

# ThoughtWorks®

*A Gentle Introduction*

---

## TDD IN GO

---

*Practice, libraries and tools for  
Test-Driven Development in the Go language*

Luciano Ramalho | [@standupdev](https://twitter.com/standupdev) | [@ramalhoorg](https://twitter.com/ramalhoorg)

# AGENDA

---

Brief introduction

Coding Dojo: Runes (randori style)

Overview of tools, techniques and libraries

A photograph of three people in a collaborative workspace, overlaid with a semi-transparent green filter. On the left, a woman with glasses and a scarf looks down. In the center, a man in a hoodie is writing on a piece of paper. On the right, a man is smiling while looking at a laptop. The background shows a wall with various papers and notes.

ThoughtWorks®

# INTRODUCTION

---

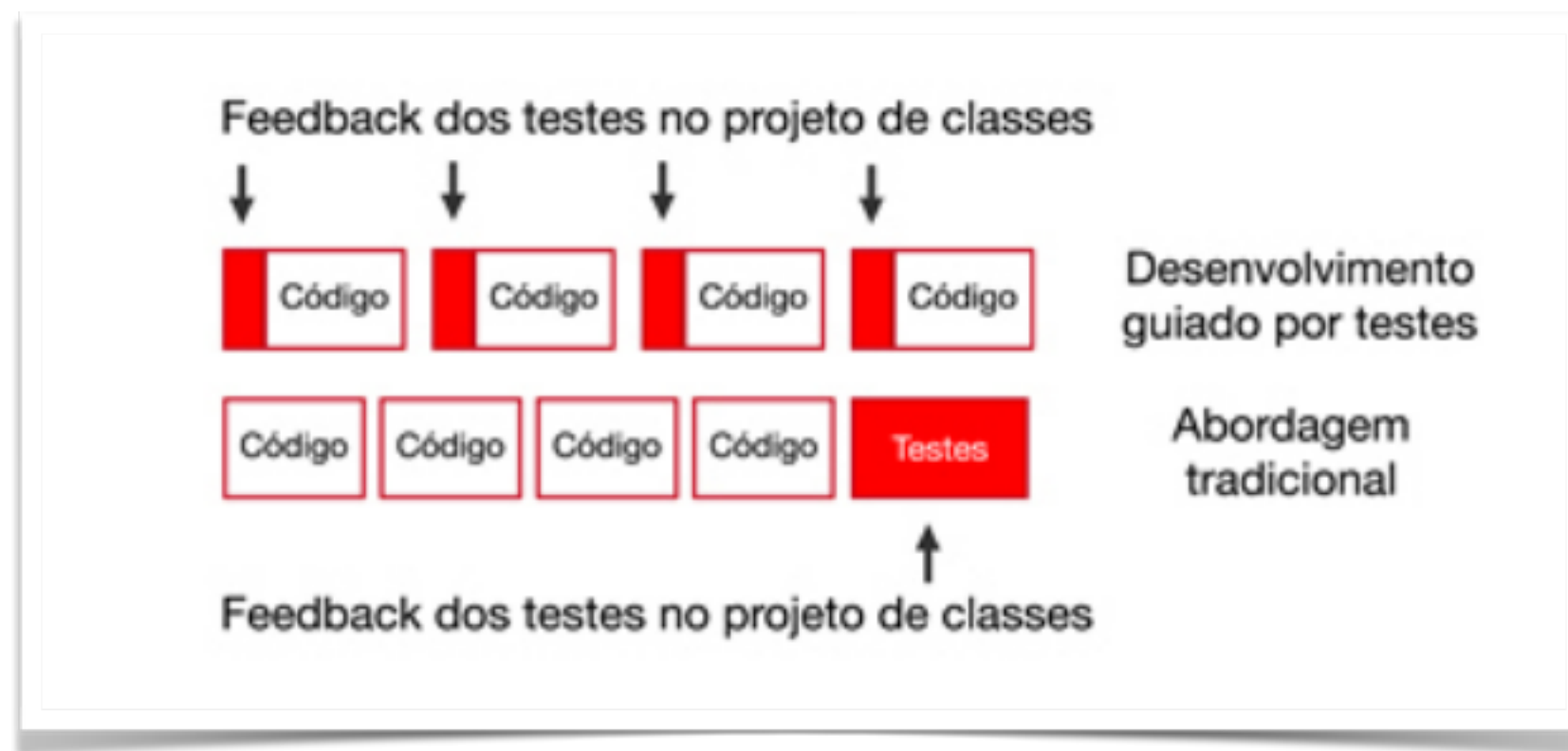
*What TDD is all about*

# ABOUT TDD

---

