



UNIVERSIDAD DE CUENCA  
Facultad de Ingeniería

Grado en Ingeniería de Sistemas  
Curso 2020-2021

# Calidad de Software

**Tema: Introducción a la Calidad de Software**

**Ing. Priscila Cedillo O. Ph.D.**

Departamento de Ciencias de la Computación  
Universidad de Cuenca, Ecuador  
email: [priscila.cedillo@ucuenca.edu.ec](mailto:priscila.cedillo@ucuenca.edu.ec)

# Introducción

- Desde antes de nacer, ya nos vemos sometidos a una serie de **mediciones** que determinan nuestra vida.
- Continuamente realizamos mediciones que nos **guian a la hora de tomar decisiones** y seleccionar la alternativa que creemos mejor, tanto en nuestra vida personal como en las diversas actividades.
- En la Ingeniería del Software hay que tener presente las **peculiaridades** que **diferencian** al software de otros productos.
- Incluso los desarrolladores más experimentados estarán de acuerdo en que **obtener software de alta calidad es una meta importante**.



# ¿Cómo se define la calidad de software?

Establece la infraestructura que da apoyo a cualquier esfuerzo de elaboración de un producto de software de alta calidad.

**“Proceso eficaz de software que se aplica de manera que crea un producto útil que proporciona valor medible a quienes lo producen y a quienes lo utilizan”**

Bessin J. The Business Value of Quality

Entrega contenido, funciones y características que el usuario final desea, de igual importancia es que entrega estos activos de forma confiable y libre de errores.

Al agregar valor para el productor y para el usuario de un producto, el software de alta calidad proporciona beneficios a la organización que lo produce y a la comunidad de usuarios finales.



# Definición de Calidad (ii)

n Principales objetivos estratégicos de las organizaciones.

n Calidad según la RAE.

1. Propiedad o conjunto de propiedades inherentes a algo, que permiten juzgar su valor.
2. Buena calidad, superioridad o excelencia.
3. Carácter, genio, índole.
4. Condición o requisito que se pone en un contrato.



La calidad suele ser transparente cuando está presente, pero resulta fácilmente reconocible cuando está ausente (por ejemplo cuando el producto falla o el servicio es deficiente).



# Vistas de la Calidad

- Vista trascendental. → Se reconoce pero no se define. Es un ideal al que se intenta llegar.
- Vista del usuario. → Adecuación al propósito. Cuantificar y medir productos, establecer objetivos a ser alcanzados.
- Vista del fabricante. → Calidad durante producción y entrega del producto. Vista centrada en el proceso.
- Vista del producto. → Unida a las características del producto en si mismo, “desde adentro”. Usuario y fabricante “desde fuera”
- Vista basada en el valor. → Cantidad que el cliente está dispuesto a pagar.



# Orígenes de la calidad

## Realizada

Persona que realiza el trabajo gracias a su habilidad en la ejecución de la tarea

Se potencia con la mejora de las habilidades personales y técnicas de los participantes en un proceso.

## Planificada

Es la que se ha pretendido obtener. Es la que aparece descrita en una especificación, en un documento de diseño o en un plano.

Se potencia con la elaboración de una especificación que sirva de buena referencia a los participantes de un proceso.

## Necesaria

Es la que el cliente exige con mayor o menor grado de concreción, o al menos, le gustaría recibir.

Se potencia con una adecuada obtención de información de la idea de calidad de los clientes.



Una buena gestión de calidad busca que estas tres coincidan lo más posible.

# Datos Históricos Importantes

Año	Hecho Importante
Épocas iniciales	<p><u>Alan Turing</u>, descifra los códigos secretos Enigma usados por los alemanes en la <u>II Guerra Mundial</u> para sus comunicaciones. Turing fue un pionero en el desarrollo de la lógica de los computadores modernos, y uno de los primeros en tratar el tema de la inteligencia artificial con máquinas.</p>
Primera era	<p>Durante los primeros años el software se hacia muy ajustado a las necesidades de cada persona.</p>
Segunda era	<p>Mitad de la época de los sesenta hasta finales de los setenta. Multi-programación y sistemas multi-usuario introdujeron nuevos conceptos de hci. Se empieza a hablar en milisegundos y ya no en minutos</p>



# Datos Históricos Importantes

Año	Hecho Importante
Tercera era	Mediados de los años 70 - más allá de una década. Sistemas distribuidos, múltiples computadoras, concurrencia, incrementó notablemente la complejidad de los sistemas informáticos. El final fue por la llegada de los microprocesadores. El microprocesador ha producido un extenso grupo de productos inteligentes, desde automóviles hasta hornos de microondas, desde robots industriales a equipos de diagnósticos de suero sanguíneo, pero ninguno ha sido más importante que la computadora personal.
Cuarta era	La cuarta era de la evolución de sistemas informáticos se aleja de las computadoras individuales y de los programas de computadoras, dirigiéndose al impacto colectivo de las computadoras y del software. Redes globales y locales, aplicaciones de software avanzadas. La industria del software ya es cuna de la economía mundial.



# La Crisis del Software (i)

- Concepto informático acuñado en 1968
  - Se refiere a la dificultad de escribir programas libres de defectos, fácilmente comprensibles y verificables.
- Problemas
  - Planificación y estimación del coste es frecuentemente muy imprecisa.
  - La “productividad” de la gente del software no se responde a la demanda de sus servicios.
  - La calidad del software no es la adecuada.



# La Crisis del Software (ii)

- Muchos proyectos de software fallaron porque:
  - No fueron entregados a tiempo
  - Se gastó mucho más de lo previsto
  - Software de baja calidad
  - Caro de mantener (corrección de fallas, modificaciones difíciles por cambio en requisitos, adaptación a nuevos dispositivos)
  - Software no cumplía las expectativas del cliente
  - No se recogen datos sobre el proceso de desarrollo de software. (No existen datos históricos).
    - Sin datos históricos como guía, la estimación no es buena y los resultados son pobres
    - No se puede evaluar la eficacia de nuevas herramientas.



# La Crisis del Software (iii)

- Para superar la crisis:
  - Aparición de metodologías concretas de desarrollo
  - Concepción de la Ingeniería del Software como disciplina
  - Trabajo en equipo y especialización (analistas, programadores, ...)
- No se ha llegado a una situación estable, sino a una evolución permanente con avances continuos en la IS, forzados por el rápido abaratamiento y aumento de capacidad del hardware.

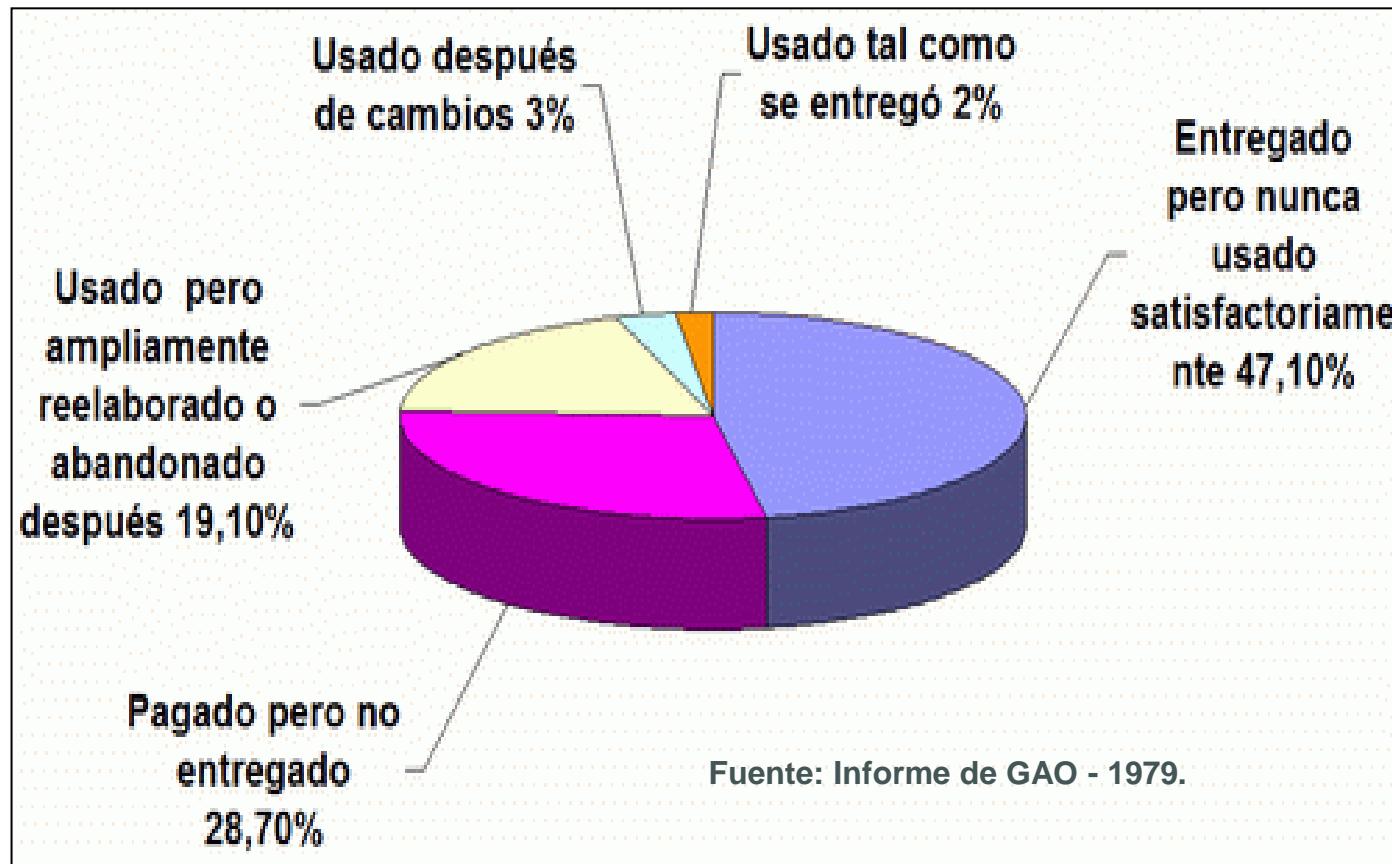


# La Crisis del Software (iv)

- Son los sucesivos fracasos de las distintas metodologías para:
  - Dominar la complejidad del software, lo que implica el retraso de los proyectos de software
  - Las desviaciones por exceso de los presupuestos fijados y la existencia de deficiencias respecto a los requisitos del cliente



# Informe GAO (1979)



Informe de GAO (Gobernment Account Office)

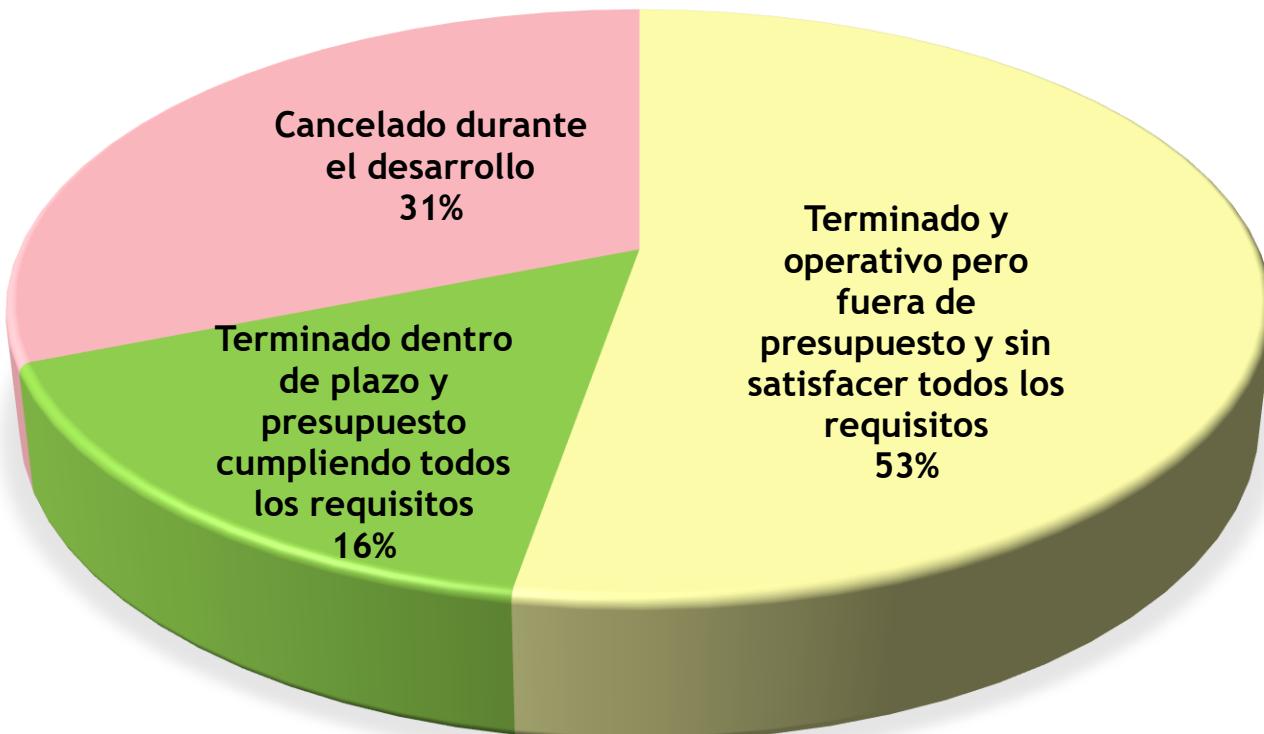


# Informe CHAOS

- Publicado cada año desde 1994
- Presenta el estado de la industria de Ingeniería del Software
- En 2015 se presentó el reporte de 50.000 proyectos alrededor del mundo
- Los resultados muestran que existe mucho trabajo aún por ser realizado en los proyectos de desarrollo de software.



# Informe CHAOS 1995



# Standish Group 2015 Chaos Report

	2011	2012	2013	2014	2015
Éxito	29%	27%	31%	28%	29%
Desafío	49%	56%	50%	55%	52%
Fracaso	22%	17%	19%	17%	19%



# CHAOS por tamaño de proyecto

	ÉXITO	DESAFIO	FRACASO
Muy Grande	2%	7%	17%
Grande	6%	17%	24%
Mediano	9%	26%	31%
Moderado	21%	32%	17%
Pequeño	62%	16%	11%
Total	100%	100%	100%



# Chaos por Metodología de Desarrollo

Tamaño	Método	Éxito	Desafío	Fracaso
Proyecto de todos los tamaños	Ágil	39%	52%	9%
	Cascada	11%	60%	29%
Proyectos Grandes	Ágil	18%	59%	23%
	Cascada	3%	55%	42%
Proyectos Medianos	Ágil	27%	62%	11%
	Cascada	7%	68%	25%
Proyectos Pequeños	Ágil	58%	38%	4%
	Cascada	44%	45%	11%



# La Complejidad del Desarrollo de Software

Sistemas de  
Software Simples

VS

Sistemas de  
Software Complejos



# La Ingeniería del Software y la Calidad

- Si se espera construir un software de alta calidad, debe **entender el problema** que se quiere resolver.
- También debe ser capaz de **crear un diseño que esté de acuerdo con el problema** y que al mismo tiempo tenga características que lleven al software a las dimensiones y **factores de calidad**.
- En la Ingeniería del Software existen **actividades sombrillas** que ayudan a controlar **la calidad**, el cambio y el riesgo del software.
  - Seguimiento y Control del Proyecto de Software
  - Gestión de Riesgo
  - Aseguramiento de la Calidad
  - Revisiones Técnicas Formales
  - Medición
  - Gestión de Configuración del Software
  - Gestión de Reutilización
  - Preparación y Producción del Producto de Trabajo



# Costos e Impactos de la Mala Calidad

## Casos Reales



**UNIVERSIDAD DE CUENCA**  
desde 1867

# Casos de Estudio - Caso 1. California DMV

- California Department of Motor Vehicles (DMV) -> Gran Proyecto para renovar licencias y proceso de registro
- En 1993 y 45 millones de dolares gastados el proyecto se canceló
- Querían adoptar una nueva tecnología
  - El Proyecto no fue soportado por una gestión ejecutiva
  - No intervino el usuario
  - Planificación pobre
  - Diseño pobre
  - Objetivos no claros
  - El Proyecto estuvo condenado desde el inicio



## Casos de Estudio - Caso 2. CONFIRM

- Problema involucró a Airlines, Budget Rent-A-Car, Marriot y Hoteles Hilton
- 1994 - CONFIRM ->Un proyecto de una empresa de alquiler de vehículos y reservaciones de hotel, colapsó en un caos.
- Gastó 165 millones de dólares.
- Problemas con la definición correcta de requisitos.
- Falta de involucramiento de los usuarios
- Constante cambios en requisitos y especificaciones



# Casos de Estudio - Caso 3. Hoteles Hyatt

- Mientras Marriot y Hilton estaban sin poder utilizar su sistema de reservación, Hyatt lo estaba haciendo.
- Hoy se puede reservar un cuarto de hotel incluso desde un avión a 35000 pies de altura, reservar el hotel de cortesía para que recojan a un huésped y tener todo listo.
- El proyecto estuvo de acuerdo a la planificación, dentro del presupuesto, con características adicionales por tan solo 15 millones de dólares.
- Tuvo todos los ingredientes para el éxito:
  - Involucramiento del usuario
  - Ayuda de una gestión ejecutiva
  - Clara descripción de requisitos
  - Planificación adecuada
  - Hitos pequeños en el desarrollo



# Casos de Estudio - Caso 4. Banco Itamarati

- Banco de Brasil->Crecimiento 51% y se desplazó del lugar 47 al 15 en la industria bancaria brasileña.
- Razones fundamentales de su éxito:
  - Visión clara con objetivos específicos documentados.
  - Alto involucramiento de todos en el proceso.
  - Manejo de una relación cercana con sus clientes.
  - No tuvo una descripción clara de los requisitos de software.
  - Tuvo una buena planificación.
  - Personal altamente calificado.
  - Utilización de buena tecnología.



# Casos de Éxito y Fracaso

Success Criteria	Points	DMW	CONFIRM	Hyatt	Itamarati
1. User Involvement	19	No (0)	No (0)	Yes (19)	Yes (19)
2. Executive Management Support	16	No (0)	Yes (16)	Yes (16)	Yes (16)
3. Clear Statement of Requirements	15	No (0)	No (0)	Yes (15)	No (0)
4. Proper Planning	11	No (0)	No (0)	Yes (11)	Yes (11)
5. Realistic Expectations	10	Yes (10)	Yes (10)	Yes (10)	Yes (10)
6. Smaller Project Milestones	9	No (0)	No (0)	Yes (9)	Yes (9)
7. Competent Staff	8	No (0)	No (0)	Yes (8)	Yes (8)
8. Ownership	6	No (0)	No (0)	Yes (6)	Yes (6)
9. Clear Vision & Objectives	3	No (0)	No (0)	Yes (3)	Yes (3)
10. Hard-Working, Focused Staff	3	No (0)	Yes (3)	Yes (3)	Yes (3)
Total	100	10	29	100	85



# Discusión en Clase

- ¿Por qué toma tanto tiempo desarrollar software?
- ¿Por qué es tan elevado su costo?
- ¿Por qué no se puede entregar programas libres de errores?
- ¿Por qué es tan costoso su mantenimiento?
- ¿Por qué resulta tan difícil constatar el progreso del desarrollo de software?
- ¿Y qué sucede con las personas que programan los sistemas, existe una sutil “anarquía” por parte de ellos?



# Conceptos asociados a la calidad

- **Política de la calidad:** intenciones y dirección de una organización relativas a la calidad tal como las expresan formalmente su alta dirección.
- **Objetivo de la calidad:** resultado a lograr en cuanto a la calidad.
- **Sistema de gestión de la calidad:** sistema de gestión para dirigir y controlar una organización con respecto a la calidad.
- **Planificación de la calidad:** parte de la gestión de la calidad enfocada al establecimiento de los objetivos de la calidad y a la especificación de los procesos operativos necesarios.
- **Control de la calidad:** parte de la gestión orientada al cumplimiento de los requisitos.
- **Aseguramiento de la calidad:** parte de la gestión de la calidad orientada a proporcionar confianza en que se cumplirán los requisitos de la calidad.
- **Mejora de la calidad:** parte de la gestión de la calidad orientada a aumentar la capacidad de cumplir con los requisitos de la calidad. La mejora continua es una actividad recurrente para mejorar el desempeño.



# Bibliografía

- La Crisis del Software, Roger S. Pressman
- Chaos Report, The Standish Group Report, Project Smart 2014
- U.S. GAO 2015 Annual Report.
- Ingeniería del software. Un enfoque práctico. R. Pressman





**Facultad de Ingeniería  
Universidad de Cuenca  
Grado en Ingeniería de Sistemas  
Curso 2020-2021**

# **Calidad de Software**

**Tema: Sistemas Híbridos**

Ing. Priscila Cedillo O. PhD.

Departamento de Ciencias de la Computación

Universidad de Cuenca, Ecuador

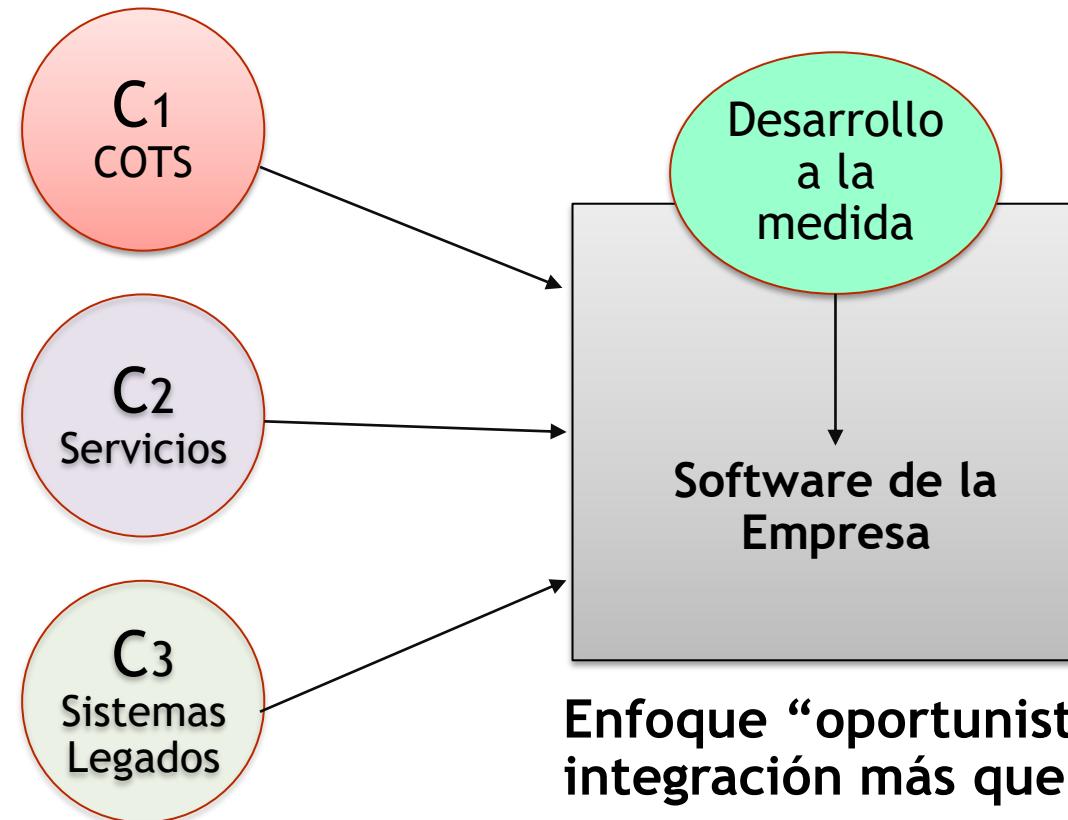
email: [priscila.cedillo@ucuenca.edu.ec](mailto:priscila.cedillo@ucuenca.edu.ec)

# Sistemas Híbridos - Generalidades

- La mayoría de sistemas de software hoy en día se pueden construir mediante la integración de componentes de diversa naturaleza.
  - Comerciales
  - Código Libre
  - Componentes
  - Legados
  - Etc.
- Así se forman arquitecturas híbridas



# Sistemas Híbridos



**Enfoque “oportunista”, orientado a la integración más que al desarrollo**

**Diversos componentes de proveedores externos a la organización que se integran con algún software hecho a medida.**



# Sistemas Híbridos

- Los componentes de software utilizados en este tipo de enfoques arquitectónicos incluyen componentes desarrollados por terceros, generalmente conocidos como componentes “*Off-The-Shelf*” (OTS)
  - Componentes comerciales (COTS)
  - Componentes gratuitos y de código abierto (FOSS)
  - Servicios Web
- Software desarrollado a la medida
- Sistemas legados



# Componentes COTS (Commercial Off-The-Shelf)

- “*Un componente es una **unidad** de composición de aplicaciones software, que posee un **conjunto de interfaces** y un conjunto de **requisitos**, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio*” *Clemens Szyperski*
- El adjetivo COTS se refiere a **un tipo particular de componente**, caracterizado por:
  - ser de índole comercial,
  - generalmente de grano grueso y
  - de bajo coste,
  - que es adquirido, seleccionado, probado, validado e integrado por desarrolladores de un sistema software basado en componentes para satisfacer ciertas necesidades del sistema, a partir de unos requisitos específicos



K. C. Wallnau, D. Carney, and B. Pollack.  
How COTS software affects the design of COTS-intensive  
systems.  
SEI Interactive, 1998.

# Beneficios del Desarrollo de Software basado en Componentes

- Reutilización de Software
- Simplifica las pruebas
- Simplifica el mantenimiento
- Mayor calidad
- Ciclos de desarrollo más cortos
- Mejor ROI
- Funcionalidad mejorada
- Independencia SW y HW



# Desventajas del Uso de Componentes

- No existe control sobre las nuevas versiones
- Costos asociados
  - Costos de licenciamiento
  - Costos de integración
  - Costos de soporte
- Licencias y propiedad intelectual
- Dependencia del fabricante
- La integración no es trivial



# Integración / Adaptación / Parametrización

- Tipos de parametrización
  - Personalización
  - Rendimiento
  - Simplemente poner el componente a trabajar
- La mayoría de componentes requieren algún tipo de parametrización:
  - MS-Office: un par de tardes
  - Sistemas ERP: varios meses

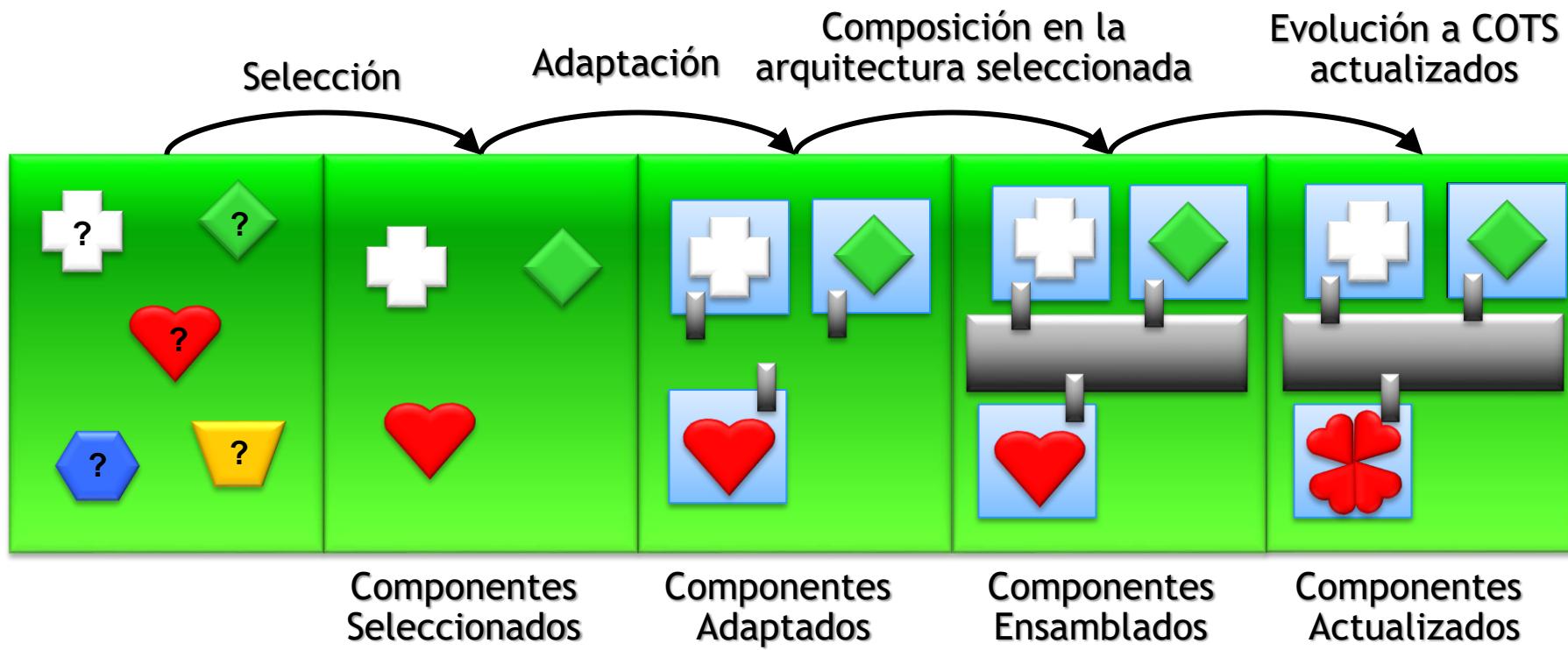


# Dificultades

- Especificación de requisitos
- Selección de componentes adecuados
- Adaptación e integración a una arquitectura común
- Identificación de las necesidades estratégicas para las cuales es demandado el software
- Agrupación de servicios relacionados en dominios atómicos que estructuran la arquitectura genérica del sistema y describen la funcionalidad mínima cubierta por cada componente



# Ciclo de Vida y Retos



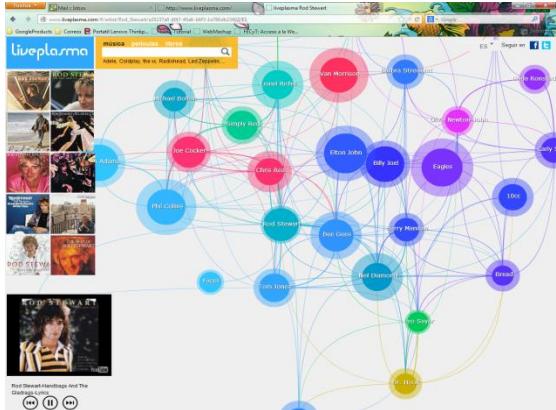
- Mercado en continua evolución:
  - Cientos de nuevos productos y actualizaciones
  - Nuevos productos no son idénticos a los anteriores
  - Reemplazo de componentes puede ser muy difícil
  - Reentrenamiento y testeо pueden ser requeridos

# Mashups

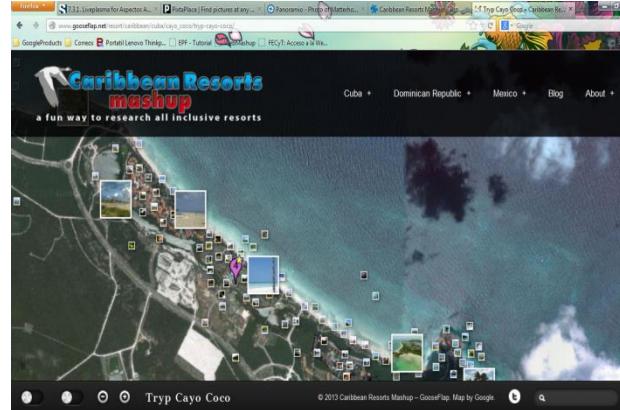
- El término mashup tiene su origen en el dominio musical refiriéndose a artistas que mezclan algunas piezas de música, usualmente desde diferentes estilos musicales, en un simple registro.
- Los mashups son aplicaciones desarrolladas para integrar contenido y funcionalidad cuya fuente es la web. Integran elementos heterogéneos disponibles en la web tales como RSS|Atom feeds, Web Services, APIs, contenido traído desde sitios de terceros o widgets (tales como Google maps).



