Design patterns criticism sources [closed]

Asked 14 years, 2 months ago Modified 2 years, 8 months ago

Viewed 8k times

19

Closed. This question is seeking recommendations for software libraries, tutorials, tools, books, or other off-site resources. It does not meet Stack Overflow guidelines. It is not currently accepting answers.

We don't allow questions seeking recommendations for software libraries, tutorials, tools, books, or other off-site resources. You can edit the question so it can be answered with facts and citations.

Closed 3 years ago.

Improve this question

I was reading the design patterns page on Wikipedia, particularly the "Criticism" section.

Could you point me to some articles or books about shortcomings of design patterns?

design-patterns

Share

Improve this question

Follow

edited Apr 12, 2018 at 10:04

Matthias Braun

34.1k ● 27 ● 152 ● 176

asked Oct 21, 2010 at 18:54

kofucii

7,643 ● 13 ● 53 ● 80

about any specific pattern (e.g. singletons) or about all Design Patterns? – jball Oct 21, 2010 at 18:59

My two cents: If a "design pattern" has to be thought about in terms of a "name" it means the task/process is too complex and/or does not easily fit into the model or paradigm exposed. (I use many "design patterns" all the time; they are nameless to me except for Academic discussions.)
– user166390 Oct 21, 2010 at 19:00

@jball, maybe more generally will be better. – kofucii Oct 21, 2010 at 19:03

5 @pst isn't that the point of naming design patterns? So that we can think and work at a higher level than the intricacies of the task/process and create better designs for our system as a whole. It's a lot easier to deal with a complex system when you can place a label on a common interaction. Sort of like when you use the word 'engine' to describe the very complex interaction of components that cause combustion.
– Matthew Vines Oct 21, 2010 at 19:07

7 Answers

Sorted by: Highest score (default)

Most of the criticisms of design patterns that I have come across relate to a distaste for the structuring and labeling

34

of what they consider to be just good object-oriented practices. Most patterns boil down to programming to interfaces, and other SOLID principles. The feeling is that when we teach patterns we cause developers, especially junior developers to try to cram all problems into the set of patterns that they have learned, which can create more obtuse and cumbersome problems than if they had taken a more 'straightforward' approach.

I tend to agree with the sentiment that once you begin to learn patterns you tend to overuse them, however, typically you very quickly move out of that stage, and into a much more productive and professional software career afterward.

As a bonus, here is a bit of mild criticism from Jeff Atwood and some critical insights from Mark Dominus

Share Improve this answer

Follow

edited Apr 12, 2018 at 10:02

Matthias Braun

34.1k ● 27 ● 152 ● 176

answered Oct 21, 2010 at 19:14

Matthew Vines

27.5k ● 7 ● 78 ● 99

I think overusing is actually good - through experience (overuse) you learn when to use them and when not.
– paul23 Jun 3, 2015 at 20:20

8

Alan Kay himself is very critical about patterns because he doesn't believe software should be so vaunted. [Here's a 2012 Dr. Dobbs interview with Kay](#).

> "The most disastrous thing about programming — to pick one of the 10 most disastrous things about programming — there's a very popular movement based on pattern languages. When Christopher Alexander first did that in architecture, he was looking at 2,000 years of ways that humans have made themselves comfortable. So there was actually something to it, because he was dealing with a genome that hasn't changed that much. I think he got a few hundred valuable patterns out of it. But the bug in trying to do that in computing is the assumption that we know anything at all about programming. So extracting patterns from today's programming practices ennobles them in a way they don't deserve. It actually gives them more cachet."

Share Improve this answer

Follow

answered Jul 31, 2016 at 15:19

Molly Wright Steenson

81 ● 1 ● 1

One big criticism against design patterns is about how "generic" some design patterns really are. For instance, the [Strategy Pattern](#) *implementation* seems to be more

7

relevant (and complex) in languages that lack lambdas/first-class functions (as we may notice here).

But I think this argument ignores the simple fact that design patterns **do exist**: we *can* perceive patterns in code, any code. Once we notice them, we can start using them as a clear, language-agnostic way of **describing** and **understanding** software architecture. So the fact that some design patterns have easier *implementations* in (or are directly supported by) certain programming languages is something to be aware of, of course; but using that as an argument *against* design patterns is absurd, in my opinion.

