



Pontificia Universidad Católica de Chile  
Escuela de Ingeniería  
Departamento de Ciencia de la Computación

# **Clase 2**

# **Introducción: Conceptos**

## **IIC3745 – Testing**

Rodrigo Saffie

rasaffie@uc.cl

17 de agosto de 2020

## 1. Clase pasada:

- Aspectos Administrativos
  - Curso y horario
  - Ayudantes
  - Objetivos y contenidos del curso
  - Canales de comunicación
  - Evaluaciones
- Introducción
  - Motivación

## 2. Introducción

- Conceptos

# Ayudantes

<b>Ignacio Acosta</b>	<b>iaacosta@uc.cl</b>
<b>Luis Fros</b>	<b>lmfros@uc.cl</b>
<b>Alexander Israel</b>	<b>adisrael@uc.cl</b>
<b>Mauricio Valdivia</b>	<b>movaldivia@uc.cl</b>
<b>Hernán Valdivieso</b>	<b>hfvaldivieso@uc.cl</b>

# Encuestas

Jueves 20 de agosto:

- Conocimientos generales
- Grupos proyectos semestral

## 1. Clase pasada:

- Aspectos Administrativos
  - Curso y horario
  - Ayudantes
  - Objetivos y contenidos del curso
  - Canales de comunicación
  - Evaluaciones
- Introducción
  - Motivación

## 2. Introducción

- Conceptos

# Calidad del Software

¿Qué entendemos por calidad?

Calidad según **David Garvin**:

- **Vista transcendental**: se percibe, pero no se puede explicar
- **Vista del usuario**: en base a los objetivos del usuarios final
- **Vista del productor**: según las especificaciones del producto
- **Vista del producto**: en función de lo que hace el producto
- **Vista del valor**: en base a lo que está dispuesto a pagar un consumidor

# Calidad del Software

¿Qué entendemos por calidad en el software?

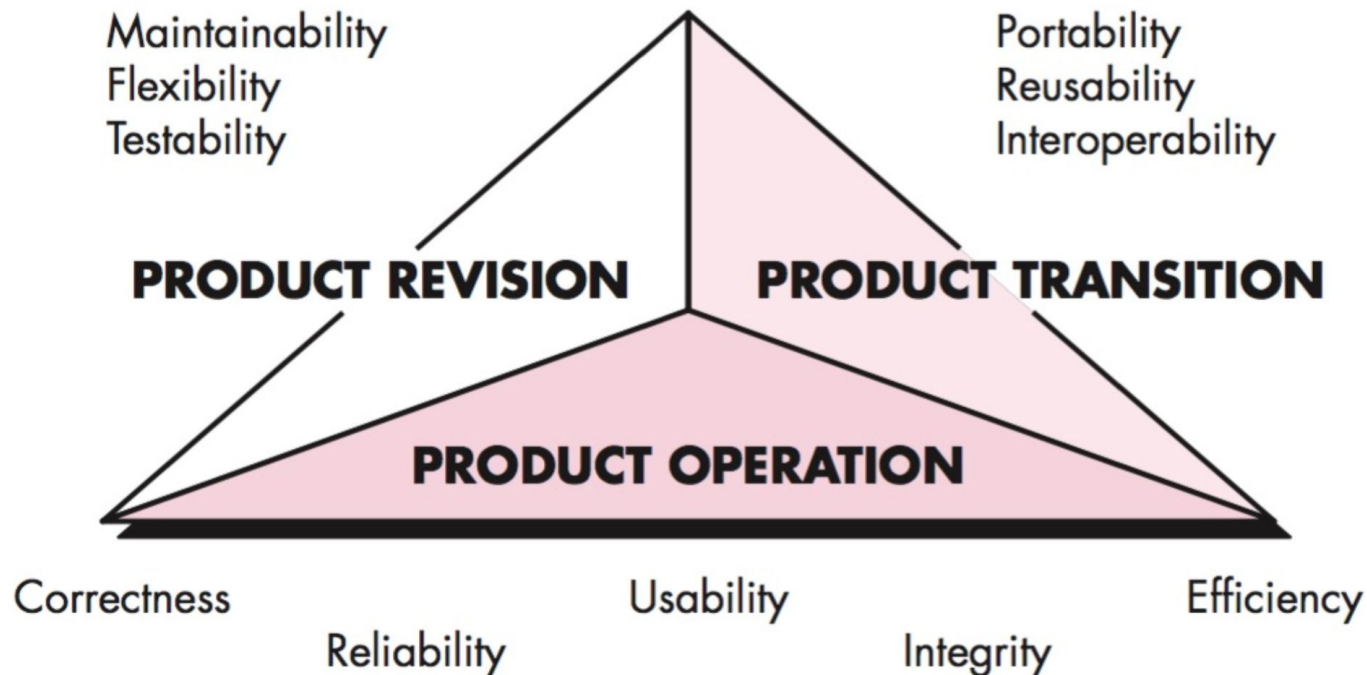
Calidad según **Roger S. Pressman**:

*“Un desarrollo de software efectivo, aplicado de una manera que crea un producto útil que provee valor cuantificable para aquellos que lo producen y aquellos que lo utilizan”*

# Calidad del Software

¿Qué entendemos por calidad en el software?

- Satisfacción total a requerimientos
- Atributos de calidad





# Calidad del Software

¿Qué es el software?

Según Pressman:

*“Software es: (1) instrucciones que cuando se ejecutan proveen funcionalidades, funciones y rendimiento deseado; (2) estructuras de datos que permiten a los programas manipular información de manera adecuada, y (3) información descriptiva tanto en forma física y virtual que describe la operación y uso del programa.”*

# Calidad del Software

¿Qué es el software?

- No se manufactura, se desarrolla
- No se fatiga/degrada, queda obsoleto

# ¿Por qué falla el software?

En los requisitos:

- Faltan requisitos
- Requisitos mal definidos
- Requisitos no realizables
- Diseño de software defectuoso

En la implementación:

- Algoritmos incorrectos
- Implementación defectuosa

# Errores, Defectos, Fallas

- **Error:** acción humana que produce un resultado incorrecto.
- **Defecto:** presencia de una imperfección que puede generar fallas.
- **Falla:** comportamiento observable incorrecto con respecto a los requisitos.

Un **error** introduce un **defecto** en el software que se manifiesta a través de una **falla** en las pruebas.

# Ejemplo

```
1  // Retorna la cantidad de ceros contenidos en un arreglo
2  function countZeros(array) {
3      var count = 0;
4      for (var i = 1; i < array.length; i++) {
5          if (array[i] === 0) {
6              count++;
7          }
8      }
9      return count;
10 }
```

```
11
12 countZeros([2, 7, 0]);
13 // Esperado 1 – Observado 1
```

**Falla:** ninguna

```
14
15 countZeros([0, 2, 7]);
16 // Esperado 1 – Observado 0
```

**Falla:** retorna 0

**Defecto:** no considera el primer elemento

**Error:** `var i = 1;`

# ¿Qué es un *bug*?


- Concepto que se utiliza informalmente para representar un defecto, un error y/o una falla
- Hace alusión a que “algo no anda bien”
- Origen:
  - ~1880: [Thomas Edison](#) para referirse a problemas de diseño
  - 1947: Reportado por [Grace Hopper](#) en proyecto Mark I de la universidad de Harvard

# ¿Qué es un *bug*?

Photo # NH 96566-KN (Color) First Computer "Bug", 1947

92

9/9

0800 Antan started  
 1000 " stopped - antan ✓  
 1300 (032) MP - MC 1.5826000  
 (033) PRO 2 2.130476415  
 convd 2.130676415  
 Relays 6-2 in 033 failed special speed test  
 in Relay " 11.00 test.  
 Relays changed  
 1100 Started Cosine Tape (Sine check)  
 1525 Started Multy Adder Test.  
 1545  Relay #70 Panel F  
 (moth) in relay.  
 First actual case of bug being found.  
 1630 Antan started.  
 1700 closed down.