## Test-Driven Development | Design

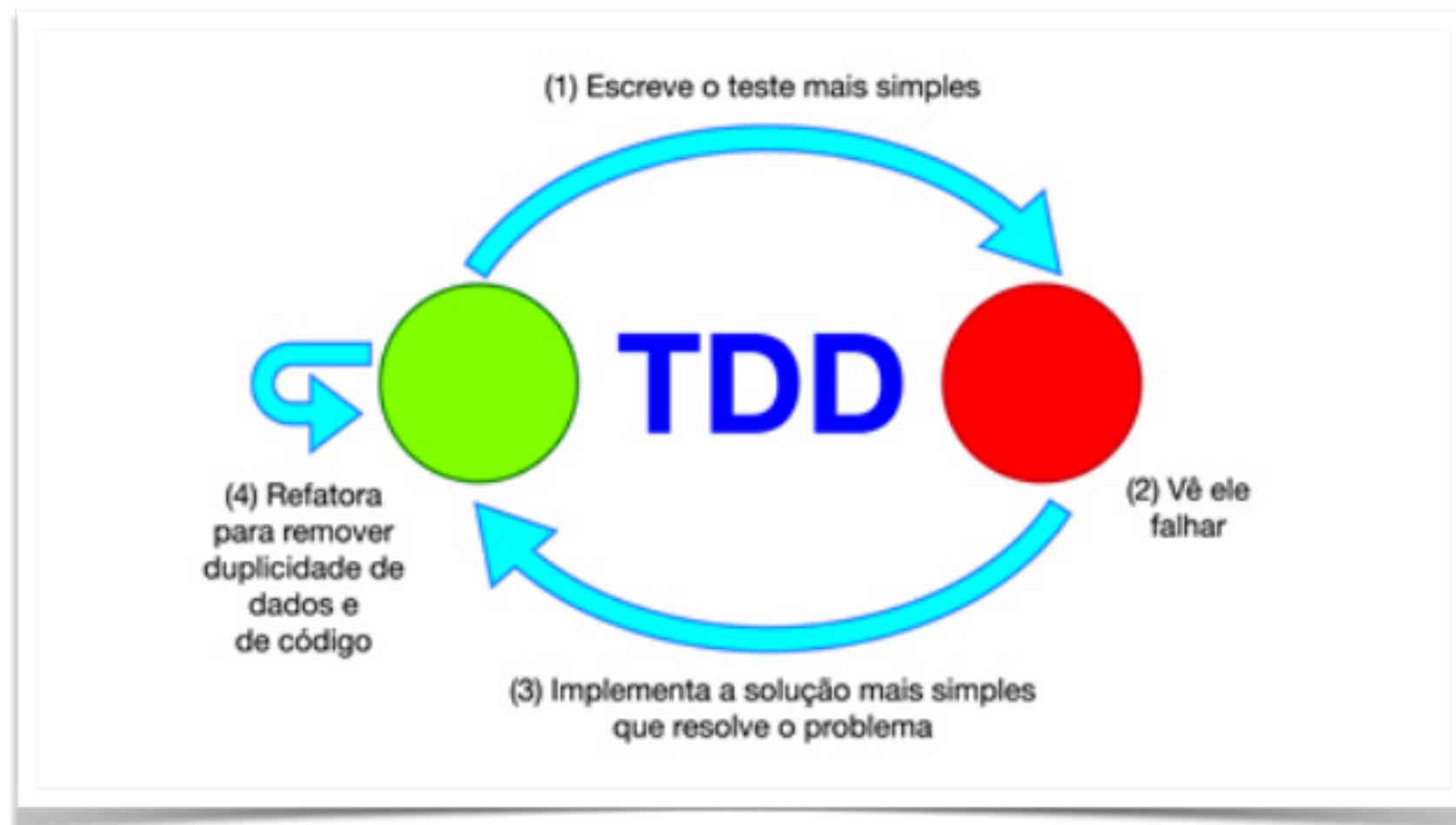
### Test-first approach



Source: **Test-Driven Development**, by Hugo Corbucci and Mauricio Aniche

# TDD CYCLE

---



Source: **Test-Driven Development**, by Hugo Corbucci and Mauricio Aniche

# Work on small increments

Like 4L on 4WD: no need to engage at all times, but good when the going is tough

At first: practice with the smallest increments you can think



# TDD BEST-PRACTICE FOR PAIRING: CALL SHOT

---

## HUGO FALA

Uma técnica divertida e muito útil quando se está usando TDD é a de, antes de rodar o teste, narrar o resultado esperado. Algo como:

- Espero que este teste falhe com uma exceção que diz que o método `maior_valor` não existe.
- Agora espero que o teste falhe dizendo que esperava 250 mas devolveu `nil`.
- Agora o teste vai passar.

