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

Clase 9
Calidad del *software* – Ingeniería Reversa

IIC2113 – Diseño Detallado de Software

Rodrigo Saffie

rasaffie@uc.cl

16 de noviembre de 2018

Interrogación 2

¿Qué entendemos por calidad?

Calidad según David Garvin (1984):

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

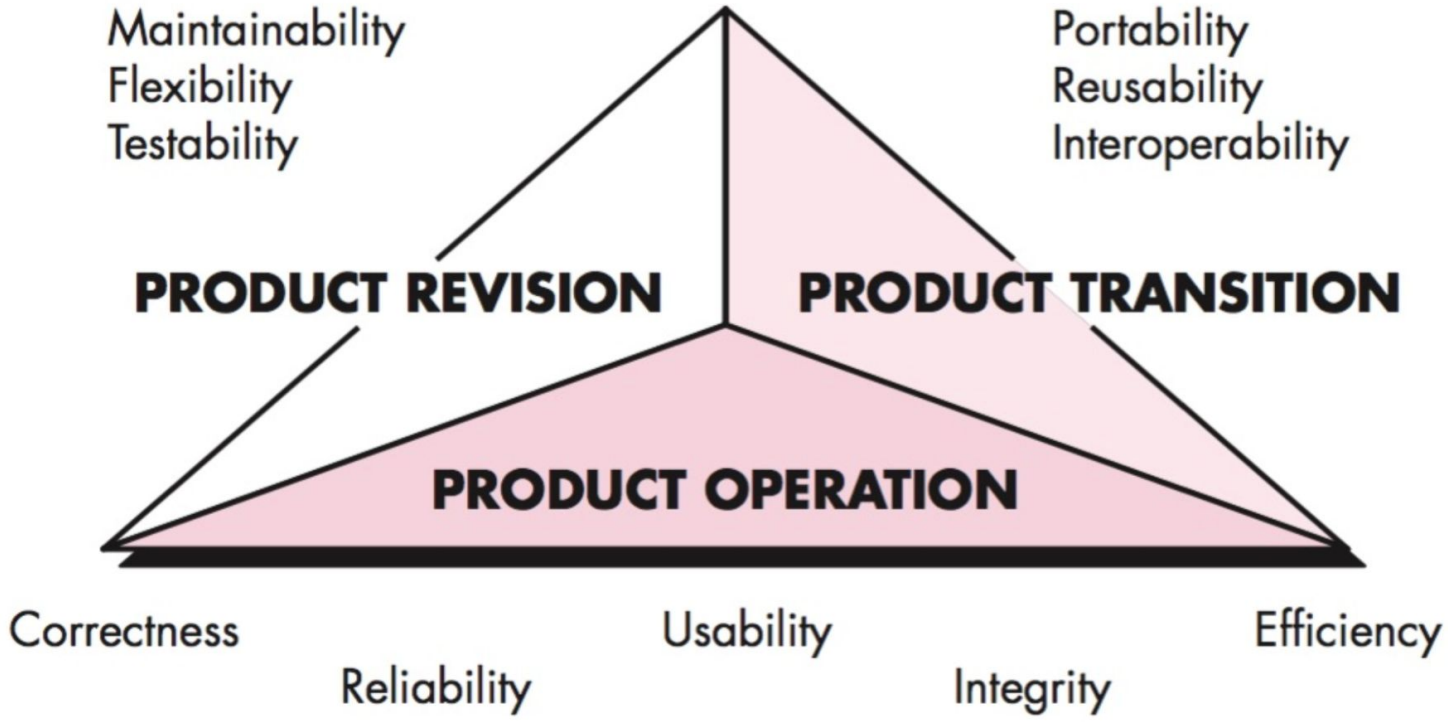
¿Qué entendemos por calidad del *software*?

Calidad según Pressman (2009):

"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."

¿Qué entendemos por calidad del *software*?