Relay  
2145  
Relay 3371

# Validación y Verificación

## Validación:

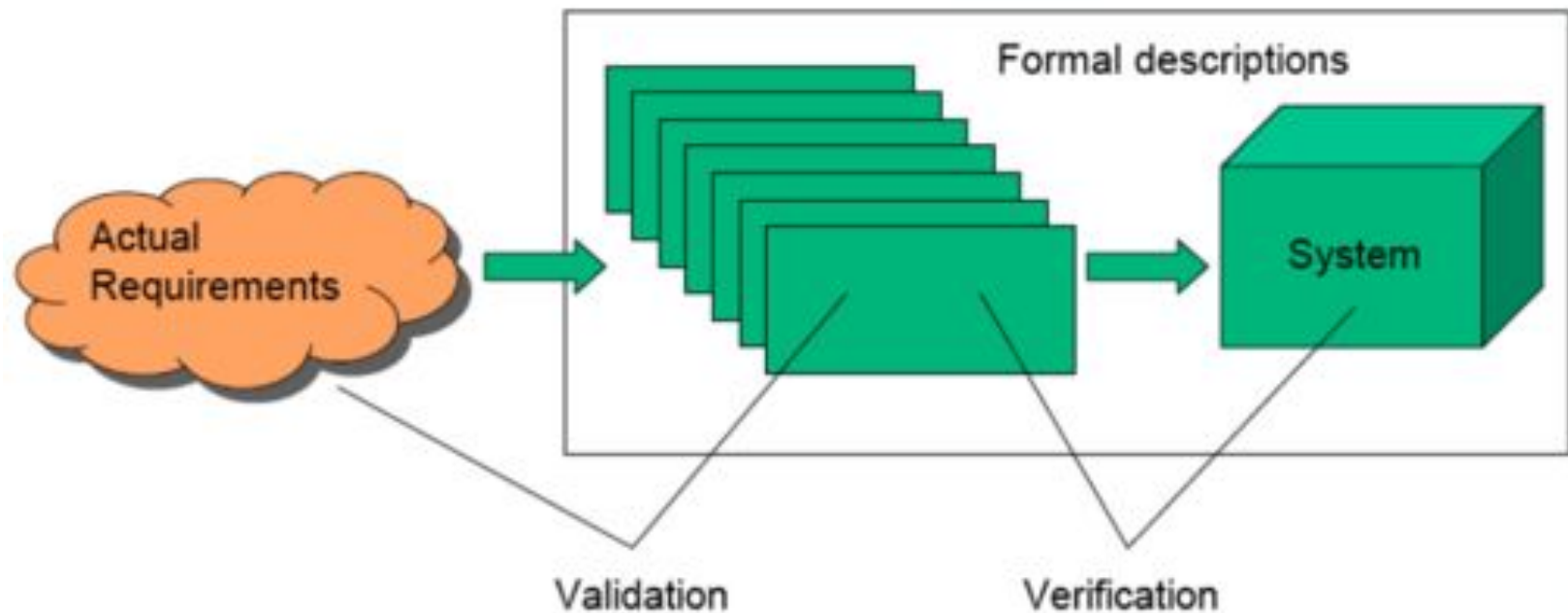
- ¿Estamos construyendo el producto **correcto**?
- El sistema hace lo que realmente se necesitaba que hiciera

## Verificación:

- ¿Estamos construyendo el producto **correctamente**?
- El sistema hace correctamente lo que se especificó



# Validación y Verificación



# Hacia ausencia de defectos

- No incorporarlos al construir software (muy difícil)
- Análisis estático
- Inspección formal de código
- *Testing*

# Análisis estático

- Análisis del código sin ejecutarlo, mediante herramientas que permiten detectar elementos sospechosos
  - Ejemplos: [rubocop](#), [brakeman](#), [reek](#), [flay](#), [bundler-audit](#), [dependabot](#)
- Se detecta gran cantidad de falsos positivos
- No se pueden detectar defectos importantes

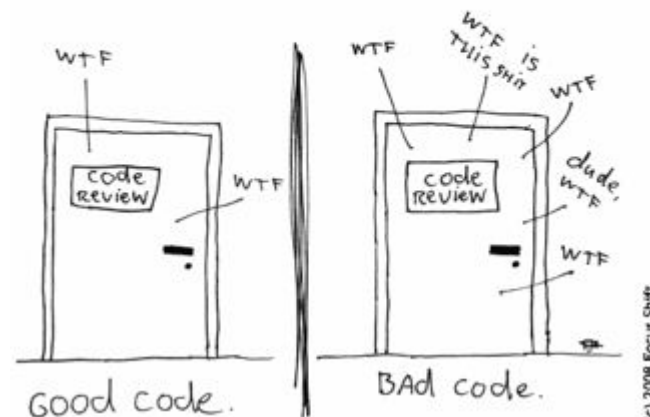
# Inspección formal de código

- Código es revisado por equipo de pares con el objetivo de detectar la mayor cantidad de defectos
- Forma efectiva para producir código de calidad
- Ejemplos: *Pull Requests* o sesiones de inspección