Apesar de o exercício parecer fútil, falar em voz alta o resultado que esperamos nos ajuda a tomar consciência do nosso erro quando o resultado não bater. Também torna a prática de programação em pares mais divertida e garante que ambas as pessoas no par estejam acompanhando uma a outra. Nesse caso, a pessoa que não escreveu o teste é a que precisa prever o que vai acontecer.



ThoughtWorks®

# CODING DOJO

---

*Let's practice*



# **CODING DOJO: RULES FOR RANDORI SESSION**

---

Rotating pairs of pilot and co-pilot.

After 7 minutes, call volunteer for co-pilot.




When tests are green, audience can make suggestions for refactoring or next test.

When a test is red, audience should only offer suggestions when requested by pair.

# RUNNING EXAMPLE

---

\$ runes cat eyes

U+1F638		GRINNING CAT FACE WITH SMILING EYES
U+1F63B		SMILING CAT FACE WITH HEART-SHAPED EYES
U+1F63D		KISSING CAT FACE WITH CLOSED EYES

\$ █

A photograph of three people in a collaborative office environment, overlaid with a semi-transparent green filter. On the left, a woman with glasses and a scarf looks down at a document. In the center, a man in a hoodie is leaning over a desk, writing on a piece of paper. On the right, another man is sitting at a desk with a laptop, smiling and looking towards the center. The background shows a wall with various papers and sticky notes.