Factores de Calidad [McCall, 1977]:



Métricas de calidad

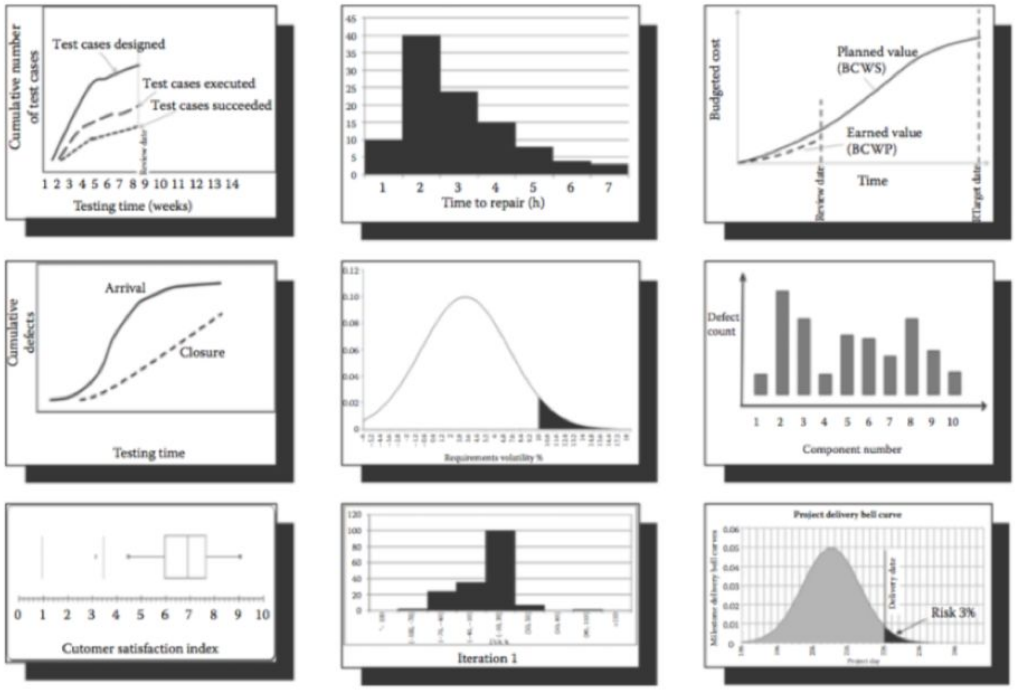
- Una métrica son datos procesados que expresan numéricamente el rendimiento sobre un criterio
  - Ejemplo: *Coverage*
- Sirven para:
  - comparar la efectividad de distintas estrategias
  - tener respaldo cuantitativo sobre un criterio

Etapas de una métrica

- **Formulación**: formalización de factores apropiados para representar el *software*
- **Recolección**: mecanismos para acumular datos a partir de la formulación
- **Análisis**: procesamiento de los valores recolectados para obtener información
- **Interpretación**: evaluación de la información para determinar mejoras
- **Retroalimentación**: recomendaciones para el equipo de desarrollo, derivadas de la interpretación

Visualización de métricas

- Las métricas se pueden representar con gráficos



Características de una buena métrica

- Simple y computable
- Intuitiva
- Consistente y objetiva
- Unidades de medición expresivas
- Independiente del contexto (equipo, lenguaje de programación)
- Reflejar recomendaciones para mejorar

Ejemplos de métricas

- Complejidad ciclomática:
  - Es una métrica basada en el cálculo del número de caminos independientes que tiene el código
  - Un buen valor referencial es 11, que representa métodos sencillos
- **A**ssignment **B**ranch **C**ondition Size (ABCSize):
  - Cuenta el número de asignaciones, ramas (o "saltos") y condicionales en un programa
  - Sirve para medir tamaño y complejidad del *software*

Ejemplos de métricas

CK Metrics Suite:

- Conjunto de métricas propuestas por Chidamber & Kemerer (1994)

CK metrics suite

WMC: **W**eighted **M**ethods per **C**lass

- Es la suma de la complejidad ciclomática de cada método de una clase
- Es un indicador predictivo del esfuerzo necesario para mantener y extender una clase

