Software Testing Guide

I grew up in the waterfall era, where testing was seen as a separate activity to programming, done by a different group of people, and carried out after programming was done. The shift towards iterative and agile approaches, particularly the influence of Extreme Programming, has changed the role of testing - raising its importance, and integrating it with the core development work.

At the core of my philosophy to testing is that we want to write self-testing code, meaning we have a suite of automated tests that be easily run against the code base. We should be confident that this suite will catch almost all bugs in the software, so that when this suite “goes green” (passes), we can release that version of the software into production. This kind of environment can both drastically improve the robustness of our software, but also enables us to use refactoring to improve our productivity and responsiveness in the coming months.

A guide to material on martinfowler.com about testing software systems.

Martin Fowler

2 Dec 2019

Self Testing Code

Specification By Example

Self-Testing Code is the name I used in Refactoring to refer to the practice of writing comprehensive automated tests in conjunction with the functional software. When done well this allows you to invoke a single command that executes the tests - and you are confident that these tests will illuminate any bugs hiding in your code.

by Martin Fowler

BLIKI
1 May 2014

Read more…

AGILE CONTINUOUS DELIVERY TESTING EXTREME PROGRAMMING PROGRAMMING STYLE REFACTORING

Test Driven Development

Test-Driven Development (TDD) is a technique for building software that guides software development by writing tests. It was developed by Kent Beck in the late 1990's as part of Extreme Programming. In essence we follow three simple steps repeatedly:

by Martin Fowler

BLIKI
11 Dec 2023

Read more…

TESTING PROGRAMMING STYLE

I was attending a workshop at XP/Agile Universe in 2002 when the phrase 'Specification By Example' struck me as a way to describe one of roles of testing in XP.

by Martin Fowler

BLIKI
18 Mar 2004

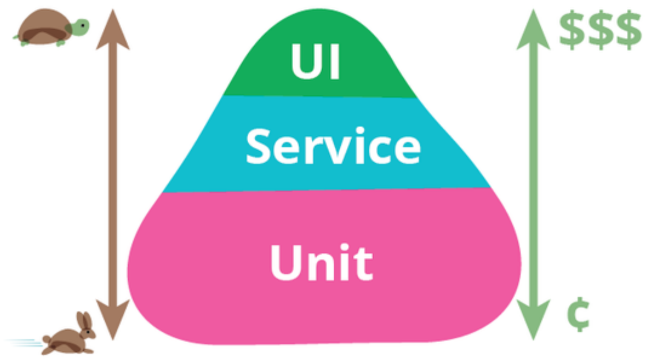
Read more…

TESTING REQUIREMENTS ANALYSIS

What kinds of tests should you have?

There are many forms of tests, depending on what they are trying to verify, the scope of the codebase they cover, and their role in the software process. A popular way to look at your test portfolio is the Test Pyramid.

Test Pyramid



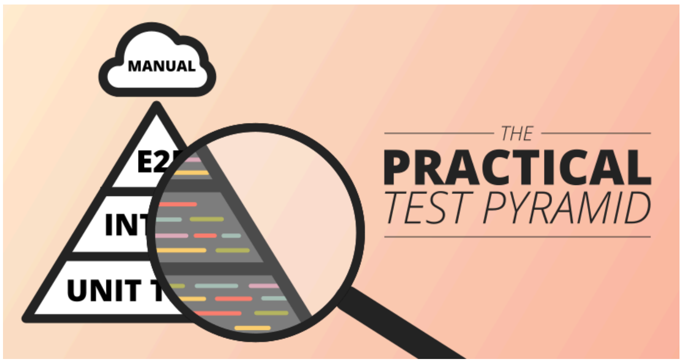
The test pyramid is a way of thinking about how different kinds of automated tests should be used to create a balanced portfolio. Its essential point is that you should have many more low-level UnitTests than high level BroadStackTests running through a GUI.

by Martin Fowler BLIKI 1 May 2012

Read more...