# ThoughtWorks®

# VARIATIONS

---

*Beyond the classic TDD cycle*

# TDD CYCLE: IMPROVE FAILING REPORTS

---

**Figure 5.2** *Improving the diagnostics as part of the TDD cycle*

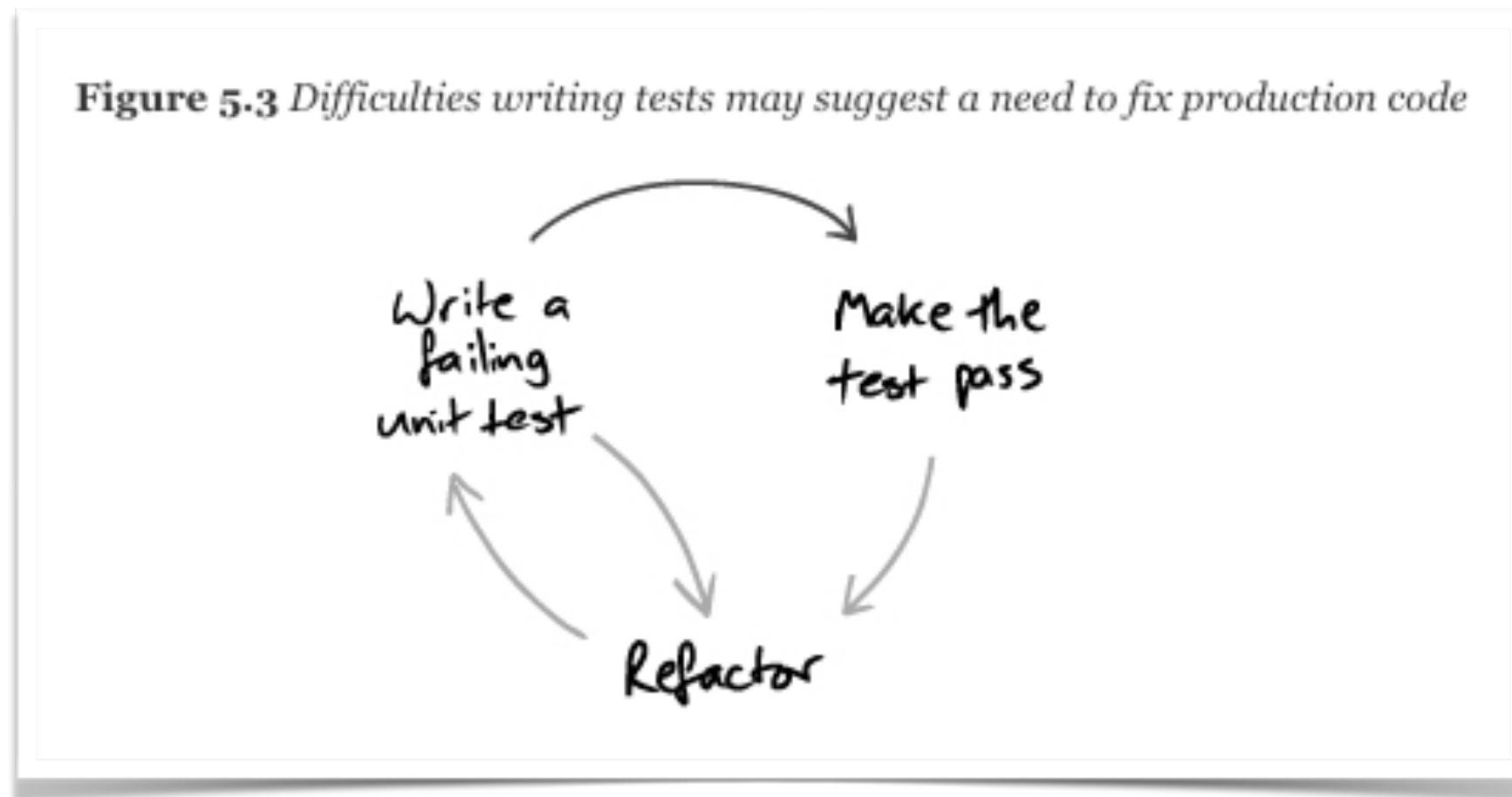


Source: **Growing Object-Oriented Software, Guided by Tests**  
by Steve Freeman, Nat Pryce