# Aplicación del Método: Casos de Estudio



*Liveplasma*



*Caribbean Resorts*



*Skypicker*

- Se han escogido del repositorio de Programmable Web.
- Pertenecen a la lista de los Mashups más populares.
- Presentan características que permiten mostrar el método de evaluación más claramente, por la variedad de recursos que los componen.



# Ejercicio en Clase 2

- Explicar el concepto de Orquestación vs Coreografía en la composición de servicios
- Investigar 1 caso de sistema híbrido y comentar lo siguiente:
  - Servicios o componentes integrados
  - Forma de integración
  - Pros y Cons de las fuentes utilizadas
- Buscar 2 mashups y analizar cómo se ha realizado su composición



# Bibliografía

- <https://msdn.microsoft.com/es-es/library/bb972268.aspx>
- J.P. Carvallo, X. Franch, “*Descubriendo la arquitectura de sistemas de software híbridos: Un enfoque basado en modelos i\**” E-prints UPC -Universitat Politecnica de Catalunya-. Recuperada en Abril 10, 2012, del sitio Web temoa : Portal de Recursos Educativos Abiertos (REA) en <http://www.temoa.info/es/node/139907>
- Carvallo J. P., Presentaciones, 2014-2015
- L. Fuentes, J.M. Troya, A. Vallecillo, *Desarrollo de Software Basado en Componentes*, Universidad de Málaga, España en  
<http://www.lcc.uma.es/~av/Docencia/Doctorado/tema1.pdf>
- P. Cedillo, A. Fernandez, E. Insfran, S. Abrahao, *Quality of Web Mashups: A Systematic Mapping Study*, ICWE 2013 International Workshops ComposableWeb, QWE, MDWE, DMSSW, EMotions, CSE, SSN, and PhD Symposium, Aalborg, Denmark, July 8-12, 2013.  
[http://link.springer.com/chapter/10.1007/978-3-319-04244-2\\_8](http://link.springer.com/chapter/10.1007/978-3-319-04244-2_8)





**Facultad de Ingeniería  
Universidad de Cuenca  
Grado en Ingeniería de Sistemas  
Curso 2020-2021**

# **Calidad de Software**

## **Capítulo 2: Introducción a la Calidad de Software**

Ing. Priscila Cedillo O. PhD.

Departamento de Ciencias de la Computación

Universidad de Cuenca, Ecuador

email: [priscila.cedillo@ucuenca.edu.ec](mailto:priscila.cedillo@ucuenca.edu.ec)

# Contenido

- El concepto de calidad
- Tipos de calidad
- El concepto de software
- La calidad de software
- Sistema de calidad, aseguramiento de calidad
- Roles de las personas, los procesos, las herramientas y la tecnología.
- El triángulo de la calidad en el software
- La calidad del producto de software
- La calidad del proceso de elaboración del software

# Contenido

- **El concepto de calidad**
- Tipos de calidad
- El concepto de software
- La calidad de software
- Sistema de calidad, aseguramiento de calidad
- Roles de las personas, los procesos, las herramientas y la tecnología.
- El triángulo de la calidad en el software
- La calidad del producto de software
- La calidad del proceso de elaboración del software

# El Concepto de Calidad (i)

- La calidad es un concepto que ha mantenido ocupado a los filósofos por más de 2000 años.
- Las raíces de describir y definir la calidad caen en la vista de la belleza según Plato.  
*R.W.: Symposium, reprint edn. Oxford World's Classics. Oxford University Press, Oxford (1998)*
- El pensaba que la belleza causaba respuestas de amor y deseo, pero que la belleza está localizada en el mismo objeto bello.
- Sobre el tiempo los filósofos se movieron a una vista contraria, en la cual la belleza es causada por el deseo.
- De ahí la belleza significa que algo es útil.

# El Concepto de Calidad (ii)

- Las discusiones anteriores pueden ser traducidas a un uso moderno de la palabra calidad.
- Originalmente esta no contenía una noción de evaluación como buena o mala, pero en el uso contemporáneo, significa que un software es bueno al decir que es un software de calidad.

## □ **Discusión:**

- ¿Está la calidad objetivamente contenida en el software?
- ¿La calidad depende siempre de la utilidad del software para un usuario específico?

# El Concepto de Calidad (iii)

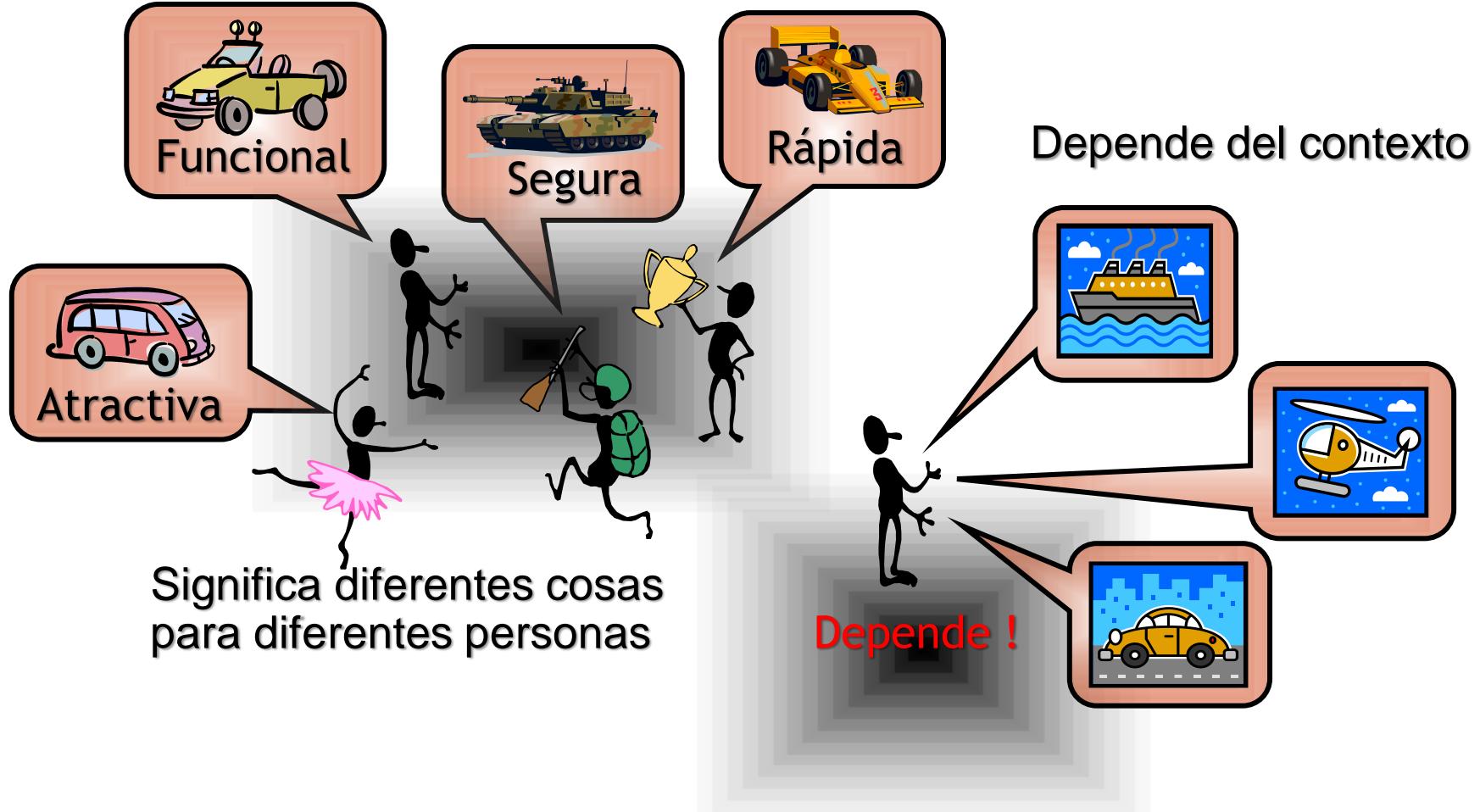


- La ISO/IEC e IEEE define la calidad de las siguientes maneras:
  - El grado con el cual un sistema, componente o proceso reúne **requisitos** específicos.
  - La habilidad de un producto, servicio, sistema, componente o proceso para reunir las **necesidades, expectativas o requisitos** de un cliente o usuario.
  - La totalidad de características de una entidad que tratan de su habilidad para satisfacer las **necesidades** implicadas.
  - La conformidad con las **expectativas** del usuario, conformidad de sus **requisitos, satisfacción** del cliente, confiabilidad y nivel de defectos presentes.
  - El grado con el cual un conjunto de características inherentes llenan las **expectativas**.
  - El grado con el cual un sistema, componente o proceso reúne las **necesidades o expectativas** del usuario.

# El Concepto de Calidad (iv)

- ❑ ¿Qué es más importante, reunir características de calidad para el usuario o para el cliente?
  - No está claro si la alta calidad significa satisfacer a la persona que usará el sistema o a la persona que pagará el sistema.
- ❑ Incluso si cumplimos con todos los requisitos explícitos, nuestro sistema no necesariamente tiene alta calidad porque no cumple con las **expectativas del usuario**.

# La Calidad es Subjetiva



Fuente Imagen: Transparencias 2014-2015, JP. Carvallo.

# Contenido

- El concepto de calidad
- Tipos de calidad
- **El concepto de software**
- La calidad de software
- Sistema de calidad, aseguramiento de calidad
- Roles de las personas, los procesos, las herramientas y la tecnología.
- El triángulo de la calidad en el software
- La calidad del producto de software
- La calidad del proceso de elaboración del software

# Concepto de Software

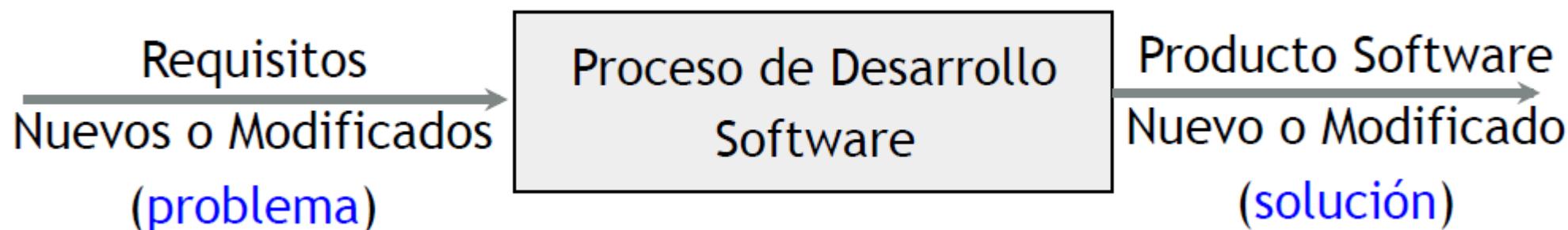
- ❑ “Instrucciones (programas de computadora) que cuando se ejecutan proporcionan la función y el rendimiento deseados”.
- ❑ “Estructuras de datos que permiten a los programas manipular adecuadamente la información”.
- ❑ “Documentos que describen la operación y el uso de programas”.
- ❑ “Es un conjunto de elementos u objetos que forman una *configuración* que incluye *Programas, Documentos y Datos*”

“Programas de computadora, procedimientos, y documentación y datos pertinentes a la operación de un sistema de cómputo”

IEEE: Standard Glossary of Software Engineering  
Terminology, 1988

# ¿Qué es un proceso de desarrollo de software?

- Definición de un conjunto de actividades cuyo objetivo es el **desarrollo** o **evolución** de un producto software.
- Actividades genéricas en todos los procesos de software.
  - Especificación: Lo que se espera que el sistema haga y las restricciones sobre ello.
  - Desarrollo: La producción misma del producto o sistema de software.
  - Validación: Comprobación con el cliente de lo que éste necesita.
  - Evolución: Cambio en respuesta de las demandas organizacionales / cliente.



# Contenido

- El concepto de calidad
- Tipos de calidad
- El concepto de software
- **La calidad de software**
- Sistema de calidad, aseguramiento de calidad
- Roles de las personas, los procesos, las herramientas y la tecnología.
- El triángulo de la calidad en el software
- La calidad del producto de software
- La calidad del proceso de elaboración del software

# La Calidad de Software

- ❑ En general se entiende que un **producto de software** posee calidad adecuada si provee **valor (satisfacción)** a los usuarios, produce una **ganancia**, genera **pocas quejas** por parte de sus clientes y contribuye de alguna manera a los **objetivos de la calidad** (o por lo menos no es opuesto). Krasner
- ❑ Calidad es el conjunto de características de un producto que **satisfacen las necesidades** de los clientes y en consecuencia, hacen satisfactorio el producto.
- ❑ Calidad consiste en no tener **deficiencias** en el producto o proceso

# La Calidad de Software

- ❑ “**Concordancia** del software producido con los **requerimientos explícitamente** establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desea el usuario” Pressman, 1998
  
- ❑ “La calidad del software es el grado con el que un sistema, componente o proceso **cumple los requerimientos especificados** y las necesidades o expectativas del cliente o usuario”. IEEE, Standard 610-1990.

# Calidad de Software

- Un producto de software ...
  - Se desarrolla, no se fabrica en el sentido clásico del mismo.
  - Se trata de un producto **lógico**, sin existencia **física**.
  - No se degrada con el uso.
  - Por la complejidad del SW y la ausencia de controles adecuados, se suele entregar el SW **conscientemente con defectos** (incluso públicamente declarados).
  - Un gran porcentaje de la producción se hace aún a **medida** en vez de emplear componentes existentes y ensamblar.
  - Es muy **flexible**. Se puede cambiar con facilidad e incluso reutilizar fragmentos.

# Trade-off

- ❑ Los atributos de calidad son frecuentemente conflictivos y aumentan los costes de desarrollo, por lo que hay una necesidad de ponderación y equilibrio (*trade-off*)





*La Ingeniería del Software tiene como objetivo el desarrollo “rentable” de software de alta calidad.*

# Contenido

- El concepto de calidad
- Tipos de calidad
- El concepto de software
- La calidad de software
- **Sistema de calidad, aseguramiento de calidad**
- Roles de las personas, los procesos, las herramientas y la tecnología.
- El triángulo de la calidad en el software
- La calidad del producto de software
- La calidad del proceso de elaboración del software

# Aseguramiento de la Calidad

## □ Definición:

- *Patrón sistemático y planificado de todas las acciones necesarias para proveer una confianza adecuada de que un ítem o producto está de acuerdo a sus requisitos técnicos.*

ISO/IEC/IEEE 24765:2010

- En consecuencia, la evaluación de la calidad incluye todas las **técnicas** que usamos **para construir productos** de una manera que incremente nuestra confianza, así como también **técnicas para analizar productos**.
- Los estándares son importantes dado que contienen la **encapsulación de las mejores prácticas**, evitan errores del pasado, son un marco para procesos y proporcionan continuidad (personal nuevo entra a laborar y entiende la organización solo conociendo los estándares que utilizan).

# Aseguramiento de la Calidad

Aseguramiento de la calidad constructiva



Todo lo usado en construir un producto de manera que reúna los requisitos de calidad.

Aseguramiento de la calidad analítica



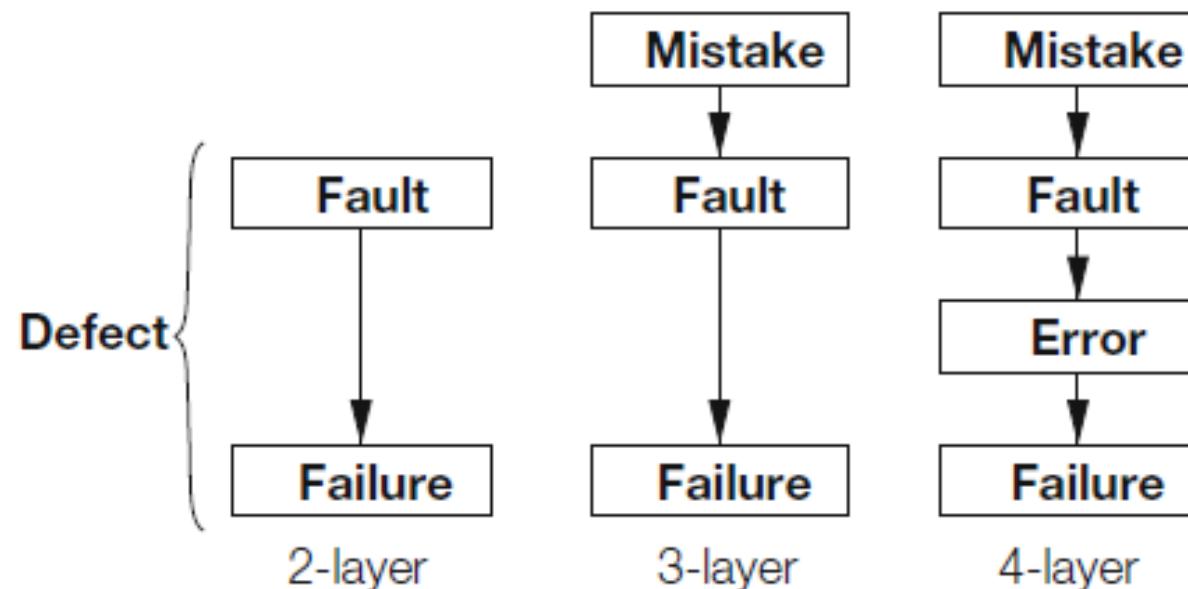
Todo lo usado para analizar la calidad del producto.

# Aseguramiento de la Calidad

- ❑ Failure-> Salida incorrecta, visible al usuario.
  - También se puede considerar a una desviación de un requerimiento.
  - Aquí es necesario incluir la expectativa del usuario.
- ❑ Fault-> Causa de un potencial fallo.
  - Conocido también como bug
  - Cuando trata de una omisión es muy difícil de detectar.
- ❑ Defect-> Los defectos son los superconjuntos de failures y faults.
- ❑ Mistake-> Acción humana que produce un fallo.

# Aseguramiento de la Calidad

- Error-> Es una parte del estado del sistema que puede ocasionar un failure



*Testing can only reveal failures while inspections find faults*

# Aseguramiento de la Calidad

- ❑ Para encontrar defectos empleamos técnicas de “**Verificación y Validación**”
- ❑ **Verificación** -> Actividad de evaluar si el resultado de un trabajo dado reúne los **requisitos especificados**.
- ❑ **Validación** -> Actividad de evaluar si el resultado de un trabajo y sus requerimientos reúnen las **expectativas** de los stakeholders.
- ❑ Análisis estático -> El proceso de evaluar un sistema o componente basado en su forma, estructura, contenido o documentación.
- ❑ Análisis dinámico -> El proceso de evaluar un sistema o componente basado en su comportamiento durante la ejecución.

# Contenido

- El concepto de calidad
- Tipos de calidad
- El concepto de software
- La calidad de software
- Sistema de calidad, aseguramiento de calidad
- **Marco de trabajo de la gestión de la calidad: roles de las personas, los procesos, las herramientas y la tecnología.**
- El triángulo de la calidad en el software
- La calidad del producto de software
- La calidad del proceso de elaboración del software

# Marco de Trabajo de la Gestión de la Calidad



- ❑ Ubicar la calidad desde una perspectiva en relación del desarrollo y adquisición de productos de software.
- ❑ Los siguientes términos y sus definiciones son provistos para establecer la base de la calidad:
  - Objeto
  - Proceso
  - Requisito / Requerimiento
  - Usuario
  - Evaluación
  - Medida y Medición
  - Calidad

# Marco de Trabajo de la Gestión de la Calidad



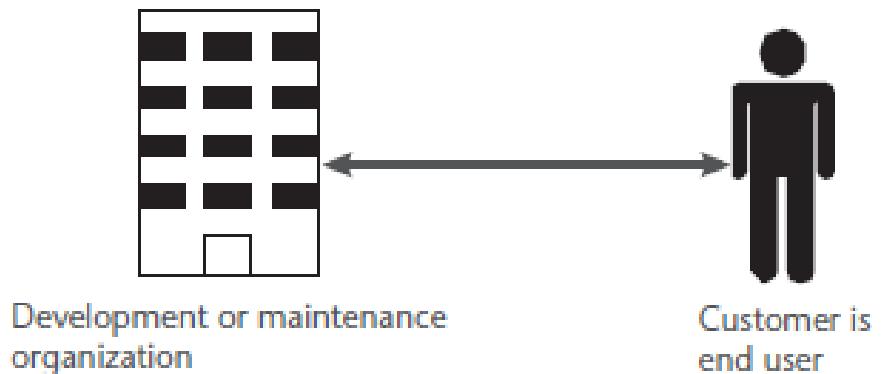
## ❑ Objeto

- Los tipos de objetos o entidades a los cuales la calidad puede ser aplicada incluyen:
  - Producto
  - Proceso
  - Servicio
  - Recurso
  - Artefacto
  - Actividad
  - Medida o métrica
  - Ambiente
  - Colección de entidades u objetos

# Marco de Trabajo de la Gestión de la Calidad

## □ Usuario

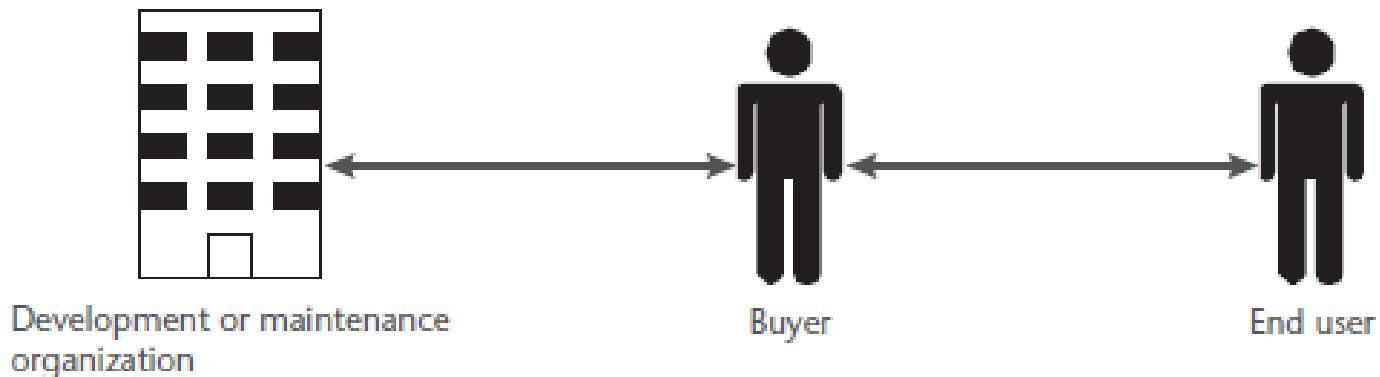
- Lo definimos como **cliente** o **usuario final**.
- Escenarios:
  - Usuario como cliente y usuario final.



# Marco de Trabajo de la Gestión de la Calidad

## □ Usuario

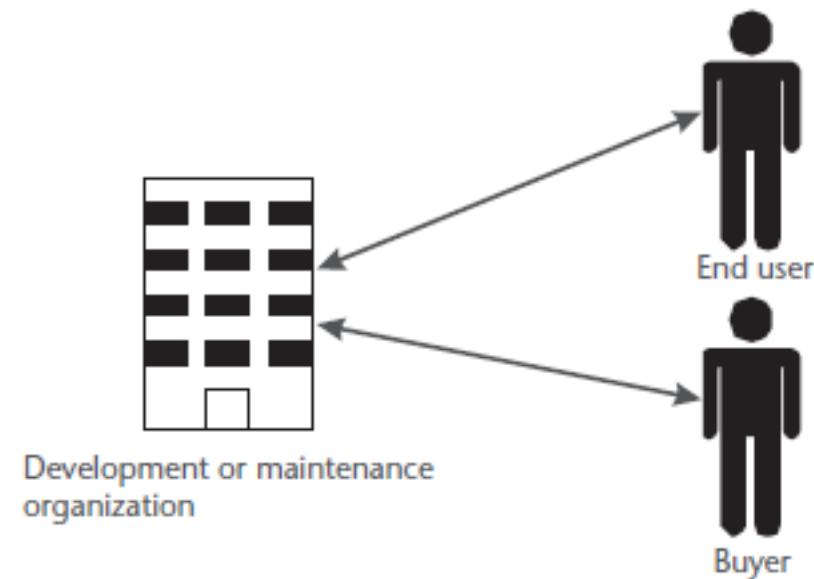
- Lo definimos como **cliente** o **usuario final**.
- Escenarios:
  - Usuario es representado por un comprador



# Marco de Trabajo de la Gestión de la Calidad

## □ Usuario

- Lo definimos como **cliente** o **usuario final**.
- Escenarios:
  - Tanto el usuario final como el comprador pueden acceder a la organización de desarrollo o mantenimiento.