TESTING

The Practical Test Pyramid



The “Test Pyramid” is a metaphor that tells us to group software tests into buckets of different granularity. It also gives an idea of how many tests we should have in each of these groups. Although the concept of the Test Pyramid has been around for a while, teams still struggle to put it into practice properly. This article revisits the original concept of the Test Pyramid and shows how you can put this into practice. It shows which kinds of tests you should be looking for in the different levels of the pyramid and gives practical examples on how these can be implemented.

by Ham Vocke ARTICLE 26 Feb 2018

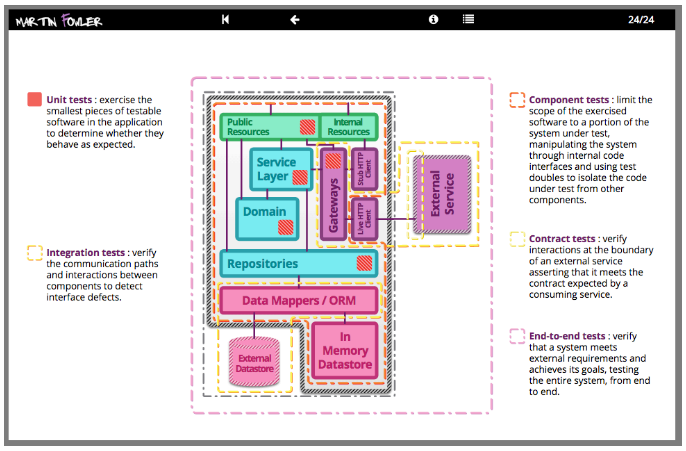
Read more...

TESTING

Testing Strategies in a Microservice Architecture

On the Diverse And Fantastical Shapes of Testing

There are arguments about whether a testing portfolio should be a pyramid or more like honeycomb. My second biggest issue with this argument is that it's rendered opaque by the fact that it's not clear what



🟥 **Unit tests** : exercise the smallest pieces of testable software in the application to determine whether they behave as expected.

⬚ **Integration tests** : verify the communication paths and interactions between components to detect interface defects.

⬚ **Component tests** : limit the scope of the exercised software to a portion of the system under test, manipulating the system through internal code interfaces and using test doubles to isolate the code under test from other components.

⬚ **Contract tests** : verify interactions at the boundary of an external service asserting that it meets the contract expected by a consuming service.

⬚ **End-to-end tests** : verify that a system meets external requirements and achieves its goals, testing the entire system, from end to end.

There has been a shift in service based architectures over the last few years towards smaller, more focussed "micro" services. There are many benefits with this approach such as the ability to independently deploy, scale and maintain each component and parallelize development across multiple teams. However, once these additional network partitions have been introduced, the testing strategies that applied for monolithic in process applications need to be reconsidered. Here, we plan to discuss a number of approaches for managing the additional testing complexity of multiple independently deployable components as well as how to have tests and the application remain correct despite having multiple teams each acting as guardians for different services.

by Toby Clemson

INFODECK
18 Nov 2014

Read more…

POPULAR TESTING INFODECKS MICROSERVICES

people see as the difference between unit and integration tests.

by Martin Fowler

ARTICLE
2 Jun 2021

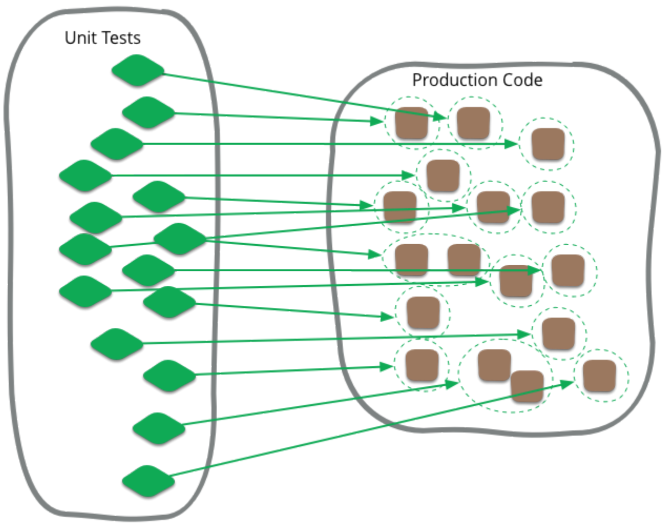
Read more…

TEST CATEGORIES

Classifying Tests

Lots of people classify tests into different categories based on their purpose and scope - but sadly few people use the same classification to mean the same thing. Here's how I classify things, and I try to keep material on this site consistent with these categories.