# TDD CYCLE: REFACTOR AFTER TEST

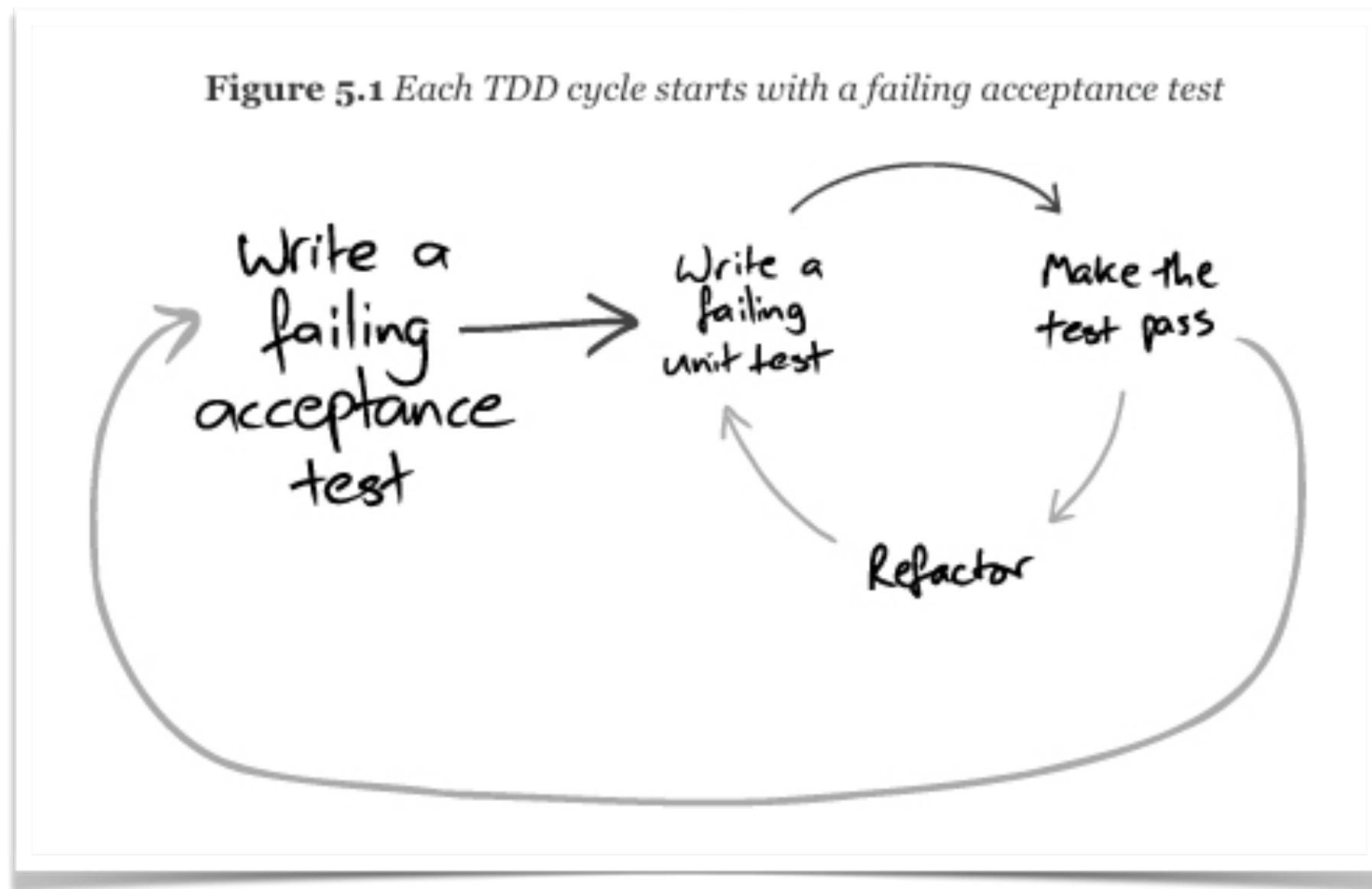
---



Source: **Growing Object-Oriented Software, Guided by Tests**  
by Steve Freeman, Nat Pryce

# TDD CYCLES: MOCKIST STYLE

---



Source: **Growing Object-Oriented Software, Guided by Tests**  
by Steve Freeman, Nat Pryce

# TDD STYLES

---

## Chicago style, a.k.a. “classic”

Mostly inside-out: from unit tests to acceptance tests

## London style, a.k.a. “mockist”

Mostly outside-in: from acceptance tests to unit tests

# MARTIN FOWLER ON TDD STYLES

Mock's Aren't Stubs

https://martinfowler.com/a

Search


MARTINFOWLER.COM

## 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.*

---

02 January 2007



Martin Fowler

Translations: [French](#) · [Italian](#) · [Spanish](#) · [Portuguese](#) · [Korean](#)  
Find **similar articles** to this by looking at these tags: [popular](#) · [testing](#)

### Contents

- Regular Tests
- Tests with Mock Objects
  - Using EasyMock
- The Difference Between Mocks and Stubs
- Classical and Mockist Testing
- Choosing Between the Differences
  - Driving TDD
  - Fixture Setup
  - Test Isolation
  - Coupling Tests to Implementations
  - Design Style
- So should I be a classicist or a mockist?

16





ThoughtWorks®

# TOOLS

---

*Libraries and utilities for TDD in Go*

# LIBRARIES FOR TESTING

---

Go testing libraries with most Github stars\*

5251 ★	stretchr/testify
3685 ★	smartystreets/goconvey
2166 ★	onsi/ginkgo
1452 ★	golang/mock
902 ★	DATA-DOG/go-sqlmock
884 ★	gavv/httpexpect
709 ★	onsi/gomega
575 ★	google/go-cmp
512 ★	franela/goblin
502 ★	h2non/baloo
496 ★	h2non/gock
404 ★	DATA-DOG/godog
387 ★	go-check/check

\*Data collected 2018-07-05





ThoughtWorks®

# TECHNIQUES

---

*Coding tips*

# SIMPLE EXAMPLE-TEST

---

```
func Example() {  
    main()  
    // Output:  
    // Please provide one or more words to search.  
}
```



# ANOTHER SIMPLE EXAMPLE-TEST

---

```
func ExampleMake() {  
    w := []string{"beta", "alpha", "gamma", "beta"}  
    s := Make(w...)  
    fmt.Println(s)  
    // Output: Set{alpha beta gamma}  
}
```

# EXAMPLE-TEST WITH NON-DETERMINISTIC OUTPUT

---

```
func ExampleSet_Channel() {  
    set := MakeFromText("beta alpha delta gamma")  
    result := []string{}  
    for elem := range set.Channel() { // order is undefined  
        result = append(result, elem)  
    }  
    sort.Strings(result) // must sort so example passes  
    fmt.Println(result)  
    // Output: [alpha beta delta gamma]  
}
```

