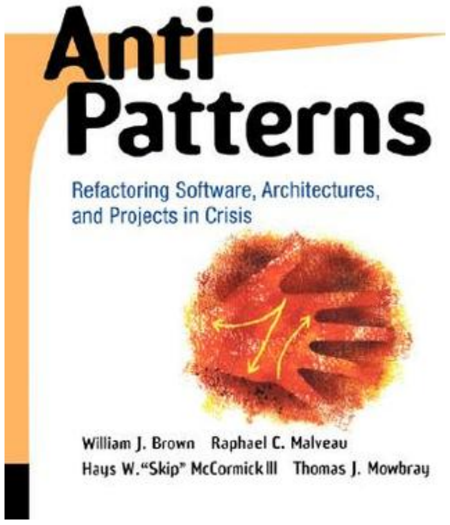
Anti
Patterns
Refactoring Software, Architectures, and Projects in Crisis
William J. Brown Raphael C. Malveau
Hays W. "Skip" McCormick III Thomas J. Mowbray

Antipatterns: Refactoring Software, Architectures, and Projects in Crisis

William J. Brown , Raphael C. Malveau , Hays W. "Skip" McCormick

Antipatterns: Refactoring Software, Architectures, and Projects in Crisis

William J. Brown , Raphael C. Malveau , Hays W. "Skip" McCormick

Antipatterns: Refactoring Software, Architectures, and Projects in Crisis William J. Brown , Raphael C. Malveau , Hays W. "Skip" McCormick
While patterns help you to identify and implement procedures, designs, and codes that work, AntiPatterns do the exact opposite; they let you zero-in on the development detonators, architectural tripwires, and personality booby traps that can spell doom for your project. Written by a team of object-oriented systems developers, AntiPatterns identifies 40 of the most common AntiPatterns in the areas of software development, architecture, and project management. The authors then show you how to detect and defuse AntiPatterns as well as supply refactored solutions for each AntiPattern presented.

Antipatterns: Refactoring Software, Architectures, and Projects in Crisis Details

Date	: Published April 3rd 1998 by Wiley
ISBN	: 9780471197133
Author	: William J. Brown , Raphael C. Malveau , Hays W. "Skip" McCormick
Format	: Paperback 336 pages
Genre	: Computer Science, Programming, Science, Technology, Software, Technical

From Reader Review Antipatterns: Refactoring Software, Architectures, and Projects in Crisis for online ebook

Bob Kozik says

For the experienced or well-read engineer I am not sure how valuable this one is going to be. Quite a few of concepts discussed in this book you'd likely have already experience, heard of, or had explained to you by a peer. What this book does give you is a much deeper, formal explanation of the problems and how to address them. That last bit is especially useful because all of us software engineers, regardless of how good we think we are, do at least one of things discussed in book or are prone to. If you have a more junior member on your team and this is a great book to lend them. You just have to hope that they read it and don't look down it because its age.

Curtis Jensen says

A little dated, but hits on ideas that are still being talked about today.

Charles says

How to avoid a rut in software development

In 1994, a book was published that caused a mini-revolution in the field of software development. The book was _Design Patterns_ by Gamma et. al. Their approach was to describe software in terms of patterns, which are abstractions that are more general than a standard algorithm. Since that time, a small but growing band of individuals have made great progress in the codification and application of patterns. Preliminary indications are that properly understood, and it is problematic that anyone really does at this time, and applied patterns will have a substantial affect on software development.
An antipattern is a pattern that has negative consequences when applied. This ranges from the antipattern that almost always leads to a negative consequence to those that are generally positive, but lead to negative results when used in the wrong context. One example is the Cut-and Paste Programming antipattern. We all have benefited from the use of cut and paste and we have all suffered when we used it in an inappropriate situation. Many such examples are given, and fortunately for us all, for each antipattern the authors provide instructions on how to recognize it, what causes it and how to cure it. Anyone who has worked in software development has experienced one or more of these problems.
In keeping with a negative often being more significant than a positive, it is quite possible that the study of antipatterns will yield more substantial results than similar effort being expended elsewhere. That is why I included this book in my list of best books of the year that appeared in the September, 1999 issue of _Journal of Object-Oriented Programming_.

Erika RS says

1 out of 5 is the score I reserve for books that I consider a waste of my reading time. It saddens me to say so, but this book deserves that score. AntiPatterns has an excellent premise: just as there are good patterns which benefit the development process, there are also bad patterns. These negative patterns can be at many levels including the level of code, the level of architecture, and the social level. As anyone who has been on a real project knows, there is plenty of material for a book with this premise. Sadly, despite the occasional glimmer of interest, this book does not deliver on that potential. Rather, it is dated, boring, and vague.

The book shows its age with frequent references to technologies that, at best, I have but vaguely heard of (CORBA, OMG IDL). Another historical artifact, at least relative to software development at my workplace, is the strict division of architects and developers. Developers, it seems, are naught but the lowly dregs, necessary only because architects cannot dirty their hands with the writing of actual code. In addition to being annoying, this division is unrealistic. In my world, you need both sets of skills, and I believe, from personal and collected anecdote, that those with both perspectives will come to better solutions.

I could forgive the dated references and social structure if the book were otherwise interesting. I quite enjoyed *The Mythical Man-Month* despite its age. However, this book was boring. The authors used a distinctly academic style. As a reader, I don't care about the general development of the field of software patterns. I don't care to read in excessive detail about who else may have named a similar antipattern or why the authors think their version is better. I want substance.

And yet, substance rarely appeared. Comically, although perhaps understandably given how definitions drift over time, the authors called the solutions to their antipatterns the "Refactored Solution". But the solutions were generally vague and unactionable. For example, one antipattern is "The Blob", that class that does everything and is the heart of your application (yup, I'm familiar with that one). The suggested solution: find cohesive components, move them into other places if such places exist, otherwise create such places. Poof! You're done. As if it's that simple.The coupling within real blobs is deep; without a description of how to manage that complexity, the refactored solution does not go beyond common sense. The other antipatterns follow this same pattern: a description of a very real problem is followed by a worthless solution.

I did get something out of this book: as with traditional design patterns, one of the best things about anti-patterns is that they name common problems, making them easier to talk about. That said, the 6 page appendix which summarizes all the patterns provides all of that value. As for the rest of the book, it was a waste of time.

Allisonperkel says

solid discussion on the patterns of software programs run amok. More importantly, it has a few ideas on how to bring the miscreant programs back on track. Its a little dated and a little on the light side at times but overall well worth the read

---

Hunter Johnson says

AntiPatterns, by William J. Brown et al. Cute idea, mirroring *Design Patterns* by Gamma et al., but ultimately forgettable.

Adam says

Still relevant, even if dated, useful even from the perspective of increasing awareness of what can go wrong.

Billy Dean says

Am I missing something? Was there a contest to see how often someone could repeat the word, "AntiPatterns" on a single page, in a single paragraph, in a single sentence?

If you're not already familiar with the concept of antipatterns, it'll be a while in this book before you get a straight answer as to what an antipattern is. If you're already familiar with the concept, then you probably don't need this book to begin with. It's packaged as a revolutionary idea, but a couple brief paragraphs would be enough to explain what it is and how to apply it. Skim this book for what you need and move on.

Adam Lydick says

Some interesting ideas in this book, but only about 60 pages of real content, stretched to fill their quota. Worth reading, but get a used copy and skim the portions where the author is rambling. A much better read (with a similar topic) is "The Pragmatic Programmer".

James Haring says

Just Hilarious. Puts a name on all those bad practices.
