My Favorite Software Books

yegor256.com/2015/04/22/favorite-software-books.html

book-review

There are plenty of books about software engineering, but only a few of them rank among my favorites. I read all of those that do over and over again, and I might just update this post in the future when I stumble upon something else that's decent.

Note that I tried to put the most important books at the top of the list.

Object Thinking by David West. This is the best book I've read about object-oriented programming, and it totally changed my understanding of it. I would recommend you read it a few times. But before reading, try to forget everything you've heard about programming in the past. Try to start from scratch. Maybe it will work for you too.

PMP Exam Prep, Eighth Edition: Rita's Course in a Book for Passing the PMP Exam by Rita Mulcahy. This book is my favorite for project management. Even though it's about the PMI approach and PMBOK in particular, it is a must-read for everyone who is interested in management. Ignore the PMBOK specifics and focus on the philosophy of project management and the role of project manager in it.

The Art of Software Testing by Glenford J. Myers et al. You can read my short review of this book here. The book perfectly explains the philosophy of testing and destroys many typical myths and stereotypes. No matter what your job description is, if you're working in the software industry, you should understand testing and its fundamental principles. This is the only book you need in order to get that understanding.

Growing Object-Oriented Software, Guided by Tests by Steve Freeman and Nat Pryce. All you need to know about your unit testing is in this book. I'm fully aware that I didn't include famous software engineer Kent Beck's book in this list because I don't like it at all. You definitely should read it, just to know what's going on, but it won't help you write good tests. Read this one instead, and read it many times.

Working Effectively With Legacy Code by Michael Feathers. This is awesome reading about modern software development, its pitfalls, and typical failures. Most of the code we're working on now is legacy (a.k.a. open source). I read this book as a novel.

Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation by Jez Humble and David Farley. This is a perfect book about software delivery, continuous integration, testing, packaging, versioning, and many other techniques involved in programming. It's definitely a must-read for anyone who is serious about software engineering.

XML in a Nutshell, Third Edition by Elliotte Rusty Harold and W. Scott Means. XML is my favorite standard. And I hated it before I read this book. I didn't understand all the strange prefixes, namespaces, XPath expressions, and schemes. Just this one book changed everything, and ever since reading it, I've used XML everywhere. It is very well written and easy to read. It's a must for everybody.

Java Concurrency in Practice by Brian Goetz et al. This is a very practical book about Java multi-threading, and at the same time, it provides a lot of theoretical knowledge about concurrency in general. I highly recommend you read it at least once.

Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14 by Scott Meyers. No matter what language you're using, this book is very interesting and very useful. It makes many important suggestions about better C++ coding. If you understand most of them, your Java/Ruby/Python/Scala coding skills will improve significantly.

Code Complete: A Practical Handbook of Software Construction, Second Edition by Steve McConnell. Consider this the bible of clean coding. Read it a few times and use it as a reference manual in debates with your colleagues. It mentions the most terrible anti-patterns and worst practices you'll see in modern programming. To be a good programmer, you must know all of them.

Software Estimation: Demystifying the Black Art by Steve McConnell. This one's an interesting read about software engineering and its most tricky part—estimations. At the least, read it to be aware of the problem and possible solutions.

Writing Effective Use Cases by Alistair Cockburn. An old and very good book, you won't actually use anything from this in your real projects, but you will pick up the philosophy of use cases, which will redirect your mind in the right direction. Don't take this book as something practical; these use cases are hardly used anywhere today, but the idea of scoping functionality this way is absolutely right.

Software Requirements, Third Edition by Karl Wiegers (author) and Joy Beatty. A superb book about requirements analysis, the first and most important activity in any software project. Even if you're not an analyst, this book is a must-read.

Version Control With Git: Powerful Tools and Techniques for Collaborative Software Development by Jon Loeliger and Matthew McCullough. This title serves as a practical guide for Git, a version control system. Read it from cover to cover and you will save many hours of your time in the future. Git is a de-facto standard in version control, and every programmer must know its fundamental principles—not from a cheat sheet but from an original source.

JavaScript: The Definitive Guide: Activate Your Web Pages by David Flanagan. JavaScript is a language of the modern Web, and this book explains it very well. No matter what kind of software you develop, you must know JavaScript. Don't read it as a practical guide (even though it's called a guide) but rather as food for thought. JavaScript offers a lot to learn for Java/Ruby/Python developers.

CSS: The Definitive Guide by Eric A. Meyer. CSS is not just about colors and shadows, and it’s not only for graphic designers. CSS is a key language of the modern Web. Every developer must know it, whether you’re working with a back-end, front-end, or desktop application in C++.

Also, check my GoodReads profile.