The Peer Code Review



The ONLY VALID MEASUREMENT  
OF CODE QUALITY: WTFs/minute



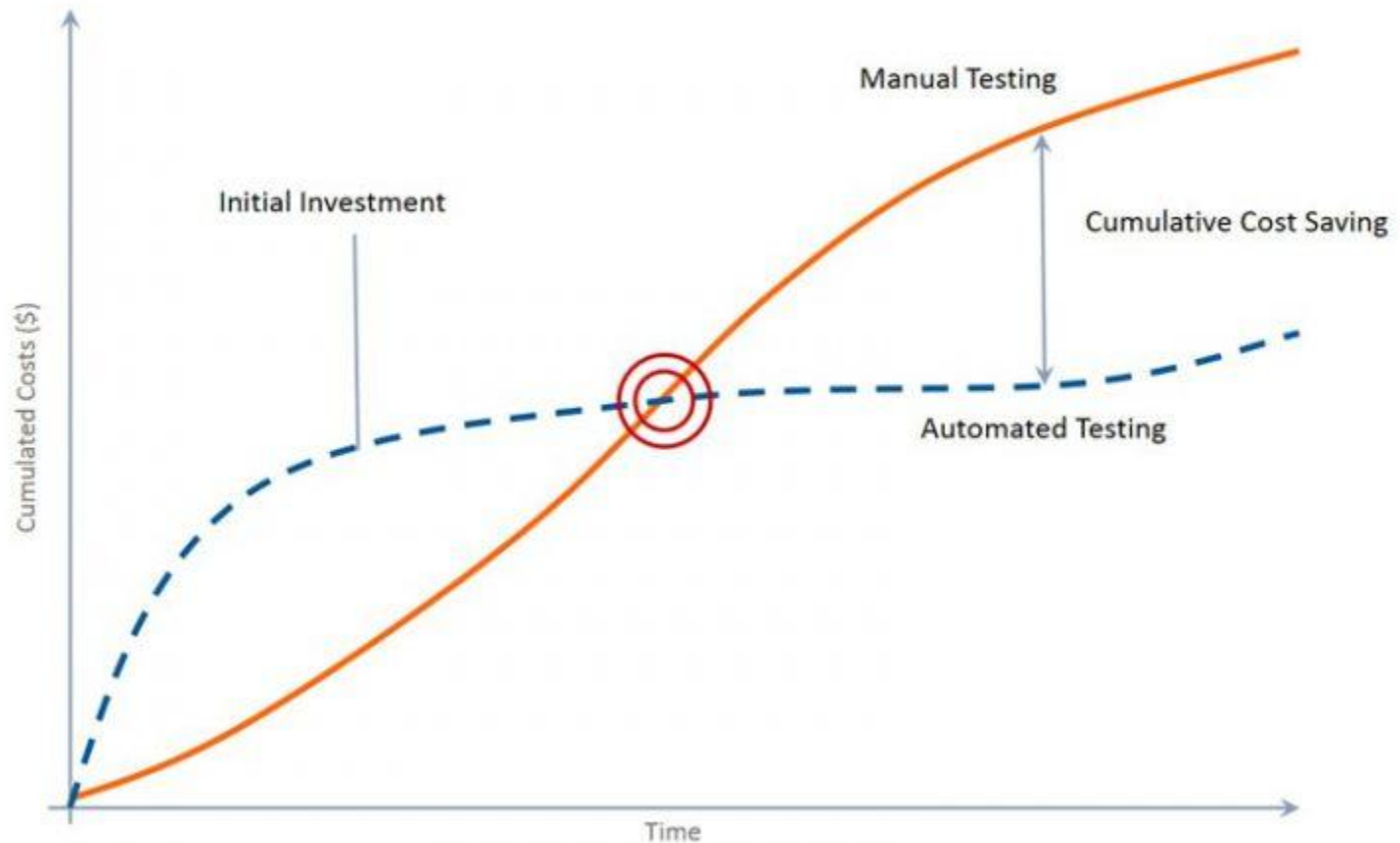
# *Testing*

- Proceso de ejecutar un programa con el objetivo de encontrar un error
- Se diseñan casos de prueba y se somete el *software* a ellas
- Un buen caso de prueba es uno con una alta probabilidad de encontrar un error oculto
- No necesariamente es automatizado: *debugging* o uso de plantillas

