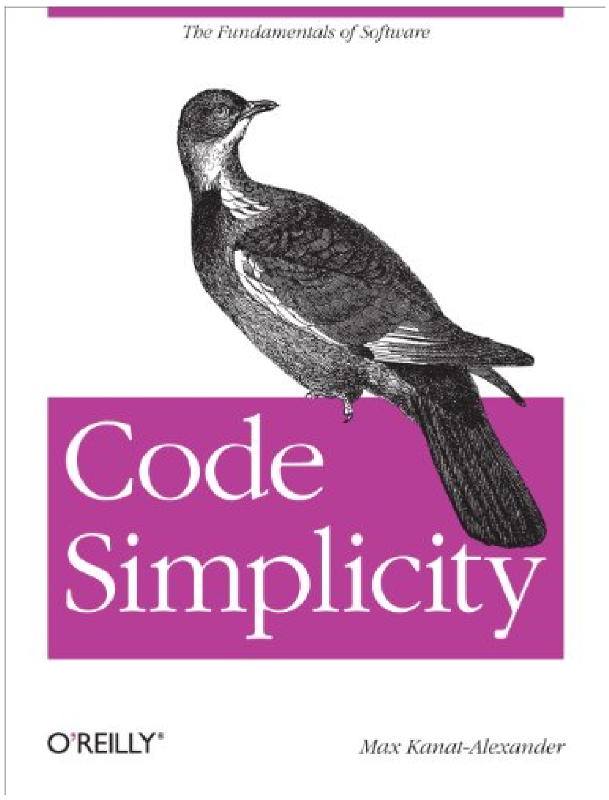
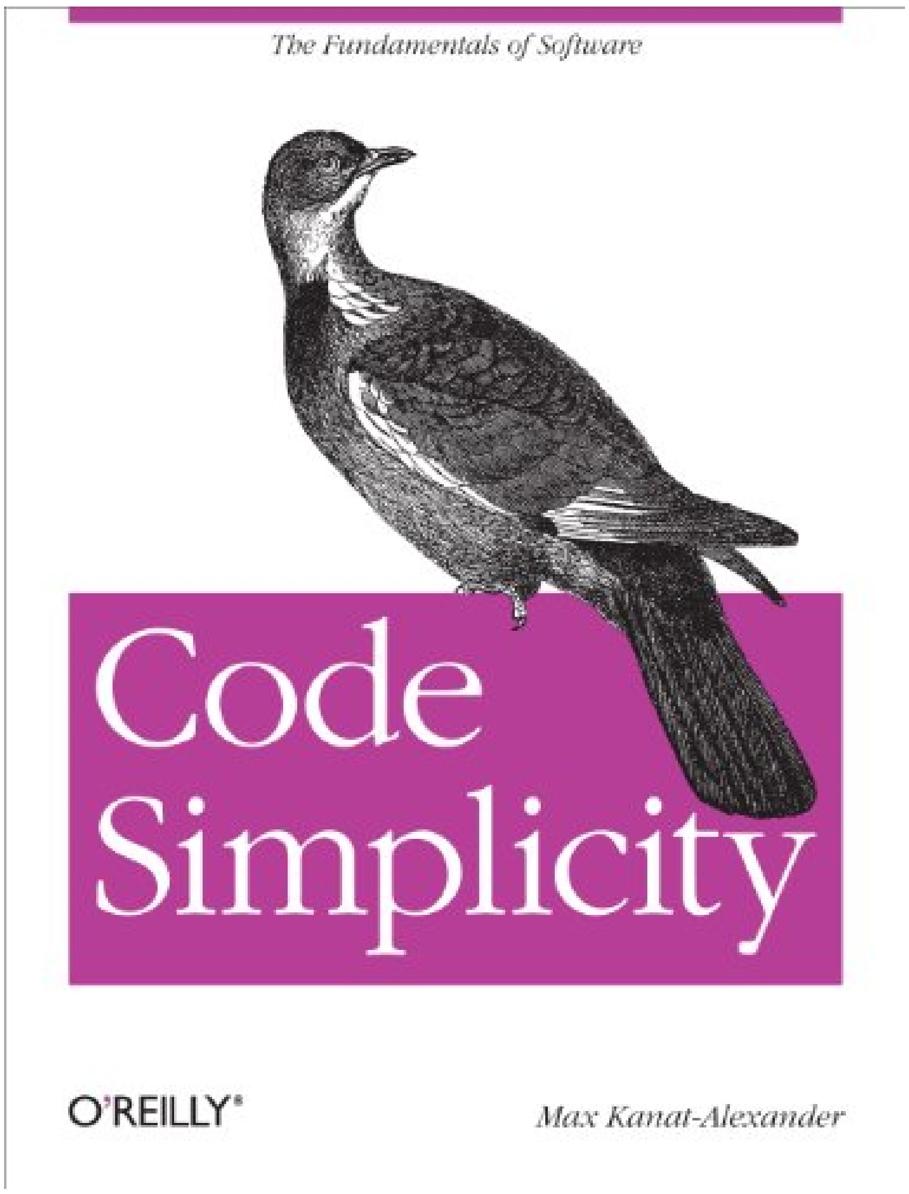
CODE SIMPLICITY: THE FUNDAMENTALS OF SOFTWARE BY MAX KANAT-ALEXANDER




