



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

# Clase 16

# Pruebas de integración / validación

**IIC3745 – Testing**

Rodrigo Saffie

[rasaffie@uc.cl](mailto:rasaffie@uc.cl)

28 de octubre de 2020

1. Anuncios
2. Recapitulación
3. Pruebas de integración
4. Pruebas de validación

- Charlas
  - ¿Sensaciones?
- Actividad 4
  - Fecha de entrega: 3/11
- Proyecto
  - GitHub Actions
  - Correcciones entrega 2
    - Avances
    - Coevaluaciones
  - Cambio de fechas
    - E3: 13/11
    - E4: 4/12
- ¿Preguntas?

1. Anuncios
- 2. Recapitulación**
3. Pruebas de integración
4. Pruebas de validación

# Recapitulación

- Conceptos
- Criterios de cobertura
  - Grafos
  - Lógica
  - Dominio
- Pruebas unitarias
  - Mocks / Stubs
  - TDD

1. Anuncios
2. Recapitulación
- 3. Pruebas de integración**
4. Pruebas de validación

# Pruebas de integración

- Consisten en probar módulos o componentes de software manera conjunta. Ya sea:
  - Pruebas unitarias sin realizar *stubs/mocks*
  - Probar la interacción entre ellos
- Sirven para verificar la especificación de requisitos funcionales.
  - También se pueden conocer como pruebas de aceptación.
  - Normalmente se realizan luego de las pruebas unitarias y antes de las pruebas de validación.
- Si bien son rápidos de implementar, cuando fallan no entregan suficiente detalle y es fácil olvidar casos de prueba.

# Pruebas de integración

- Diferentes enfoques de realización:
  - *Bottom-up*
  - *Top-down*
  - *Sandwich*
  - *Big bang*
  - *Risky-hardest*
- En RSpec + RoR:
  - *Request Spec*
  - *Feature Spec*

1. Anuncios
2. Recapitulación
3. Pruebas de integración
- 4. Pruebas de validación**

# Pruebas de validación

- Con las pruebas unitarias y de integración tenemos cierto nivel de certeza que la aplicación “funciona”.

**¿Funciona como los usuarios esperan/quieren?**

- Puede que la lógica de negocio esté correcta, pero es **inútil** si es que los usuarios no saben/pueden interactuar con ella.

# Pruebas de validación

- Consisten en evaluar un sistema con usuarios representativos del producto/servicio.
- Es un concepto fundamental en el proceso de *User Experience (UX) design*.
- No es algo exclusivo al *software*, siempre ha existido en el mundo del *marketing*:
  - *Focus Groups*
  - Encuestas de satisfacción
  - *Net Promoter Score (NPM)*

# Pruebas de validación

## lab51

¿De 1 a 10, cuán probable es que le recomiendes Lab51 a otra persona?



Nada probable

Muy probable

# Customer experience

- Son las interacciones entre un cliente y un proveedor.

## Modern Customer Journey



# *Customer experience*

- Los proveedores pueden ofrecer mejores productos/servicios al entender las interacciones con sus clientes:
  - Se analiza toda la experiencia, no solamente la compra.
  - Se enfocan los esfuerzos en lo que los usuarios necesitan realmente.
  - Se pueden detectar puntos de contacto con falencias.