# Contenido

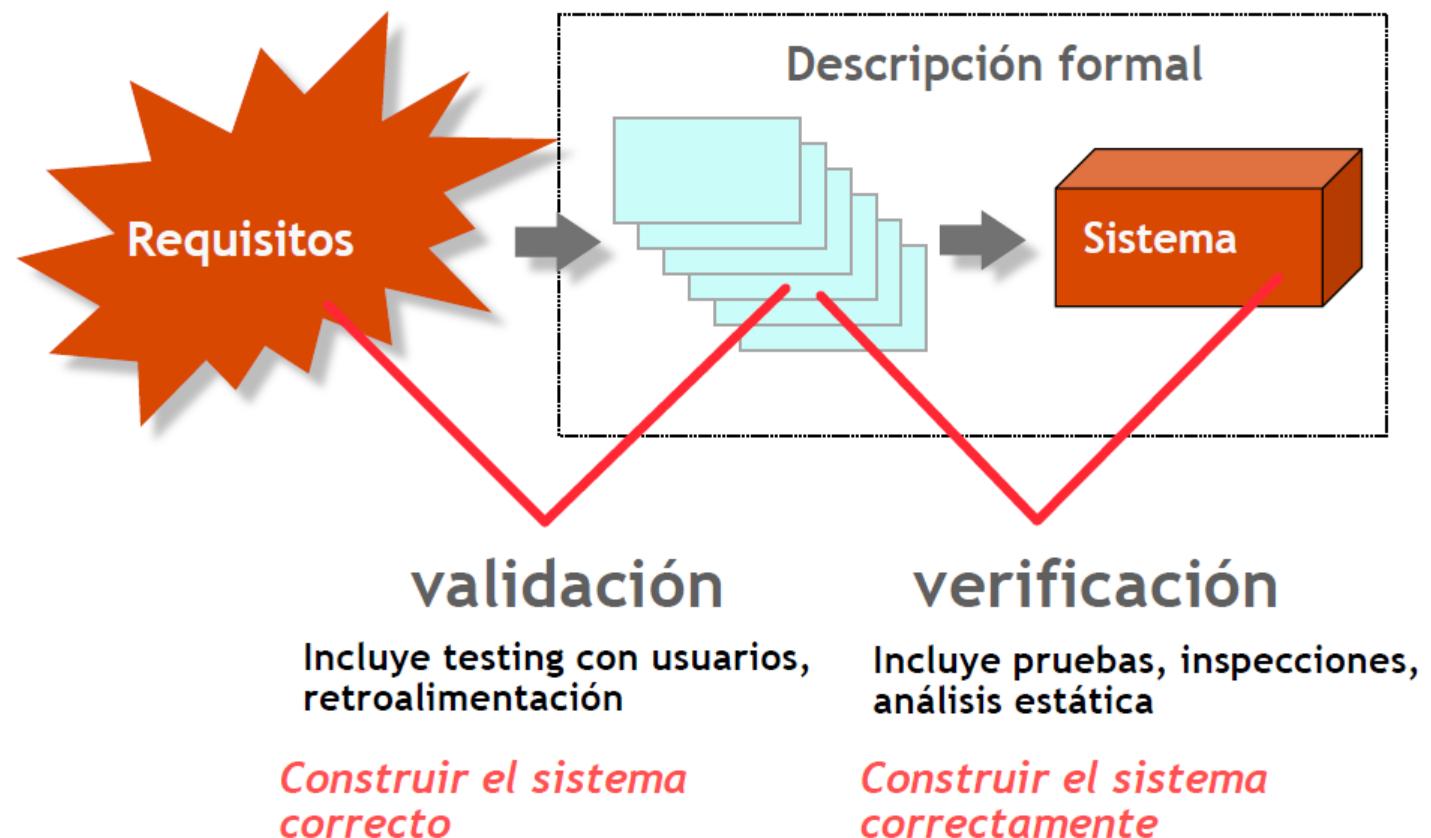
- El concepto de calidad
- **Tipos de calidad**
- El concepto de software
- La calidad de software
- Sistema de calidad, aseguramiento de calidad
- Roles de las personas, los procesos, las herramientas y la tecnología.
- El triángulo de la calidad en el software
- La calidad del producto de software
- La calidad del proceso de elaboración del software

# Tipos de Calidad

- ❑ Existen 3 tipos de calidad relacionados entre sí:
- ❑ Calidad Necesaria:
  - Calidad que pide el cliente y la que le gustaría recibir.
- ❑ Calidad Programada:
  - Es el nivel de calidad que se propone obtener el fabricante.
- ❑ Calidad Realizada:
  - Es la calidad que se puede obtener debido a las personas que realizan el trabajo o a los medios utilizados



# Validación y verificación



Fuente Imagen: Calidad de Software. S. Abrahao, UPV

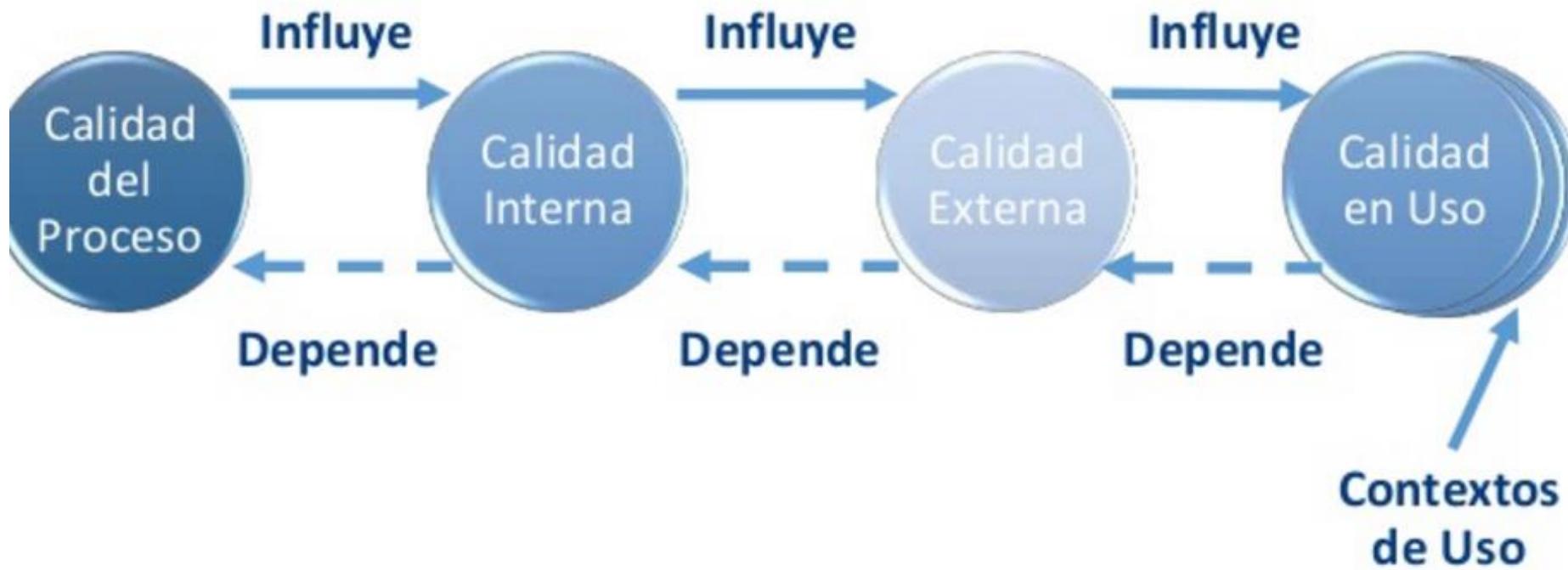
# Contenido

- El concepto de calidad
- Tipos de calidad
- El concepto de software
- La calidad de software
- Sistema de calidad, aseguramiento de calidad
- Roles de las personas, los procesos, las herramientas y la tecnología.
- **El triángulo de la calidad en el software**
- La calidad del producto de software
- La calidad del proceso de elaboración del software

## PROCESO

## PRODUCTO

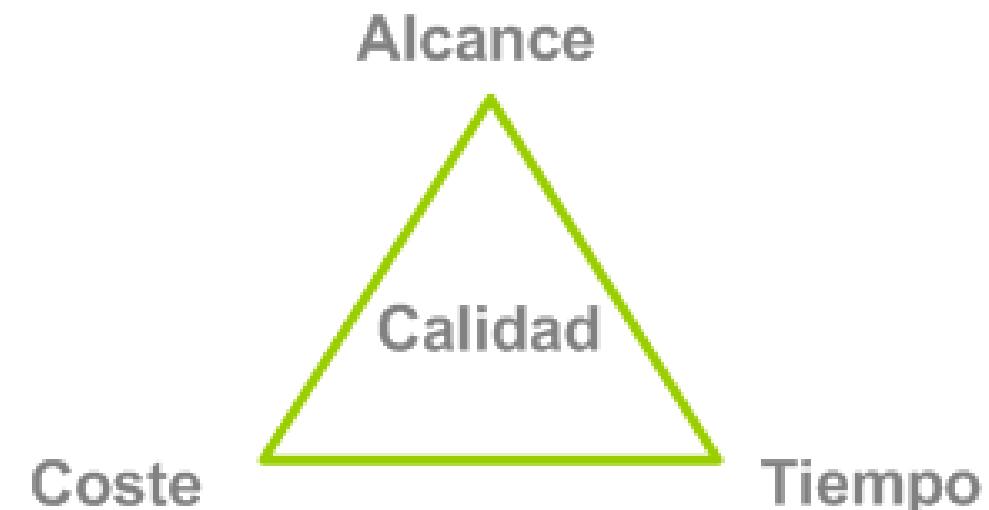
## EFECTO DEL PRODUCTO



# El Triángulo de la Calidad del Software

## ❑ El Triangulo de Hierro

- En todo proyecto existen 3 variables relacionadas, es el llamado “triangulo de hierro”
  - El **alcance**: cuántos requisitos o tareas hay que realizar
  - El **tiempo** o planificación: cuánto durará el proyecto
  - El **coste** o recursos: cuantos dinero, personas, etc



# El Triangulo de la Calidad del Software (ii)



1. Un proceso eficaz de software establece la infraestructura que da apoyo a cualquier esfuerzo de elaboración de un producto de software de alta calidad.
  - Los aspectos de administración del proceso generan las verificaciones y equilibrios que ayudan a evitar que el proyecto caiga en el caos.
  - Las prácticas de Ingeniería del Software ayudan al desarrollador analizar el problema y diseñar una solución sólida.
  - Las actividades sombrilla (administración del cambio y revisiones técnicas)
2. Un producto útil
  - Entrega contenido, funciones y características que el usuario final desea.
  - Entrega estos activos en forma confiable y libre de errores.
  - Satisface los requisitos

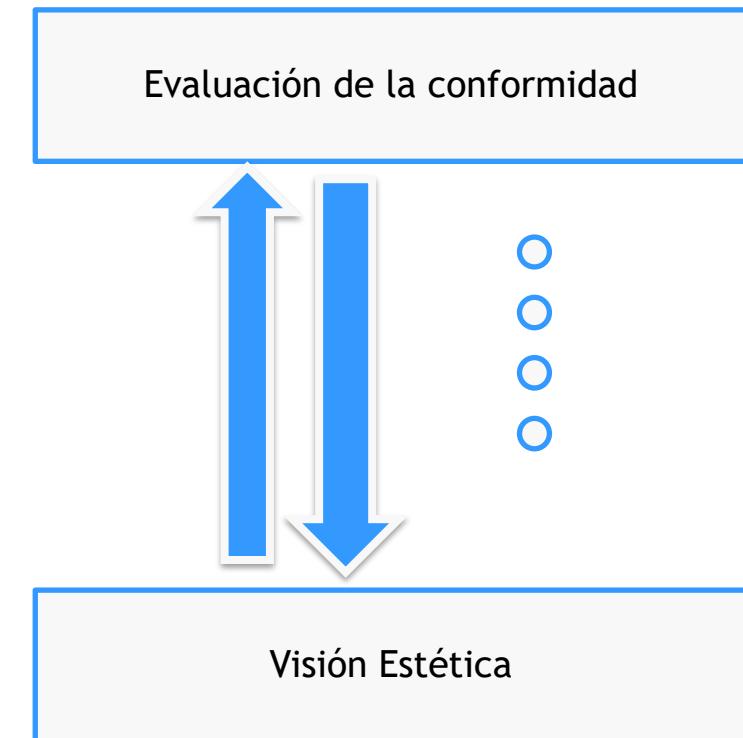
# El Triangulo de la Calidad del Software (iii)



- 3. Al agregar **valor para el productor** y para el usuario de un producto, el software de alta calidad proporciona **beneficios a la organización** que lo produce y a la comunidad de usuarios finales. La organización obtiene valor agregado porque el software de alta calidad requiere un menor esfuerzo de mantenimiento, menos errores que corregir y poca asistencia al cliente.

# Dimensiones de la Calidad de Gavin

- La calidad se aborda desde un punto de vista multidimensional.



# Dimensiones de Calidad de Garvin

Calidad del Desempeño



¿El software entrega todo el contenido, las funciones y las características especificadas como parte del modelo de requerimientos, de manera que da valor al usuario final?

Calidad de las Características



¿El software tiene características que sorprenden y agradan la primera vez que lo emplean los usuarios finales?

Confiabilidad



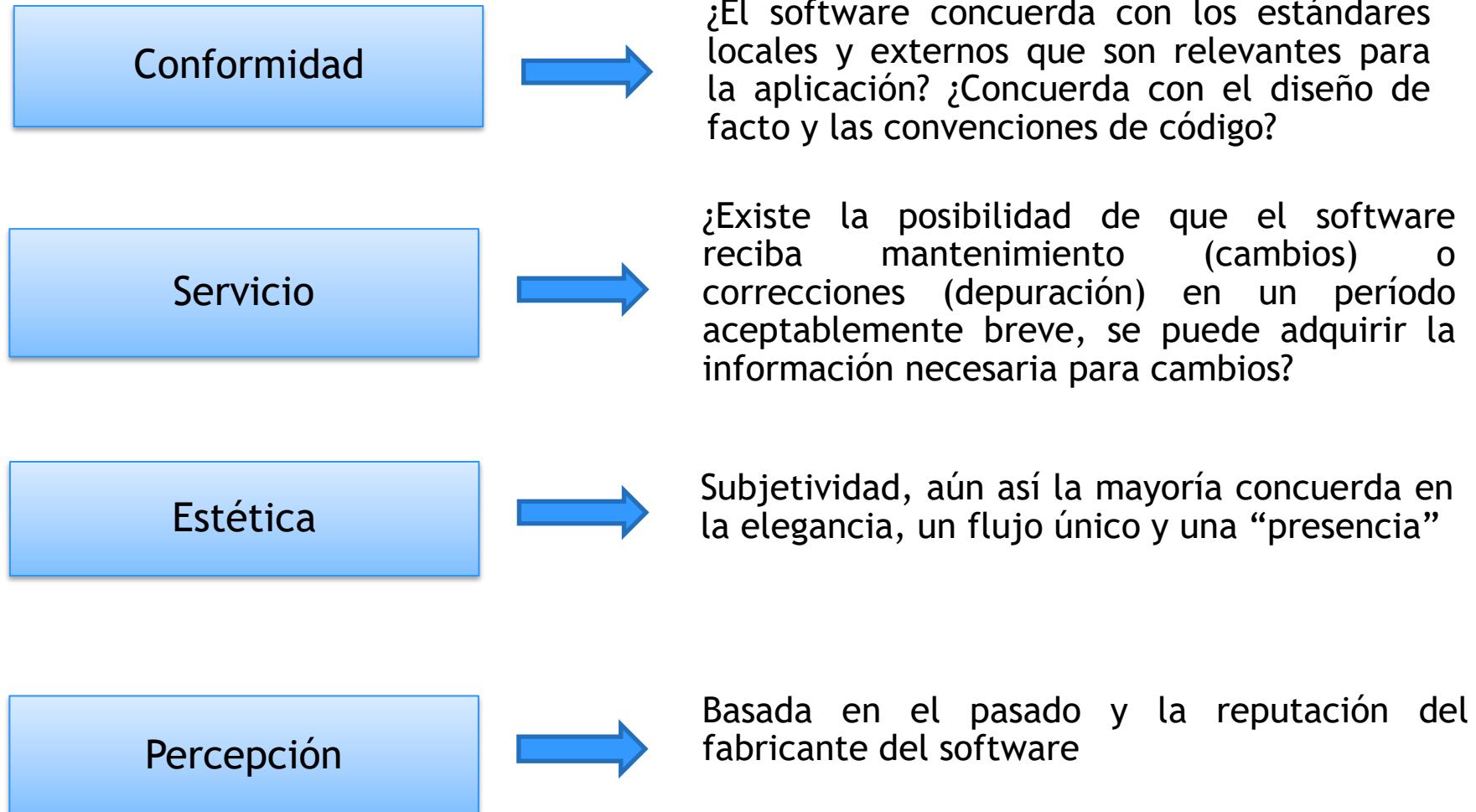
¿El software proporciona todas las características y capacidades sin fallar? ¿Está disponible cuando se necesita? ¿Entrega la funcionalidad libre de errores?

Durabilidad



¿El software puede recibir mantenimiento (cambiar) o corregirse (depurarse) sin la generación inadvertida de eventos colaterales? ¿Los cambios ocasionarán que la tasa de errores o la confiabilidad disminuyan con el tiempo?

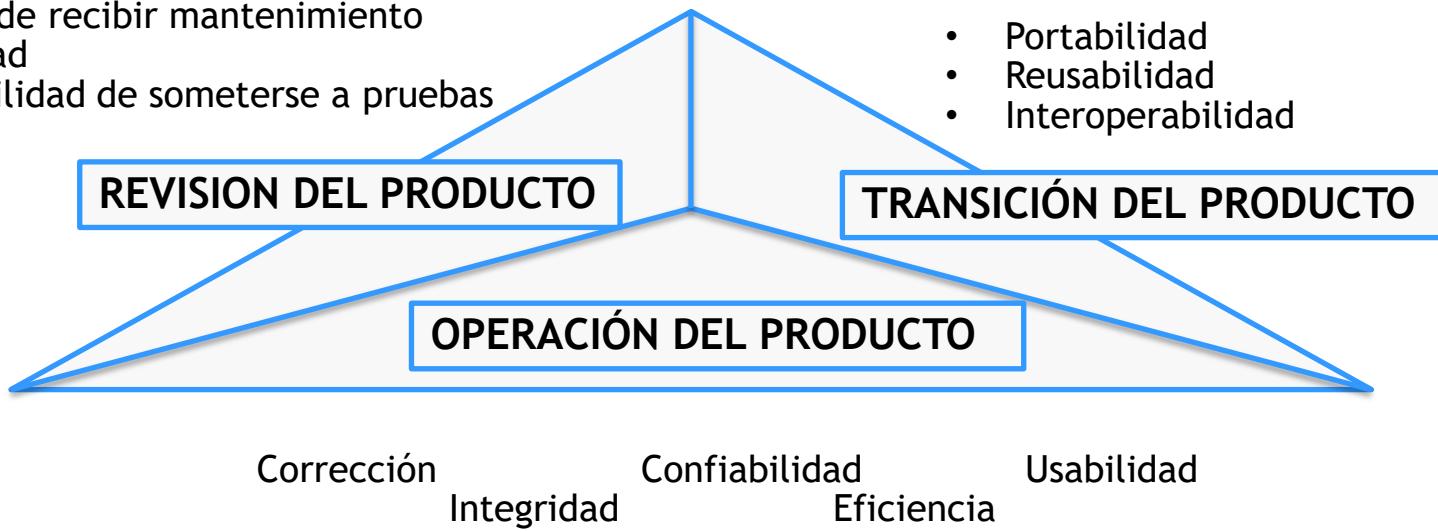
# Dimensiones de Calidad de Garvin



# Factores de Calidad de McCall

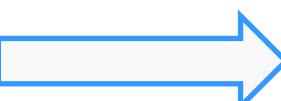
- Facilidad de recibir mantenimiento
- Flexibilidad
- Susceptibilidad de someterse a pruebas

- Portabilidad
- Reusabilidad
- Interoperabilidad



- McCall, Richards y Walters proponen una clasificación útil de los factores que afectan la calidad del software.

# Factores de Calidad de McCall

- |                   |  |   |
|-------------------|--|---|
| Corrección        |    | Grado en el que un programa satisface sus especificaciones y en el que cumple con los objetivos de la misión del cliente. |
| Confiabilidad     |    | Grado en el que se espera que un programa cumpla con su función y con la precisión requerida.                             |
| Eficiencia        |    | Cantidad de recursos de cómputo y de código requeridos por un programa para llevar a cabo su función.                     |
| Integridad        |   | Grado en el que es posible controlar el acceso de personas no autorizadas al software o a los datos                       |
| Interoperabilidad |  | Esfuerzo requerido para acoplar un sistema con otro   |

# Factores de Calidad de McCall

Usabilidad	→	Esfuerzo que se requiere para aprender, operar, preparar las entradas e interpretar las salidas de un programa
Facilidad de recibir mantenimiento	→	Esfuerzo requerido para detectar y corregir un error de un programa (concepto limitado)
Flexibilidad	→	Esfuerzo necesario para modificar un programa que ya opera
Susceptibilidad de someterse a pruebas	→	Esfuerzo que se requiere para probar un programa a fin de garantizar que realiza la función que se pretende
Portabilidad	→	Esfuerzo que se necesita para transferir el programa de un ambiente de sistema de hardware o software a otro
Reusabilidad	→	Grado con el que un programa (o partes de uno) pueden volverse a utilizar en otras aplicaciones

# Contenido

- El concepto de calidad
- Tipos de calidad
- El concepto de software
- La calidad de software
- Sistema de calidad, aseguramiento de calidad
- Roles de las personas, los procesos, las herramientas y la tecnología.
- El triángulo de la calidad en el software
- **La calidad del producto de software**
- La calidad del proceso de elaboración del software

# Calidad del Producto de Software

- ❑ La calidad el producto significa que el producto debe reunir los **requisitos de su especificación**.
  - El software debe ser entregado con la funcionalidad requerida (functional requirements) con los atributos de calidad requeridos (non-functional requirements)
- ❑ Puede existir tensión entre los requisitos **de calidad de los clientes** (eficiencia, confiabilidad, ...) y **los requisitos del desarrollador** (mantenibilidad, reusabilidad)
- ❑ Las especificaciones de software a menudo son incompletas e inconsistentes.
- ❑ Los atributos de calidad están frecuentemente en conflicto e incrementan los costos

# Calidad de Producto de Software

- ❑ Los factores que intervienen de una u otra manera en la calidad del producto son:
  - La calidad del proceso
    - Un buen proceso es usualmente produce un buen producto
    - En la manufactura de productos es vital un buen proceso
  - Calidad del personal
    - Para pequeños proyectos, las capacidades de los desarrolladores son determinantes.
    - Tamaño del proyecto x Calidad de personal = Constante?
  - Desarrollo de tecnología
    - Es particularmente significativo para proyectos pequeños
  - Presupuesto y calendario
    - En todos los proyectos si una planificación de tiempo es no realista, la calidad del producto sufre sustancialmente.

# Contenido

- El concepto de calidad
- Tipos de calidad
- El concepto de software
- La calidad de software
- Sistema de calidad, aseguramiento de calidad
- Roles de las personas, los procesos, las herramientas y la tecnología.
- El triángulo de la calidad en el software
- La calidad del producto de software
- **La calidad del proceso de elaboración del software**

# Calidad de Proceso

- Hay **poca evidencia** en que cumplir un modelo de procesos asegure la calidad del producto, la estandarización de los procesos garantiza la **uniformidad en la salida de los mismos** lo que puede incluso **institucionalizar la creación de malos productos.**

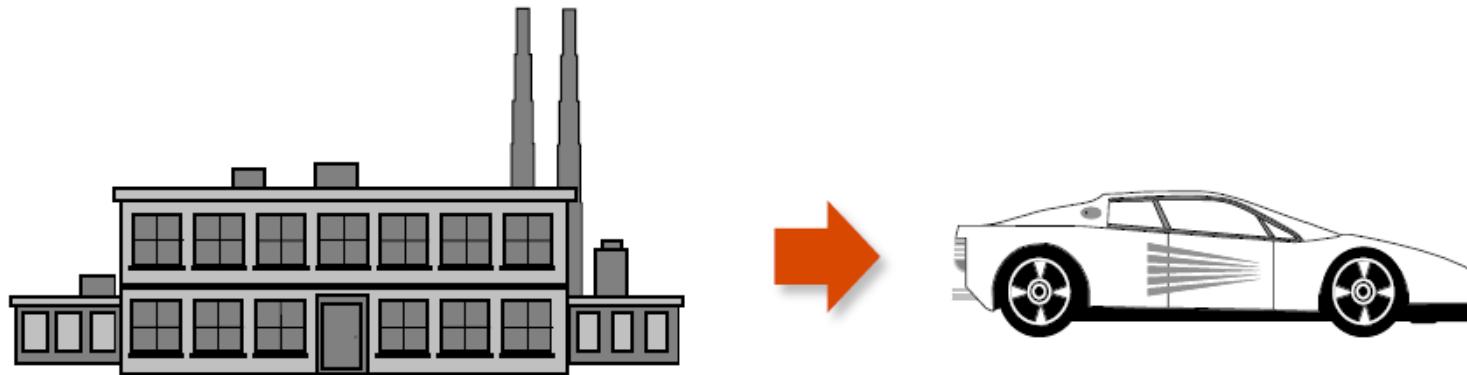
- Kitchenham, B. y Pfleeger, S. L. (1996). "Software Quality: The Elusive Target." IEEE Software 20(1): 12-21.

# Calidad de Proceso (ii)

- ❑ Se puede ver como una colección estructurada de prácticas que describen las características de un proceso efectivo.
- ❑ Se usa para:
  - Definir las prioridades y objetivos de mejora
  - Guía para la mejora
  - Definir un lenguaje común.
- ❑ Áreas del proceso relacionadas con la gestión de la calidad
  - Aseguramiento de la calidad en el proceso y en el producto
  - Verificación del proceso
  - Validación del proceso

"El proceso de Software define como se organiza, gestiona, mide, soporta y mejora el desarrollo, independientemente de las técnicas y métodos usados.  
(Derniame et al, 1999)"

# Calidad de Proceso vs. Calidad de Producto



## Calidad de Proceso

- El objetivo es introducir características de propósito especial para controlar:
  - Coste
  - Tiempo de Entrega
  - Calidad
  - Competitividad

## Calidad de Producto

- Se centra en evaluar la calidad de los productos sin importar el proceso detrás de ellos.
- Los productos software deben ser capaces de satisfacer los **requisitos explícitos e implícitos de los clientes**.

Fuente Imagen: Calidad de Software. S. Abrahao, UPV

# Factores que afectan la calidad de productos (1)

Tecnologías de Desarrollo

Proceso de Desarrollo de Software

Calidad de proceso

Calidad de producto

Calidad de personas

Coste, tiempo y calendario

Fuente Imagen: Calidad de Software. S. Abrahao, UPV

# Bibliografía

- ❑ Galin D., Software Quality Assurance From theory to implementation, 2004
- ❑ Chappell, D. (2012). THE THREE ASPECTS OF SOFTWARE QUALITY : FUNCTIONAL , STRUCTURAL , AND PROCESS Sponsored by Microsoft Corporation. *David Chappel & Associates, 1.0.* Retrieved from [http://www.davidchappell.com/writing/white\\_papers/The\\_Three\\_Aspects\\_of\\_Software\\_Quality\\_v1.0-Chappell.pdf](http://www.davidchappell.com/writing/white_papers/The_Three_Aspects_of_Software_Quality_v1.0-Chappell.pdf)
- ❑ O'Regan, G. (2014). *Introduction to Software Quality*. <http://doi.org/10.1007/978-3-319-06106-1>
- ❑ Software, D., Rosa, V., & Zepeda, V. (2012). Metodología para el Aseguramiento de la Calidad en la Adquisición del Software ( proceso y producto ) y servicios.
- ❑ Wagner, S. (2013). *Software Product Quality Control*. <http://doi.org/10.1007/978-3-642-38571-1>
- ❑ Carvallo J. P., Presentaciones, 2014-2015.



UNIVERSIDAD DE CUENCA

Facultad de Ingeniería  
Universidad de Cuenca  
Grado en Ingeniería de Sistemas  
Curso 2020

# Calidad de Software

## Capítulo 3: Modelos de Calidad de Software

Departamento de Ciencias de la Computación

Universidad de Cuenca, Ecuador

email: [priscila.cedillo@ucuenca.edu.ec](mailto:priscila.cedillo@ucuenca.edu.ec)

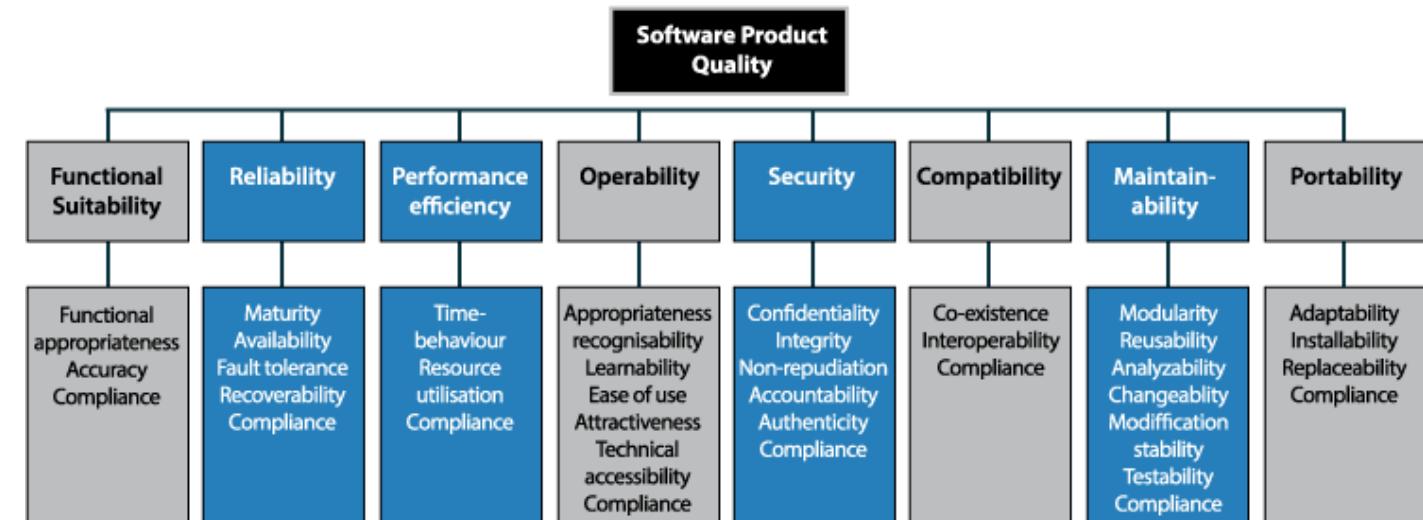
# Contenido



- **Introducción**
- Que es un modelo de calidad del software
- Estructura de los modelos de calidad del software
- Tipos de modelos de calidad
- Estándares de modelos de calidad del software
- Aplicaciones de los modelos de calidad del software

# Introducción

- Los modelos de calidad han sido un tópico de investigación durante algunas décadas.
- Los modelos de calidad son medios aceptados para **soportar el control de calidad** de los sistemas de software.
- El ISO/IEC 25010 es usado principalmente para definir la calidad, utilizado para **evaluar la calidad** de un sistema dado.