The Fundamentals of Software
Code Simplicity
O'REILLY
Max Kanat-Alexander

CODE SIMPLICITY: THE FUNDAMENTALS OF SOFTWARE BY MAX KANAT-ALEXANDER PDF

Nevertheless, checking out the book **Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander** in this site will lead you not to bring the published publication all over you go. Just store guide in MMC or computer disk as well as they are available to read any time. The prosperous system by reading this soft data of the Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander can be leaded into something brand-new habit. So now, this is time to prove if reading can enhance your life or not. Make Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander it surely work and also get all benefits.

About the Author

Max Kanat-Alexander, Chief Architect of the open-source Bugzilla Project, Google Software Engineer, and writer, has been fixing computers since he was eight years old and writing software since he was fourteen. He is the author of http://www.codesimplicity.com/ and http://www.fedorafaq.org, and is currently living in Northern California.

CODE SIMPLICITY: THE FUNDAMENTALS OF SOFTWARE BY MAX KANAT-ALEXANDER PDF

How if there is a website that allows you to look for referred book **Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander** from all over the globe author? Instantly, the site will certainly be extraordinary finished. Numerous book collections can be discovered. All will certainly be so simple without difficult thing to relocate from website to site to obtain the book Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander really wanted. This is the site that will certainly give you those requirements. By following this website you could obtain lots numbers of book Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander collections from versions sorts of writer and also publisher popular in this globe. Guide such as Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander as well as others can be gotten by clicking wonderful on link download.

Reviewing *Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander* is an extremely valuable passion as well as doing that can be gone through any time. It suggests that checking out a publication will not limit your task, will certainly not compel the moment to invest over, as well as won't invest much money. It is a quite economical and obtainable point to acquire Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander Yet, with that said very economical point, you could get something brand-new, Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander something that you never do and also get in your life.

A brand-new experience could be gained by checking out a book Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander Also that is this Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander or various other publication collections. We offer this book because you could locate much more things to urge your ability and also knowledge that will certainly make you better in your life. It will certainly be additionally valuable for individuals around you. We advise this soft file of the book right here. To recognize the best ways to obtain this book Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander, find out more right here.

CODE SIMPLICITY: THE FUNDAMENTALS OF SOFTWARE BY MAX KANAT-ALEXANDER PDF

Often when we discuss a programming decision, we talk about our feelings or opinions. Wouldn't it be better if instead we had a series of rules and laws for software design, and we could base our discussions and decisions on those?

Well, there are laws of software design, they can be known, and you can know them. Based on extensive research and broad experience, this concise guide boils down software to its true fundamentals--simple principles that any programmer or technical manager can apply to guide the way towards sustainable, well-designed systems.

This isn't a book that will tell you what to do with your software. Rather, it's a guide that will help you understand how to think about design choices and make the right decision for your situation.

- Learn what differentiates great programmers from poor programmers
- Understand the ultimate purpose of software and the goals of good software design
- Determine the value of your decisions now and in the future
- Examine real-world examples that demonstrate how a system changes over time
- Learn how to write software that stands up to unpredictable future requirements
- Make easier changes in the future by keeping your code simpler now
- Understand principles behind test writing and how to choose what to test

- Sales Rank: #538001 in eBooks
- Published on: 2012-03-23
- Released on: 2012-03-23
- Format: Kindle eBook

About the Author

Max Kanat-Alexander, Chief Architect of the open-source Bugzilla Project, Google Software Engineer, and writer, has been fixing computers since he was eight years old and writing software since he was fourteen. He is the author of http://www.codesimplicity.com/ and http://www.fedorafaq.org, and is currently living in Northern California.

Most helpful customer reviews

39 of 42 people found the following review helpful.
Made me want to code.
By Erez Zukerman
This review is part of the O'Reilly Blogger Review Program. I got the book for free, but I didn't have to write a positive review about it. Fortunately, it's a great book.

I've been scripting, hacking, and patching things together for years now. I first started with VBA, then AutoHotkey, then Ruby, with a bit of PHP and JavaScript here and there. But I've never really considered myself a proper coder -- I just hack things together until I get something works. I knew "proper coding" can be beautiful, but I didn't really understand it until I finished reading Code Simplicity.