# *Testing*: ejemplo de plantilla

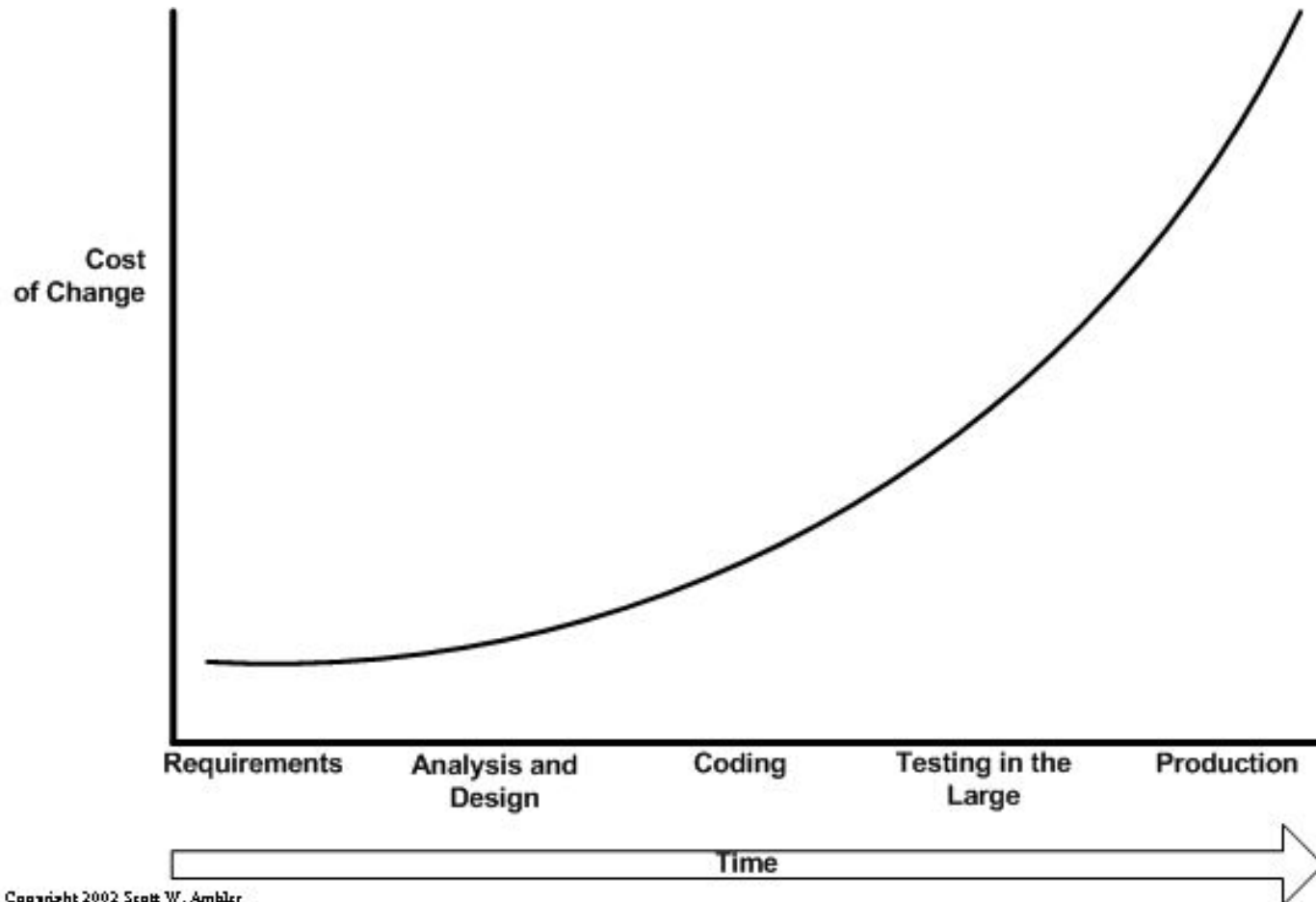
Nombre	
ID	
Casos de uso asociados	
Descripción	
Precondiciones	
Escenarios	
Resultados esperados	
Resultados obtenidos	
Errores detectados	
Comentarios	