CK metrics suite

DIT: **D**epth of **I**nheritance **T**ree

- Es la distancia máxima de una clase base a una 'hoja' de la jerarquía
- Refleja la complejidad del diseño

CK metrics suite

NOC: **N**umber **O**f **C**hildren

- Cantidad de subclases directas de una clase
- Refleja el nivel de abstracción del diseño

CK metrics suite

CBO: **C**oupling **B**etween **O**bject classes

- Cantidad de clases con las que colabora una clase
- Altos niveles reflejan un alto acoplamiento, lo que dificulta modificar el código

CK metrics suite

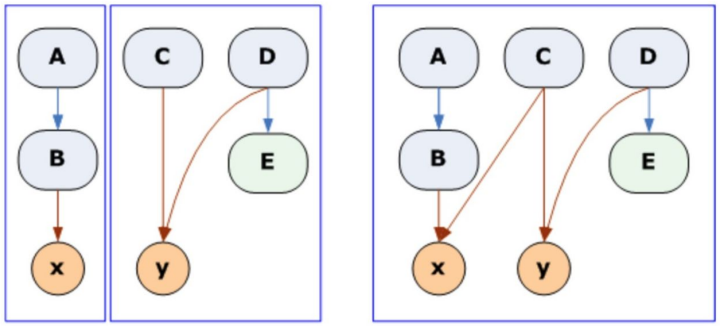
RFC: **R**esponse **F**or a **C**lass

- Cantidad de métodos únicos que son invocados desde una clase (propios y externos)
- Altos niveles reflejan alta complejidad, lo que dificulta entender y probar el código

CK metrics suite

LCOM: **L**ack of **C**ohesion **O**f **M**ethods

- Cantidad de grupos de métodos que acceden a 1 o más atributos en común de una clase
- Altos niveles reflejan baja cohesión, lo que significa que la responsabilidad de una clase no está bien definida



Ingeniería Reversa

- El término ingeniería reversa (o inversa) proviene del mundo del *hardware.*
- Una empresa obtiene, desarma y analiza un producto de la competencia, con el fin de desvelar sus secretos.
- Este proceso se realiza porque no se tiene acceso a las especificaciones ni a la documentación del producto.

Ingeniería Reversa

Samsung Smartphones **BEFORE** iPhone	Apple's iPhone (announced Jan. 2007)	Samsung Smartphones **AFTER** iPhone
		

Ingeniería Reversa

Ejemplos de utilización:

- Espionaje militar o comercial
- Análisis de seguridad
- Crear copias/alternativas
- Mejorar especificaciones de un producto indocumentado

Ingeniería Reversa

- Objetivo en *software*: descubrir el diseño de una aplicación.
- Es el proceso de analizar un programa con el fin de crear una representación de más alto nivel.
- Generalmente se aplica de forma interna sobre un producto *legacy*, por lo que pareciera ser ajeno a la empresa.

Ingeniería Reversa

Nivel de extracción:

- Se refiere a la sofisticación del diseño extraído de un *software*. A mayor nivel, más información se puede extraer.
- Los distintos niveles son:
  - Diseño de procesos
  - Diseño estructural de datos
  - Modelo del sistema
  - Relación entre componentes

Ingeniería Reversa

Completitud:

- Dado un nivel de extracción, se refiere a la cantidad de detalle que se puede obtener.
- Generalmente es inversamente proporcional al nivel de extracción.

Ingeniería Reversa

Beneficios:

- Reducir la complejidad del sistema
- Recuperar o actualizar información
- Identificar el alcance del producto
- Facilitar la reutilización de productos

Ingeniería Reversa

Ejemplo:

Unbundling Pokemón Go

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

Clase 9
Métricas de calidad – Ingeniería Reversa

IIC2113 – Diseño Detallado de Software

Rodrigo Saffie

rasaffie@uc.cl

16 de noviembre de 2018