# Ventajas de los Modelos de Calidad

- Corregir los procesos de software.
- Certificar la competitividad internacional requerida para competir en los mercados.
- Cambiar la actitud del personal de la empresa.
- Desarrollar y mejorar el nivel del personal.
- Lograr competitividad en una empresa de software.
- Reducir los costos en los procesos.
- Asegurar la satisfacción de los clientes.
- Tener productos de software con un valor agregado.
- Tener aceptación de los clientes.



# Contenido



- Introducción
- Que es un modelo de calidad del software
- Estructura de los modelos de calidad del software
- Tipos de modelos de calidad
- Estándares de modelos de calidad del software
- Aplicaciones de los modelos de calidad del software

# Definición



- Un modelo de calidad es:
  - “El **conjunto de características** y las relaciones entre ellas que proveen la base para la especificación de los requisitos de calidad y la evaluación de la calidad.”
- Los modelos de calidad permiten:
  - Definición estructurada de criterios de evaluación
  - Especificación de requisitos con relación a ellos
  - Descripción de componentes en un marco común
  - Definición de métricas y prioridades

ISO/IEC 8402.



# Métricas - Concepto

- Las **métricas** del producto se dividen en dos clases:
  - Las **métricas dinámicas**, que son recogidas por las mediciones hechas en un programa en ejecución.
  - Las **métricas estáticas**, que son recogidas por las mediciones hechas en las representaciones del sistema como el diseño, el programa o la documentación.
- Las **métricas** del software se pueden clasificar en **MEDIDAS DIRECTAS e INDIRECTAS**.
  - **Directas**: Una métrica de un atributo que no depende de ninguna métrica de otro atributo.
  - **Indirecta**: Se deriva de una o más métricas de otros atributos.

# Ejemplo de métricas directas

- **Longitud del Texto del Cuerpo de una Página**
  - Medido por cantidad de palabras, etc.
- **Cantidad de Enlaces Rotos Internos**
  - Medidos por la presencia de errores del tipo 404, (410 ?)
- **Cantidad de Imágenes con Texto Alternativo**
  - Medido por la presencia de la etiqueta ALT (con texto no nulo) en cada una de las imágenes vinculadas a las páginas de un sitio Web

# Ejemplo de métricas indirectas

## ⭐ Porcentaje de Enlaces Rotos de un Sitio

$$PorcentajeEnlacesRotos = \frac{CantidadEnlacesRotosInternos + CantidadEnlacesRotosExternos}{CantidadTotalEnlaces} \times 100$$

## ⭐ Porcentaje de Presencia de la propiedad ALT.

$$PorcentajePresenciaALT = \frac{CantidadImágenesALT}{CantidadTotalImágenes} \times 100$$

# Contenido



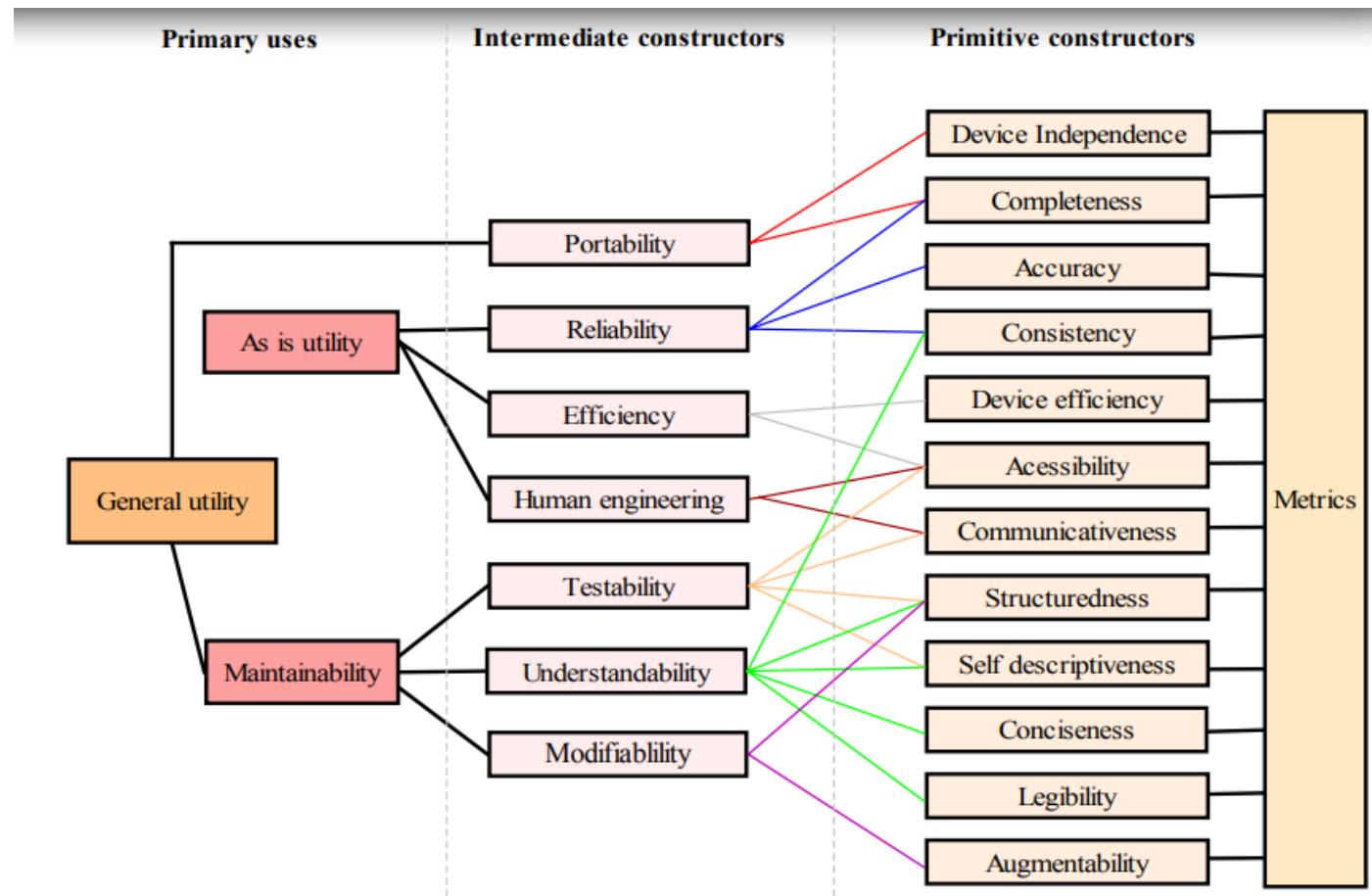
- Introducción
- Que es un modelo de calidad del software
- **Estructura de los modelos de calidad del software**
- Tipos de modelos de calidad
- Estándares de modelos de calidad del software
- Aplicaciones de los modelos de calidad del software



# Modelos de Calidad: Estructura

- Todos los modelos de calidad comparten:
  - Un **catalogo de factores de calidad** (fijo? desecharable?)
  - Diferentes **niveles de abstracción** (Número de capas? jerarquía? grafo?)
- Algunos autores recomiendan su descripción en forma de un modelo conceptual que describa:
  - La forma del modelo
  - Propiedades de las métricas
  - Elementos medibles
  - Aspectos de formalización (definiciones)

# Modelos de Calidad (Boehm - 1978)



# Contenido



- Introducción
- Que es un modelo de calidad del software
- Estructura de los modelos de calidad del software
- **Tipos de modelos de calidad**
- Estándares de modelos de calidad del software
- Aplicaciones de los modelos de calidad del software

# Tipos de Modelos de Calidad

- Existen algunos tipos de modelos de calidad:
  - Modelos de Calidad Jerárquicos
  - Modelos de Calidad Basados en Meta-Modelos
  - Modelos de Calidad Implícitos



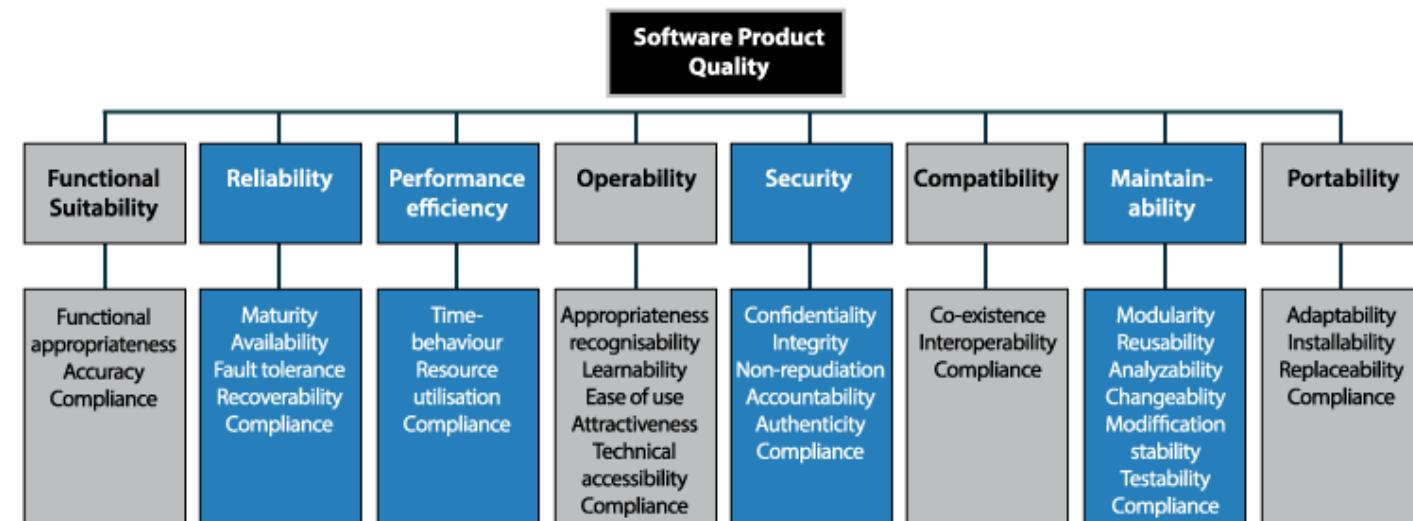


# Modelos de Calidad Jerárquicos

- El primero publicado fue en 1970.
- Usan una descomposición Jerárquica en factores de calidad (Mantenibilidad, Confiabilidad)
- Uno de los más populares es el modelo FURPS
  - Funcionalidad - Functionality
  - Usabilidad - Usability
  - Confiabilidad - Reliability
  - Rendimiento - Performance
  - Soporte - Supportability
- La principal idea es que se pueda descomponer la calidad a un nivel donde ésta pueda ser medida y de ahí evaluada.

# Modelos de Calidad Jerárquicos

- Esta clase de modelos trajeron las bases para el estándar ISO / IEC 9126 en 1991.
- Define un estándar de descomposición en características de calidad y sugiere un pequeño número de métricas para medirlas.
- Le sigue el ISO/IEC 25010, mantiene una nueva clasificación pero guarda la descomposición jerárquica general.



# Modelo de Calidad FURPS

- FURPS es un modelo de **definición jerárquica**.
- Los primeros cuatro factores de calidad son **dirigidos al operador** y al usuario del software.
- Los últimos son más dirigidos a **los desarrolladores, testers y gente de mantenimiento**.
- El principal objetivo de FURPS **es una descomposición y checklist** para requisitos de calidad.
- Ayuda a **definir la calidad** como base para los requisitos.

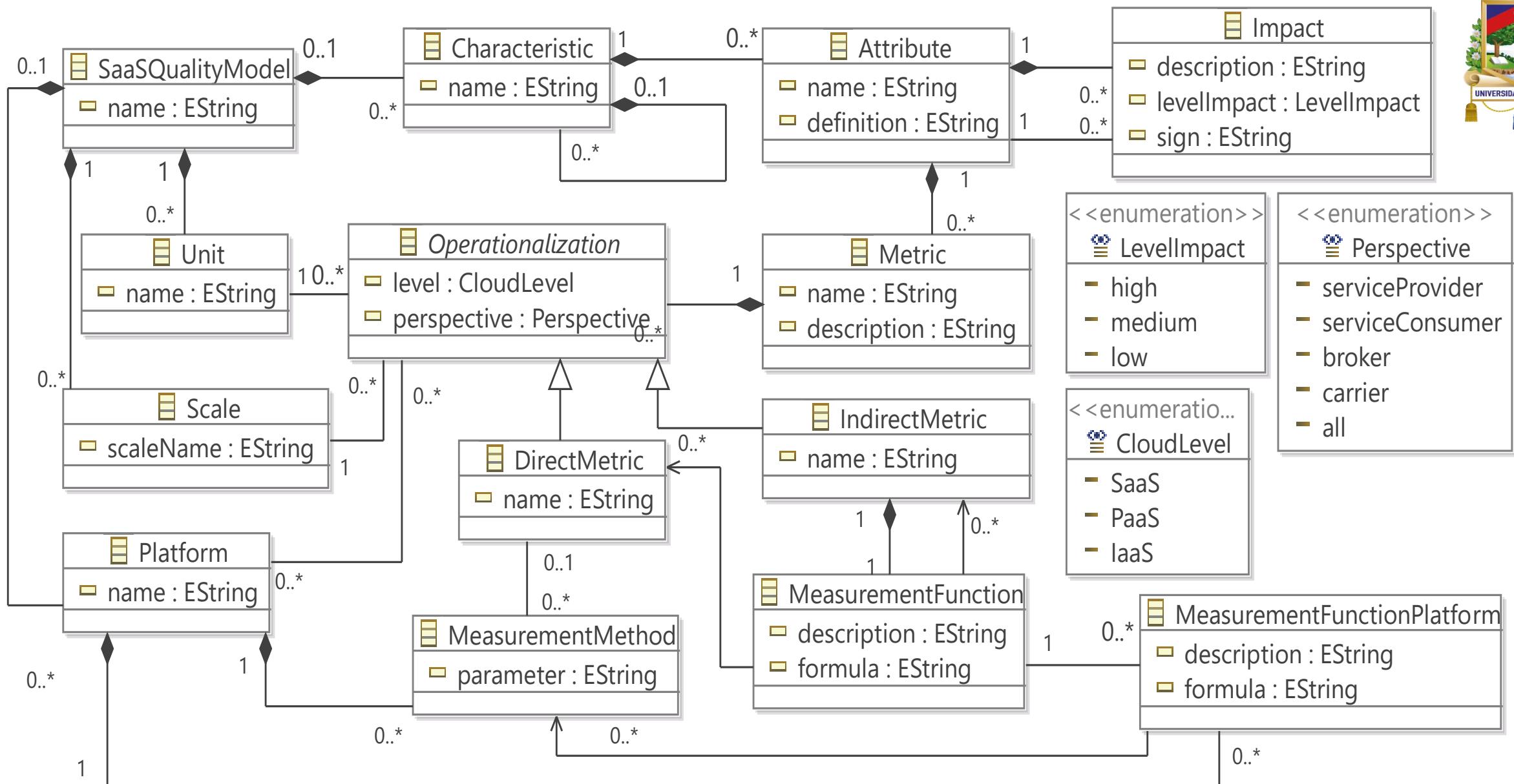
n FURPS se descompone en:

- Functionality
  - Feature set
  - Capabilities
  - Generality
  - Security
- Usability
  - Human factors
  - Aesthetics
  - Consistency
  - Documentation
- Reliability
  - Frequency/severity of failure
  - Recoverability
  - Predictability
  - Accuracy
  - Mean time to failure
- Performance
  - Speed
  - Efficiency
  - Resource consumption
  - Throughput
  - Response time
- Supportability
  - Testability
  - Extensibility
  - Adaptability
  - Maintainability
  - Compatibility
  - Configurability
  - Serviceability
  - Installability
  - Localisability
  - Portability

# Modelos de Calidad Basados en Meta-modelos



- Inicios de los 90's, los investigadores han propuesto modos más elaborados de descomponer las características de calidad.
- Describen cómo modelos de calidad válidos son estructurados.
- Incluyen mediciones y evaluaciones.
- Los modelos de calidad basados en meta-modelos muestran el concepto complejo de las necesidades de calidad más estructuras en modelos de calidad que abstraen características y métricas.

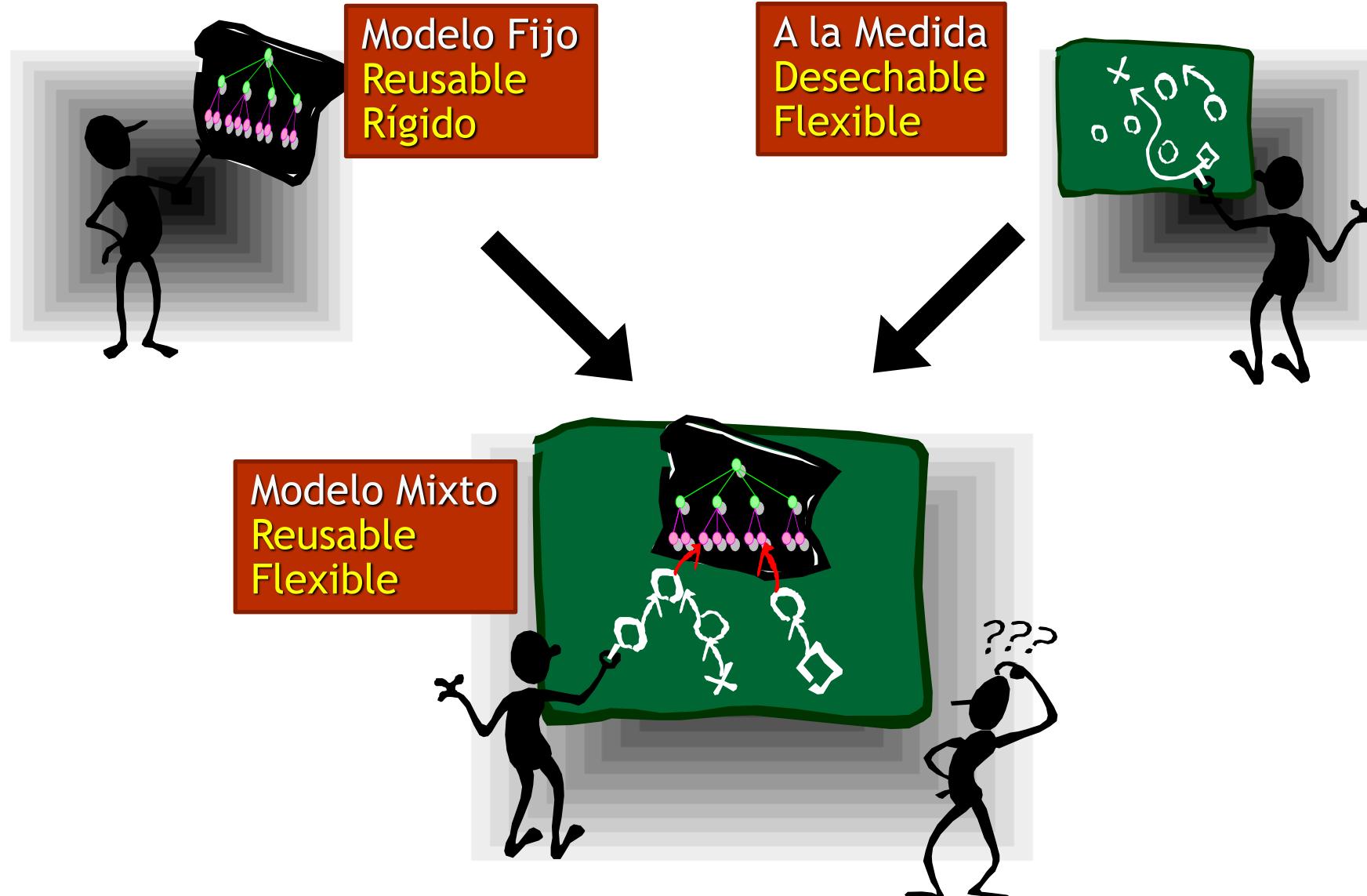




# Modelos de Calidad Estadísticos e Implícitos

- Capturan las propiedades del producto, proceso u organización.
- Estiman y predicen esos factores de calidad.
- Un ejemplo de esos modelos son los *reliability growth models* o los *Maintainability index (MI)*, un modelo de regresión desde las métricas de código o *Vulture*, un modelo de aprendizaje de máquina basado en bases de datos de vulnerabilidad y archivos de versión.

# Tipos de modelos de calidad





# Tipos de modelos de calidad

## n Modelos fijos:

- Existe un catalogo de partida del cual se elige un subset de características de calidad
- Pros: reutilizable, comparable, rápido de utilizar
- Contras: inflexible
- Ejm: Modelo de McCall, Boehm, FURPS

## n Modelos a la medida :

- Determinación de factores de calidad basada en necesidades del contexto
- Pros, contras: Lo contrario del caso anterior
- IEEE 1061(1998), Goal Question Metric (GQM)

## n Modelos mixtos:

- Un modelo de alto nivel que puede ser refinado
- Pros, contras: balanceados



# Modelos de Definición

- Son usados en varias fases de un proceso de desarrollo de software.
  - **Durante la Ingeniería de Requisitos:** Definen factores de calidad y requisitos para sistemas de software. Constituyen un método para acordar con el cliente la calidad.
  - **Durante la Implementación:** Sirven como base para modelar y codificar. Proveen recomendaciones directas sobre la implementación y constituyen enfoques constructivos para conseguir alta calidad de software.
- Enfoques constructivos para conseguir alta calidad en el software.
- Los defectos de calidad que son encontrados durante el aseguramiento de la calidad son clasificados usando el modelo de calidad.



# Modelos de Evaluación

- Extienden los modelos de definición.
- Evalúa la calidad del modelo de definición
- Los modelos de evaluación pueden ser usados durante la ingeniería de requisitos para especificar y controlar los requisitos de calidad.
- Durante la implementación este modelo de calidad puede ser la base para las mediciones (medición de producto, actividades y ambiente).
- Constituyen la piedra angular para las certificaciones de calidad.
- Ejemplo: EMISQ (modelo basado en el estándar 14598 para evaluación de producto).

# Modelos Predictivos



- Sirven para predecir el número de defectos de un sistema o módulos específicos, tiempos medios entre fallos, tiempo de reparación y esfuerzos de mantenimiento.
- Ejemplo: Modelo RGMs emplean detección de defectos desde las fases de prueba y operación para predecir la futura confiabilidad de los sistemas de software.



# Modelos Multi-Propósito

- Modelos de calidad que integran los 3 propósitos.
- Su ventaja es que se evalúa y predice en el mismo modelo los requisitos de calidad.
- Asegura una alta consistencia.
- Ejemplo: COQUAMO

# Contenido



- . Introducción
- . Que es un modelo de calidad del software
- . Estructura de los modelos de calidad del software
- . Tipos de modelos de calidad
- . **Estándares de modelos de calidad del software**
- . Aplicaciones de los modelos de calidad del software



# Modelos y Estándares de Calidad

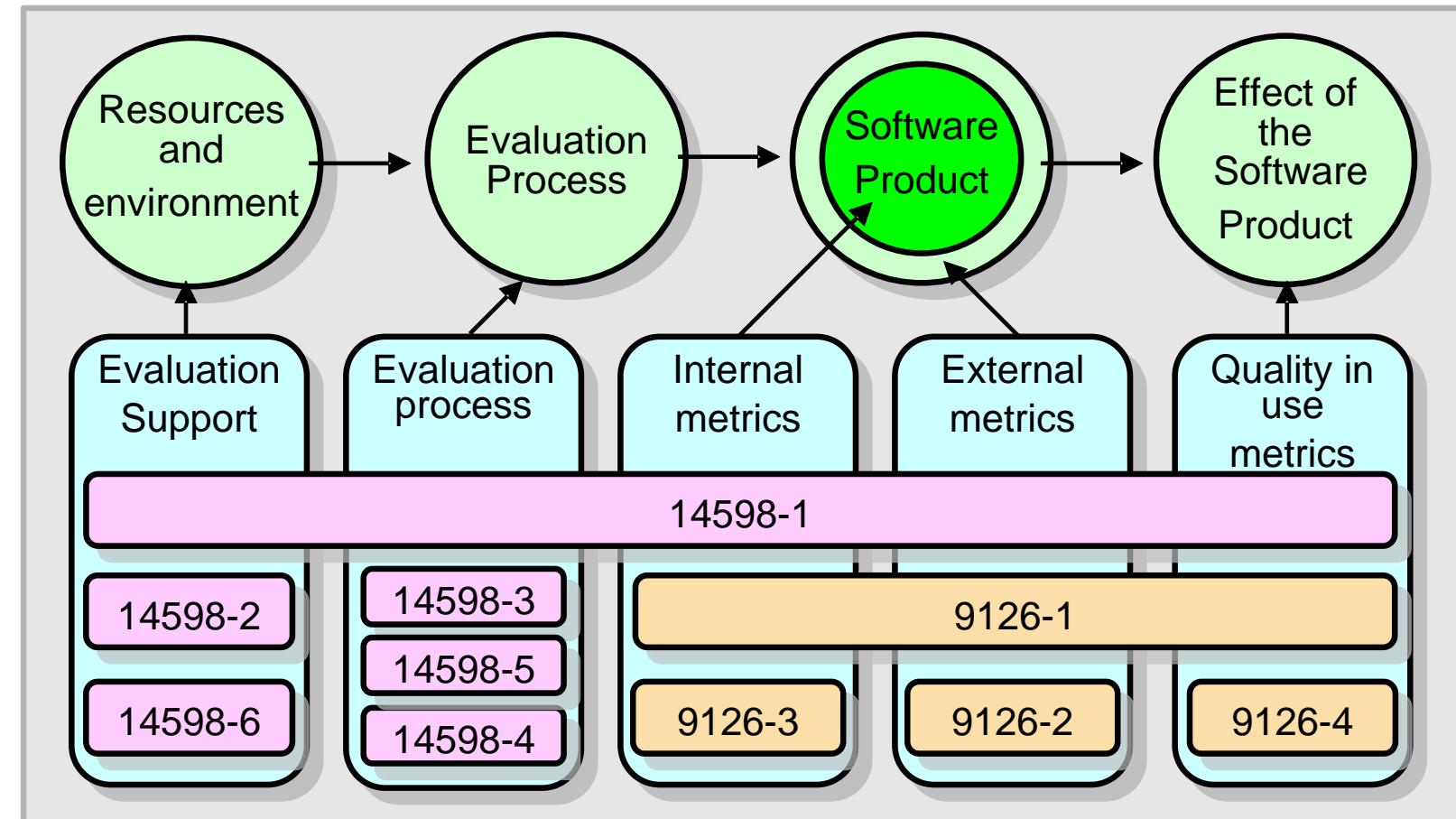
Nivel de Calidad	Modelo de Calidad	Estándar de Calidad
Proceso	CMMi, TickIT, Bootstrap, Personal SW Process (PSP), Team SW Process (TSP), Practical SW Measurement (PSM), Six Sigma for Software	ISO 90003, ISO 12207, ISO 15504 (SPICE), IEEE/EIA 12207, ISO 20000, ITIL, Cobit 4.0
Producto	Gilb, GQM, McCall, Furps, Bohem, SATC, Dromey, C-QM, Metodología SQA, Web EQM	ISO 9126-1, ISO 25000 (SQuaRE), IEEE Std 1061-1998



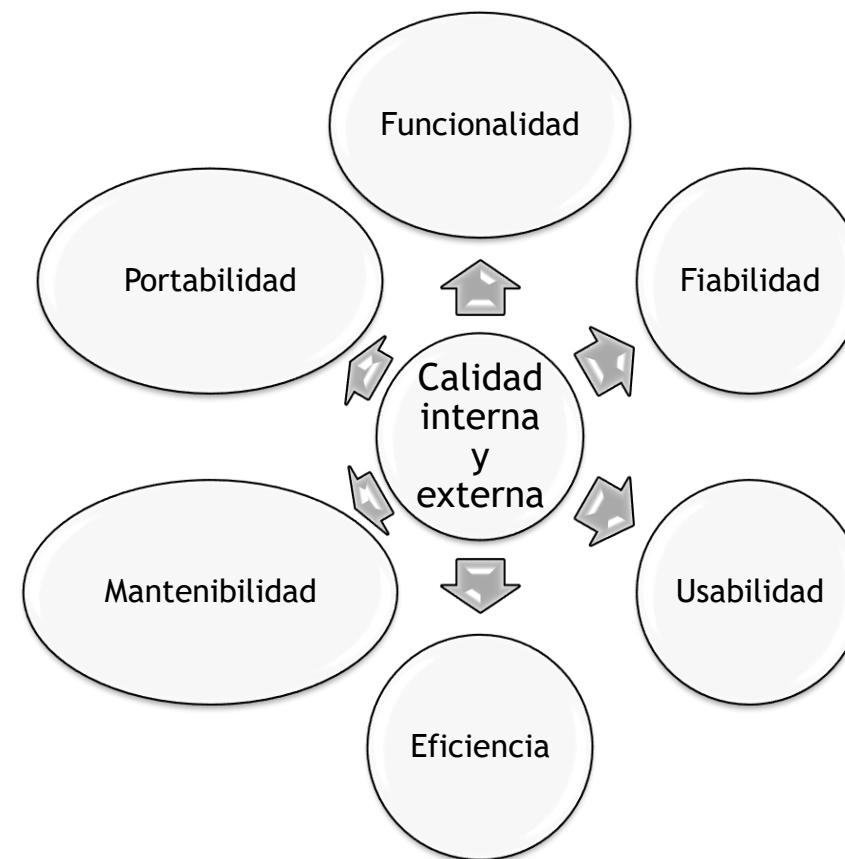
## Estándares de modelos de calidad: ISO/IEC 9126