Unit Test



Unit testing is often talked about in software development, and is a term that I've been familiar with during my whole time writing programs. Like most software development terminology, however, it's very ill-defined, and I see confusion can often occur when people think that it's more tightly defined than it actually is.

by Martin Fowler

BLIKI
5 May 2014

Read more…

TEST CATEGORIES EXTREME PROGRAMMING

Broad Stack Test

A broad-stack test is a test that exercises most of the parts of a large application. It's often referred to as an **end-to-end test** or **full-stack test**. It lies in contrast to a ComponentTest, which only exercises a well-defined part of a system.

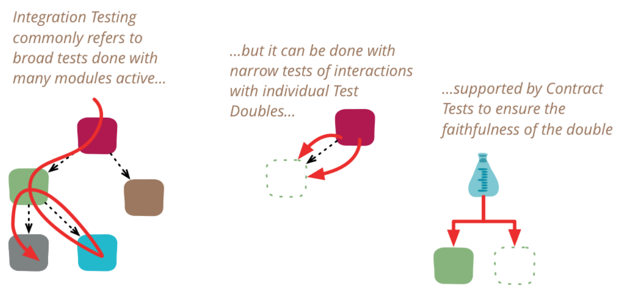
by Martin Fowler

BLIKI
22 Apr 2013

Read more…

TEST CATEGORIES

Integration Test


Integration Testing commonly refers to broad tests done with many modules active...
...but it can be done with narrow tests of interactions with individual Test Doubles...
...supported by Contract Tests to ensure the faithfulness of the double

Integration tests determine if independently developed units of software work correctly when they are connected to each other. The term has become blurred

Story Test

Story tests are BusinessFacingTests used to describe and verify the software delivered as part of a UserStory. When a story is elaborated the team creates several story tests that act as acceptance criteria for the story. The story tests can be combined into a regression suite for the software and provide traceability from the requirements (user stories) to tests and (through execution) to the behavior of the system. Story tests are usually BroadStackTests.

by Martin Fowler

BLIKI
24 Apr 2013

even by the diffuse standards of the software industry, so I've been wary of using it in my writing. In particular, many people assume integration tests are necessarily broad in scope, while they can be more effectively done with a narrower scope.

by Martin Fowler

BLIKI

16 Jan 2018

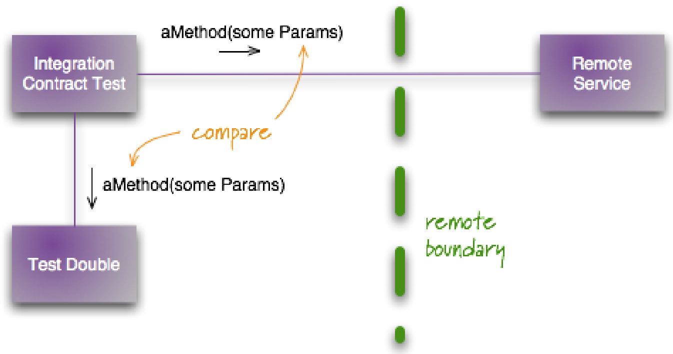
Read more…

TEST CATEGORIES

Read more…

TEST CATEGORIES BAD THINGS

Contract Test



One of the most common cases of using a TestDouble is when you are communicating with an external service. Typically such services are being maintained by a different team, they may be subject to slow, and unreliable networks, and maybe unreliable themselves. That's why a test double is handy, it stops your own tests from being slow and unreliable. But testing against a double always raises the question of whether the double is indeed an accurate representation of the external service, and what happens if the external service changes its contract?

by Martin Fowler

BLIKI

12 Jan 2011

Read more…

TEST CATEGORIES

Component Test

A component test is a test that limits the scope of the exercised software to a portion of the system under test. It is in contrast to a BroadStackTest that's intended to exercise as much of the system as is reasonable.