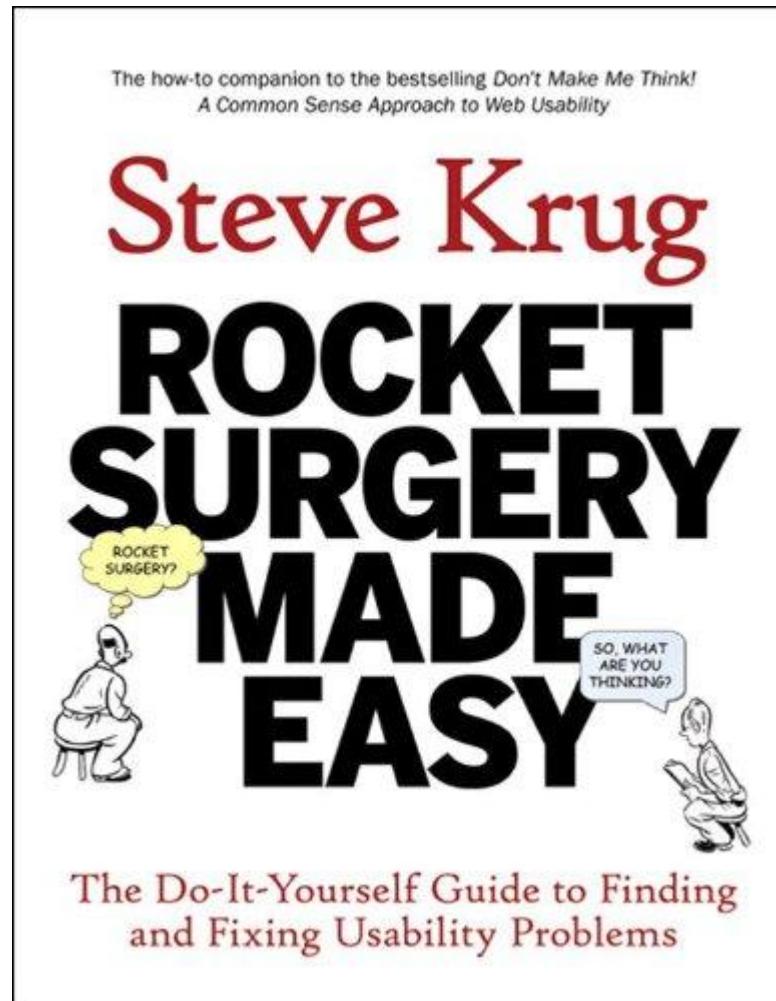
# Pruebas de validación

- Pruebas de usuarios
- *A/B testing*
- *Alpha/Beta testing*

# Pruebas de usuarios

- Se enfocan en analizar cómo los usuarios interactúan con la aplicación.
- Se definen tareas que los usuarios deben completar para así detectar dificultades o sensaciones de estos.
- No se está evaluando a los usuarios, sino que si la aplicación cumple con sus expectativas.

# Pruebas de usuarios



# Beneficios

- Validan si los usuarios pueden completar ciertas tareas sin obstáculos dentro de tiempos razonables.
- Evalúan la satisfacción de los usuarios con el sistema.
- Detectan oportunidades de mejora en base a las necesidades de los usuarios.

# ¿Cuándo se prueba?

- Se pueden realizar pruebas antes, durante o después.
  - *Sketches*
  - *Wireframes*
  - Prototipos de vistas
  - Aplicación funcional
- Mientras más seguido se realicen más temprano se detectarán mejoras, por lo que serán más fáciles de implementar.
- Una buena práctica es por lo menos una vez al mes, aunque depende del contexto.

# ¿Con quién se prueba?

- Con personas que sean representativas de los usuarios:
  - No sirve hacer pruebas con gente que no utilizará el sistema ni está familiarizada con los conceptos.
  - Un *subset* de usuarios representativos sirve para detectar casi los mismos problemas que para todos los usuarios.
- Las pruebas serán tan buenas como los usuarios lo sean:
  - representativos
  - dispuestos a entregar *feedback*
- Se deben invertir recursos en seleccionar con cuidado a estos usuarios.

