**10 Books to Read for Software Engineers**

Being a software engineer does not mean all you need to do is to code. The modern software engineer cannot be a lone genius with a single-minded focus on only coding. We must work well in the team, understand business, have a sharp focus on delivering business values, practice continuous delivery and improvement, design and implement resilient and scalable systems and, last but not least, code well.

To become a wholesome software engineer, we need to continuously learn and broaden our minds. Books are the best resources for us to learn. Here are the 10 books I recommend for you to take your software engineering skills to the next level. I split them into different categories, hoping to give you a little bit of direction.

1. **Programming Practice**

[**Clean Code: A Handbook of Agile Software Craftsmanship**](https://www.amazon.com.au/Clean-Code-Handbook-Software-Craftsmanship-ebook/dp/B001GSTOAM) **by Robert C. Martin**

If you are not certain about which book to pick up first, this is the one. As a software engineer, we spend a lot of time writing code. Understanding the clean code principles makes your code much more bug-free, scalable and maintainable. The examples are based on Java. It has a lot of focus on OOP. However, the principles can be applied to any programming paradigm. It will make you not only a better programmer, but also a better peer reviewer. By understanding these principles, you can be articulate about the reason why a certain block of code can be improved. My recommendation is to read through this book and apply these principles when you write code every day. Clean code can be easier said than done. But, practice makes it perfect.

[**The Clean Coder: A Code of Conduct for Professional Programmer**](https://www.amazon.com.au/Clean-Coder-Conduct-Professional-Programmers-ebook/dp/B0050JLC9Y) **by Robert C. Martin**

Some argues this is more important than Clean Code, which I believe has some truth to it. Software engineering is a team sport. You need to be a good communicator and be a nice person to hang out with. This book will further help you to be a good team member in the context of an agile development. It gives you a guideline to work with tight deadline, estimate your work as accurately as possible, say yes or no in a certain situation, resolve conflicts and be effective agile team member. These skills are in fact more valuable than being a genius programmer in the modern software development.

[**The Pragramatic Programmer: From Journeyman to Master**](https://www.amazon.com.au/Pragmatic-Programmer-Journeyman-Master-ebook/dp/B003GCTQAE) **by Andrew Hunt and David Thomas**

Software engineer’s job is to deliver business values, not arguing over which syntax is better. Our job does not end when we commit code to master because merging code to master itself does not add any business value unless it is delivered to the end user. We, programmers, tend to focus too much on our code and not the business value we are delivering. To deliver business values, we need to be more wholesome and this is the perfect book.

[**Working Effectively With Legacy Code**](https://www.amazon.com/Working-Effectively-Legacy-Michael-Feathers/dp/0131177052) **by Michael Feathers**

Working with legacy code is difficult. To be honest with you, it really sucks. It may use outdated technologies or libraries that are no longer supported. It may be badly written without any unit test. You don’t even know what side-effect it causes when you touch an existing function. Refactoring to make legacy code unit-testable can be extremely challenging, too. The book will equip you with the tools and techniques to work with legacy code effectively. When working with legacy code brings you down a little, it is time to pick up this book. You may even feel excited to work with your legacy code.

1. **Design and Implementation**

[**Domain-Driven Design: Tackling Complexity in the Heart of Software**](https://www.amazon.com.au/Domain-Driven-Design-Tackling-Complexity-Software-ebook/dp/B00794TAUG) **by Eric Evans**

A classic textbook for domain-driven software design. It is not technology specific. It introduces the concept of domain-driven design and contains a plenty of real-world examples that you can apply for your project. If you are interested in designing and implementing an enterprise application, it’s a great book to read. You will probably not get everything after reading it once. It is one of those books that you can refer as you go through your software projects. It is a good reference book to have in your bookshelf.

[**Clean Architecture**](https://www.amazon.com.au/Clean-Architecture-Craftsmans-Software-Structure-ebook/dp/B075LRM681/) **by Robert C. Martin**

Another Martin’s classic software textbook. The clean architecture principles presented in this book can help you to design software that is maintainable and scalable. It covers all the basics like decoupling, separating software by layers, componentisation, data management and programming disciplines. It covers fundamental as well as advanced concepts. You can revisit it many times in a various stages of your software engineering career and it will always teach you something new.

[**Release It!: Design and Deploy Production-Ready Software**](https://www.amazon.com.au/Release-Design-Deploy-Production-Ready-Software-ebook/dp/B079YWMY2V) **by Michael T. Nygard**

The new edition includes the topics around DevOps, microservices and cloud-native application and distributed systems. It also covers project management and continuous deployment. As the subtitle goes, it is all about design and deploy production-ready software in an agile manner. It is a lighter read than Domain-Driven Design and Clean Architecture.

[**Don’t Make Me Think, Revisited: A Common Sense Approach to Web Usability (Voices That Matter)**](https://www.amazon.com/Dont-Make-Think-Revisited-Usability-ebook-dp-B00HJUBRPG/dp/B00HJUBRPG/) **by Steve Krug**

Even if you are a back-end developer, understanding UI usability is important. If you want a book about web usability, this is the one you should pick up. The book is funny and a good read.

1. **Continuous Delivery and DevOps**

[**Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation**](https://www.amazon.com.au/Continuous-Delivery-Deployment-Automation-Addison-Wesley-ebook/dp/B003YMNVC0) **by Jez Humble and David Farley**

The classic book on continuous delivery and deployment automation. DevOps is a set of practice to build, test and deploy software without manual process. This will increase the speed of delivery and feedback loop so that we can iterate quickly to deliver better software. It will equip you with a variety of tools and techniques to deliver your software continuously.

1. **Project Management**

**[The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win](https://www.amazon.com.au/Phoenix-Project-DevOps-Helping-Business-ebook/dp/B078Y98RG8) by Gene Kim, Kevin Beher and George Spafford**

It is based on a fictitious IT project. Some scenarios and characters feel a little over the top, but it is still a easy read with a lot of lessons you can learn from. I have seen many IT projects gone wrong because a few fundamentals are not covered. [The Unicorn Project](https://www.amazon.com.au/Unicorn-Project-Developers-Disruption-Thriving-ebook/dp/B07QT9QR41) is the follow up of this book, which is written from the developer’s perspective. It is also a good read, too. Personally, I still think the Phoenix Project is more interesting to read for an engineer because it is written from the business person’s perspective and will broaden your perspective.

That’s it. There are so many books out there. Choosing what books to read can cause analysis-paralysis sometimes. I hope the list helps you to get started with reading books that help your career.

**We have more…**

The list is compiled, but there are other books that I really enjoyed reading and helped me to be a better software engineer. I am going to list two more books for those who want more.

[**The Art of UNIX Programming**](https://www.amazon.com.au/UNIX-Programming-Addison-Wesley-Professional-Computing-ebook/dp/B003U2T5BA) **by Eric S. Raymond**

It was recommended to me by a senior developer when I was starting out. It influenced me a lot. It is about the philosophy of the UNIX development, which can be applied to any programming and software development projects. The one thing stuck to my mind is that programming should be fun. Linux still attracts a large number of contributors. One of the reasons is to program Linux is fun. The way the OS is architected and coded makes it challenging enough but not too painful for programmers to keep contributing Linux eco-systems. What Linux made successful is a set of practice and philosophy that any programmer can use in their day to day work.

[**How Google Works**](https://www.amazon.com.au/How-Google-Works-Eric-Schmidt-ebook/dp/B00J379F3O) **by Eric Schmidt and Jonathan Rosenburg**

If you are interested in moving into the management position, this is a great book to read. It provides a lot of insider insight on how to build a company that is truly innovative.