by Martin Fowler

BLIKI

22 Apr 2013

Read more…

TEST CATEGORIES

Business Facing Test

A business-facing test is a test that's intended to be used as an aid to communicating with the non-programming

Subcutaneous Test

I use *subcutaneous test* to mean a test that operates just under the UI of an application. This is particulary members of a development team such as customers, users, business analysts and the like. When automated, they describe the system in domain-oriented terms, ignoring the component architecture of the system itself. Business-facing tests are often used as acceptance criteria, having such tests pass indicates the system provides the functionality that the customer expects.

by Martin Fowler

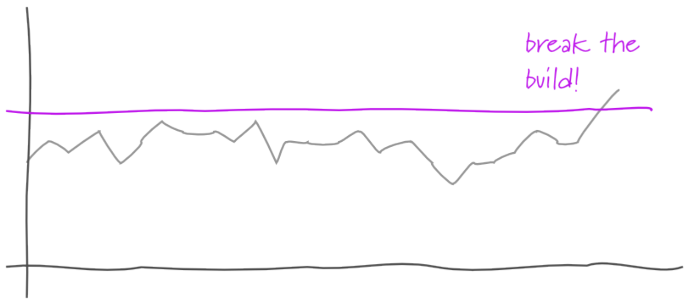
BLIKI

24 Apr 2013

Read more...

TEST CATEGORIES

Threshold Test



A threshold test is a test inserted into a DeploymentPipeline that monitors some measurable phenomenon by comparing the value in the current build against a threshold value. Should the current build's value pass the threshold, the test fails, failing the build.

by Martin Fowler

BLIKI

12 Sep 2013

Read more...

TEST CATEGORIES

valuable when doing functional testing of an application: when you want to test end-to-end behavior, but it's difficult to test through the UI itself.

by Martin Fowler

BLIKI

14 Feb 2011

Read more...

TEST CATEGORIES

Implementing Tests

A large part of a good testing culture is designing the testing infrastructure to make it easy to make tests that are easy to write and efficient.

Test Double

Gerard Meszaros is working on a book to capture patterns for using the various Xunit frameworks. One of the awkward things he's run into is the various names for stubs, mocks, fakes, dummies, and other things that people use to stub out parts of a system for testing. To deal with this he's come up with his own vocabulary which I think is worth spreading further.

by Martin Fowler BLIKI 17 Jan 2006

Read more…

TESTING

Xunit

XUnit is the family name given to bunch of testing frameworks that have become widely known amongst software developers. The name is a derivation of JUnit, the first of these to be widely known.

by Martin Fowler BLIKI 17 Jan 2006

Read more…

TESTING

Mocks Aren't Stubs

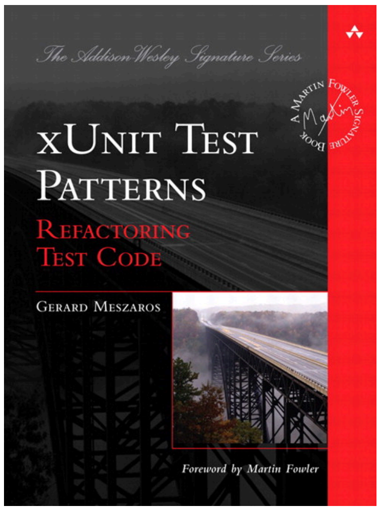
The term 'Mock Objects' has become a popular one to describe special case objects that mimic real objects for testing. Most language environments now have frameworks that make it easy to create mock objects. What's often not realized, however, is that mock objects are but one form of special case test object, one that enables a different style of testing. In this article I'll explain how mock objects work, how they encourage testing based on behavior verification, and how the community around them uses them to develop a different style of testing.

by Martin Fowler ARTICLE 2 Jan 2007

Read more…

POPULAR TESTING

xUnit Test Patterns


The Addison-Wesley Signature Series
A Martin Fowler Signature Book
xUnit Test Patterns
Refactoring Test Code
Gerard Meszaros
Foreword by Martin Fowler

This century has seen a significant rise in programmer testing, where programmers take the responsibility for writing tests of their own code. We've found this helps reduce bugs, makes it easier to think about API design, and enables Refactoring. A large part of making this happen is the xUnit family of simple testing frameworks, originating in the work of Kent Beck and Erich Gamma.