It's a short read, but it sometimes feels like it was written in blood. The author isn't afraid of making bold assertions, and calling his findings "laws." You don't have to know how to code to read this book: There are no code samples. It's all high-level concepts, from the Equation of Software Design helping you figure out whether or not to implement a change, to The Three Flaws that coders make when they're trying to change their software (mistakes that I have, of course, done in the past), to a fascinating chapter about simplicity, including how simple you really have to be (stupid, dumb simple), and, why that's important, and what _is_ simplicity, anyway.

If you have any interest in programming, are thinking of getting into it, or if you already code and feel like you're missing some of the philosopical underpinnings of the subject, this is really a must-read. Highly recommended.

31 of 33 people found the following review helpful.
a decent weekend read for the journeyman programmer
By R. Friesel Jr.
"Code Simplicity" by Max Kanat-Alexander (published by O'Reilly, 2012) is the kind of book you might give to a junior or journeyman programmer and say: "Read this over the weekend, and then on Monday we'll talk about your design." There are many quotable passages, pithy aphorisms, and axioms that take the form of definitions, facts, rules, and laws. Kanat-Alexander uses a conversational tone that takes this already common-sense study on the subject and makes it even more approachable and straightforward. And just what is the subject here? As the title dictates, the subject of the book is code simplicity: dealing with complexity, identifying areas where complexity is likely to creep in, and strategies for eliminating or reducing that complexity.

At a high level, Kanat-Alexander's discussions of the component parts of this subject are deft and lucid. He is able to evoke familiar situations and scenarios (e.g., coding under a deadline; e.g., dealing with large legacy code bases) and uses those to frame and present his recommended methodologies for keeping code "simple". A lot of the techniques and suggestions will all seem like common sense to anyone who has been programming for a non-trivial interval: reduce maintenance effort before trying to reduce implementation effort; the larger your change, the more likely you are to break something; don't "fix" things unless you know that you have (and have evidence of) a problem -- and so on. Again, to experienced programmers (and, arguably, also to someone who has just sat in on 4+ years worth of computer science lectures?) these suggestions will all seem like conventional wisdom, like the elements of craftsmanship that they perform and preach every day. To those programmers, I say: Give it a whirl for the sake of the reminder, and if nothing else, at least be familiar with it so you'll know whether it is a good tract to hand out to your young and eager recruits.

Again, the book has a conversational style and is filled with pithy aphorisms and witticisms that make it easy to absorb and retain Kanat-Alexander's message. I found myself frequently transcribing passages into my notebook for later reference and for sharing with peers. Many stand out, but two in particular that I wanted to share. First:

«Having good comments in code is a big part of making it readable. However, you generally should not add comments that say what a piece of code is doing. That should be obvious from reading the code. If it isn't obvious, the code should be made simpler. Only if you can't make code simpler should you have a comment explaining what it does.»

I felt this was worth calling out because it was highly illustrative, and emblematic of the theme. Every programmer has had in-depth discussions about comments; and every programmer has committed code that should have been commented, and wasn't; and every programmer has committed code that was commented unnecessarily; etc. This is something that we've all done, and that we'll all do again. Thus, it is easy to identify with the message embedded in this lesson. But what makes it particularly demonstrative of the book's theme is that Kanat-Alexander is trying to teach you first how to do *without* the comments. "If it isn't obvious, the code should be made simpler. Only if you can't make code simpler should you have a comment explaining what it does." And yes, he does immediately go on (in the next paragraph) to state that comments are really more about capturing the ephemeral *why* of a piece of code, but again the message remains: Consider first what you may add in value by taking away what is written and committed.

Kanat-Alexander's take on the comments is interesting one to me because it underscores the book's message about taking the time to engage with the code, to execute on thoughtful designs, and to take the time to understand what you're "doing" -- the problem you're trying to solve, the technologies that you've chosen to use, the sacrifices that you're making by choosing one approach instead of another, etc. He talks quite a bit about understanding, about taking the time to read and fully comprehend code before changing it; about taking the time to engage with the problem space and *design* a solution instead of simply stabbing at one. Not taking the time to arrive at that understanding is (he asserts) a disastrous source of recursive complexity:

«Programmers who don't fully understand their work tend to develop complex systems. It can become a vicious cycle: misunderstanding leads to complexity, which leads to further misunderstanding, and so on.»

Which reminds me of a joke we have going at the office:

«Anger leads to hate. Hate leads to suffering. Suffering leads to re-factoring.»

But this is a point which seems to prove itself: that "*of course* if you don't understand [the problem domain / the programming language / the library / the requirements / the legacy code / the customers / etc.] then *of course* you're just going to create an at-best mistake-riddled solution." And to those "experienced programmers" out there, stroking your beards and condescendingly shaking your heads with your unspoken "I told you so": I've seen you do it, too.

