

Mitos sobre unit testing

Rigoberto Vides
rigovides@gmail.com

Experimentación

- Puntos de vista
- Evidencia
- Aneccdata



Agenda


- **Glosario**
- **Top 5 Mitos de unit testing**
- **Cierre**
- **QA**

Unit Testing (Pruebas de programador): Un pedazo de código que prueba otro pedazo de código.

```
func sum(frist: Int, second: Int) -> Int {  
    return frist + second  
}  
  
func testSum() {  
    let a = 2  
    let b = 3  
  
    let c = sum(frist: a, second: b)  
  
    assert(c == 5)  
}
```


TDD (Test Driven Development): Técnica de programación donde se escriben pruebas unitarias **antes** de escribir **código de producción**.

```
func testDrivenSum() {  
    let a = 3  
    let b = 2  
  
    let c = sum(first: a, second: b)  
}
```



```
func sum(first: Int, second: Int) -> Int {  
    return 0  
}
```

```
func testDrivenSum() {  
    let a = 3  
    let b = 2  
  
    let c = sum(first: a, second: b)  
  
    assert(c == 5)  
}
```



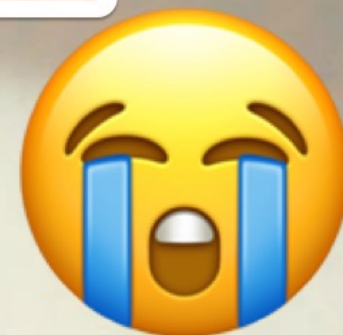
```
func sum(first: Int, second: Int) -> Int {  
    return first + second  
}
```

Code Coverage: Cuántas líneas de código de producción recorren las pruebas.

```
func isEven(number: Int) -> Bool {  
    if number % 2 == 0 {  
        return true  
    }  
    return false  
}  
  
func testIsEven() {  
    assert(isEven(number: 3) == false)  
}
```



TOP 5 MITOS DE UNIT TESTING!!!!!! !!!!!!1



#5

**El UI no debe/
puede probarse**

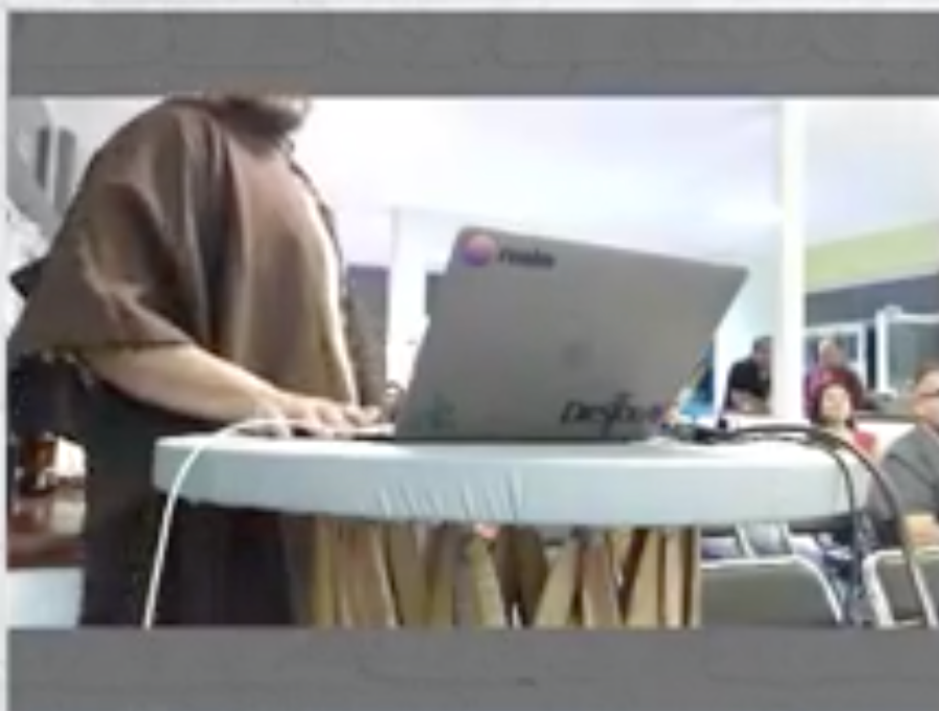
```
func makeContinueButton() -> UIButton {  
    let frame = CGRect(x: 20, y: 20, width: 30, height: 45)  
    let button = UIButton(frame: frame)  
    button.setTitle("Continue", for: .normal)  
    button.backgroundColor = .blue  
  
    return button  
}
```

```
func testMakeContinueButton() {  
    let button = makeContinueButton()  
  
    assert(button.frame.size.height == 45)  
    assert(button.backgroundColor == .blue)  
}
```

```
testMakeContinueButton()
```