But although xUnit is a simple tool, it takes experience and skill to use it effectively. Gerard Meszaros was an early adopter of these tools and of the idea of using patterns to help people learn about software design techniques. It provides a vocabulary to help us communicate about how we use tests, and a foundation for experienced practitioners to pass their knowledge onto a new generation.

by Gerard Meszaros

BOOK
2007

Read more…

Goto Fail, Heartbleed, and Unit Testing Culture

Two computer security flaws were discovered in early 2014: Apple's "goto fail" bug and OpenSSL's "Heartbleed" bug. Both had the potential for widespread and severe security failures, the full extent of which we may never know. Given their severity, it is important for the software development profession to reflect on how they could have been detected so we can improve our ability to prevent these kinds of defects in the future. This article considers the role unit testing could play, showing how unit tests, and more importantly a unit testing culture, could have identified these particular bugs. It goes on to look at the costs and benefits of such a culture and describes how such a culture was instilled at Google.

by Mike Bland

ARTICLE
3 Jun 2014

Read more…

TESTING

Eradicating Non-Determinism in Tests

An automated regression suite can play a vital role on a software project, valuable both for reducing defects in production and essential for evolutionary design. In talking with development teams I've often heard about the problem of non-deterministic tests - tests that sometimes pass and sometimes fail. Left uncontrolled, non-deterministic tests can completely destroy the value of an automated regression suite. In this article I outline how to deal with non-deterministic tests. Initially quarantine helps to reduce their damage to other tests, but you still have to fix them soon. Therefore I discuss treatments for the common causes for non-determinism: lack of isolation, asynchronous behavior, remote services, time, and resource leaks.

by Martin Fowler

ARTICLE
14 Apr 2011

Read more…

CONTINUOUS DELIVERY TESTING

Test Cancer

As my career has turned into full-time authorship, I often worry about distancing myself from the realities of day-to-day software development. I've seen other well-known figures lose contact with reality, and I fear the same fate. My greatest source of resistance to this is

Test Coverage

Thoughtworks, which acts as a regular dose of reality to keep my feet on the ground.

Thoughtworks also acts as a source of ideas from the field, and I enjoy writing about useful things that my colleagues have discovered and developed. Usually these are helpful ideas, that I hope that some of my readers will be able to use. My topic today isn't such a pleasant topic. It's a problem and one that we don't have an answer for.

by Martin Fowler

BLIKI
6 Dec 2007

Read more…

TESTING BAD THINGS

Test Coverage
Find Untested Code
~~Quality Target~~

From time to time I hear people asking what value of test coverage (also called code coverage) they should aim for, or stating their coverage levels with pride. Such statements miss the point. Test coverage is a useful tool for finding untested parts of a codebase. Test coverage is of little use as a numeric statement of how good your tests are.

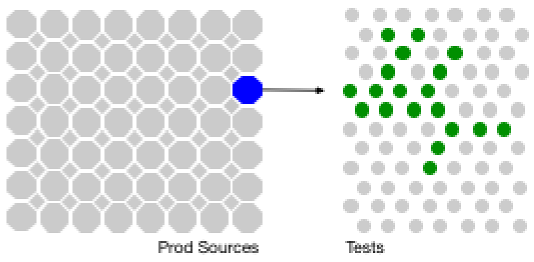
by Martin Fowler

BLIKI
17 Apr 2012

Read more…

TESTING METRICS

The Rise of Test Impact Analysis



Test Impact Analysis (TIA) is a modern way of speeding up the test automation phase of a build. It works by analyzing the call-graph of the source code to work out which tests should be run after a change to production code. Microsoft has done some extensive work on this approach, but it's also possible for development teams to implement something useful quite cheaply.

by Paul Hammant

ARTICLE
22 Aug 2017

Read more…

TESTING

Is TDD Dead?