As for "Code Simplicity" itself: I do not actually have all that much in the way of critique. As I have said, it is a short text, pithy and aphorismic -- but this is indicative that it has accomplished its goal. So what would I have liked to see, or see more of? First off, there is effectively no code presented at all; not that Kanat-Alexander promises us any -- just the opposite, he eschews code to drive home the philosophical points. Perhaps specific code examples would detract from that -- but there is a part of me that would liked to have seen something concrete in this area. My other nit to pick was that the book's sub-heading is "The Science of Software Development", and though Kanat-Alexander presents things as "facts" and "laws", I didn't exactly see... science. I read a lot of anecdotes, but I didn't see too many experiments of empirical methods. Given Kanat-Alexander's pedigree, and given the common-sense nature of so much of the advice contained in the book, I'll grant him a pass on this -- but I cannot in good conscience call it all scientific, even if I find myself vigorously agreeing with (say...) 99% of it.

So where do I land on this one? I'd heartily recommend it to any junior or journeyman programmer that is looking for some insights into how to improve their craft and how to cultivate simplicity in their designs. I know I'll likely be recommending it to folks that I mentor in the future -- again: as a quick weekend read, but with the caveat that you'll want to keep a notebook and jot down those really good parts.

54 of 61 people found the following review helpful.
Vague and naive
By Kent R. Spillner
This book was a terrible disappointment. I was excited by the novel prospect that the author managed to create an original science of software design, but in reality this book is just a vague, rambling argument in favor of Agile software development. In fact, every idea in this book has already been presented in far better books by Kent Beck, Martin Fowler, Robert C. Martin, etc.

I applaud the author's ambition in wanting to create a science of software design, but I think he was incredibly naive to think he could do so without more data, evidence, and rigor. The most confusing aspect of the whole book is that he spends several pages in chapter 2 talking about what a science is, and what the necessary characteristics of a science of software design must look like, but then throughout the rest of the book he doesn't make any attempt to adhere to this model. Instead, he always proceeds directly from vague generalizations and observations, or "data" from contrived examples, to his "laws" and "facts" about software design. For example, in chapter 4 he argues about optimizing design decisions to reduce the future cost of maintenance at the expense of greater initial implementation cost, and the only evidence he offers in support of this position is a series of tables showing different hypothetical situations with different costs of effort and value. It's not that his conclusion is necessarily flawed or invalid -- indeed, making decisions to reduce the future cost of maintenance is a very reasonable and pragmatic approach -- but that his argument suffers from lack of evidence, and specificity, and rigorous application of the scientific method.

In the whole book the only external evidence offered in support of one of his conclusions is a table in chapter 5 that shows some statistics about how five different files changed over time (in terms of line count, change count, number of lines added and deleted, etc.). But he doesn't identify any of the files, or the project(s) from which they came, or the time period in which he analyzed when creating that table. He uses this arbitrary collection of information about five random files to build and support his entire case that developers should write code that is easy to change in the future, should not write code that they don't need right now, should not write code that is too abstract, etc. Again, these are all good rules of thumb and useful lessons for every software developer to learn, but it is incredibly naive of him to label this as science given the flimsy evidence used as a basis to support his claims. Laughably, he closes this section by writing "there is a lot more interesting analysis the could be done on these numbers. You're encouraged to dig into this data and see what else you can learn." Yes, there is a lot more interesting analysis that could be done, but until you're more forthcoming with details about where your data and evidence comes from we can't verify or refute any of your claims!

I think this book was published too early in its development, and would be well served by a major rewrite (or three). The author needs to spend a lot more time and effort building a solid foundation for his "science," and should spend less time with the hand-wavy, anecdotal summaries from his own personal experience.

See all 35 customer reviews...

CODE SIMPLICITY: THE FUNDAMENTALS OF SOFTWARE BY MAX KANAT-ALEXANDER PDF

You could find the link that we offer in site to download Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander By buying the inexpensive rate and get completed downloading and install, you have actually finished to the first stage to get this Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander It will certainly be absolutely nothing when having purchased this publication and do nothing. Review it and disclose it! Invest your couple of time to merely review some sheets of web page of this book **Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander** to check out. It is soft documents as well as simple to check out wherever you are. Enjoy your new routine.

About the Author

Max Kanat-Alexander, Chief Architect of the open-source Bugzilla Project, Google Software Engineer, and writer, has been fixing computers since he was eight years old and writing software since he was fourteen. He is the author of http://www.codesimplicity.com/ and http://www.fedorafaq.org, and is currently living in Northern California.

Nevertheless, checking out the book **Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander** in this site will lead you not to bring the published publication all over you go. Just store guide in MMC or computer disk as well as they are available to read any time. The prosperous system by reading this soft data of the Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander can be leaded into something brand-new habit. So now, this is time to prove if reading can enhance your life or not. Make Code Simplicity: The Fundamentals Of Software By Max Kanat-Alexander it surely work and also get all benefits.