- **No tiene valor**
- **Setup complicado**
- **El UI cambia mucho**
- **E2E**
- **Pyramid of Testing** 😂



#TDDMeets

Chrome File Edit View History Bookmarks People Windows Help Oct 31 7:28 PM

Training Programs - Google Slides CD 808 - Google Slides TDD - Google Slides

Secure | https://docs.google.com/presentation/d/1TTP12xT0VjwRv4z4Z48_VjgpaAUE26ZFS-huJed8H0de-40-g248676d724_0_36

App Tempa Pause CC

About those drawings you see when you google "TDD"

Unit Integration E2E

Red Green Refactor

#4

**Una prueba unitaria por
función (assert por
prueba)**

```
func makeContinueButton() -> UIButton {
    let frame = CGRect(x: 20, y: 20, width: 30, height: 45)
    let button = UIButton(frame: frame)
    button.setTitle(localizedTitle(), for: .normal)
    button.backgroundColor = .blue

    return button
}

private func localizedTitle() -> String {
    return NSLocalizedString("continue-button", comment: "")
}

func testMakeContinueButtonStyles() {
    let button = makeContinueButton()

    assert(button.frame.size.height == 45)
    assert(button.backgroundColor == .blue)
}

func testMakeContinueButtonLocalization() {
    let button = makeContinueButton()

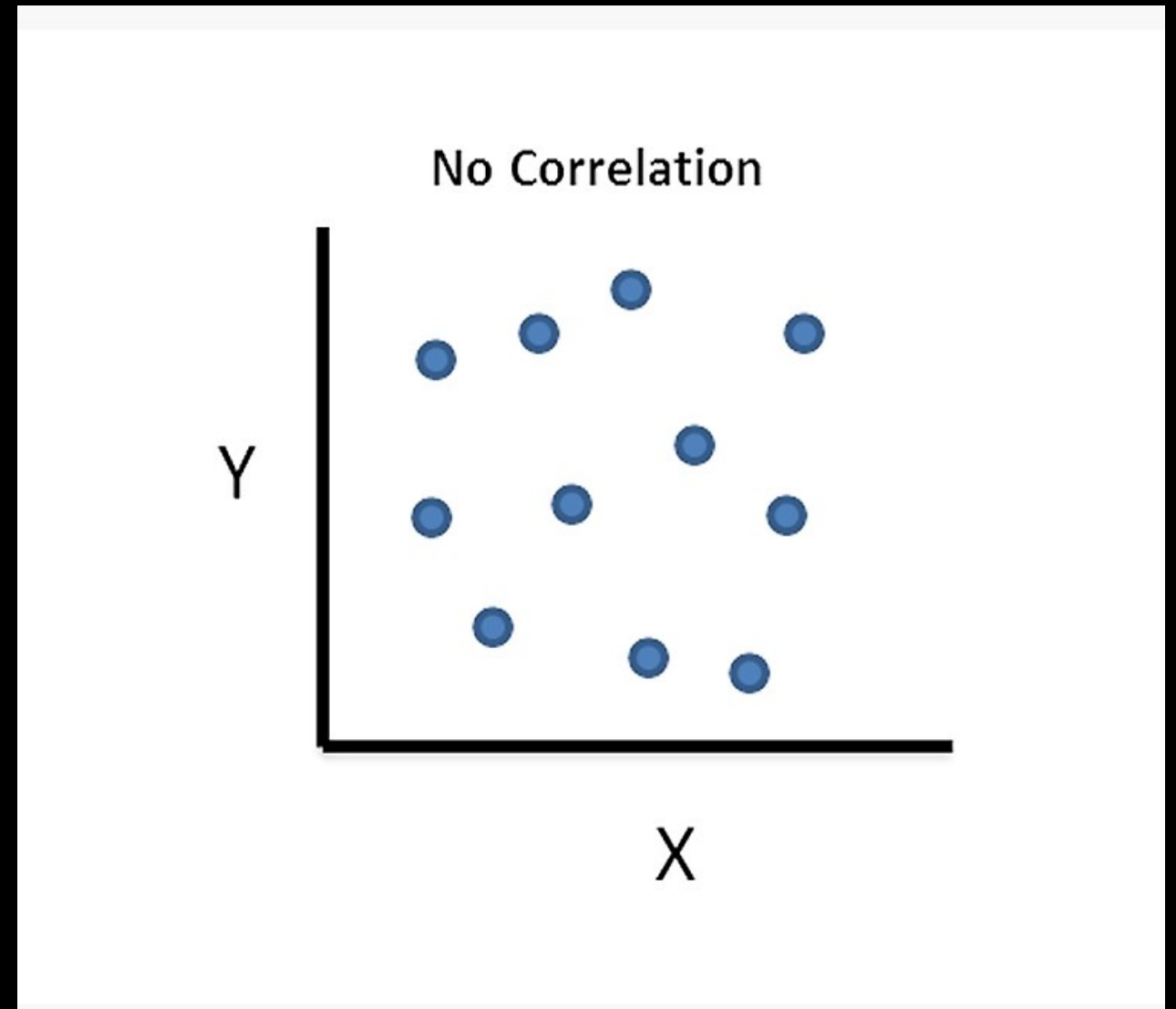
    assert(button.title(for: .normal) == "Siguiente")
}
```