David Heinemeier Hansson, the creator of Ruby on Rails, gave a keynote at RailsConf where he declared that TDD is Dead. This caused a predictably large amount of controversy in both the Rails and wider software development community. It also led to some interesting conversations between David, Kent, and myself. We decided that these conversations were interesting enough that others might like to watch them too, so recorded a series of video hangouts where we discuss the role of TDD in software development.

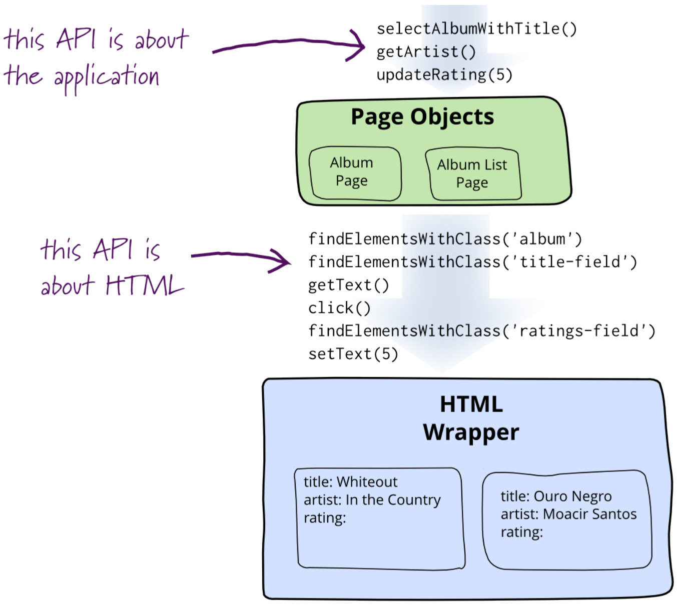
Kent Beck, Martin Fowler, and David Heinemeier Hansson

VIDEO
9 May 2014

More…

TESTING RUBY

Page Object



```
selectAlbumWithTitle()
getArtist()
updateRating(5)
```

```
findElementsWithClass('album')
findElementsWithClass('title-field')
getText()
click()
findElementsWithClass('ratings-field')
setText(5)
```

When you write tests against a web page, you need to refer to elements within that web page in order to click links and determine what's displayed. However, if you write tests that manipulate the HTML elements directly your tests will be brittle to changes in the UI. A page object wraps an HTML page, or fragment, with an application-specific API, allowing you to manipulate page elements without digging around in the HTML.

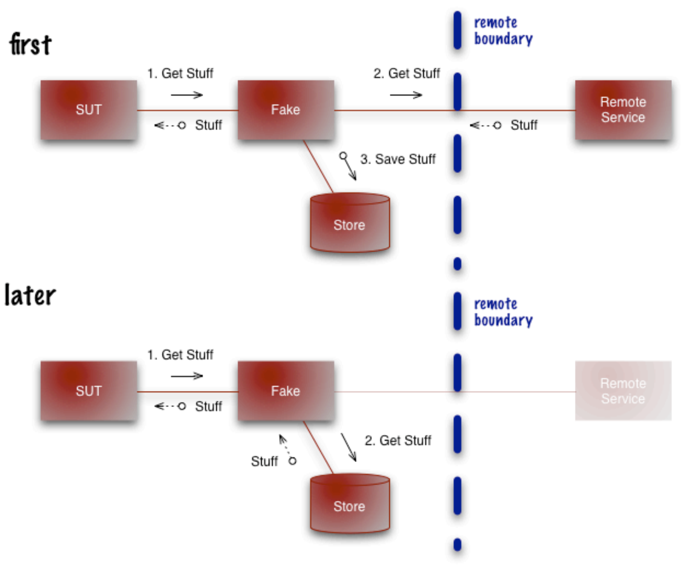
by Martin Fowler

BLIKI
10 Sep 2013

Read more...

TESTING ENCAPSULATION WEB DEVELOPMENT

Self Initializing Fake



One of the classic cases for using a TestDouble is when you call a remote service. Remote services are usually slow and often unreliable, so using a double is a good way to make your tests faster and more stable.