# EXAMPLE-TEST WITH FAKE COMMAND-LINE ARGUMENTS

---

```
func Example_2WordQuery() { // ①
    oldArgs := os.Args // ②
    defer func() { os.Args = oldArgs }()
    os.Args = []string{"", "cat", "smiling"}
    main() // ③
    // Output:
    // U+1F638      🐱      GRINNING CAT FACE WITH SMILING EYES
    // U+1F63A      😸      SMILING CAT FACE WITH OPEN MOUTH
    // U+1F63B      🐱     SMILING CAT FACE WITH HEART-SHAPED EYES
}
```

# TABLE TEST WITH SUB-TESTS

---

```
func TestMake(t *testing.T) {
    testCases := []struct {
        elems    []string
        wantLen int
    }{
        {[]string{}, 0},
        {[]string{"a"}, 1},
        {[]string{"a", "b"}, 2},
        {[]string{"a", "b", "a"}, 2},
    }
    for _, tc := range testCases {
        t.Run(fmt.Sprintf("%v gets %d", tc.elems, tc.wantLen), func(t *testing.T) {
            s := Make(tc.elems...)
            assert.Equal(t, tc.wantLen, s.Len())
        })
    }
}
```



# TEST WITH FAKE ENVIRONMENT VARIABLE

---

```
func restore(nameVar, value string, existed bool) {
    if existed {
        os.Setenv(nameVar, value)
    } else {
        os.Unsetenv(nameVar)
    }
}

func TestGetUCDPATH_isSet(t *testing.T) {
    pathBefore, existed := os.LookupEnv("UCD_PATH")
    defer restore("UCD_PATH", pathBefore, existed)
    ucdPath := fmt.Sprintf("./TEST%d-UnicodeData.txt", time.Now().UnixNano())
    os.Setenv("UCD_PATH", ucdPath)
    got := getUCDPATH()
    if got != ucdPath {
        t.Errorf("getUCDPATH() [set]\nwant: %q; got: %q", ucdPath, got)
    }
}
```

# TEST WITH HTTP SERVER DOUBLE

---

```
func TestFetchUCD(t *testing.T) {
    srv := httptest.NewServer(http.HandlerFunc(
        func(w http.ResponseWriter, r *http.Request) {
            w.Write([]byte(lines3Dto43))
        })
    defer srv.Close()

    ucdPath := fmt.Sprintf("./TEST%d-UnicodeData.txt", time.Now().UnixNano())
    done := make(chan bool) // ❶
    go fetchUCD(srv.URL, ucdPath, done) // ❷
    _ = <-done // ❸
    ucd, err := os.Open(ucdPath)
    if os.IsNotExist(err) {
        t.Errorf("fetchUCD did not save:%v\n%v", ucdPath, err)
    }
    ucd.Close()
    os.Remove(ucdPath)
}
```

# SLOW TEST THAT CAN BE SKIPPED

---

```
func TestOpenUCD_remote(t *testing.T) {  
    if testing.Short() { // ❶  
        t.Skip("skipped test [-test.short option]") // ❷  
    }  
    ucdPath := fmt.Sprintf("./TEST%d-UnicodeData.txt", time.Now().UnixNano())  
    ucd, err := openUCD(ucdPath)  
    if err != nil {  
        t.Errorf("openUCD(%q):\n%v", ucdPath, err)  
    }  
    ucd.Close()  
    os.Remove(ucdPath)  
}
```

---

A photograph of three people in a meeting, overlaid with a green tint. A woman on the left is pointing at a document. A man in the center is leaning over a table, writing on a piece of paper. A man on the right is sitting at a desk with a laptop, smiling. The background shows a wall with many sticky notes.

ThoughtWorks®

# REFERENCES

---

*Where to learn more*



# REFERENCES

---



# MORE REFERENCES

---

## Books

Kent Beck: **Test Driven Development: By Example** <https://tgo.li/2NvBfcX>

Steve Freeman, Nat Pryce:

**Growing Object-Oriented Software, Guided by Tests** <https://tgo.li/2tV8QoK>

## Posts | Videos

Martin Fowler: **Mocks Aren't Stubs** <https://tgo.li/2IUqTXv>

Martin Fowler: **Is TDD Dead?** <https://tgo.li/2IWOAYn>

Michael Feathers , Steve Freeman:

**Test Driven Development: Ten Years Later** <https://tgo.li/2KD2Gnm>

Stanislav Pankevich:

**Notes on "TDD by Example" by Kent Beck** <https://tgo.li/2ufKWDN>

# THANK YOU

*Let's connect!*

*Luciano Ramalho  
@standupdev | @ramalhoorg  
luciano.ramalho@thoughtworks.com*

**ThoughtWorks®**