Of course, I do also agree that many enterprise design patterns would not fit in a pure functional language. But I also believe the functional world has its own set of design patterns too (like the Monad).

Finally, one interesting discussion on design patterns (even if a *bit* chaotic) can be found on this Hacker News thread. This post from user @thom presents a similar opinion to mine:

> It's important to note the difference between the technical mechanism and the motivation for a design. Yes, lots of languages can pass functions around so perhaps you think the Strategy pattern doesn't apply to you. And yet I see huge numbers of APIs that still take enormous option maps and don't offer configuration with some sort

> of callback, however simple the host language might make it. Patterns capture a *design choice* as much as a *technical implementation*.

Share Improve this answer Follow

edited Mar 29, 2022 at 17:06

answered Oct 21, 2010 at 19:36

rsenna
12k ● 1 ● 56 ● 60

3

Design patterns were hyped a lot about ten years ago; it seems to me that most of the criticism is about overarchitecting applications and applying all the patterns you can think of wherever you can. This heated debate is pretty boring when you take the rant factor out - yes, too much of anything is not good and to inexperienced programmer with a hammer everything looks like a nail.

From time to time, somebody will discover that something he's been doing for the whole time has a name and comment that it doesn't deserve to have a name (missing the point that design patterns are about naming sometimes obvious things so that you can talk about them).

Apart from that, you're basically left with the fact that some languages have a few patterns built-in, and some other don't, and academic debate about how with time some patterns become programming language features.

I haven't seen much valid criticism related to design patterns beside that. They definitely exist, sometimes they are useful, you don't have to know all of them when somebody wakes you up at 3 AM, and that's about it. :)

Share Improve this answer

Follow

answered Oct 21, 2010 at 20:00

Domchi

10.8k ● 7 ● 56 ● 64

3

Design patterns are usually presented as a set of tricks in a specific programming language, usually Java or C++. It is usually not explained when and why a pattern needs to be used, and when it is best not to use it. It is usually not explained what would happen in a totally different programming language.

So one gets an impression that 1) design patterns come from the sky, invented by geniuses, 2) the GoF book needs to be memorized, 3) patterns need to be applied always and ever more in every situation.

In my view, design patterns are highly language specific (LISP has a totally different set of "design patterns" than Java) and problem specific (depending on the project, you use or don't use a pattern even though it is "theoretically" applicable to this place in your code). The GoF book is a useful companion to a Java language manual, but not a set of eternal principles about "programming in general".

Share Improve this answer

Follow

answered Oct 3, 2012 at 7:38

winitzki

3,337 ● 25 ● 34

> Design patterns are language agnostic. Language-specific patterns, also known as implementation patterns, are called **idioms**. Unfortunately these terms are often conflated.
> – jaco0646 Oct 11, 2018 at 19:46

1

Having lived through the times when the GoF book was published (ca 1995), I recall reading that a major impetus (or at least a reason for so much attention), was that OOD was still new to most developers. There was a dearth of understanding of how to make Objects at all.

So this book came out showing some common patterns of making classes and how they worked together.

So a criticism to this: it may not be as relevant now as it was then. Back then, and for a few years afterward there were not as many frameworks (especially in Java) as there would come to be in the years afterward. Now, you can find a lot of examples of good Object Oriented code. Wouldn't it be nice if some authors took snapshots of very good OO codebases aside, and wrote books about why they think they are so wonderful?

To me, the best use of these patterns is to know them when you see them, understand the drawbacks (see 'Visitor Pattern' and note which parts of the system are

more easily expanded), and take those lessons as you write your own code.

Share Improve this answer

Follow

answered Dec 8, 2019 at 19:41

Javaneer

41 • 5

-1

I am sure that design patterns help to structure educational courses on programming techniques but I don't think it helpful that they become a lingua franca in the software industry. They can be counter productive as a way of communicating design because they force the design to conform with established patterns, they allow the design to be described too casually and they force anyone trying to understand the design to read a book on design patterns.

Share Improve this answer

Follow

answered Aug 4, 2014 at 10:22

John Morrison

1 • 1