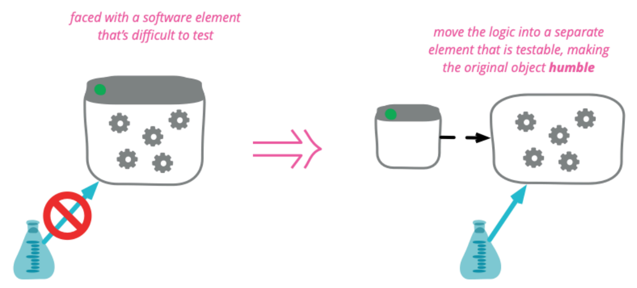
by Martin Fowler

BLIKI
4 Aug 2009

Read more...

TESTING

Humble Object


faced with a software element that's difficult to test
move the logic into a separate element that is testable, making the original object humble

Some program elements are inherently difficult, or even impossible to test. Any logic in these elements is thus prone to bugs and difficult to evolve. To mitigate this problem, move as much as logic as possible out of the

Modern Mocking Tools and Black Magic

The positive effect modern mocking tools can have on our ability to work with legacy code and the possible negative implications of using those tools.

Brett L. Schuchert

ARTICLE
10 Sep 2012

Read more...

TESTING

hard-to-test element and into other more friendly parts of the code base. By making untestable objects humble , we reduce the chances that they harbor evil bugs.

by Martin Fowler BLIKI
29 Apr 2020

Read more…

TESTING

Clock Wrapper

If you need to get the current date or time in your code, don't access the system routines for that data directly. Put some form of wrapper around it that allows you to override it by setting the "current date/time" to a particular value. This is important to simplify testing.

by Martin Fowler BLIKI
26 Mar 2005

Read more…

TESTING

Object Mother

An object mother is a kind of class used in testing to help create example objects that you use for testing.

by Martin Fowler BLIKI
24 Oct 2006

Read more…

TESTING

Testing Resource Pools

I was digging through some old notes, and came across a simple but useful tip that Rich Garzaniti gave me.

by Martin Fowler BLIKI
12 Aug 2004

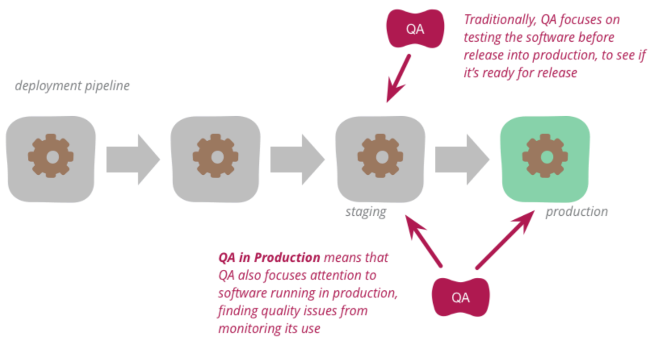
Read more…

TESTING

Automated testing is not enough

Although I advocate automated self-tests as a core part of an effective development effort - such tests cannot do all you need to do in testing or quality assurance. Automated tests provide a fine bug catching net, but you need exploratory testing to figure out if the net really covers all you need it to. While testing used to be something you should complete before code goes into production, we now see monitoring and observability as serious tools to determine the health of our running software.

QA in Production


QA
Traditionally, QA focuses on testing the software before release into production, to see if it's ready for release
deployment pipeline
staging
production
QA in Production means that QA also focuses attention to software running in production, finding quality issues from monitoring its use
QA

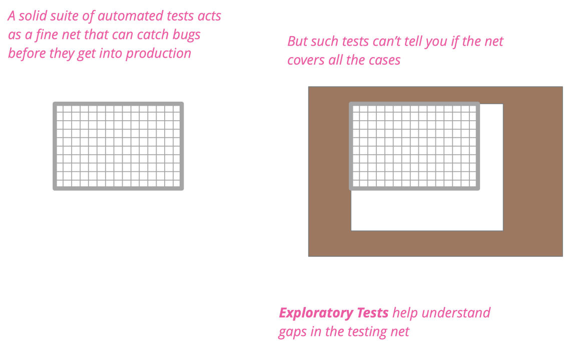
Traditionally, QA focuses on testing the software before release into production to see if it's ready for such release. But increasingly, modern QA organizations are also focusing attention onto the software running in production. By analyzing logs and other monitoring tools, they find quality problems to highlight to the development organization. This approach works particularly well with organizations that use continuous delivery to put new versions of the software into production rapidly and reliably.

by Rouan Wilsenach ARTICLE 4 Apr 2017

Read more...