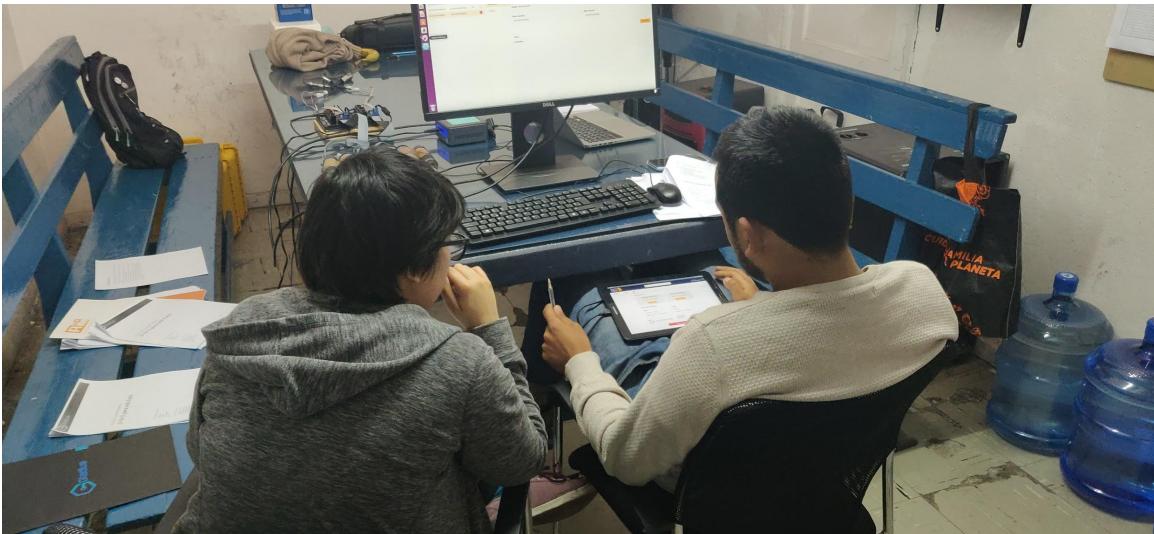
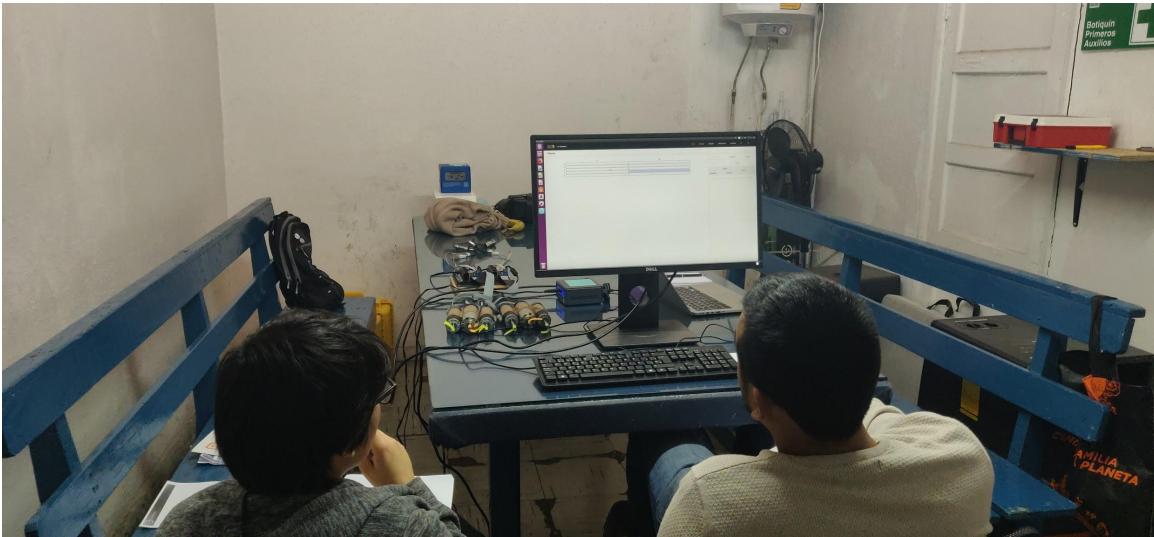
# ¿Con quién se prueba?

- Problema:
  - ¿Cómo se definen los usuarios representativos?
- La audiencia objetiva suele ser más amplia de lo que uno imagina.
- Es mejor probar con “usuarios comunes” que esperar al “usuario modelo”.
  - Se debe ser consciente de los sesgos que estos pueden tener.
- Evitar repetir rondas de pruebas con usuarios repetidos.

# ¿Qué se prueba?

- Se deben listar las tareas más importantes a evaluar:
  - Las más críticas para el negocio
  - Las menos probadas
  - Las con más criticadas por los usuarios
- Luego, se deben plantear estas como escenarios para que los usuarios interactúen con el sistema.

# La prueba misma



# La prueba misma

- Un facilitador
  - Guía al usuario en las tareas a realizar.
  - Pregunta constantemente qué piensa y siente el usuario al interactuar.
  - Es importante mantenerse neutral para no sesgar los resultados.
- Para cada escenario:
  - Cronometrar tiempo.
  - ¿El usuario logra ejecutarlo?
  - ¿Cuál es su sensación, qué opina?

# La prueba misma

- Gente observando
  - Se puede grabar la pantalla, las interacciones del usuario, donde dirigió su vista, entre otros.
  - Mientras más gente analice las pruebas más perspectivas se obtendrán del uso de la aplicación.
  - Es más fácil convencer a la organización de que un cambio es necesario si es que participaron de la prueba que lo detectó.
- *Hotjar*

# Reporte de pruebas

- Listar los problemas más frecuentes y graves que se detectaron entre los usuarios.
- Diseñar opciones para solucionarlos
  - Priorizar las soluciones simples y baratas, que los usuarios decidan cuál es mejor.

# A/B testing

- Ampliamente utilizado en sitios web.
- Se prueban dos versiones distintas de una página y se evalúa cual resulta mejor:



50 % visitors  
see variation A



23%  
conversion



50 % visitors  
see variation B



11%  
conversion

# *Alpha/Beta testing*

- Pruebas de la aplicación simulando ambiente real de utilización.
- *Alpha testing:*
  - Usuarios o desarrolladores prueban el producto en el sitio de desarrollo bajo la dirección de un encargado de las pruebas.
- *Beta testing:*
  - Número de usuarios controlado prueba libremente el producto en su propio ambiente de uso.





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

# Clase 16

# Pruebas de integración / validación

**IIC3745 – Testing**

Rodrigo Saffie

[rasaffie@uc.cl](mailto:rasaffie@uc.cl)

28 de octubre de 2020