go out-of-date soon, so the book has very long shelf-life.”

The Problem with Software: Why Smart Engineers Write Bad Code

Early-career engineers who join an existing codebase often marvel at some of the poor design choices. But having been on both sides of those codebases, senior engineers realize that those decisions were made with different goals and contexts in mind. If you haven’t internalized this, *The Problem with Software* is a good starting point. It makes the case that software is rarely *intended* to be a mess but shows some of the reasons it becomes one anyway.

“There’s a great deal of thought-provoking material here and I recommend the book highly. You could use this as a good course book in an undergraduate or masters-level software engineering course.”

Software Design Decoded: 66 Ways Experts Think

This little book is largely pictures, but it’s one of those resources I like to keep close at hand for whenever I have a few minutes. Thumbing through the pages will be a great reminder of some fundamental practices that expert software developers employ, and you’ll (hopefully) employ as your career progresses.

“This is a great little book. Full of insights abstracted from working software designers. You’ll find yourself nodding at times, making notes on practices to try, and thinking of all the people you want to read this.”

Classics

The books here have stood the test of time - many dating back 20 or more years. While you might think software engineering has changed so much in that time as to make any reading material irrelevant, you’d be surprised how little really has changed. We’re still arguing over how to estimate projects, when to test them, and where to draw abstractions between our classes.

Design Patterns: Elements of Reusable Object-Oriented Software (1994)

The so-called *Gang of Four* book introduces several “design patterns” that are still widely used in software engineering today. The examples are a bit tough to read if you’re used to working in a loosely-typed language like JavaScript, but the fundamental concepts are applicable to any object-oriented language.

“Study it, learn from from it, implement things they way it suggests - then learn that it is not dogmatic. Simply use it to help shape your software solutions into recognisable forms that can be maintained and evolved over time.”

The Mythical Man-Month: Essays on Software Engineering (1975)

One thing that non-technical people don’t realize is that throwing more software engineers at a problem rarely solves it faster. *The Mythical Man-Month* teaches the opposite: cherish the days when you just have a few people working on the project because that’s when it’s likely to move fastest. Even though this book comes from the days of punch cards, it’s highly relevant to engineers and project managers today.

“Impressive how companies repeat the same mistakes identified on this book, written almost half-century ago. The same will hold within half a century from now, as people and companies continue to ignore contents from great books such as this one.”

The Pragmatic Programmer: Your Journey To Mastery (1999)

There’s a fine line between pragmatism and laziness in programming. *The Pragmatic Programmer* shares examples and analogies that helped me figure out how to spot the difference. The authors have updated the book, but the core ideas stay just as true today as they were in 1999.

“If there was only one book I could recommend for other programmers to read it would have to be this one. You and your coworkers will thank you in the future.”

Refactoring: Improving the Design of Existing Code (1999)

I think one of the hardest ideas for new engineers to grasp is refactoring. In theory, we all get that code needs to be revisited over time, but *actively* changing it for the better throughout a project’s life takes discipline and some concrete tactics. This book will give you some patterns and examples that make refactoring more tangible, no matter your experience level.

“For those who don’t know what to do when handed a bunch of legacy code, but don’t know where to start, this book is for you.”

Engineering Leadership

Whether you’re considering a move into management or you’re just taking on a technical lead role, the books in this category should be on any software engineer’s reading list. They might be a little too abstract for junior developers, but after a few years in the industry, you’ll start to see some of these patterns in the real world. These books will help you understand and manage relationships with your team better.

The Manager’s Path: A Guide for Tech Leaders Navigating Growth and Change

If you decide to go into engineering management, your career path, goals, and day-to-day tasks change completely. Most new managers fail to realize this and still try to spend time coding and keeping up with every technical challenge. *The Manager’s Path* gives you a guide to this new career path and is well worth a read if you’re applying for your first management roles now.

“Fournier’s book is a comprehensive overview of all the roles on the career path of modern technical management (starting from “senior engineer mentoring an intern” all the way up to CTO) and how to deal with the challenges at every step of the way.”

The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win

Ever since reading *The Goal* a few years ago, I’ve liked narrative fiction around business topics. *The Phoenix Project* is aimed at IT and DevOps leaders who see projects consistently falling behind but aren’t sure why. In it, you’ll see all the archetypes and ways that you can handle each situation.

“I’m a Linux sysadm in an operations team. The book is pretty much about my daily life, all the struggles and problems. Half way through the book, I started considering leaving my job and open a kebab shop instead.”

Startup Engineering Management

While *The Phoenix Project* lends itself to leaders at large technology organizations, *Startup Engineering Management* specifically deals with growing, funded businesses who are bringing on their first few leaders. Much of the book is very tactical and gives you some good starting points for management practices like one-on-ones, performance evaluations, and setting up your team for success.

Managing Humans: Biting and Humorous Tales of a Software Engineering Manager

Another story-based book, *Managing Humans* will give you some insight into the real problems that engineering leaders face. Even if you don’t go into management, this will provide you a little bit of empathy for the challenges your boss deals with every day.

“No matter where you are in your career, read this book. As an employee, you’ll understand your boss and other teams. As a leader, you’ll understand your role a little better and probably pick a few nuggets up.”

Up-And-Comers

Finally, I wanted to highlight some relatively new books that I think are worth considering. Time will tell if these stick around long enough to fall into the “Classics” category someday, but whether they do or not, add them to your reading list.

Staff Engineer: Leadership Beyond the Management Track

You could go through the standard “management” track, but you don’t have to have the title of “manager” to be a leader on your team. Senior engineers who want to be recognized for their technical expertise and years of gained wisdom will benefit from reading *Staff Engineer*.

“I really like how Will writes: he explores relevant systems and tried to distill them. As a newly minted staff eng, this book really helped jumpstart my modeling of how to be effective.”

The DevelopHer Playbook: 5 Simple Steps to Get Ahead

Diversity is a huge problem in software engineering. I’ve been fortunate enough to work on teams with pretty good gender distributions, so from talking to a lot of women in our field, I know that they face some unique challenges. Following the *Lean In* model, *The DevelopHer Playbook* gives women in tech some insights into how they can advance their careers by being their own best advocates.

“As a woman in technology I go to conferences and hear a lot of motivational speeches. I get really pumped up but most of those don’t give me a way to attain my goals. This book is the answer for that. It gave me small goals which I can implement.”

Breaking the Code: Five Steps to a Life-Changing Software Development Job

Breaking the Code shares the oft-overlooked secret of marketing yourself to build a better career. I’m a huge advocate for this method, but a lot of developers think that technical skills alone will be the thing that sets them apart. Not so, and this book will show you why.

“After reading this book, I understand why I am having such difficulty getting interviews. I have missed all five steps the author outlines as essential to finding your first coding role.”

Have your own recommendations? Send them to me on Twitter.

Read more like this in Software Engineering, Books