- **Probar métodos privados con interfaz pública**
- **Probar (assert) todos los efectos secundarios**
- **Testing Patterns (AAA, Given/Then/When)**

#3

**El TDD es un
zero sum game**

- **Herramienta de diseño, no de testing**
- **TDD es el infierno en ciertos ambientes**
- **Soluciones obvias**
- **Soluciones complejas**



#2

**Las pruebas sólo agregan
valor al autor, y al
momento de ser escritas**

- **Documentación**
- **Code reviewing**
- **Integración Continua**
- **Seguridad ante el cambio**

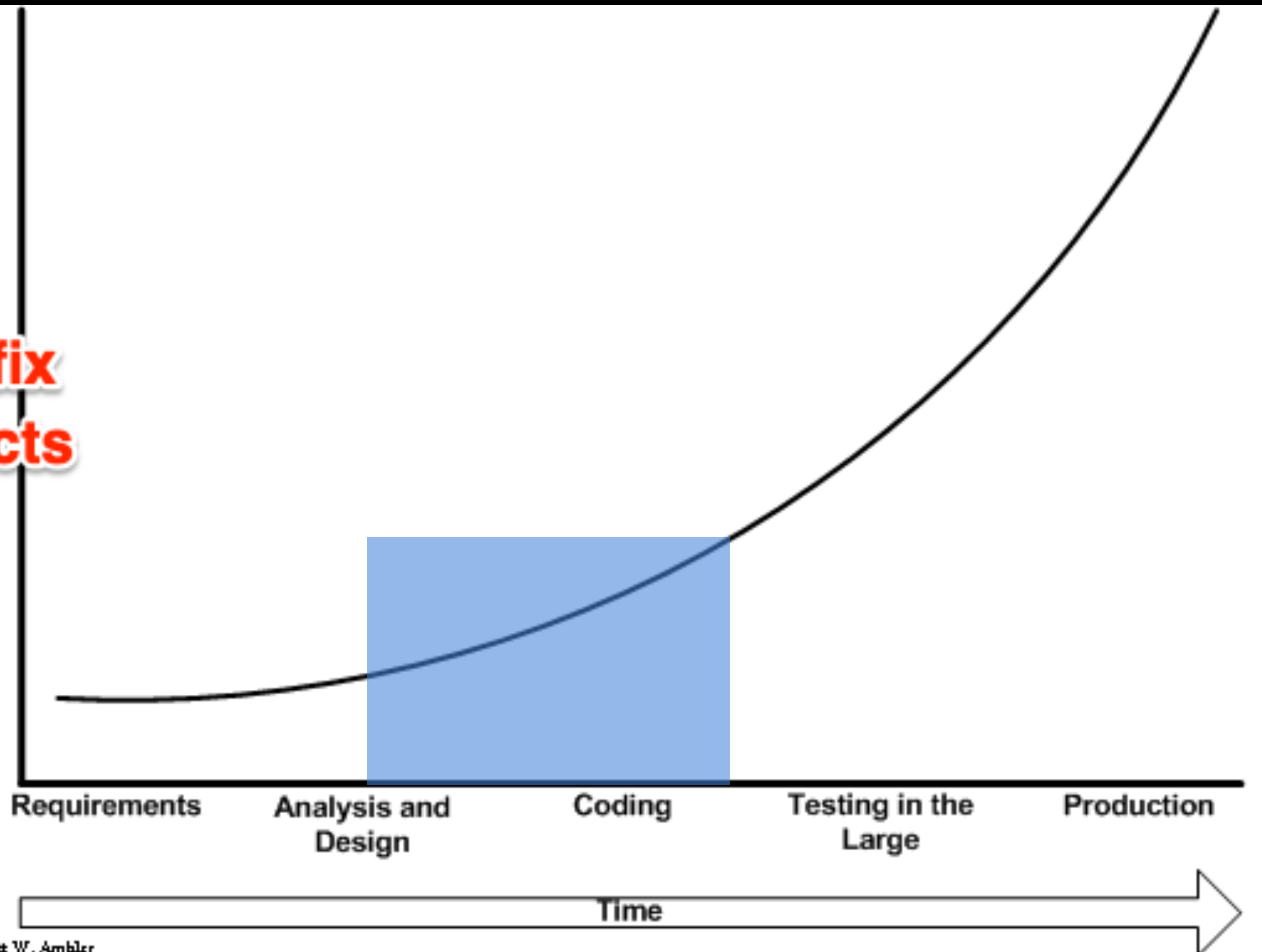
#1

LaS PrUeBAs Te haCeN LEntO



Cost
of Change

**and fix
defects**



- **Mismo número de defectos, menor esfuerzo**
- **Escribir pruebas es difícil?**
- **Mantener código testeado es difícil?**
- **Tiempo depurando vs tiempo escribiendo código**