CONTINUOUS DELIVERY TESTING

Exploratory Testing


A solid suite of automated tests acts as a fine net that can catch bugs before they get into production
But such tests can't tell you if the net covers all the cases
Exploratory Tests help understand gaps in the testing net

Exploratory testing is a style of testing that emphasizes a rapid cycle of learning, test design, and test execution. Rather than trying to verify that the software conforms to a pre-written test script, exploratory testing explores the characteristics of the software, raising discoveries that will then be classified as reasonable behavior or failures.

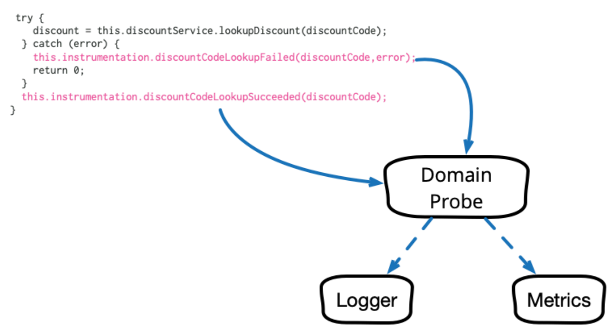
by Martin Fowler BLIKI 18 Nov 2019

Read more...

TESTING

Domain-Oriented Observability

Synthetic Monitoring



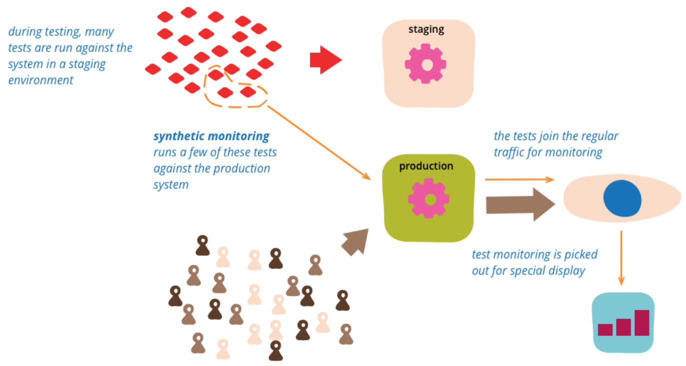
```
 try {
   discount = this.discountService.lookupDiscount(discountCode);
  } catch (error) {
   this.instrumentation.discountCodeLookupFailed(discountCode,error);
   return 0;
  }
  this.instrumentation.discountCodeLookupSucceeded(discountCode);
}
```

Observability in our software systems has always been valuable and has become even more so in this era of cloud and microservices. However, the observability we add to our systems tends to be rather low level and technical in nature, and too often it seems to require littering our codebase with crufty, verbose calls to various logging, instrumentation, and analytics frameworks. This article describes a pattern that cleans up this mess and allows us to add business-relevant observability in a clean, testable way.

by Pete Hodgson ARTICLE
9 Apr 2019

Read more…

CONTINUOUS DELIVERY PROGRAMMING STYLE APPLICATION ARCHITECTURE TESTING


during testing, many tests are run against the system in a staging environment
staging
synthetic monitoring runs a few of these tests against the production system
production
the tests join the regular traffic for monitoring
test monitoring is picked out for special display

Synthetic monitoring (also called semantic monitoring) runs a subset of an application's automated tests against the live production system on a regular basis. The results are pushed into the monitoring service, which triggers alerts in case of failures. This technique combines automated testing with monitoring in order to detect failing business requirements in production.

by Flávia Falé and Serge Gebhardt BLIKI
25 Jan 2017

Read more…

CONTINUOUS DELIVERY TESTING