# *Testing*: manual vs automatizado



# Costo del cambio

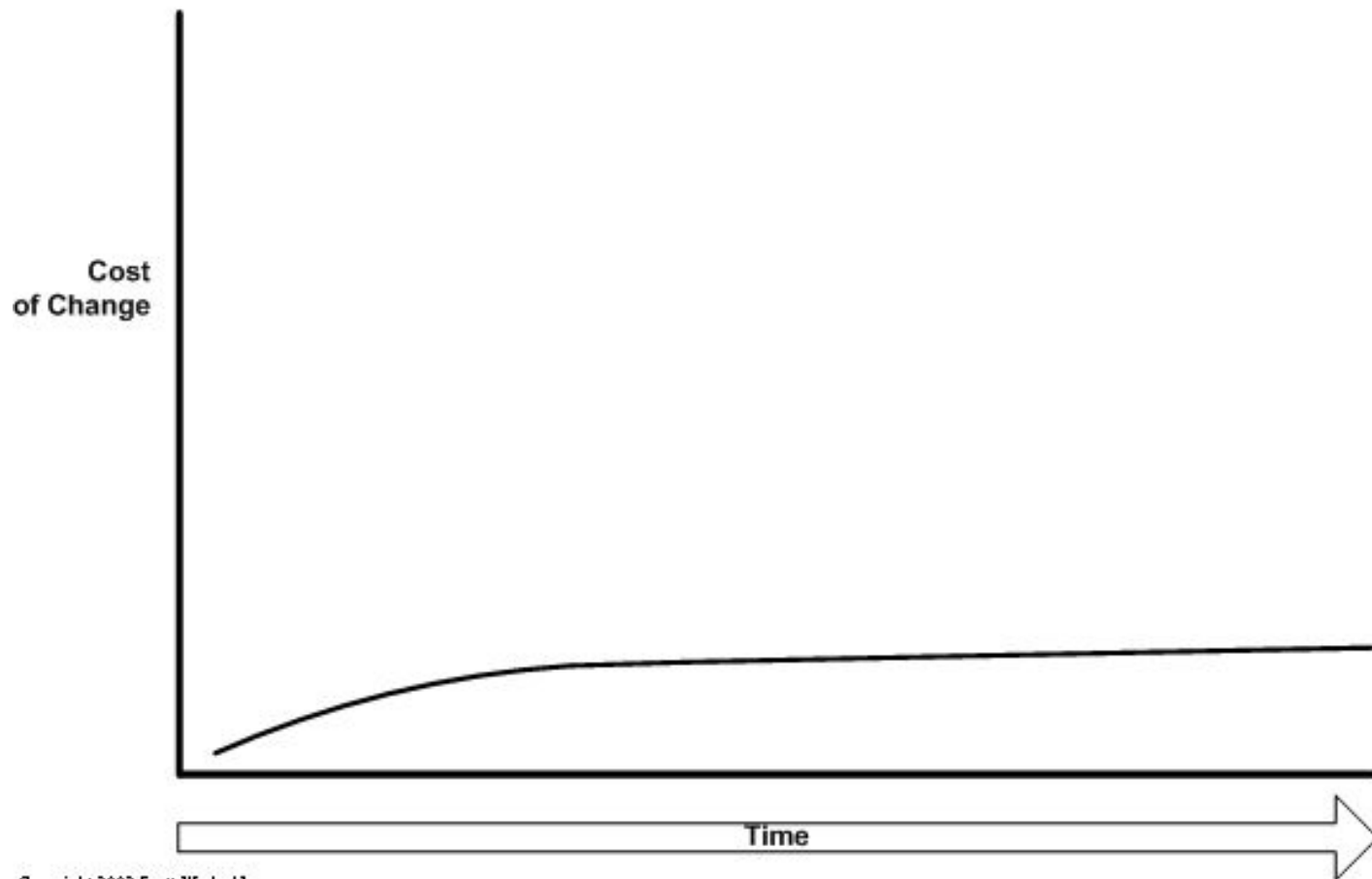
## Desarrollo Tradicional





# Costo del cambio

## Desarrollo Ágil



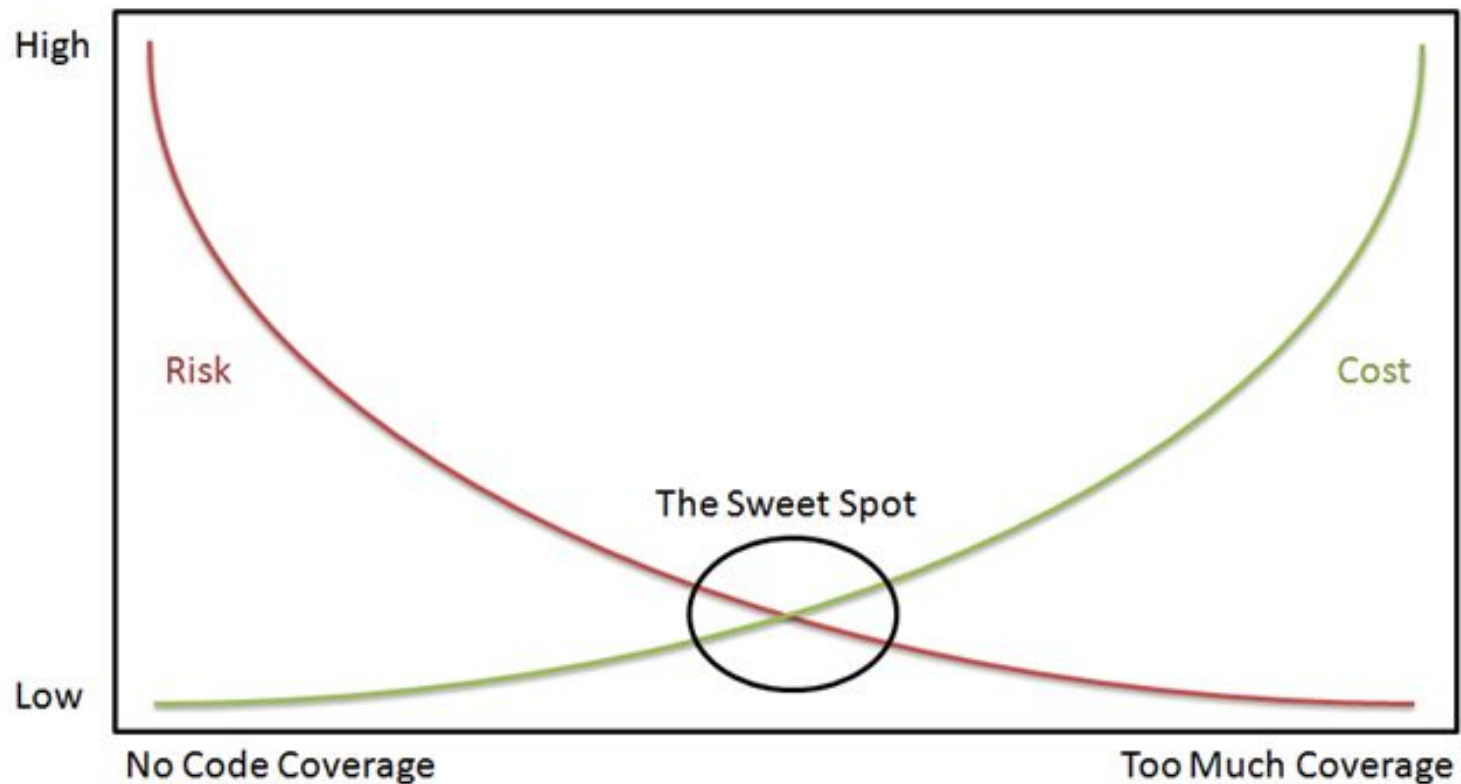
# Costo del *Testing*

- ¿Hasta cuándo *testear*?

```
1  function printer(i) {  
2    while (i < 10) {  
3      console.log(i);  
4      i++;  
5    }  
6  }  
7  
8  printer(a);
```