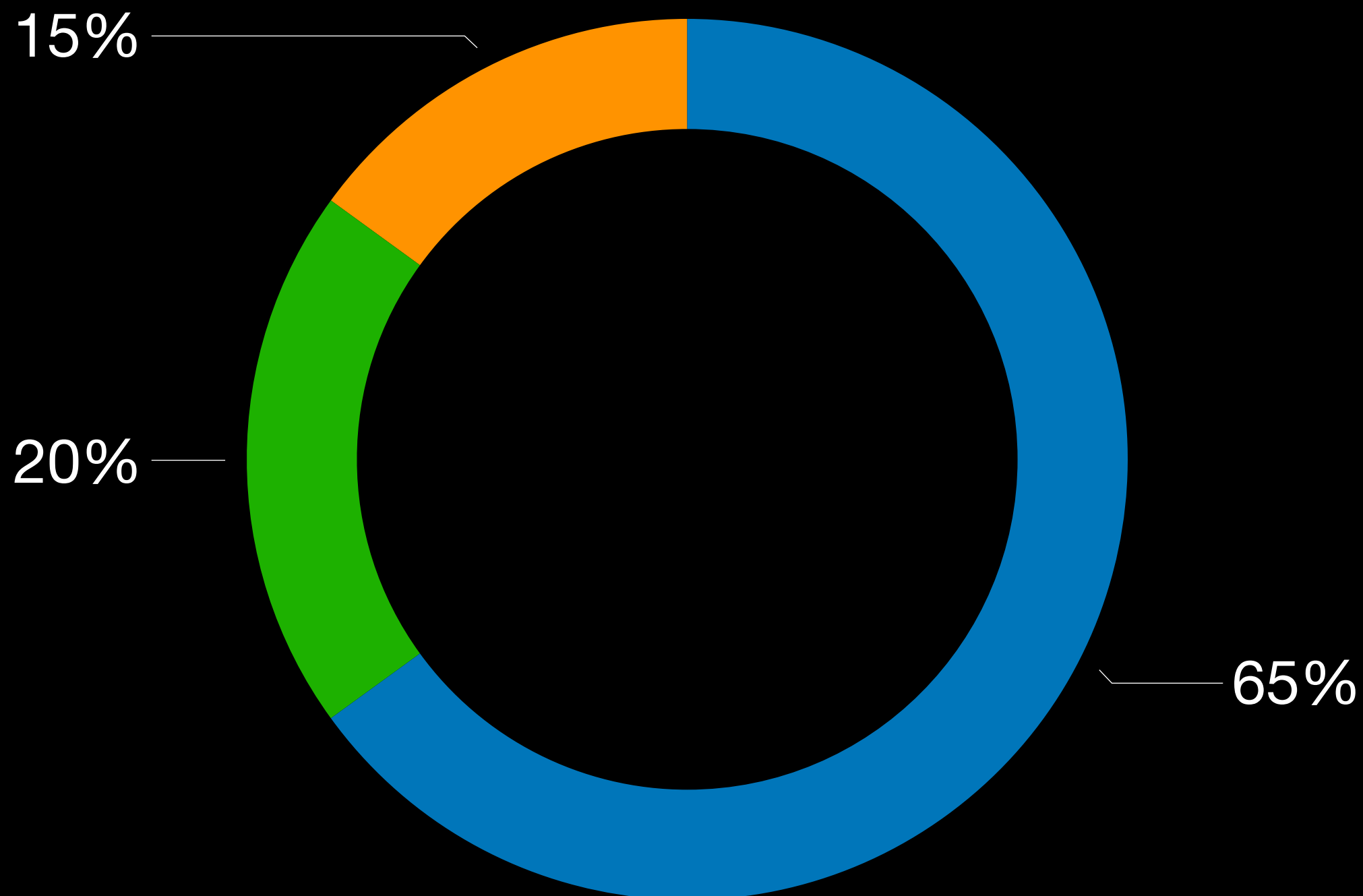
CHOOSE WISELY

**¿Porqué no nos hemos
puesto de acuerdo?**

**“Somos muy malos en lo que
hacemos”**

**“Puede que el software sea la
creación más compleja que haya
hecho el cerebro humano hasta
ahora”**

● **Riesgo bajo** ● **Riesgo Medio** ● **Crítico**



- **Menor riesgo menor consecuencia del error**
- **El fallo se vuelve la norma**
- **Habilidad de desarrollo == Habilidad de arreglar errores**
- **Podemos hacerlo mejor**

En resumen

- **Maneras más efectivas de probar UI**
- **Está OK no usar TDD todo el tiempo (o no usarlo nunca)**
- **Una prueba valida diferentes efectos secundarios**
- **Las pruebas son útiles aún cuando no se ejecutan**
- **Las pruebas mantienen los costos bajos**

**Experimenta &
Falla &
Aprende &
Repite.**

Preguntas?

rigovides@gmail.com