What does “DAMP not DRY” mean when talking about unit tests?

Asked 13 years, 6 months ago Modified 11 months ago

Viewed 103k times

448

I heard someone say that unit tests (e.g. nUnit, jUnit, xUnit) should be

> DAMP not DRY

(E.g. unit tests should contain "damp code" not "dry code")

What are they talking about?

unit-testing

Share

Improve this question

Follow

edited May 6, 2013 at 12:58

Kristopher Johnson

82.4k • 55 • 251 • 306

asked Jun 23, 2011 at 11:22

Ian Ringrose

51.9k • 57 • 217 • 321

7 There is nothing special about unit tests that warrants non-DRY code. Writing non-DRY tests is an excuse by lazy programmers to attempt to carve out territory for their laziness. Simply put, DRYness and readability are orthogonal concerns. – Asclepius Feb 13, 2017 at 14:36

15 DRYness increases code navigation distance which in turn results in higher mental load to understand. This holds in a "normal" text based environment. A projectional editor could reduce the orthogonality of code but not in every case. – Peter Dec 21, 2019 at 10:16

2 I recommend this article: enterprisecraftsmanship.com/posts/dry-damp-unit-tests – Vladimir Jun 17, 2020 at 12:23

9 Answers

Sorted by: Highest score (default)

815

+100

It's a balance, not a contradiction

DAMP and DRY are not contradictory, rather they balance two different aspects of a code's *maintainability*. Maintainable code (code that is easy to change) is the ultimate goal here.

DAMP (Descriptive And Meaningful Phrases) promotes the *readability* of the code.

To maintain code, you first need to understand the code. To understand it, you have to read it. Consider for a moment how much time you spend *reading* code. It's a

lot. *DAMP increases maintainability by reducing the time necessary to read and understand the code.*

DRY (Don't repeat yourself) promotes the *orthogonality* of the code.

Removing duplication ensures that every concept in the system has a single authoritative representation in the code. A change to a single business concept results in a single change to the code. *DRY increases maintainability by isolating change (risk) to only those parts of the system that must change.*

So, why is duplication more acceptable in tests?

Tests often contain inherent duplication because they are testing the same thing over and over again, only with slightly different input values or setup code. However, unlike production code, this duplication is usually isolated only to the scenarios within a single test fixture/file. Because of this, the duplication is minimal and obvious, which means it poses less risk to the project than other types of duplication.

Furthermore, removing this kind of duplication reduces the readability of the tests. The details that were previously duplicated in each test are now hidden away in some new method or class. To get the full picture of the test, you now have to mentally put all these pieces back together.

Therefore, since test code duplication often carries less risk, and promotes readability, its easy to see how it is considered acceptable.

As a principle, favor DRY in production code, favor DAMP in test code. While both are equally important, with a little wisdom you can tip the balance in your favor.

Share Improve this answer

Follow

edited Jan 7, 2014 at 13:04

Ian Ringrose

51.9k ● 57 ● 217 ● 321

answered Aug 7, 2012 at 1:02

Chris Edwards

8,680 ● 1 ● 16 ● 9

32 This is a great, concise summary. I also like to point out that a DAMP test is more resilient in the face of changing requirements, and the measuring the obviousness of a test is a tremendous benefit when someone else is tasked with rewriting your tests to fit the new requirements. Jesper Lundberg also has a good treatise on this subject. – Jason Apr 29, 2014 at 21:01

3 @Jason, Btw is there a link to *"Jesper Lundberg also has a good treatise on this subject"*? – Pacerier Dec 3, 2015 at 23:54

2 @JohnSaunders, you can avoid some of that duplication by using the test data builder pattern: natpryce.com/articles/000714.html – si618 Jul 8, 2016 at 2:06

5 DRYing out test code has the potential to create an obscure test by introducing a mystery guest – jayeff Apr 1, 2017 at 12:38

2 I'd also add, that well-written tests are essentially the documentation / comments for your application. So being more descriptive helps explain your intent to other developers. And as the OP says, they are self contained in each test so the danger to your application is minimal. Worse case scenario is you have a redundant test or test setup and it takes longer to run the test suite. I'd rather err on the side of good test coverage. – Lee McAlilly Jan 29, 2018 at 19:36

67

> DAMP - Descriptive And Meaningful Phrases.

"DAMP not DRY" values readability over code re-use. The idea of DAMP not DRY in test cases is that tests should be easy to understand, even if that means test cases sometimes have repeated code.

See also Is duplicated code more tolerable in unit tests? for some discussion on the merits of this viewpoint.

It may have been coined by Jay Fields, in relation to Domain Specific Languages.

Share Improve this answer

Follow

edited May 23, 2017 at 10:31

Community Bot

1 ● 1

answered Jun 23, 2011 at 11:26

Dominic Rodger

99.6k ● 36 ● 202 ● 216

1 Good answer and link to related question. There is no perfect DAMP vs DRY choice. We want code that is as dry as possible and in testing that means not so dry that the test becomes difficult to understand. When a test fails I want the reason to be obvious so the developer can get started on fixing the SUT that means I lean towards DAMP code in tests. Like most programming concepts it is always possible to take something too far. If your unit test code is so dry that it takes an extended time to determine how and why the test failed it might be "too dry". – Gerald Davis Apr 28, 2015 at 14:34

29

"DRY" is "Don't repeat yourself"

This is a term which is used to tell people to write code that is reusable, so that you don't end up writing similar code over and over again.