# Costo del *Testing*

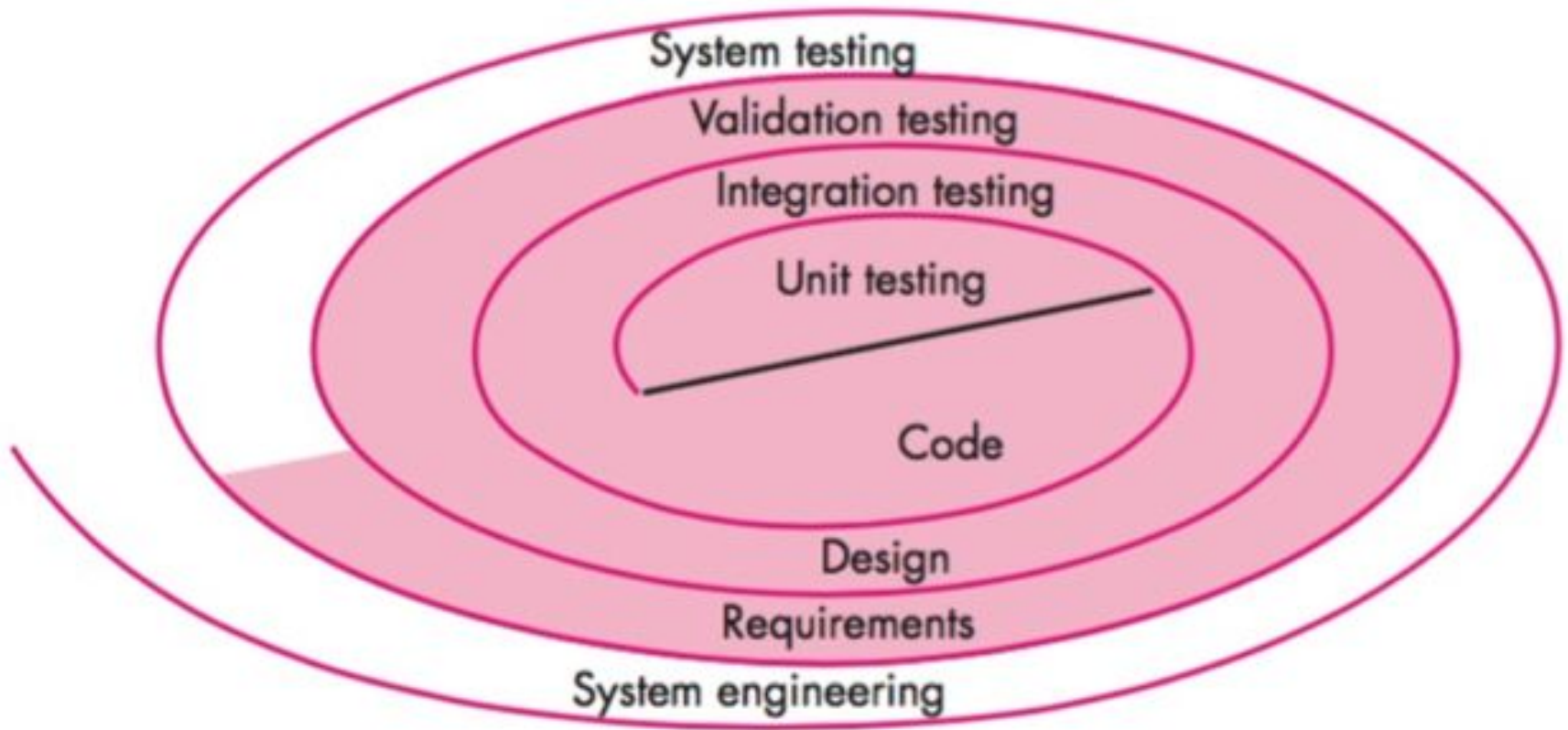
- ¿Se debe *testear* todo el *software*?



# ¿Por qué hacer pruebas?

- Mejorar calidad / disminuir riesgo de defectos
- Reducir costos a causa de errores no detectados
- Garantizar nivel de servicio
- Generar confianza en el *software*

# Niveles de *Testing*





Pontificia Universidad Católica de Chile  
Escuela de Ingeniería  
Departamento de Ciencia de la Computación

# **Clase 2**

# **Introducción: Conceptos**

## **IIC3745 – Testing**

Rodrigo Saffie

rasaffie@uc.cl

17 de agosto de 2020