- Modelo mixto con un catalogo de partida mas elaborado:
  - 6 características, 27 subcaracterísticas...
  - ... descomponibles en atributos (jerarquía multi-nivel)
  - Grupo de métricas propuestas
- Antiguamente un estándar único:
  - ISO/IEC 9126, 1991
- Actualmente un estándar multiparte:
  - ISO/IEC 9126: Software quality (part1 1, 2001; 2&3, 2003; 4: 2004)
  - ISO/IEC 14598: Software Product Evaluation (6 partes)
- Recientemente remplazado:
  - ISO/IEC CD 25000, SQuaRE (Software Quality Requirements and Evaluation)

# ISO / IEC 9126 y 14598



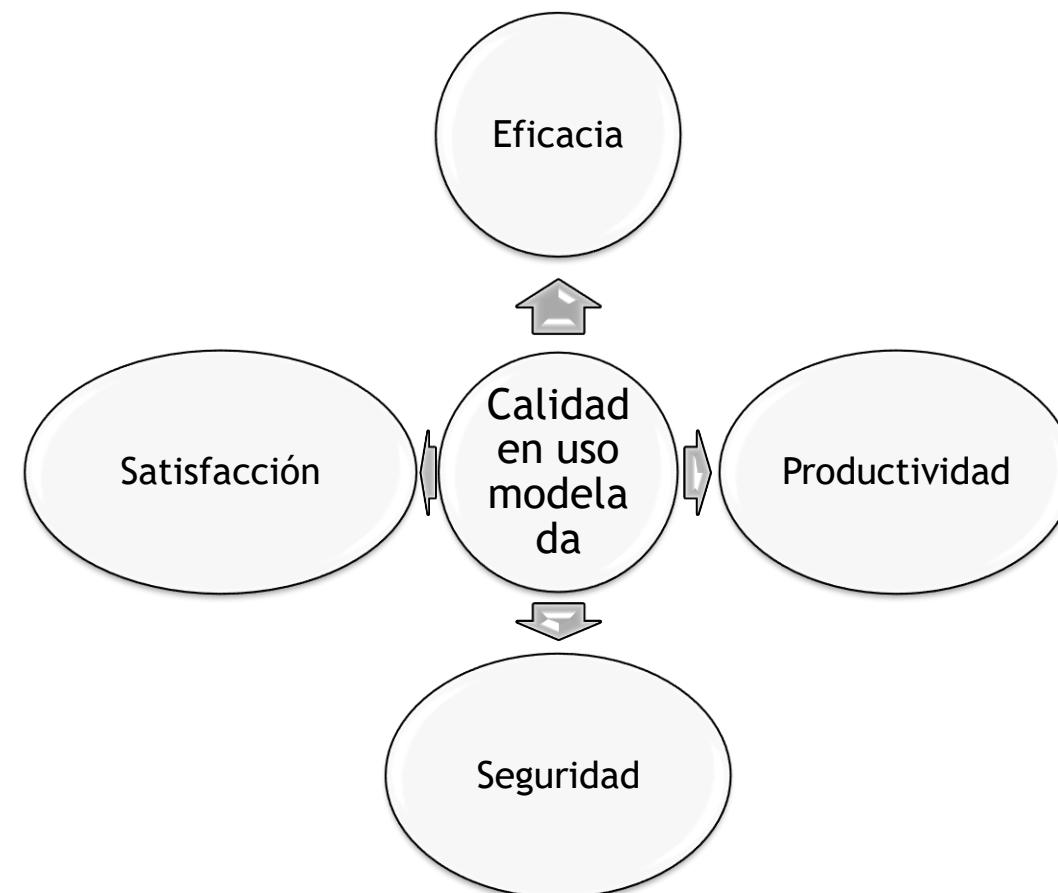
# ISO IEC 9126-1



# ISO IEC 9126 - 1



## n Calidad en uso modelada



## ISO / IEC 9126-2

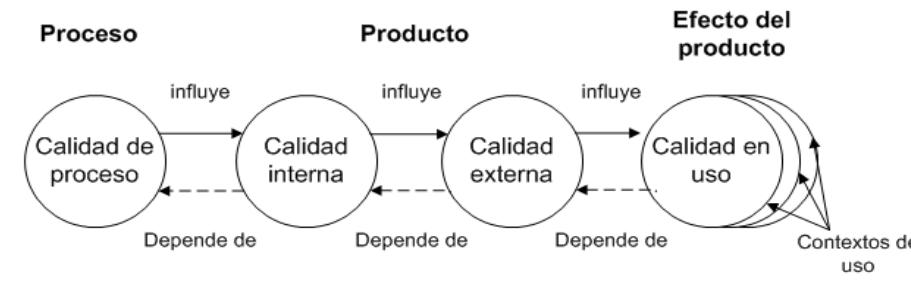
- Describe las métricas externas que son utilizadas para especificar o evaluar el comportamiento del software cuando es operado por el usuario

## ISO / IEC 9126-3

- Esta parte describe las métricas internas que se pueden utilizar para crear describir propiedades internas, que puede ser evaluadas por la inspección sin poner en funcionamiento el software.

## ISO / IEC 9126-4

- Esta parte describe las métricas de calidad en uso que se pueden utilizar para especificar o evaluar el efecto del producto software cuando son operados por el usuario en determinados contextos de uso.



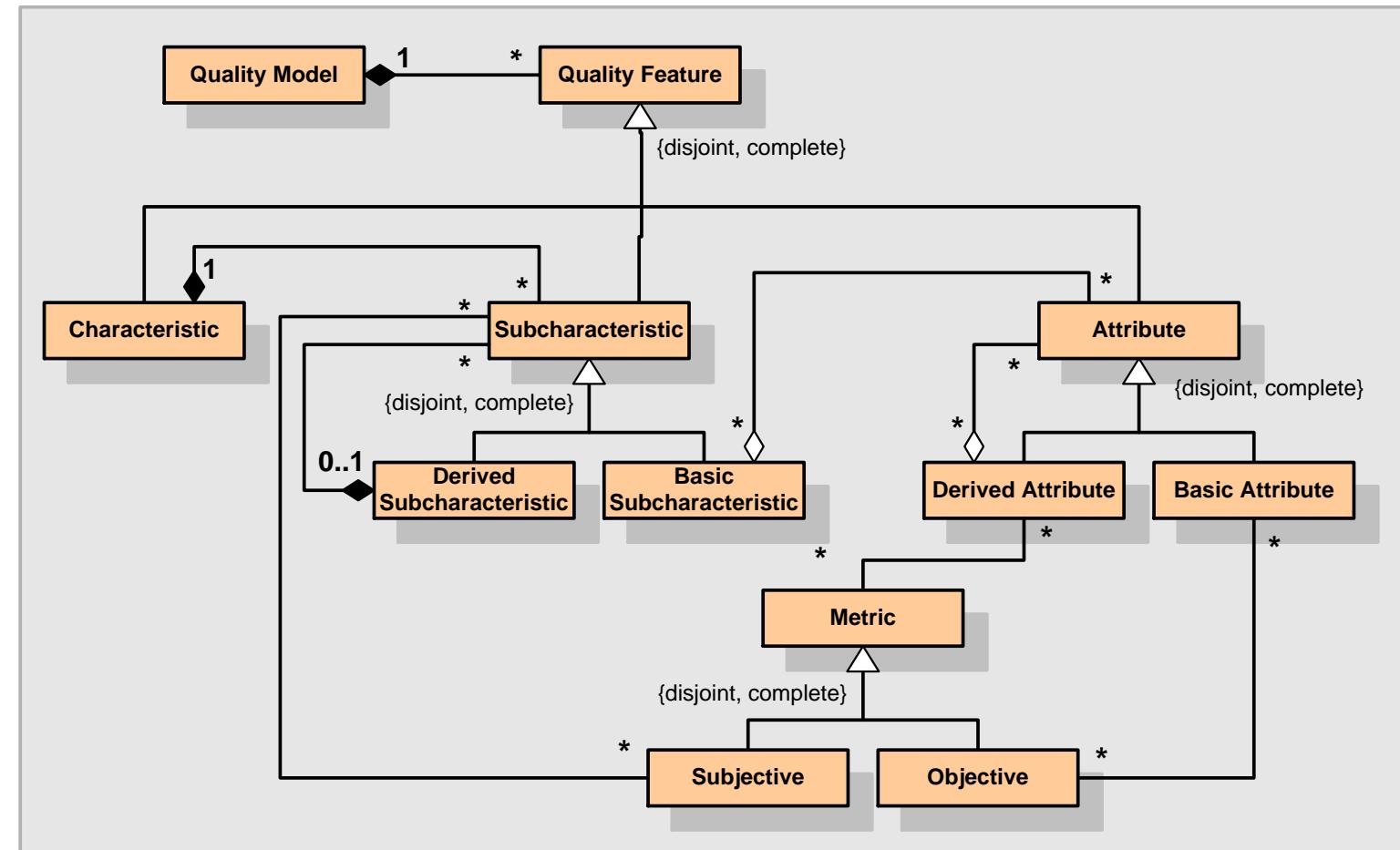
# ISO / IEC 14598



ISO/IEC 14598-1	Visión general de todo el estándar y explicación de las diferencias entre la evaluación del producto software y el modelo de calidad definido en la ISO / IEC 9126.
ISO/IEC 14598-2	Requisitos y guías para las funciones de planificación y gestión de la evaluación del producto.
ISO/IEC 14598-3	Requisitos y guías para la evaluación del producto software cuando la evaluación se lleva a cabo en paralelo al desarrollo del mismo.
ISO/IEC 14598-4	Requisitos y guías para la evaluación del producto software cuando este ha sido adquirido y se requiere reutilizar un producto existente o pre-desarrollado.
ISO/IEC 14598-5	Requisitos y guías para la evaluación del producto cuando esta es llevada a cabo por evaluadores independientes.
ISO/IEC 14598-6	Provee las guías para la documentación del módulo de evaluación.

# Ejercicio

- Analizar y entender la propuesta conceptual de ISO / IEC 9126





# Estándar ISO 25000 (SQuaRE)

- Revisa el ISO / IEC 9126 e incorpora las mismas características de calidad con algunas enmiendas.



# Ejercicio



- Realizar un breve análisis de cada una de las partes del estándar ISO 25000

# Contenido



- Introducción
- Que es un modelo de calidad del software
- Estructura de los modelos de calidad del software
- Tipos de modelos de calidad
- Estándares de modelos de calidad del software
- **Aplicaciones de los modelos de calidad del software**



# Aplicaciones de los Modelos de Calidad

■ Aplicaciones exploradas por diversos autores:

- Especificaciones de software
- Diseño arquitectónico del software
- Soporte a la implementación del software
- Soporte a la evaluación del software
- Soporte para la certificación del software
- Identificación de riesgos
- Otros:
  - Soporte a decisiones económicas en relación al rendimiento del software



# Bibliografía

- Galin D., Software Quality Assurance From theory to implementation, 2004
- Chappell, D. (2012). THE THREE ASPECTS OF SOFTWARE QUALITY : FUNCTIONAL , STRUCTURAL , AND PROCESS Sponsored by Microsoft Corporation. *David Chappel & Associates, 1.0.* Retrieved from [http://www.davidchappell.com/writing/white\\_papers/The\\_Three\\_Aspects\\_of\\_Software\\_Quality\\_v1.0-Chappell.pdf](http://www.davidchappell.com/writing/white_papers/The_Three_Aspects_of_Software_Quality_v1.0-Chappell.pdf)
- O'Regan, G. (2014). *Introduction to Software Quality*. <http://doi.org/10.1007/978-3-319-06106-1>
- Software, D., Rosa, V., & Zepeda, V. (2012). Metodología para el Aseguramiento de la Calidad en la Adquisición del Software ( proceso y producto ) y servicios.
- Wagner, S. (2013). *Software Product Quality Control*. <http://doi.org/10.1007/978-3-642-38571-1>
- Carvallo J. P., Presentaciones, 2014-2015.



**Facultad de Ingeniería  
Universidad de Cuenca  
Grado en Ingeniería de Sistemas  
Curso 2020-2021**

# **Calidad de Software**

## **Capítulo 4: Construcción de Modelos de Calidad de Software**

Departamento de Ciencias de la Computación  
Universidad de Cuenca, Ecuador  
email: [priscila.cedillo@ucuenca.edu.ec](mailto:priscila.cedillo@ucuenca.edu.ec)

# Contenido



- Introducción
- Alternativas para abordar la construcción de modelos de calidad
- Factores técnicos (funcionales y no funcionales)
- Factores no técnicos (proveedor, políticos, económicos, etc.)
- Solapamiento de características
- Características de calidad internas y externas
- Interdependencias entre características de calidad
- Métricas del software
- Métodos de construcción de modelos de calidad

# Contenido



- **Introducción**
- Alternativas para abordar la construcción de modelos de calidad
- Factores técnicos (funcionales y no funcionales)
- Factores no técnicos (proveedor, políticos, económicos, etc.)
- Solapamiento de características
- Características de calidad internas y externas
- Interdependencias entre características de calidad
- Métricas del software
- Métodos de construcción de modelos de calidad



# Introducción

- „ La calidad **no** es una **propiedad fija** de un sistema de software.
- „ Esta depende de las **necesidades** y **objetivos** de los stakeholders.
- „ Se deben **mapear** esas necesidades a **propiedades técnicas**.
- „ No solamente planearemos **que** calidad queremos tener sino adicionalmente **cómo** la construiremos y la aseguraremos.

# Introducción



Stakeholders



Necesidades y Objetivos



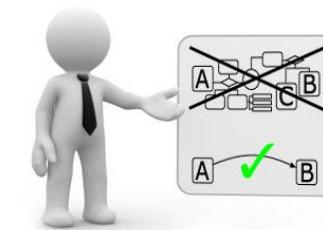
Propiedades Técnicas



Qué



Cómo





# Contenido

- Introducción
- Alternativas para abordar la construcción de modelos de calidad
- Factores técnicos (funcionales y no funcionales)
- Factores no técnicos (proveedor, políticos, económicos, etc.)
- Solapamiento de características
- Características de calidad internas y externas
- Interdependencias entre características de calidad
- Métricas del software
- Métodos de construcción de modelos de calidad

# Alternativas para abordar la construcción de modelos de calidad

- Los requisitos no-funcionales describen las propiedades que no representan la funcionalidad primaria del sistema
- Son características que hacen al producto atractivo, usable, rápido, confiable
- El problema es cómo eliciar y evaluar los requisitos de calidad en una forma estructurada y comprensible
- Los principales enfoques para eliciar requisitos de calidad son:
  - Chequear diferentes tipos de requisitos y construir prototipos
  - Usar escenarios positivos y negativos (Casos de Uso)
  - Refinar metas de calidad por gráficos de objetivos

# Alternativas para abordar la construcción de modelos de calidad

- Un ejemplo para chequear tipos de requisitos son los checklists para SLAs.
- En los casos de uso, los requisitos de calidad son adjuntados como información adicional (ej. máximo tiempo de respuesta del sistema en tal caso de uso)
- Gráficos de meta son a menudo descompuestos de metas de calidad a sentencias lógicas para luego operacionalizarlas como requisitos funcionales.





# Alternativas para abordar la construcción de modelos de calidad

- Especificar requisitos de calidad y construir modelos de calidad pueden fusionarse en un solo proceso.
- Mientras añadimos nueva información a un modelo de calidad cuando lo construimos, especificamos requisitos y escogemos las partes necesarias de modelos existentes.



# Contenido

- Introducción
- Alternativas para abordar la construcción de modelos de calidad
- Factores técnicos (funcionales y no funcionales)
- Factores no técnicos (proveedor, políticos, económicos, etc.)
- Solapamiento de características
- Características de calidad internas y externas
- Interdependencias entre características de calidad
- Métricas del software
- Métodos de construcción de modelos de calidad



# Factores Técnicos

- Descomposición de factores en factores de más bajo nivel de abstracción
- Sólo ocasionalmente a la eliminación o modificación de algún factor ya existente
- Los factores técnicos se refieren a la calidad intrínseca del producto de software.

# Factores No-Técnicos

- n Carvallo, Franch y Quer destacan que se pueden incluir en los modelos de calidad factores no-técnicos, y ellos así lo hacen; como ejemplo tenemos:
  - Proveedor: Características del proveedor que pueden influenciar en la calidad del producto de software
  - Negocio: Características del contrato entre proveedor y cliente que pueden influenciar la calidad del producto del software
  - Producto: Características de los aspectos comerciales del producto que pueden influenciar su calidad.



# Contenido

- Introducción
- Alternativas para abordar la construcción de modelos de calidad
- Factores técnicos (funcionales y no funcionales)
- Factores no técnicos (proveedor, políticos, económicos, etc.)
- **Métricas del software**
- Solapamiento de características
- Características de calidad internas y externas
- Interdependencias entre características de calidad
- Métodos de construcción de modelos de calidad



# Métricas del Software

- Las métricas son un buen método para medir, entender, monitorizar, predecir y probar el desarrollo de software y los proyectos de mantenimiento (Briand et al., 1996)
- La medición persigue tres objetivos fundamentales:
  - Entender qué ocurre durante el desarrollo y el mantenimiento.
  - Controlar qué es lo que ocurre en los proyectos de desarrollo.
  - Mejorar los procesos y productos.
- Se pueden ver a las medidas como medios para asegurar la calidad en los producto de software.

# Métricas del Software

## n Problemas

- Todavía se falla en dar objetivos medibles cuando desarrollamos productos de software.
  - Ejm. Se dice que será usable sin especificar esa usabilidad en términos medibles.
- Fallamos en medir diferentes componentes para calcular costes reales de proyectos.
  - Ejm. No sabemos cuánto tiempo fue invertido en el diseño comparado con las pruebas.
- No intentamos cuantificar la calidad de los productos que producimos.
  - Ejm. No podemos decir a un usuario cómo de fiable va a ser un producto en términos de fallos en un período de uso dado.



# Métricas del Software

## n Problemas

- Solemos ver informes que hacen afirmaciones como que el 80% de los costes del software son de mantenimiento o que hay una media de 55 errores en cada 1.000 líneas de código.
- Pero no se dice
  - Cómo se han obtenido esos resultados
  - Cómo se han ejecutado los experimentos
  - Qué entidades fueron medidas y cómo
  - Los márgenes de error.



# Atributos Medibles

- En software hay **tres clases de entidades** cuyos atributos podemos medir:
  - **Procesos:** actividades del desarrollo
  - **Productos:** entregables, artefactos o documentos generados en el ciclo de vida del software
  - **Recursos:** todos aquellos elementos que hacen de entrada a la producción de software.

# Atributos a medir

## n Procesos:

- El tiempo (duración del proceso)
- El esfuerzo (asociado al proceso)
- El número de incidentes de un tipo específico

## n Productos:

- La fiabilidad del código
- La entendibilidad de un documento de especificación
- La mantenibilidad de un código fuente.
- La longitud, funcionalidad, modularidad o corrección sintáctica de los documentos de especificación

## n Recursos:

- El personal
- Los materiales
- Las herramientas y métodos
- El coste
- La productividad



# Utilidad de las Métricas del Software

- Estimación del esfuerzo
- Modelos y medidas de productividad
- Modelos y medidas de calidad
- Modelos de fiabilidad
- Modelos de evaluación y rendimiento
- Métricas de complejidad estructural

# Ejemplo

- Se quiere contratar software as a service (SaaS) de supermercados.
- El software debe estar disponible. La intensidad de fallos del software es 1 fallo por 100 CPU/hr. El sistema especializado ejecuta 20 CPU/hr por semana y hay 800 clientes que usarán el servicio.
- Se requiere proporcionar a los clientes un servicio de reparación.
- Cada empleado puede realizar 4 llamadas de servicio por día y el servicio estará disponible 5 días por semana.
- ¿Cuántos empleados necesitamos contratar?

- Usando el valor de intensidad de fallos, cada sistema experimenta 0.2 fallos por 20 horas de operación o (0.2 fallos por semana en promedio)
- El total de fallos para 800 clientes es de 160 por semana o 32 por día.
- Cada empleado puede realizar 4 visitas por día. Así el número de empleados debería ser  $32/4=8$ .



# Contenido

- Introducción
- Alternativas para abordar la construcción de modelos de calidad
- Factores técnicos (funcionales y no funcionales)
- Factores no técnicos (proveedor, políticos, económicos, etc.)
- Solapamiento de características
- Características de calidad internas y externas
- Interdependencias entre características de calidad
- Métricas del software
- **Métodos de construcción de modelos de calidad**



# Pasos para construir el modelo a través de la Ingeniería de Requisitos

1. Identificar a los stakeholders relevantes
2. Definir metas generales
3. Analizar documentos relevantes y solicitar nueva información
4. Escoger y definir actividades
5. Definir metas de calidad
6. Identificar artefactos afectados y escoger entidades
7. Escoger entidades y analizar material relevante
8. Escoger factores y definir nuevas características de producto
9. Especificar requisitos de calidad

# Otras propuestas..

- Por su complejidad se requiere de una guía para su construcción.
- Existen propuestas generales:
  - Bohem-78: 6 pasos
  - GQM: 5 pasos
  - Dromey: 5 pasos
- Método IQMC (Individual Quality Model Construction)
  - Diseñado para producir modelos compatibles con el estándar ISO/IEC 9126-1

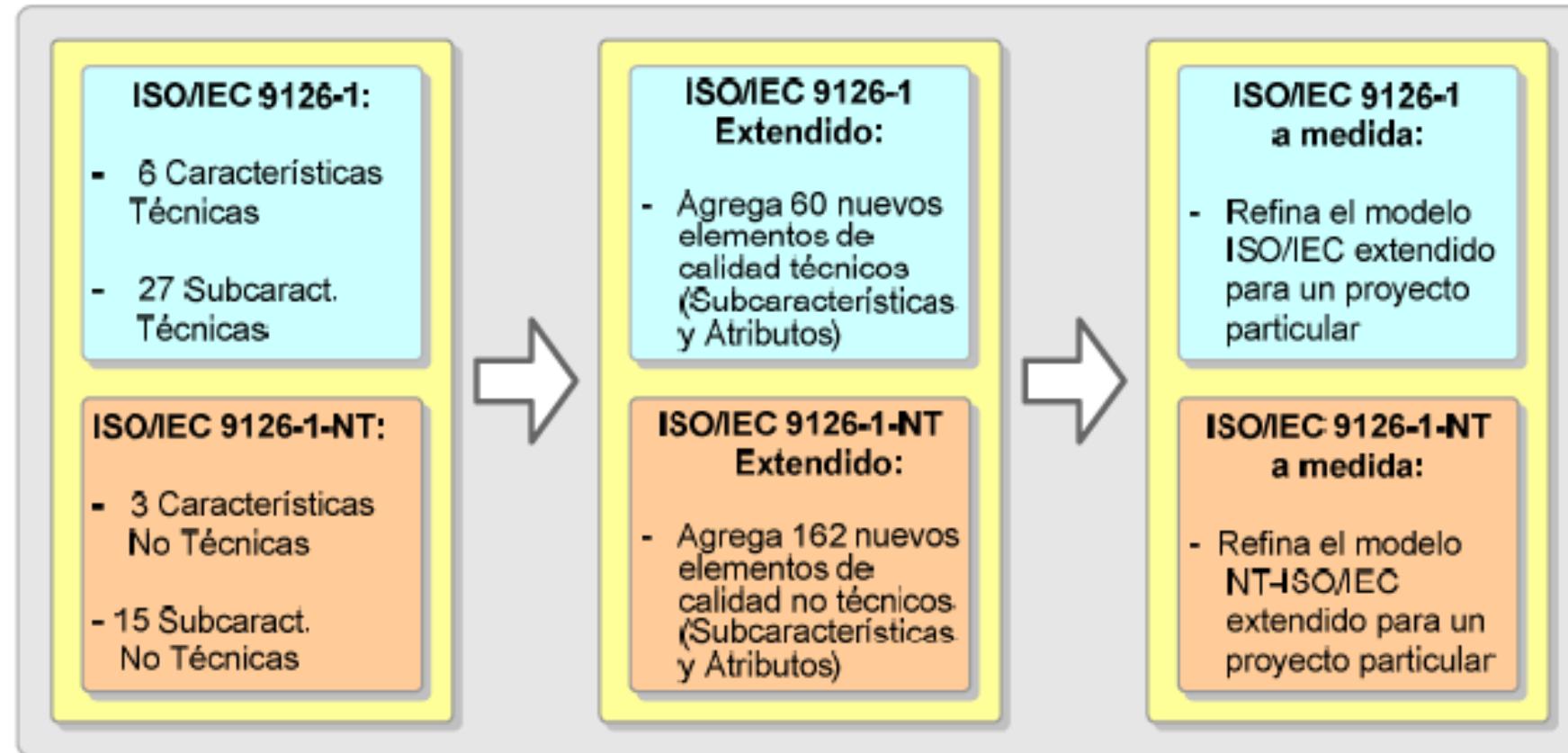


# El método IQMC para la construcción de modelos de calidad

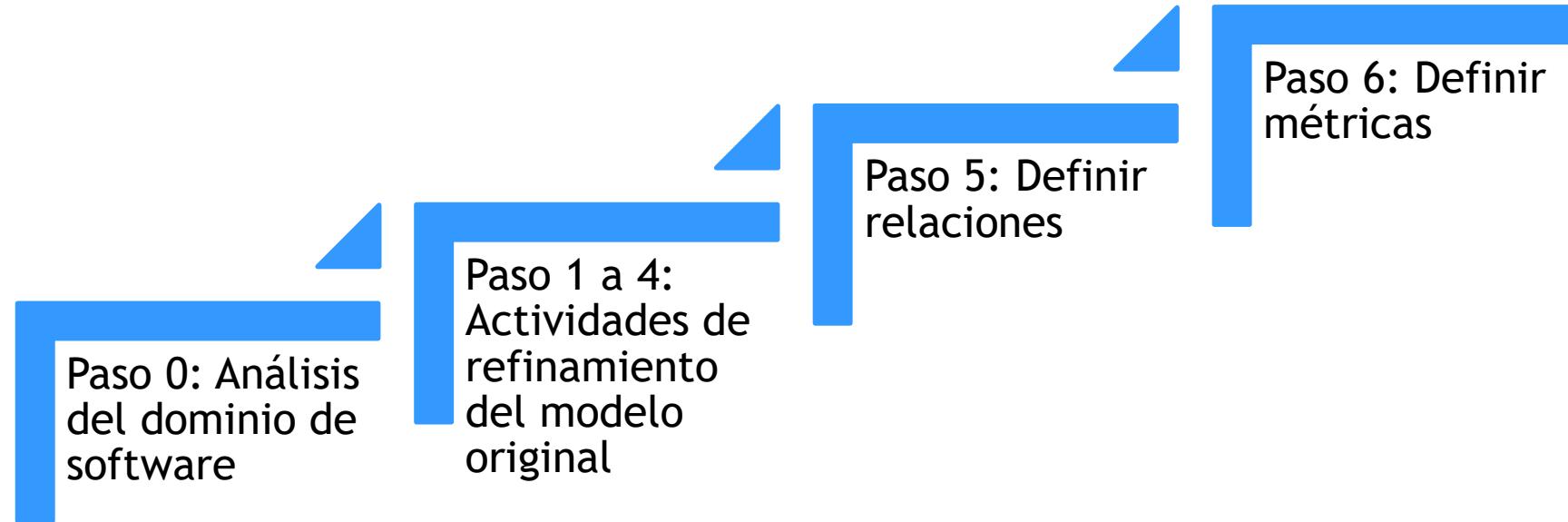
- Circunstancias que intervienen en la construcción de un modelo de calidad:
  1. El equipo realiza la construcción del modelo, en el caso de que este equipo **no tenga experiencia** en la construcción de modelos o en el contexto del dominio.
  2. El dominio para el que se construye el modelo, para el que en muchas ocasiones **no existe una terminología común**.
  3. **Factores metodológicos**, ya que es difícil conocer el nivel de profundidad hasta el que es necesario descomponer los modelos.

**IQMC** : Conjunto de guías y técnicas para la identificación de los factores de calidad apropiados que deben ser incluidos en un modelo de calidad que permita analizar la calidad de componentes pertenecientes a un cierto dominio de software.

# Extensión del ISO 9126-1



# Método IQMC



- El estándar IQMC consiste en 7 pasos que aunque se presentan como secuenciales, pueden ser simultáneos de ser necesario.
- Es importante hacer hincapié en la diferencia de detalle que existe en el catálogo entre los factores técnicos y no-técnicos.