"DAMP" is "Descriptive And Meaningful Phrases".

This term is intended to tell you to write code which can easily be understood by someone who is looking at it. If you are following this principle, you will have long and descriptive variable and function names, etc.

Share Improve this answer

Follow

answered Jun 23, 2011 at 11:39

Spudley

168k ● 39 ● 237 ● 308

20 AIUI, DRY isn't just a matter of saving time through reusability - it also prevents different code paths getting "out of sync". If you copy-paste he same logic across multiple

classes, every instance of that code will need to be updated when a change is required. (And inevitably one of them won't, and will blow up when exercised.) – Andrzej Doyle Jun 24, 2011 at 15:38

28

Damp = 'Descriptive And Meaningful Phrases' - your unit tests should be able to be 'read':

> Readability is more important than avoiding redundant code.

From the article:

> DAMP stands for “descriptive and meaningful phrases” and is the opposite of DRY, not in the sense that it says “everything should look like a trash heap and be impossible to read”, in that readability is more important than avoiding redundant code.
>
> What does this mean and where to use it?
>
> DAMP mostly applies when writing test code. Test code should be very easy to understand to the point that some redundancy is acceptable.

Share Improve this answer Follow

edited Jun 20, 2020 at 9:12

Community Bot

1 • 1

answered Jun 23, 2011 at 11:26

stuartd

73.2k ● 16 ● 138 ● 167

14

There are several answers here already, but I wanted to add another as I didn't think they necessarily explained it as well as they could.

The idea of DRY (Don't repeat yourself) is that in your *application* code you want to avoid redundant or reptetive code. If you've got something that your code needs to do multiple times you should have a function or class for it, rather than repeating similar code in several places.

This is a fairly well known programming concept.

DAMP (Descriptive and Meaninful Phrases) is for your unit tests. The idea here is that your unit test method names should be long and descriptive -- effectively short sentences that describe what you're testing.

eg: `testWhenIAddOneAndOneIShouldGetTwo() { .... }`

When you read a DAMP method name like this, you should understand exactly what the test writer was trying to acheive, without even having to read the test code (although the test code can also follow this concept as well of course with wordy variable names, etc).

This is possible because a unit test method has very specific input and expected output, so the DAMP principle works well for them. Methods in your main application

code are unlikely to be specific enough to warrant names like this, especially if you've written it with the DRY principle in mind.

DAMP and DRY do not contradict each other -- they cover different aspects of how your code is written -- but nonetheless they aren't typically used together because methods written with the DRY principle in mind would be general-purpose and unlikely to be suited to highly specific method name. In general therefore, as explained above, your application code should be DRY and your unit test code DAMP.

I hope that helps explain it a bit better.

answered Dec 8, 2011 at 13:42

SDC

14.2k ● 2 ● 38 ● 49

10

DAMP stands for “descriptive and meaningful phrases” and is the opposite of DRY, not in the sense that it says “everything should look like a trash heap and be impossible to read”, in that readability is more important than avoiding redundant code.

http://codeshelter.wordpress.com/2011/04/07/dry-and-damp-principles-when-developing-and-unit-testing/

Share Improve this answer

Follow

answered Jun 23, 2011 at 11:29

Peter

27.9k ● 8 ● 68 ● 85

5

I agree with Chris Edwards in that you need to strike a balance between the two. Another thing to note is that if, in an attempt to remove duplication, you end up adding a lot of additional structure in your unit test code (i.e. when taking DRY to extremes), you run the risk of introducing bugs in there. In such a situation, you would either have to unit test your unit tests or leave bits of structure untested.

Share Improve this answer

Follow

answered Mar 27, 2014 at 11:34

Philip Atz

938 ● 1 ● 10 ● 26

0

I don't wish to duplicate the effort here, but you can have tests that are DAMP but have the benefit of DRY. On the

flip side, DRY tests won't satisfy DAMP tests in some cases.

I've blogged about DRY vs DAMP which includes some examples.

Neither approach should be your only solution, sometimes DAMP is overkill, other times a very nice addition.

As a general rule you should apply the rule of three. If you spot duplication a third time, it may be worth looking into writing DAMP style tests, but even then not all duplication is bad. Context matters.

Share Improve this answer Follow

answered Apr 13, 2015 at 17:31

Finglas

15.7k ● 11 ● 59 ● 90

downvoting as the links are broken. – Ramkumar Singh Jun 7, 2023 at 19:52

0

A colleague and I defined DAMP as Descriptive And Minimally Pasted. Production code should be DRY, but test code should be DAMP. Expressiveness in test code should trump duplication, but you should still strive for minimal duplication.

Jay Fields mentions Descriptive And Maintainable Procedures in his book Working Effectively with Unit Tests.

Share Improve this answer Follow

edited Jan 12 at 17:29

answered Jan 12 at 17:13

Clint Shank

1 • 1