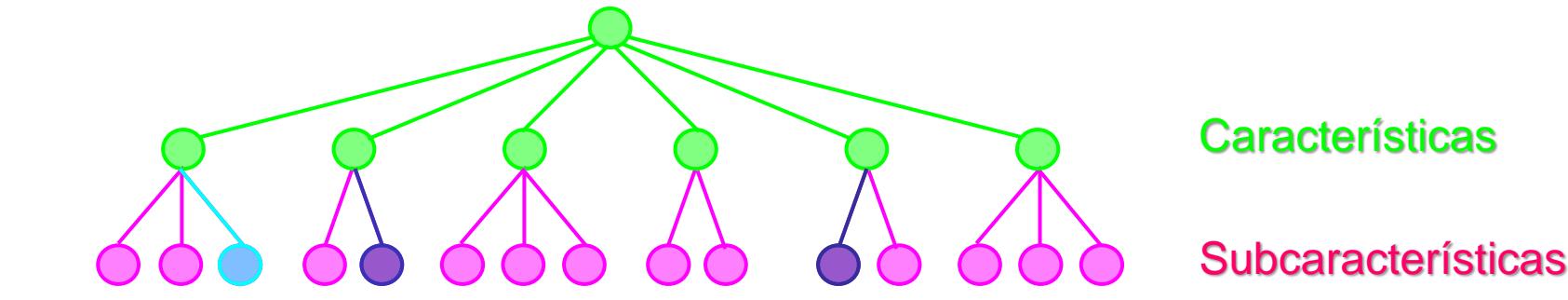


# Paso 0

- Estudio del ámbito del software.
- Paso opcional que puede soslayarse en caso de poseer el conocimiento suficiente.
- Puede ser necesario realizar algún tipo de modelización para realizar una unificación de la terminología identificada en las distintas fuentes de información de cara a los pasos siguientes.
- Tener cuidado con la terminología a ser empleada, muchas personas utilizan diferentes términos para un mismo concepto o lo que es peor el mismo término para diferentes conceptos.

## Paso 1

- Determinación de **sub-características** de calidad. Teniendo en cuenta que partimos del catálogo ISO/IEC 9126-1 extendido, el añadido de sub-características no será muy habitual.
- Puede pasar que alguna sub-característica deba **reformularse** ligeramente para **adaptarla al dominio de interés**, o **eliminarse**.



## Paso 2

- Refinamiento de la jerarquía de sub-características. Se descomponen las sub-características del más bajo nivel de abstracción formando jerarquías de sub-características.
- Al igual que en el paso anterior, el añadido de sub-características no será muy habitual, excepto en el caso de la descomposición de la sub-característica.
- En cuanto a sub-características no-técnicas, lo que se realizará es un purgado de las sub-características que no interesen al proyecto en cuestión.

# Paso 3



- Refinamiento de sub-características en atributos.
- Descomposición de esos atributos de calidad en unos más concretos.
- Objetivos: llegar a tener descompuestas las sub-características en atributos medibles ya sea de forma directa o indirecta a través del valor de otro atributos básicos.
- Un atributo puede estar en más de una sub-característica.



## Paso 4

- Atributos básicos y derivados
- Refinamiento de atributos derivados en básicos. Se descomponen los atributos derivados hasta obtener los atributos básicos.
- Una vez realizado lo anterior los atributos básicos pueden ser medidos de forma directa.

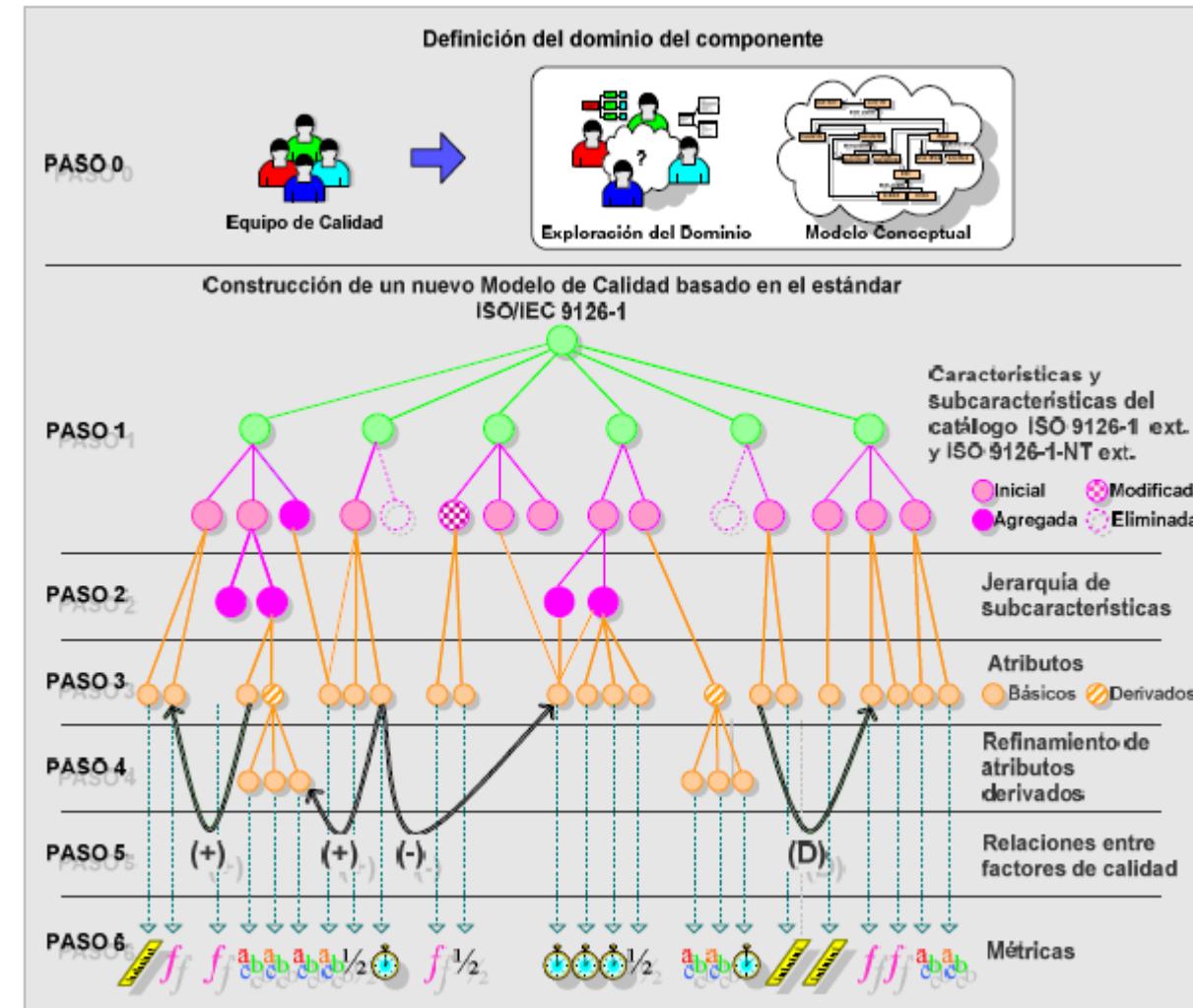
# Paso 5



- n Determinar las métricas para los atributos básicos.
- n Establecimiento de relaciones entre factores de calidad.
- n Permiten conocer las dependencias entre los distintos factores de calidad del modelo.
- n Colaboración, daño o dependencia

# Paso 6

- n Determinación de métricas para los atributos. Se determinan las métricas para los atributos identificados.



# Goal Question-Metric

- Técnica definida por Basili y Weiss (1984) y Rombach (1990), para seleccionar y generar métricas tanto del proceso como de los resultados del proyecto.



- Propone definir un objetivo, refinarlo en preguntas y definir métricas que intenten dar información para responder estas preguntas.

# Fases del método GQM

## n Planificación

- Se selecciona, define, caracteriza y planifica un proyecto para la aplicación de la medición, obteniéndose como resultado un **plan del proyecto**.

## n Definición

- Se **define y documenta el programa de la medición** (objetivos, preguntas, métricas e hipótesis).

## n Recopilación de Datos

- Se reúnen los **datos reales de la medición**.

## n Interpretación

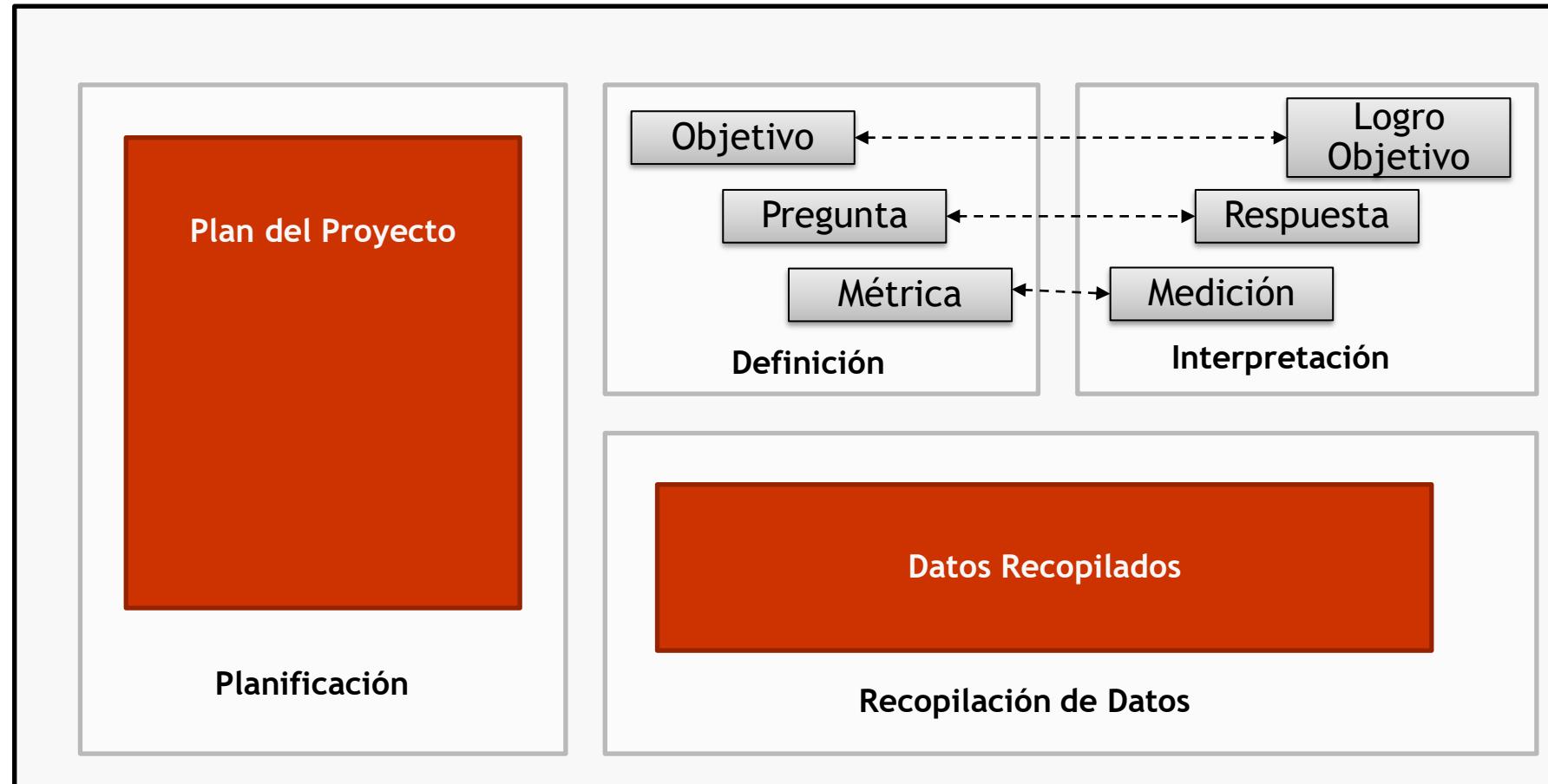
- Se **procesan los datos** recopilados respecto a las métricas definidas en forma de resultados de medición, que proporcionan **respuestas a las preguntas** definidas a partir de las cuales se puede evaluar el logro del objetivo planteado.



# Planificación

- Recopilar toda la información necesaria para un inicio exitoso de un proyecto de medición.
- Motivación y preparación de los miembros de la organización para llevar a cabo el programa de medición.
- Plan de proyecto - producto principal de esta fase.
- Incluye:
  - Documentos
  - Procedimientos
  - Calendarios
  - Objetivos del programa de medición.

# El proceso GQM



El Proceso GQM (van Soligen y Berghout, 1999)

# Planificación

## n Establecer el equipo GQM

- Garantiza la continuidad de los programas de medición.
- Cuando existe apuro se descuida, por lo cual GQM debería tener las siguientes cualidades:
  - Ser independiente de los equipos del proyecto.
  - No ser “parte interesada”
  - Poseer conocimiento previo sobre los objetos de la medición.
  - Conciencia de su rol.
  - Mentalidad de orientación a la mejora.



# Planificación

## n Seleccionar Áreas de Mejora

- Problemas evidentes a los que se enfrenta la organización.
- Areas a mejorar identificadas, en base a los objetivos del negocio.
- Problemas que podrían ocurrir.
- Influencias externas.
- Tecnologías.
- Leyes.
- Procesos y productos.
- Experiencia en medición de las personas implicadas.



# Planificación

- Seleccionar el proyecto de aplicación y establecer un equipo del proyecto.
  - Depende de la voluntad, motivación y entusiasmo de los miembros del equipo de proyecto.
  - Los objetivos de la medición deben estar alineados con las ideas de mejora del proyecto.

# Planificación

## n Crear el plan del proyecto

- Resumen ejecutivo: presentar en 20 líneas aprox. el programa de medición.
- Introducción: Presenta el alcance del programa de medición. Relación entre los objetivos de la mejora y los objetivos del proyecto de desarrollo de software.
- Calendario: incluye planificación temporal, entregables, asignación de recursos y análisis coste-beneficio del programa de medición.
- Organización: Estructuras organizacionales del proyecto y equipo GQM que son relevantes para el programa de medición.
- Procesos de gestión: Contiene prioridades, procedimientos de generación de informes de gestión así como actividades de control de riesgos.
- Formación y promoción: Plan para la formación de los miembros del equipo del proyecto y la comunicación de los resultados de la organización.

# Definición

- Definir los objetivos de la medición
  - Se obtiene una definición formal y bien estructurada de los objetivos, para lo cual se utilizan plantillas como la que se muestra:

<b>Analizar</b>	el objeto bajo medición
<b>Con el propósito de</b>	entender, controlar, o mejorar el objeto
<b>Con respecto a</b>	el enfoque de calidad del objeto en el que se centra la medición
<b>Desde el punto de vista de</b>	las personas que miden el objeto
<b>En el contexto de</b>	el entorno en el que la medición tiene lugar

Definir preguntas e hipótesis

Con la respuesta a las preguntas planteadas, se deberá poder concluir si se cumple un determinado objetivo.

Definir las métricas

Deben proporcionar información cuantitativa que permita responder las preguntas planteadas.



# Definición

Revisar o elaborar los modelos de proceso software

-Dan soporte a la definición de las mediciones

Realizar entrevistas GQM

-Se extrae del equipo toda la información relevante

Definir y revisar preguntas e hipótesis

Definir las métricas

# Ejemplo

## ■ Métricas para las BD Relacionales

## ■ Objetivo GQM:

- Analizar: Bd Relacionales
- Con el propósito de: Asegurar
- Con respecto a: La mantenibilidad
- Desde el punto de vista de: BD Relacional

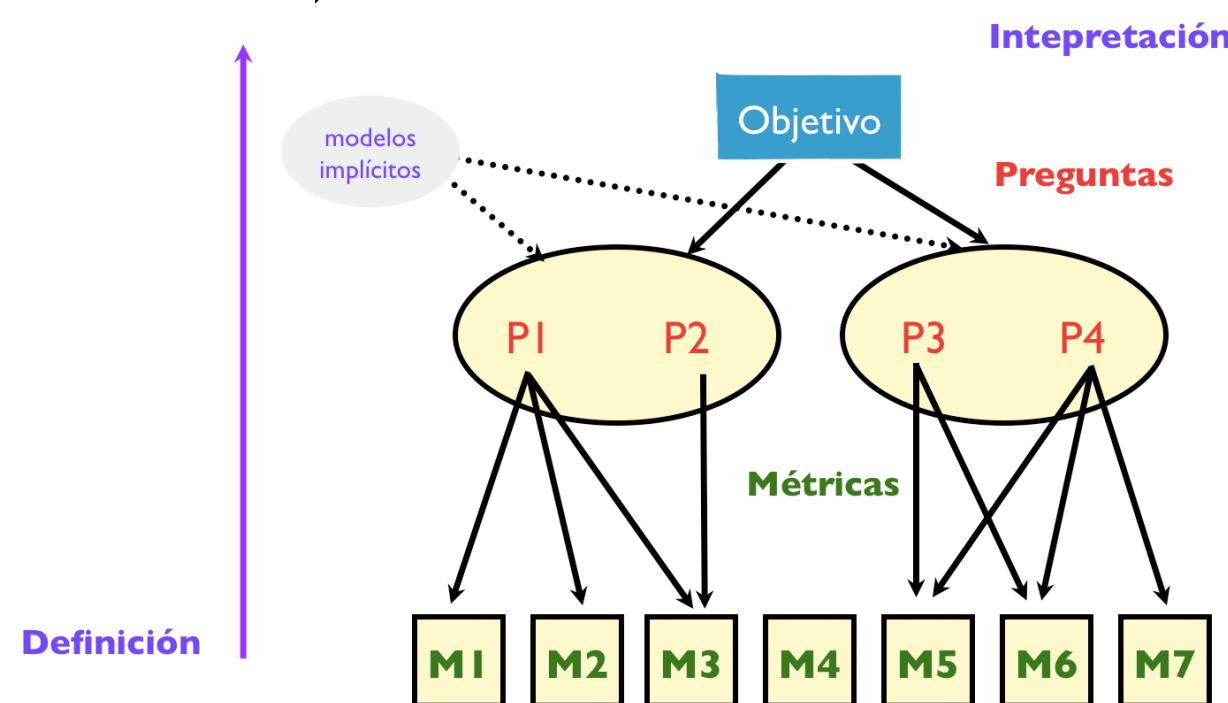
## ■ Preguntas

- Cómo influye la **complejidad de las tablas** en la mantenibilidad de las BD Relacionales.
- Cómo influye la **complejidad entre tablas** en la mantenibilidad de las BD Relacionales

# Niveles GQM

Se constituyen 3 niveles dentro de esta técnica

- Nivel Conceptual (Objetivos-Goals)
- Nivel Operacional (Preguntas - Questions)
- Nivel Cuantitativo (Métricas - Metrics)



# Ejemplo

## n Métricas

### ▪ Pregunta 1

- **NA(T)** - NÚMERO DE ATRIBUTOS DE UNA TABLA
- **NFK(T)** - NÚMERO DE CLAVES AJENAS
- **RFK(T)** - RATIO DE CLAVES AJENAS DE UNA TABLA

$$RFK(T) = \frac{NFK(T)}{NA(T)}$$

### ▪ Pregunta 2

- **NT** - NÚMERO DE TABLAS
- **NA** - NÚMERO DE ATRIBUTOS
- **NFK** - NÚMERO DE CLAVES AJENAS (NFK)



# Bibliografía

- Galin D., Software Quality Assurance From theory to implementation, 2004
- Chappell, D. (2012). THE THREE ASPECTS OF SOFTWARE QUALITY : FUNCTIONAL , STRUCTURAL , AND PROCESS Sponsored by Microsoft Corporation. *David Chappel & Associates, 1.0.* Retrieved from [http://www.davidchappell.com/writing/white\\_papers/The\\_Three\\_Aspects\\_of\\_Software\\_Quality\\_v1.0-Chappell.pdf](http://www.davidchappell.com/writing/white_papers/The_Three_Aspects_of_Software_Quality_v1.0-Chappell.pdf)
- O'Regan, G. (2014). *Introduction to Software Quality*. <http://doi.org/10.1007/978-3-319-06106-1>
- Software, D., Rosa, V., & Zepeda, V. (2012). Metodología para el Aseguramiento de la Calidad en la Adquisición del Software ( proceso y producto ) y servicios.
- Wagner, S. (2013). *Software Product Quality Control*. <http://doi.org/10.1007/978-3-642-38571-1>
- Carvallo J. P., Presentaciones, 2014-2015.



Facultad de Ingeniería  
Universidad de Cuenca  
Grado en Ingeniería de Sistemas  
Curso 2020-2021

# Métricas

## Capítulo 5: Métricas y la Ontología de la Medición del Software

Presentación del trabajo de García et al. (2004)

Departamento de Ciencias de la Computación  
Universidad de Cuenca, Ecuador  
email: [priscila.cedillo@ucuenca.edu.ec](mailto:priscila.cedillo@ucuenca.edu.ec)

# Contenidos



- Introducción
- Términos del Glosario
- Diagrama Ontológico
- Ejemplo de Métricas



# Contenido

- **Introducción**
- Términos del Glosario
- Diagrama Ontológico
- Ejemplo de Métricas



# Introducción

- La medición de software es una disciplina relativamente joven y no existe aún un consenso general sobre la definición exacta de los conceptos y terminología que maneja.
- En las siguientes diapositivas se presenta cuatro conceptos fundamentales:
  1. La forma de medir (concepto de measurement de ISO-15939, y que va a cubrir: el método de medición para métricas directas, la función de cálculo utilizada en métricas directas, y el modelo de análisis utilizado en los indicadores)
  2. La acción de medir (denominado medición)
  3. El resultado de la medición (denominado medida)
  4. El concepto de métrica (una forma de medir + una escala)



# GLOSARIO



# Categoría de Entidad

- Una colección de entidades caracterizadas por **satisfacer un cierto predicado común**. Se pueden definir jerarquías entre categorías de entidad.
- Relaciones
  - Una categoría de entidad puede “**incluir a**” una o varias categorías de entidad, y puede “estar incluida en” una o varias categorías de entidad.
  - Una categoría de entidad tiene uno o varios atributos.
  - Una categoría de entidad puede tener definidos varios modelos de calidad.
- Ejemplos
  - Programas, programas en C, componentes software, componentes COTS, componentes de software para comunicaciones, etc.
  - Procesos productos, servicios, proyectos o recursos son ejemplos de categorías de entidad.



# Entidad

- Un objeto que va a ser caracterizado mediante una medición de sus atributos.
- Una entidad puede ser un proceso, un producto, un servicio, un proyecto, un servicio o un recurso concreto.ç
- Una entidad puede ser física -tangible- (p.e. un ordenador) o abstracta (p.e. un programa en C)
- Relaciones
  - Una entidad puede pertenecer a una o más categorías de entidad.
  - Una medición se realiza sobre los atributos de una entidad.
- Ejemplos
  - El programa “holaMundo.c”



# Atributo

- „ Una **propiedad medible** (mensurable), **física o abstracta**, que comparten todas las entidades de una categoría de entidad.
- „ Relaciones:
  - Un atributo puede pertenecer a una categoría de entidad.
  - Una medición se realiza sobre los atributos de una entidad.
  - Un atributo tiene definida cero, una o varias métricas.
  - Un atributo está relacionado con uno o más conceptos medibles.
- „ Ejemplos
  - El atributo “**tamaño de código fuente**”, como atributo de categoría de entidad “**programas en C**” va a ser diferente del atributo “**tamaño del código fuente**” de la categoría de entidad “**programa en ADA**”.
  - El **tamaño de un “modulo en C”** no es el mismo atributo (aunque tenga el mismo nombre) que el **“tamaño de un diagrama de clases UML”**.
  - Tamaño de código fuente, precio.



# Forma de medir

- Conjunto de operaciones cuyo **objeto** es determinar el **valor de una medida**. Una forma de medir puede ser un **método de medición, función de cálculo o modelo de análisis**.
- Relaciones:
  - Una forma de medir es ejecutada en cada medición, dependiendo de la métrica que calcula.
- Ejemplos:
  - Véase los ejemplos de método de medición, función de cálculo o modelo de análisis, ya que la forma de medir es una generalización de ellos.

# Métrica

n Una forma de medir (método de medición, función de cálculo o modelo de análisis) **y una escala**, definidas para realizar mediciones de uno o varios atributos.

## Relaciones

- Una métrica está definida para uno o más atributos.
- Dos métricas pueden relacionarse mediante una función de transformación, El tipo de dicha función de transformación va a depender del tipo de escala de ambas métricas.
- Una métrica puede expresarse en una unidad (sólo para métricas cuya escala sea de tipo intervalo o ratio).

## Ejemplos

- La métrica “líneas de código” puede ser definida para realizar mediciones del “tamaño” de un “programa en ADA”.

# Métrica Directa

- Una métrica de la cual se pueden realizar mediciones sin depender de ninguna otra métrica y cuya forma de medir es un método de medición.
- Relaciones
  - La forma de medir una métrica directa es un método de medición.
  - La métrica directa puede ser utilizada en funciones de cálculo.
- Ejemplos
  - LCF (líneas de código fuente escritas)
  - HPD (horas-programador diarias).
  - CHP (coste por hora-programador, en unidades monetarias).

# Métrica Indirecta

- Una métrica cuya forma de medir es una función de cálculo, es decir las mediciones de dicha métrica utilizan las medidas obtenidas en mediciones de otras métricas directas o indirectas.
- Relaciones
  - La forma de medir una métrica indirecta es una función de cálculo.
  - Una métrica indirecta puede usarse en una función de cálculo.
- Ejemplos
  - HPT (Horas-programador totales)
  - LCFH (Líneas de código fuente por hora de programador)
  - CTF (coste total actual del proyecto, en unidades monetarias)
  - CLCF (coste por líneas de código fuente)

# Indicador

- Una métrica cuya forma de medir es un modelo de análisis, es decir, las mediciones de dicha métrica utilizan las medidas obtenidas en las mediciones de otras métricas (directas, indirectas o indicadores) junto con criterios de decisión.
- Relaciones
  - Un indicador satisface necesidades de información.
  - Un indicador es definido por un modelo de análisis.
- Ejemplos
  - PROD (productividad de los programadores)
  - CAR (carestía del proyecto).



# Medición (Acción de Medir)

- La acción que permite obtener el valor de una medida para un atributo de una entidad, usando una forma de medir.
- Relaciones
  - Cada medición produce una medida.
  - Una medición usa una métrica, la cual debe estar definida para el atributo objeto de la medición.
  - Una medición es llevada a cabo usando una forma de medir. Esta forma de medir es la que define la métrica usada en la medición.
  - Una medición se realiza para un atributo de un atributo de una entidad. El atributo ha de estar definido para la categoría a la que pertenece dicha entidad.
- Ejemplos
  - Acción consistente en usar la forma de medir “contar el número de líneas de código” para obtener la medida del atributo “tamaño” de la entidad “módulo nominas.c”.

# Medida

- Resultado de una medición.
- Ejemplos
  - 35.000 líneas de código, 200 páginas, 50 clases.
  - 5 meses desde el comienzo al fin del proyecto.
  - 0,5 fallos por cada 1.000 líneas de código.



# Método de Medición

■ La forma de medir una métrica directa. Secuencia lógica de operaciones, descritas de forma genérica usadas para realizar mediciones de un atributo respecto de una escala específica.

## ■ Relaciones

- Un método de medición define una o más métricas directas.
- Un método de medición puede usar Instrumentos de Medición.

## ■ Ejemplos

- Contar líneas de código.
- Anotar cada día las horas dedicadas por los programadores al proyecto.
- Valorar el grado de dificultad de un problema.



# Instrumento de Medición

- Instrumento que asiste o es útil a un método de medición.
- Relaciones
  - Un instrumento de medición asiste a uno o más métodos de medición.
- Ejemplos
  - Un reloj es un instrumento de medición que asiste al método de medición para contar el paso del tiempo.
  - Una herramienta CASE que sirva para contar líneas de código es un instrumento de medición que asiste al método de medición contar líneas de código.

# Necesidad de información

- Información necesaria para gestionar un proyecto (sus objetivos, hitos, riesgos y problemas).
- Relaciones
  - Una necesidad de información se asocia con un concepto medible.
  - Una necesidad de información se satisface con uno o más indicadores.
- Ejemplos
  - Conocer el nivel de productividad de los programadores del proyecto en comparación con lo habitual en otros proyectos de la organización (ver ejemplo desarrollado).
  - Determinar si los recursos del proyecto son adecuados para satisfacer sus objetivos.
  - Evaluar el rendimiento de la actividad de codificación.
  - Evaluar si un producto software satisface las expectativas del cliente.



# Concepto Medible

- Relación abstracta entre atributos y necesidades de información.
- Relaciones
  - Un concepto medible se asocia a una o varias necesidades de información.
  - Un concepto medible puede incluir otros conceptos medibles.
  - Un concepto medible relaciona uno o más atributos.
  - Un concepto medible está incluido en uno o más modelos de calidad.
- Ejemplos
  - Ratio de productividad de un equipo de desarrollo frente a un grado de productividad objetivo.
  - Adecuación de la tecnología.

# Modelo (de calidad)

## n Definición:

- Un marco conceptual que especifica una serie de conceptos medibles y sus relaciones, para una determinada categoría de entidad.

## n Relaciones:

- Un modelo (de calidad) está definido para una determinada categoría de entidad.
- Un modelo (de calidad) evalúa uno o varios conceptos medibles.

## n Ejemplos

- Modelo de calidad para productos de software de ISO 9126.
- Factores de calidad de McCall.

# Unidad

■ Una cantidad particular, definida y adoptada por convenios, con la que se puede comparar otras cantidades de la misma clase para expresar sus magnitudes respecto a esa cantidad particular.

## Relaciones

- Una unidad sirve para expresar una o varias métricas cuyo tipo de escala sea intervalo o ratio.

## Ejemplos

- Kilómetros, metros, millas.
- Líneas de código, páginas, persona-mes.
- Número de módulos, número de clases.
- Dolares, euros, horas, días, meses, años.

# Escala

- Un conjunto de valores con propiedades definidas.
- Relaciones
  - Toda escala es de un cierto “Tipo de Escala”.
- Ejemplos
  - Los valores que puede tomar la métrica “lenguaje de Programación usado en un proyecto”: Pascal, C, Java (Nominal).
  - El nivel de madurez CMMI: 1,2,3,4 (Ordinal)
  - Tamaño de un código software expresado en líneas de código: Conjunto de los números naturales (Ratio).
  - La temperatura expresada en grados centígrados o grados Farengheit (Intervalo)



# Tipo de escala

- Indica la naturaleza de la relación entre los valores de la escala [ISO 15939]
- Relaciones
  - A un tipo de escala pertenecen una o más escalas.
- Ejemplos
  - Nominal, ordinal , intervalo, ratio y absoluta.

# Función de cálculo

## n Definición

- La forma de medir una métrica indirecta. Algoritmo o cálculo realizado para combinar dos o más métricas directas y/o indirectas.

## n Relaciones

- Una función de cálculo usa cero o más métricas directas.
- Una función de cálculo usa cero o más métricas indirectas.
- Una función de cálculo utiliza al menos una métrica (sea directa o indirecta)
- Una función de cálculo define una o más métricas indirectas.

$$LCFH = \frac{LCF}{HPT}$$
 [Métrica indirecta definida en base a 2 métricas directas]

$$HTP = \sum_{días} HPD$$

[Métrica indirecta definida en base a sólo una métrica directa]

# Criterio de Decisión

- Valores umbrales, objetivos, o patrones usados para determinar la necesidad de una acción o investigación posterior, o para describir el nivel de confianza de un resultado dado.

## Relaciones

- Un criterio de decisión es utilizado en uno o más modelos de análisis.

- $LCFH/LCFH_{vm} < 0'70 \Rightarrow PROD = \text{'muy baja'}$ .
- $0'70 \leq LCFH/LCFH_{vm} < 0'90 \Rightarrow PROD = \text{'baja'}$ .
- $0'90 \leq LCFH/LCFH_{vm} < 1'10 \Rightarrow PROD = \text{'normal'}$ .
- $1'10 \leq LCFH/LCFH_{vm} < 1'30 \Rightarrow PROD = \text{'alta'}$ .
- $1'30 \leq LCFH/LCFH_{vm} \Rightarrow PROD = \text{'muy alta'}$ .

# Modelo de Análisis

n La forma de medir un indicador.  
Algoritmo o cálculo realizado para combinar una o más métricas (directas, indirectas o indicadores) con criterios de decisión asociados.

## Relaciones

- Un modelo de análisis define uno o más indicadores.
- Un modelo de análisis utiliza uno o más criterios de decisión.
- Un modelo de análisis usa una o más métricas.

- Modelo de Análisis para obtener la métrica PROD. Utiliza los valores de las métricas LCF, HPT, LCFH y CTP para establecer un valor cualitativo de la productividad de los programadores en este proyecto. El modelo se basa en extraer de una base histórica de proyectos de la organización los valores medios de LCF, HPT, LCFH (LCFHvm) y CTP del subconjunto de proyectos similares (aquellos que tienen LCF entre el 80% y el 120%)



# GRACIAS POR SU ATENCIÓN

**Escuela Técnica Superior de Ingeniería Informática  
Universitat Politècnica de València  
Grado en Ingeniería Informática, Curso 2013-2014**

# **Calidad de Software**

## **Tema 3: Métricas del Software**

Departamento de Sistemas Informáticos y Computación  
Universitat Politècnica de València, España  
email: [sabrahao@dsic.upv.es](mailto:sabrahao@dsic.upv.es)



# Objetivos

- Presentar de forma intuitiva el concepto de métricas del software
- Presentar una ontología de medición que formaliza el conocimiento sobre la medición del software.
- Presentar el framework QGM (Goal-Question-Metrics)
- Presentar los principales métodos de validación teórica de métricas
- Presentar los principales métodos de validación empírica de métricas



# Contenido

- Tema 1: Introducción
  - Fundamentos de la Calidad del Software

- Tema 2: Perspectivas de Calidad

## 2.1 Calidad de Producto

- Estándares de Calidad de Producto
- Modelos de Calidad
- Calidad de Arquitecturas Software

## 2.2 Calidad de Datos

- Características de calidad de datos

## 2.3 Calidad de Proceso

- Modelos y Estándares de Evaluación y Mejora de Proceso

## 2.4 Calidad en Uso

- Estándares de Calidad en Uso
- Métodos de Evaluación de Usabilidad



# Contenido

## ■ Tema 3: Métricas del Software

- El paradigma Objetivo/Pregunta/Métrica (GQM)
- Definición de Métricas de Calidad
- Validación Teórica
- Validación Empírica

## ■ Tema 4: Gestión, Aseguramiento y Control de la Calidad

- Gestión de la Calidad del Software
- Aseguramiento de la Calidad del Software: actividades y métodos
- Control de la Calidad



# Contenido

- Tema 5: Estimación de Proyectos de Software
  - Conceptos
  - Problemática de la estimación de proyectos
  - Métodos de Medición de Tamaño Funcional
    - FPA, COSMIC
  - Generación de indicadores de productividad, calidad, etc.
  - COCOMO II



# Métricas del Software

- Introducción
- Goal-Question-Metric (GQM)
- Validación Teórica
- Validación Empírica



UNIVERSIDAD  
POLITECNICA  
DE VALENCIA

# Métricas del Software

- Las métricas son un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo de software y los proyectos de mantenimiento (Briand et al., 1996).
- En general, la **medición** persigue tres objetivos fundamentales (Fenton y Pfleeger, 1997):
  - entender qué ocurre durante el desarrollo y el mantenimiento
  - controlar qué es lo que ocurre en los proyectos de desarrollo
  - mejorar los procesos y productos
- Las métricas pueden ser utilizadas para que los profesionales puedan tomar las mejores decisiones.



**MEDIDAS COMO MEDIOS PARA ASEGURAR LA CALIDAD EN LOS PRODUCTOS SOFTWARE**



# Métricas del Software

- Sin embargo, la **medición** ha sido ignorada en la Ingeniería del Software:
  - Todavía fallamos en dar **objetivos medibles** cuando desarrollamos productos software.
    - Por ejemplo, se dice que será amigable, fiable y mantenible, sin especificar qué significa esto en términos medibles.
  - Fallamos al medir diferentes componentes que permiten calcular los costes reales de los proyectos software.
    - Por ejemplo, normalmente no sabemos cuánto tiempo fue realmente invertido en el diseño, comparado con las pruebas.
  - No intentamos cuantificar la calidad de los productos que producimos.
    - Por ejemplo, no podemos decir a un usuario cómo de fiable va a ser un producto en términos de fallos en un periodo dado de uso.



# Métricas del Software

- Solemos ver informes que hacen afirmaciones como que el 80% de los costes del software son de mantenimiento o que hay una media de 55 errores en cada 1.000 líneas de código.
- Sin embargo, no se dice:
  - Cómo se obtuvieron esos resultados,
  - Cómo se diseñaron y ejecutaron los experimentos,
  - Qué entidades fueron medidas y cómo y
  - Cuales fueron los márgenes de error.
- Sin estos datos no podemos **repetir las mediciones de forma objetiva** en otros entornos para tener comparaciones con los estándares de la industria.



# Métricas del Software

- Todos estos problemas derivados de una medición insuficiente se agravan por una falta de una aproximación rigurosa a la medición.
- En general, la producción software está en crisis, tiene costes excesivos, baja productividad y poca calidad.
- Se ha llegado a sugerir que esto es debido a que no medimos.
- En software hay **tres clases de entidades** cuyos **atributos** podemos medir:
  - **Procesos:** actividades del desarrollo del software que normalmente conllevan el factor tiempo.
  - **Productos:** entregables, artefactos o documentos generados en el ciclo de vida del software.
  - **Recursos:** todos aquellos elementos que hacen de entrada a la producción del software.



# Atributos a medir

## ■ Procesos:

- El tiempo (duración del proceso)
- El esfuerzo (asociado al proceso)
- El número de incidentes de un tipo específico que se dan durante el proceso (Ej., el número de errores de requisitos encontrados durante la construcción de la especificación)

## ■ Productos:

- La fiabilidad del código
- La entendibilidad de un documento de especificación
- La mantenibilidad del código fuente
- La longitud, funcionalidad, modularidad o corrección sintáctica de los documentos de especificación

## ■ Recursos:

- El personal
- Los materiales
- Las herramientas y métodos
- El coste
- La productividad



# Ámbito/Utilidad de las Métricas del Software

- **Estimación de Esfuerzo y Costes:** proceso de predecir la cantidad de esfuerzo requerida para construir un producto software. Ejemplo: COCOMO, SLIM. etc.
- **Modelos y medidas de productividad:** el ratio de salida por unidad de entrada.  $Productividad = \text{tamaño} / \text{esfuerzo}$  o  $Productividad = LOC / \text{personas-mes}$ .
- **Modelos y medidas de calidad:** medición de calidad de software. Modelo de McCall, ISO/IEC 9126, SQUARE, etc..
- **Modelos de fiabilidad:** Plot de cambios de intensidad de fallos por tiempo. Ejemplo: Modelo exponencial básico, Modelo logaritmo Poisson.
- **Modelos y evaluación de rendimiento:** uso de características de rendimiento observables (tiempo de respuesta y tasa de complejidad).
- **Métricas de complejidad estructural:** complejidad ciclomática,
- ...



# Ejemplo

- Una empresa quiere iniciar un nuevo servicio de descargas de vídeos y juegos bajo demanda. El servicio será proporcionado a usuarios que tengan PCs y se registran al servicio.
- Los clientes deben usar software especializado para bajar vídeos y juegos del servidor. La **intensidad de fallos** del software es de **1 fallo por 100 CPU/hr**. El sistema software especializado ejecuta **20 CPU/hr por semana en cada máquina cliente** y hay **800 clientes** que usarán el servicio.
- Se quiere proporcionar a los clientes un servicio de reparación. Cada empleado puede realizar 4 llamadas de servicio por día y el servicio está disponible 5 días por semana.
- ¿Cuántos empleados necesitamos contratar?



# Ejemplo

- ¿Cuántos empleados necesitamos contratar?
- Usando el valor de la intensidad de fallos, cada sistema experimenta **0,2 fallos por 20 horas de operación** (o 0,2 fallos por semana de media).
- El total de fallos para 800 clientes es de 160 por semana o 32 por día.
- Cada empleado puede realizar 4 visitas por día. Así, el número de personal es de  **$32/4 = 8$** .



# ¿Qué es la Medición del Software?

- **Medición** es el proceso por el cuál se asignan números o símbolos (**medidas**) a los **atributos** de las **entidades (objetos)** del mundo real.

$$Object = \left\{ \begin{array}{ll} attribute_1 & (value_{11}, value_{12}, \dots) \\ attribute_2 & (value_{21}, value_{22}, \dots) \\ \dots & \dots \\ attribute_n & (value_{n1}, value_{n2}, \dots) \end{array} \right\}$$



# ¿Qué es la Medición del Software?

## ■ Medición del Software:

“Una función que toma como entrada cierta información del software que se está midiendo, y que devuelve como salida un valor numérico, el cual es interpretado como el grado en que el producto software posee un atributo dado que afecta a su calidad”. (IEEE:1992)

## ■ Métrica:

“Un **método** de medición y una **escala cuantitativos** que pueden ser usados para determinar el valor que toma cierta característica en un producto software concreto”.

(ISO/IEC 14598-1:1999)

# ¿Qué es la Medición del Software?

- La medición de software es una disciplina relativamente joven, y no existe consenso general sobre la definición exacta de los conceptos y terminología que maneja.
- Creación de una **ontología de medición** entre:
  - Universidad de Castilla-La Mancha
  - Universidad de Málaga
  - Universidad Nacional de La Pampa
  - Universidad Politécnica de Cataluña
  - Universidad Politécnica de Valencia

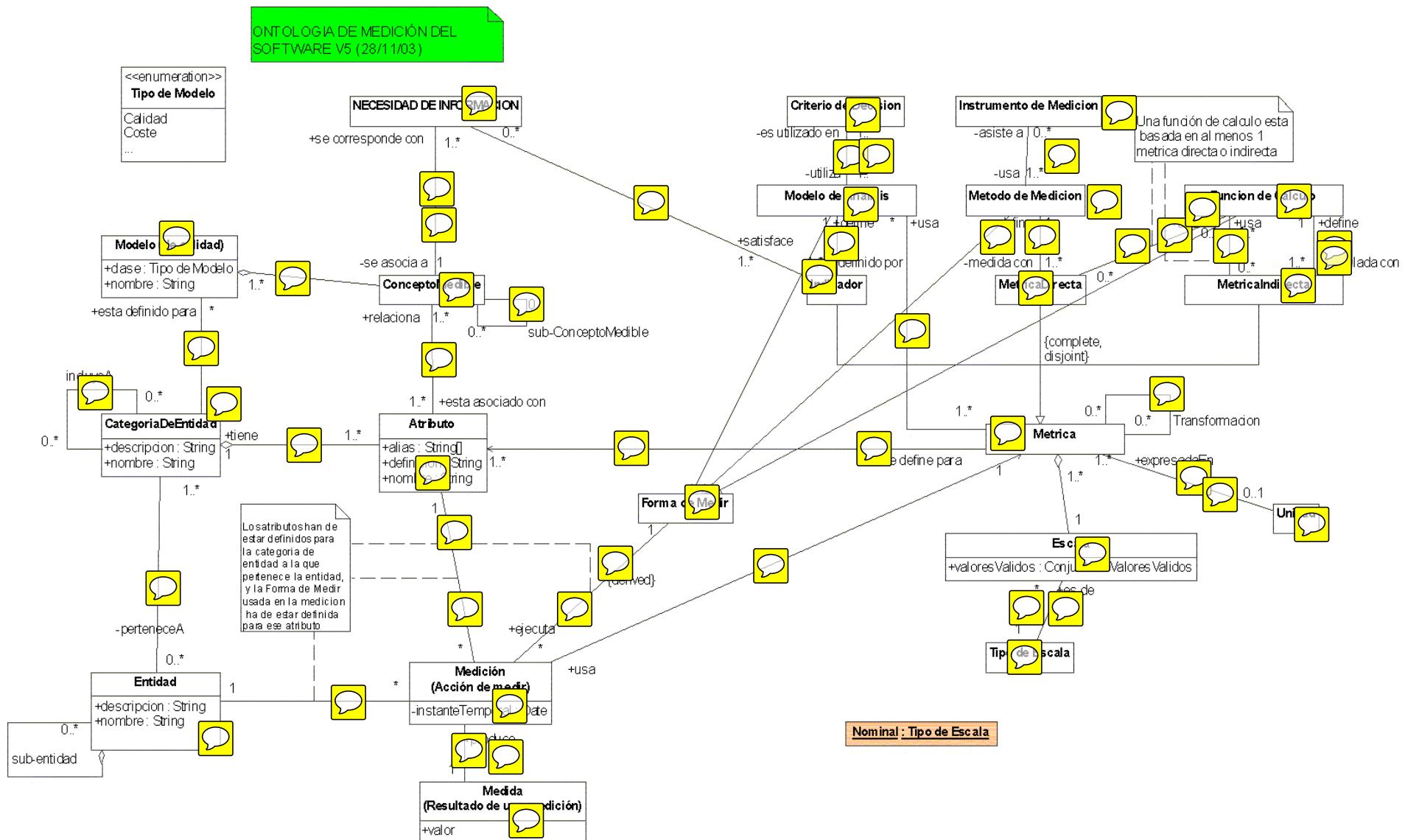


# Ontología de la Medición

- Qué es una ontología? (Gruber, 1995)
  - Una ontología es una especificación de una conceptualización.
- En otras palabras:
  - Mediante la definición de ontologías se pretende reunir y formalizar el conocimiento sobre un determinado dominio de problema.
  - Se mejora el entendimiento y la comunicación mediante el establecimiento de vocabularios comunes, lo que facilita la reutilización y la interoperabilidad de los sistemas software.



# Ontología de la Medición



# Ontología de la Medición

- **Concepto.** ATRIBUTO
- **Definición.** Una propiedad mensurable, física o abstracta, que comparten todas las entidades de una categoría de entidad.
- **Relaciones.**
  - Una medición se realiza sobre los atributos de una entidad
  - Un atributo tiene definida cero, una o varias *métricas*.
  - Un atributo sólo puede pertenecer a una categoría de entidad.
  - Un atributo está relacionado con uno o más conceptos medibles.
- **Ejemplos**
  - El *atributo* “**tamaño de código fuente**”, de “**programas en C**” que es diferente del *atributo* de “**programa en Ada**”.



# Ontología de la Medición

- **Concepto:** MÉTRICA
- **Definición.** Una forma de medir y una escala, definidas para realizar mediciones de uno o varios atributos
- **Relaciones:**
  - Una métrica está definida para uno o más atributos
  - Dos *métricas* pueden relacionarse mediante una función de transformación.
  - El tipo de dicha función de transformación va a depender del tipo de escala de ambas métricas.
  - Una métrica puede expresarse en una unidad (sólo para métricas cuya escala sea de tipo intervalo o ratio)
- **Ejemplos**
  - “**líneas de código**” para el “tamaño” de un “módulo en C” o de un “programa en Ada”.



# Ontología de la Medición

- **Concepto.** MEDIDA
- **Definición.** Resultado de una *medición*.
- **Relaciones**
  - Una medida es el resultado de una medición
- **Ejemplos**
  - 35.000 líneas de código, 200 páginas, 50 clases.
  - 5 meses desde el comienzo al fin del proyecto.
  - 0,5 fallos por cada 1.000 líneas de código.



# Ontología de la Medición

## ■ Ejemplo 1:

- Atributo: tamaño de código fuente
- Entidad: programa C
- Métrica: Líneas de Código (LCF)
- Medida: 10.000 LCF

## ■ Ejemplo 2:

- Atributo: tamaño
- Entidad: diagrama de clases UML
- Métricas: Número de clases.
- Medida: 30 clases

■ **Métricas Directas:** una métrica de un atributo que no depende de ninguna métrica de otro atributo

- LCF (líneas de código fuente escritas)
- HPD (horas-programador diarias)
- CHP (coste por hora-programador, unid. monetarias)
- #def (numero de defectos en la fase de pruebas)

■ **Métricas Indirectas:** una métrica de un atributo que se deriva de una o más métricas de otros atributos.

- LCFH: líneas de código fuente por hora de programador
- CLCF (coste por línea de código fuente) =  $LCF / CTP$
- EffDef (Eficiencia detección defectos) =  $\#def \text{ detectados} / \#total \text{ defectos}$



# Métricas

- Existe un gran número de métricas aunque muy pocas van más allá de su definición y no se usan en la industria.
- Esto se debe a múltiples problemas, entre ellos:
  - Las métricas no se definen siempre en el contexto del objetivo de interés industrial que se pretende alcanzar.
  - No siempre es posible realizar una **validación teórica** adecuada de la métrica porque el atributo que queremos medir no siempre está bien definido.
  - Un gran número de métricas nunca se ha **validado empíricamente**

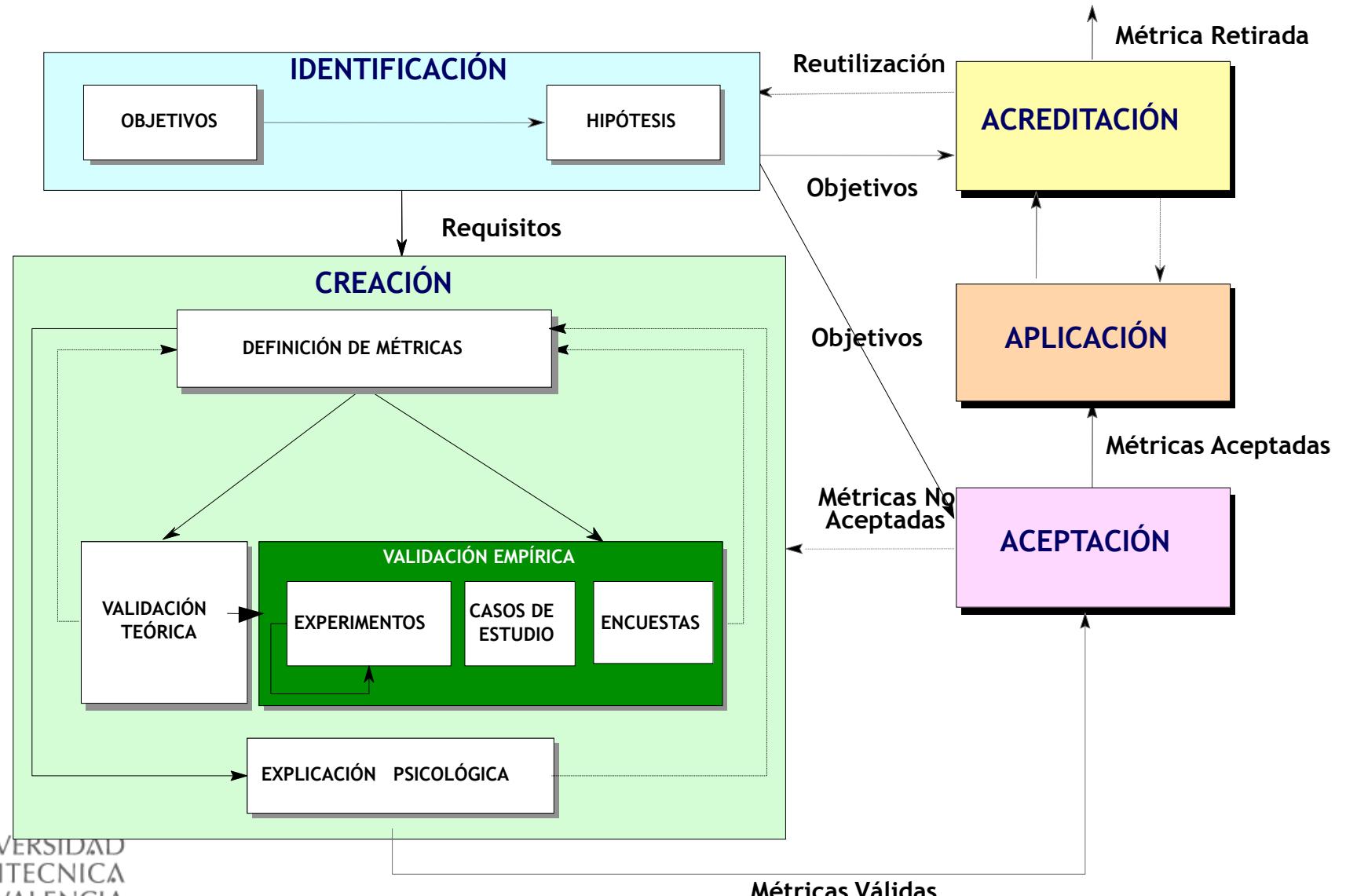


# Métricas

- Esta situación ha conducido ambigüedad en:
  - las definiciones,
  - propiedades y
  - asunciones de las métricas,
- haciendo que:
  - el uso de las mismas sea difícil,
  - la interpretación peligrosa y
  - los resultados contradictorios.
- Para evitarlo es necesario contar con un **método de definición de métricas**.



# Método de Definición de Métricas



# Método de Definición Identificación

- Se **definen los objetivos** que se persiguen a la hora de crear la métrica.
- Se **plantean las hipótesis** de cómo se llevará a cabo la medición.
- Sobre los elementos de esta etapa (objetivos e hipótesis) se basarán todas las etapas siguientes.
- Como resultado de esta etapa se generan los requisitos que debe cumplir la métrica.



# Método de Definición Creación

- Hay que tener en cuenta las características del sistema junto con la experiencia de los diseñadores.
- Hay que tener un **objetivo concreto** para evitar la definición de métricas que en realidad no capturen el objetivo perseguido (GQM: *Goal-Question-Metric*<sup>\*</sup>).

\* Basili and Rombach's Goal-Question-Metrics paradigm, *IEEE Transactions on Software Engineering*, 1988 paper on the TAME project.

# Goal-Question Metric (GQM)

- Introducción
- Ejemplo



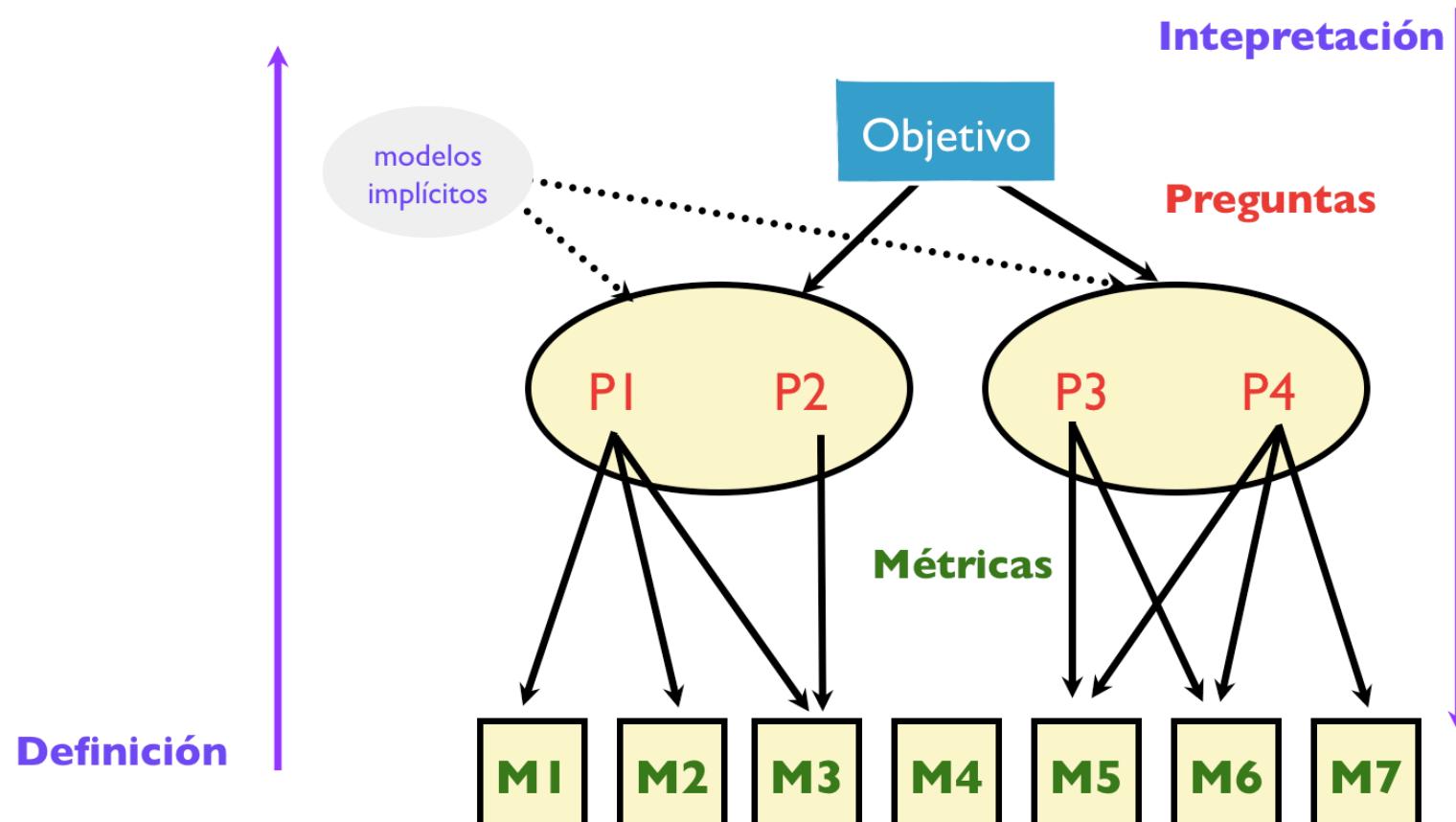
UNIVERSIDAD  
POLITECNICA  
DE VALENCIA

# Goal Question Metric (GQM)

- Es una técnica definida por Basili y Weiss (1984) y Rombach (1990), para seleccionar y generar métricas tanto del proceso como de los resultados de un proyecto.
- Principio básico: la medición debe ser realizada siempre orientada a un objetivo.
- Define un **objetivo**, el cual se refina en **preguntas** y define **métricas** que intentan dar información para responder a estas preguntas.
- Conformado por 3 niveles:
  - Nivel conceptual (Objetivos - *Goals*)
  - Nivel operacional (Preguntas - *Questions*)
  - Nivel cuantitativo (Métricas - *Metrics*)



# Goal Question Metric (GQM)



# Goal-Question-Metric

## ■ Definir los objetivos de la medición

<b>Analizar</b>	el objeto bajo medición
<b>Con el propósito de</b>	entender, controlar, o mejorar el objeto
<b>Con respecto a</b>	el enfoque de calidad del objeto en el que se centra la medición
<b>Desde el punto de vista de</b>	las personas que miden el objeto
<b>En el contexto de</b>	el entorno en el que la medición tiene lugar

## ■ Definir preguntas e hipótesis

- Con la respuesta a las preguntas planteadas, se debería poder concluir si se cumple un determinado objetivo.

## ■ Definir las métricas

- Deben proporcionar información cuantitativa que permita responder las preguntas planteadas.



# Ejemplo 1

## Métricas para BD Relacionales

### ■ Objetivo GQM:

- |   |                          |
|---|--------------------------|
| <input type="checkbox"/> <i>Analizar</i>                | BD Relacionales          |
| <input type="checkbox"/> <i>Con el propósito de</i>     | Asegurar                 |
| <input type="checkbox"/> <i>Con respecto a la</i>       | Mantenibilidad           |
| <input type="checkbox"/> <i>Desde el punto de vista</i> | De los diseñadores de BD |
| <input type="checkbox"/> <i>En el contexto de</i>       | BD Relacional            |

### ■ Preguntas:

- Pregunta 1: ¿Cómo influye la complejidad de las tablas en la mantenibilidad de las Bases de Datos relacionales?
- Pregunta 2: ¿Cómo influye la complejidad entre tablas en la mantenibilidad de las Bases de Datos relacionales?



# Ejemplo 1

## Métricas para BD Relacionales

### ■ Métricas:

#### ▪ **Pregunta 1**

- **NA(T)** - NÚMERO DE ATRIBUTOS DE UNA TABLA
- **NFK(T)** - NÚMERO DE CLAVES AJENAS
- **RFK(T)** - RATIO DE CLAVES AJENAS DE UNA TABLA

$$RFK(T) = \frac{NFK(T)}{NA(T)}$$

#### ▪ **Pregunta 2**

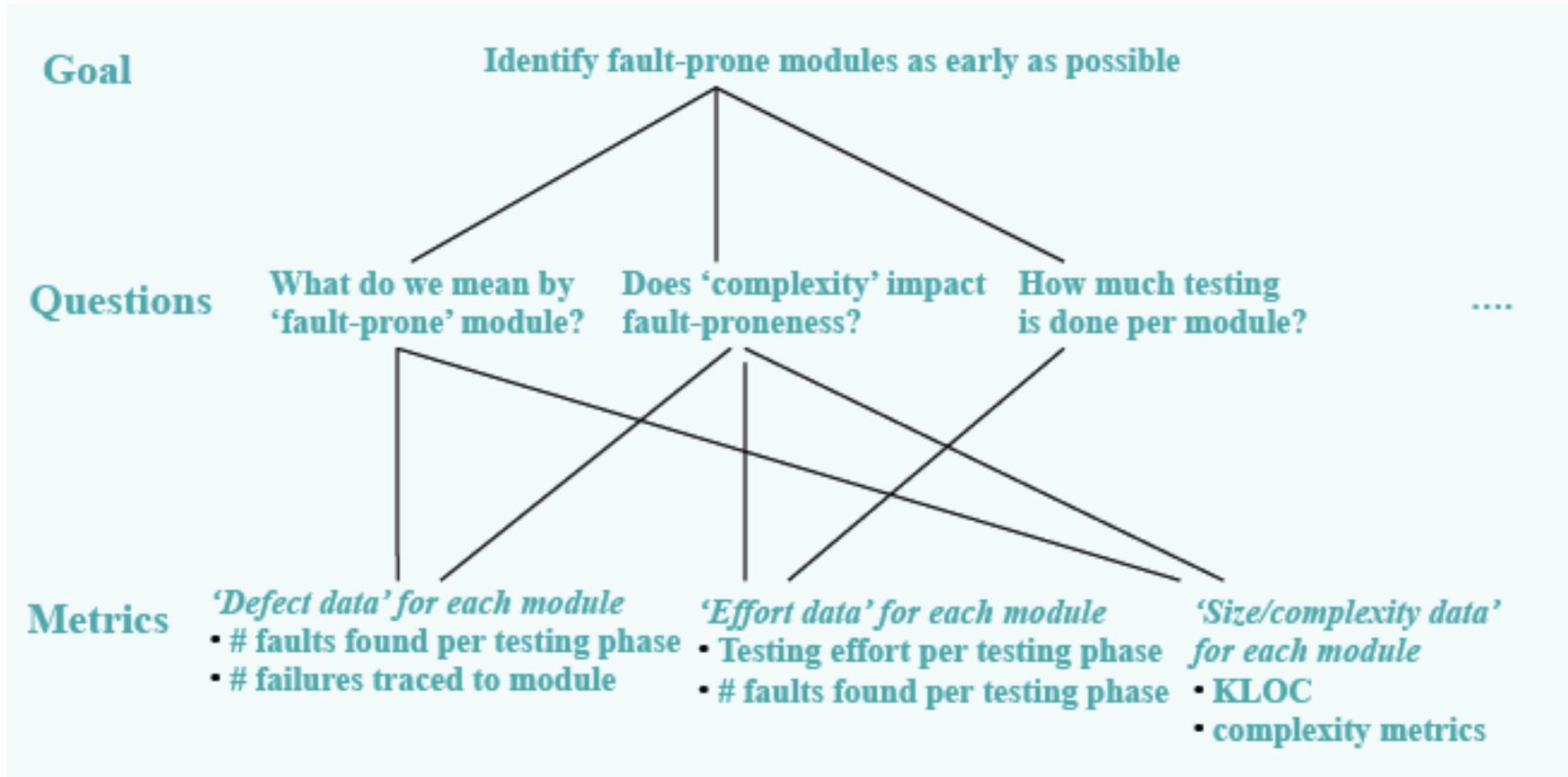
- **NT** - NÚMERO DE TABLAS
- **NA** - NÚMERO DE ATRIBUTOS
- **NFK** - NÚMERO DE CLAVES AJENAS (NFK)



UNIVERSIDAD  
POLITECNICA  
DE VALENCIA

# Ejemplo 2

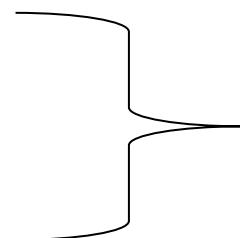
## Identificación de módulos propenso a errores



# Las Escalas de Medida

## Cuatro grandes tipos:

- Nominal
- Ordinal
- Intervalo
- Razón



- El tipo de escala de medida que adoptan los datos, determina el **tipo de operaciones aritméticas** que se pueden realizar con ellos y, por tanto, también el **tipo de análisis estadístico**.
- Las escalas de intervalo y de razón aparecen, a veces, definidas como “cuantitativas”.



UNIVERSIDAD  
POLITECNICA  
DE VALENCIA

# Escala Nominal

- Sólo permiten la **clasificación** o **diferenciación** de los objetos.
- Permiten establecer relaciones de igualdad o desigualdad entre dos o más objetos.
- Las variables que adoptan este nivel de medida se denominan “cualitativas”.
- Con los “números” de este tipo de variables **no** se pueden efectuar operaciones aritméticas.
- Ejemplo:
  - Clasificación de fallos en el software (fallo de especificación, fallo de diseño, fallo de codificación)

# Escala Ordinal

- Los objetos son **jerarquizados** conforme algún criterio.
- Los números que se utilizan para codificar las distintas categorías de una variable sólo permiten establecer relaciones de igualdad/desigualdad y de **orden**.
- No se puede precisar la diferencia exacta que existe entre dos objetos.
- Con los “números” de este tipo de variables **no** se pueden efectuar operaciones aritméticas.
- Ejemplo:
  - Medición de complejidad de módulos de software considerando 5 clases de complejidad: trivial, simple, moderada, compleja, incomprensible.



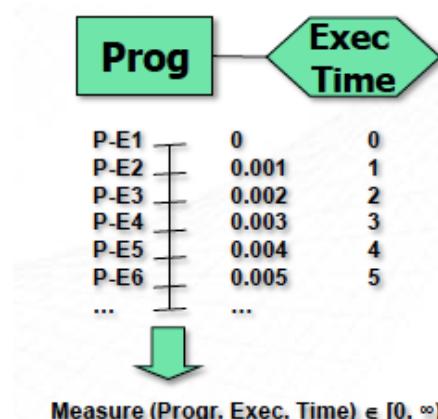
# Escala de Intervalo

- Se pueden observar ciertas **diferencias cuantitativas** entre las unidades.
- Se pueden establecer relaciones de igualdad/desigualdad, de orden y además **los intervalos entre los distintos números o valores son iguales**.
- No tiene principio ni final (no existe cero absoluto). Son números arbitrarios en cuanto al origen.
- Con los números de este tipo de variables, se pueden realizar operaciones aritméticas como la suma o la resta, pero no la división ni la multiplicación.
- Ejemplo: *Medición de duración de las distintas fases de un proyecto*
  - *Project scheduling*: Análisis de Requisitos 3 semanas, Diseño 4 semanas, Codificación 4 semanas, Pruebas empieza cuando termina la codificación. ¿Cuándo empieza las pruebas? *Después de 11 semanas*.



# Escala de Razón

- Se pueden establecer relaciones de igualdad/desigualdad, de orden, los intervalos entre los distintos valores son iguales y se cuenta con un **verdadero punto cero absoluto**, en relación con el cual se expresan todos los demás valores.
- El valor cero representa el **origen empírico** de la variable, la carencia total de cierta característica.
- Se pueden efectuar todas las operaciones aritméticas (suma, resta, división y multiplicación).
- Ejemplos:
  - Medición del tiempo de ejecución de un programa
  - Líneas de Código (como medidas del tamaño de un programa)



# Validación Teórica

- Introducción
- Aproximaciones Axiomáticas
- Aproximaciones basadas en la Teoría de la Medición



UNIVERSIDAD  
POLITECNICA  
DE VALENCIA

# Método de Definición

## Validación Teórica

- Nos ayuda a saber cuándo y cómo aplicar las métricas.
- Nos proporciona información relativa a las **operaciones realizables con la métrica**.
- Tendencias:
  - Aproximaciones basadas en propiedades
  - Aproximaciones basadas en la teoría de la medición
- Desafortunadamente, no disponemos de un estándar (Van Den Berg y Van Den Broek, 1996).



# Método de Definición

## Validación Teórica

### Aproximaciones Axiomáticas

- Son más sencillas pero de menor utilidad.
- Se limitan a suministrar condiciones necesarias pero no suficientes para los conceptos que definen.
- Sirven para clasificar las métricas por tipos.
- Los marcos formales más conocidos dentro de este tipo son los propuestos por Weyuker (1988), Briand et al. (1996), Morasca y Briand (1997) y Poels and Dedene (1999).



# Método de Definición Validación Teórica

## ■ Marco de Briand et al. (1996) y (1997)

Tamaño
Longitud
Complejidad
Cohesión del sistema
Acoplamiento del sistema



# Método de Definición

## Validación Teórica

- Un sistema es caracterizado por sus *elementos* y *relaciones* entre ellos. Un sistema  $S$  se representa como un par  $\langle E, R \rangle$ .
- El **tamaño de un sistema**  $S$  es una función  $Tamaño(S)$  que queda caracterizada por las siguientes propiedades:
  - **Propiedad 1. No negatividad**  
El tamaño de un sistema  $S = \langle E, R \rangle$  es no negativo.
  - **Propiedad 2. Valor nulo**  
El tamaño de un sistema  $S = \langle E, R \rangle$  es nulo si está vacío.
  - **Propiedad 3. Aditividad de módulos**  
El tamaño de un sistema  $S = \langle E, R \rangle$  es igual a la suma de los tamaños de dos de sus módulos  $m_i = \langle E_{mi}, R_{mi} \rangle$  y  $m_j = \langle E_{mj}, R_{mj} \rangle$  tal que cualquier elemento de  $S$  es un elemento de  $m_i$  o de  $m_j$ .



# Método de Definición

## Validación Teórica

- La **longitud de un sistema\***  $S$  es una función Longitud( $S$ ) caracterizada por las siguientes propiedades:
  - **Propiedad 1. No negatividad**  
La longitud de un sistema  $S = \langle E, R \rangle$  es no negativa
  - **Propiedad 2. Valor nulo**  
La longitud de un sistema  $S = \langle E, R \rangle$  es nula si  $E$  está vacío.

### \*Métricas de Halstead

Ejemplos de cálculo para las métricas de Halstead:  
<http://cnx.org/content/m18034/latest>

# Método de Definición

## Validación Teórica

### Propiedad 3. Monotonicidad no incremental para componentes conexos.

Sea  $S$  un sistema y  $m$  un módulo de  $S$  tal que  $m$  está representado como un componente conexo del grafo de  $S$ . La adición de relaciones entre elementos de  $m$  no incrementa la longitud de  $S$ .

### Propiedad 4. Monotonicidad no decreciente para componentes no conexos.

Sea  $S$  un sistema y  $m_1$  y  $m_2$  dos módulos de  $S$  tal que  $m_1$  y  $m_2$  están representados por dos componentes conectados separados en el grafo que representa  $S$ . La adición de relaciones desde elementos de  $m_1$  a elementos de  $m_2$  no decrementa la longitud de  $S$

### Propiedad 5. Módulos disjuntos

La longitud de un sistema  $S = \langle E, R \rangle$  compuesto de dos módulos disjuntos  $m_1, m_2$  es igual al máximo de las longitudes de  $m_1$  y  $m_2$ .

# Ejemplo de Métrica de Longitud

## Métricas de Halstead

- Su teoría está basada en un conteo (fácil de automatizar) de **número de operadores y operandos en un programa**:
  - los operadores son las palabras reservadas del lenguaje, tales como IF-THEN, READ, FOR,...; los operadores aritméticos +, -, \*,..... los de asignación y los operadores lógicos AND, EQUAL TO,....
  - los operandos son las variables, literales y las constantes del programa.
- Halstead distingue entre el # operadores y # operandos únicos y el # total de operadores y operandos. Se utiliza la notación:
  - $n_1$  - número de operadores únicos que aparecen en un programa
  - $N_1$  - número total de ocurrencias de operadores
  - $n_2$  - número de operandos únicos que aparecen en un programa
  - $N_2$  - número total de ocurrencias de operandos



# Ejemplo de Métrica de Longitud

## Métricas de Halstead

- Las métricas de Halstead para cualquier programa escrito en cualquier lenguaje pueden ser derivadas de estas cuatro cuentas ( $n_1$ ,  $N_1$ ,  $n_2$  y  $N_2$ ). A partir de ellas han sido elaboradas diferentes medidas para diversas propiedades de los programas: longitud, volumen, etc...
- La **longitud,  $N$ , de un programa**, se calcula como:  
$$N = N_1 + N_2$$
- El **volumen,  $V$ , de un programa**, se calcula como:  
$$V = N \times \log_2(n)$$
  
donde  $n = n_1 + n_2$

# Ejemplo calculo Métricas de Halstead

```
{  
    for (i=2;i<=n;i++)  
        for (j=1;j<=i;j++)  
            if (x[i] < x[j])  
            {  
                aux = x[i];  
                x[i] = x[j];  
                x[j] = aux;  
            }  
}
```

- Operadores:  
 $\{..\} \rightarrow 2, \text{for}(;) \rightarrow 2, = \rightarrow 5, \text{if} \rightarrow 1, ; \rightarrow , (..) \rightarrow 1, < \rightarrow 1, <= \rightarrow 2, ++ \rightarrow 2, [] \rightarrow 4$
- # total de operadores ( $n_1$ ) son 10 y el # de operadores ( $N_1$ ) son 23.
- Operandos:  
 $\rightarrow 7, n \rightarrow 1, j \rightarrow 6, x \rightarrow 6, \text{aux} \rightarrow 2$
- # total de operandos ( $n_2$ ) son 5 y el # total ( $N_2$ ) son 22.
- La longitud es:  $N = 23 + 22 = 45$
- El volumen es:  $45 \times \log_2(10+5) = 175,8$



# Método de Definición

## Validación Teórica

- La **complejidad** de un sistema  $S$  es una función  $\text{Complejidad}(S)$  que está caracterizada por las siguientes propiedades:
  - **Propiedad 1. No negatividad**  
La complejidad de un sistema  $S = \langle E, R \rangle$  es no negativa.
  - **Propiedad 2. Valor nulo**  
La complejidad de un sistema  $S = \langle E, R \rangle$  es nula si  $R$  está vacío.



# Método de Definición

## Validación Teórica

### Propiedad 3. Simetría

La complejidad de un sistema  $S = \langle E, R \rangle$  no depende de la convención elegida para representar las relaciones entre sus elementos.

### Propiedad 4. Monotonidad de módulos

La complejidad de un sistema  $S = \langle E, R \rangle$  no es menor que la suma de las complejidades de cualquiera dos de sus módulos sin relaciones en común.

### Propiedad 5. Aditividad de módulos disjuntos

La complejidad de un sistema  $S = \langle E, R \rangle$  compuesto de dos módulos disjuntos  $m_1$  y  $m_2$  es igual a la suma de las complejidades de los dos módulos.

# Ejemplo de Métrica de Complejidad

## Complejidad ciclomática de McCabe

- Se basa en el recuento del número de caminos lógicos individuales contenidos en un programa. Para calcular la métrica, McCabe utilizó la teoría y flujo de grafos.
- Para hallar la complejidad ciclomática, el programa se representa como un grafo, y cada instrucción que contiene, un nodo del grafo.
- Las posibles vías de ejecución a partir de una instrucción (nodo) se representan en el grafo como aristas.
- Ejemplo:

[http://jbravomontero.files.wordpress.com/2012/07/grafica\\_complejidad\\_ciclomatica\\_metodo.png](http://jbravomontero.files.wordpress.com/2012/07/grafica_complejidad_ciclomatica_metodo.png)

# Ejemplo de Métrica de Complejidad

## Complejidad ciclomática de McCabe

```
1 if (condicion){  
2   if (condicion){  
3     A;  
4     B;  
5   } else {  
6     C;  
7     D;  
8   }  
9 }
```

- Si se realizase el grafo, se observaría que se encuentran 3 caminos posibles para llegar de la sentencia 1 a la sentencia 6:
  - Camino 1 (si ambos IF's son verdad): Sentencias 1, 2, 3, 6
  - Camino 2 (si el primer IF es verdad y el segundo es falso): Sentencias 1, 4, 6
  - Camino 3 (si el primer IF es falso): Sentencias 1, 6
- Este programa tiene una complejidad ciclomática de 3.
- La complejidad ciclomática se puede calcular de otras maneras, e.j., con la fórmula:

$$v(G) = e - n + 2$$

donde  $e$  representa el #de aristas y  $n$  el #de nodos.



# Método de Definición

## Validación Teórica

La **cohesión\*** de un módulo  $m = \langle E_m, R_m \rangle$  de un sistema modular  $MS$  es una función  $Cohesión(m)$  caracterizada por las siguientes propiedades.

### Propiedad 1. No negatividad y normalización

La cohesión de un módulo  $m = \langle E_m, R_m \rangle$  de un sistema modular  $MS = \langle E, R, M \rangle$  pertenece a un intervalo especificado.

### Propiedad 2. Valor nulo

La cohesión de un módulo  $m = \langle E_m, R_m \rangle$  de un sistema modular  $MS = \langle E, R, M \rangle$  es nula si  $R_m$  está vacío.

\*Se refiere al grado de "**adhesivo interno**" con el que se ha construido el módulo/componente. Un modulo es cohesivo si todos sus elementos están orientados a la realización de una única tarea y son esenciales para llevarla a cabo.



# Método de Definición

## Validación Teórica

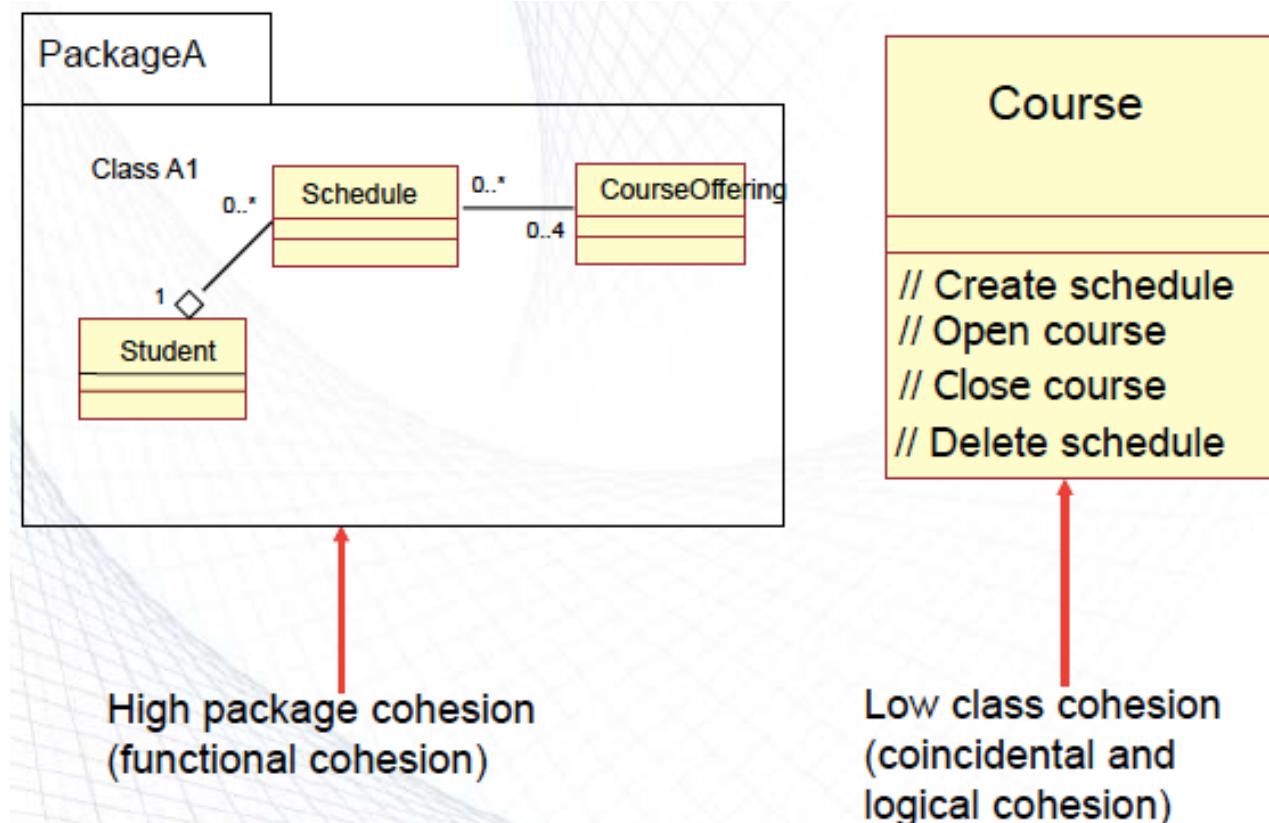
### Propiedad 3. Monotonidad

Sea  $MS' = \langle E, R', M' \rangle$  y  $MS'' = \langle E, R'', M'' \rangle$  dos sistemas modulares (con el mismo conjunto de elementos  $E$ ) tal que existen dos módulos  $m' = \langle E_m, R'_m \rangle$  y  $m'' = \langle E_m, R''_m \rangle$  (con el mismo conjunto de elementos  $E_m$ ) pertenecientes a  $M'$  y  $M''$  respectivamente, tal que  $R' - R'_m = R'' - R''_m$ , y  $R'_m \subseteq R''_m$  (que implica que  $IR' \subseteq IR''$ ). Entonces,  $Cohesión(m') \leq Cohesión(m'')$

### Propiedad 4. Módulos con cohesión

Sean  $MS' = \langle E, R, M' \rangle$  y  $MS'' = \langle E, R, M'' \rangle$  dos sistemas modulares (con el mismo sistema subyacente  $\langle E, R \rangle$ ) tal que  $M'' = M' - \{m'_1, m'_2\} \cup \{m''\}$ , con  $m'_1 \in M'$ ,  $m'_2 \in M'$ ,  $m'' \notin M'$ , y  $m'' = m'_1 \cup m'_2$ . (Los dos módulos  $m'_1$  y  $m'_2$  son sustituidos por el módulo  $m''$ , unión de  $m'_1$  y  $m'_2$ ). Si no existen relaciones entre elementos pertenecientes a  $m'_1$  y  $m'_2$ , esto es,  $InputR(m'_1) \cap OutputR(m'_2) = \emptyset$  e  $InputR(m'_2) \cap OutputR(m'_1) = \emptyset$ , entonces  $\max\{Cohesión(m'_1), Cohesión(m'_2)\} \geq Cohesión(m'')$

# Ejemplos de Cohesión



\*Se refiere al grado de **"adhesivo" interno** con el que se ha construido el módulo/componente. Un modulo es cohesivo si todos sus elementos están orientados a la realización de una única tarea y son esenciales para llevarla a cabo.



# Método de Definición

## Validación Teórica

El **acoplamiento\*** de un módulo  $m = \langle E_m, R_m \rangle$  de un sistema modular  $MS$  es una función  $\text{Acoplamiento}(m)$  caracterizada por las siguientes propiedades:

### Propiedad 1. No negatividad

El acoplamiento de un módulo  $m = \langle E_m, R_m \rangle$  de un sistema modular es no negativo.

### Propiedad 2. Valor nulo.

El acoplamiento de un módulo  $m = \langle E_m, R_m \rangle$  de un sistema modular es nulo si  $\text{OuterR}(m)$  está vacío.

### Propiedad 3. Monotonidad.

Sean  $MS' = \langle E, R', M' \rangle$  y  $MS'' = \langle E, R'', M'' \rangle$  dos sistemas modulares (con el mismo conjunto de elementos  $E$ ) tal que existen dos módulos  $m' \in M'$ ,  $m'' \in M''$  tal que  $R' - \text{OuterR}(m') = R'' - \text{OuterR}(m'')$ , y  $\text{OuterR}(m') \subseteq \text{OuterR}(m'')$ . Entonces,  $\text{Acoplamiento}(m') \leq \text{Acoplamiento}(m'')$

\*Se dice que dos módulos, componentes (o clases) están **altamente acoplados** cuando existe mucha dependencia entre ellos. Los módulos **poco acoplados**, tienen algunas dependencias, pero son débiles.

# Método de Definición

## Validación Teórica

### Propiedad 4. Fusión de módulos

Sean  $MS' = \langle E', R', M' \rangle$  y  $MS'' = \langle E'', R'', M'' \rangle$  dos sistemas modulares tales que  $E' = E''$ ,  $R' = R''$ , y  $M'' = M' - \{m'_1, m'_2\} \cup \{m''\}$ , donde  $m'_1 = \langle E_{m'_1}, R_{m'_1} \rangle$ ,  $m'_2 = \langle E_{m'_2}, R_{m'_2} \rangle$ , y  $m'' = \langle E_{m''}, R_{m''} \rangle$  con  $m'_1 \in M'$ ,  $m'_2 \in M'$ ,  $m'' \notin M'$ , y  $E_{m''} = E_{m'_1} \cup E_{m'_2}$  y  $R_{m''} = R_{m'_1} \cup R_{m'_2}$ . (Los dos módulos  $m'_1$  y  $m'_2$  son sustituidos por el módulo  $m''$ , cuyos elementos y relaciones son la unión de los de  $m'_1$  y  $m'_2$ ). Entonces  $\text{Acoplamiento}(m'_1) + \text{Acoplamiento}(m'_2) \geq \text{Acoplamiento}(m'')$

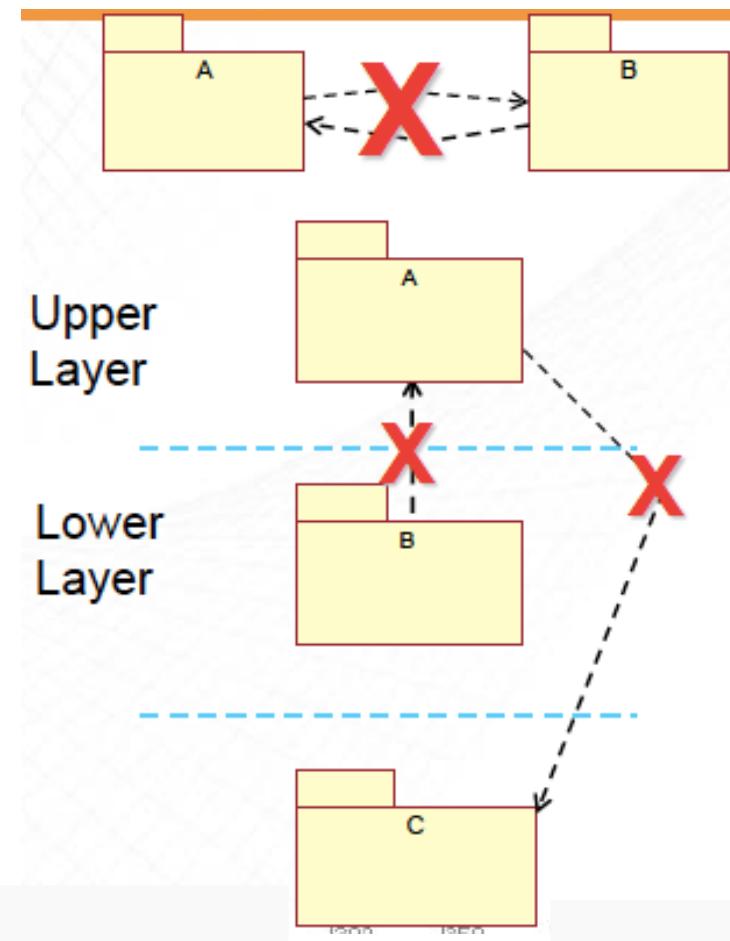
### Propiedad 5. Aditividad de módulos disjuntos

Sean  $MS' = \langle E, R, M' \rangle$  y  $MS'' = \langle E, R, M'' \rangle$  dos sistemas modulares (con el mismo sistema subyacente  $\langle E, R \rangle$ ) tal que  $M'' = M' - \{m'_1, m'_2\} \cup \{m''\}$ , con  $m'_1 \in M'$ ,  $m'_2 \in M'$ ,  $m'' \notin M'$ , y  $m'' = m'_1 \cup m'_2$ . Si no existen relaciones entre elementos pertenecientes a  $m'_1$  y  $m'_2$ , esto es,  $\text{InputR}(m'_1) \cap \text{OutputR}(m'_2) = \emptyset$  e  $\text{InputR}(m'_2) \cap \text{OutputR}(m'_1) = \emptyset$ , entonces  $\text{Acoplamiento}(m'_1) + \text{Acoplamiento}(m'_2) = \text{Acoplamiento}(m'')$



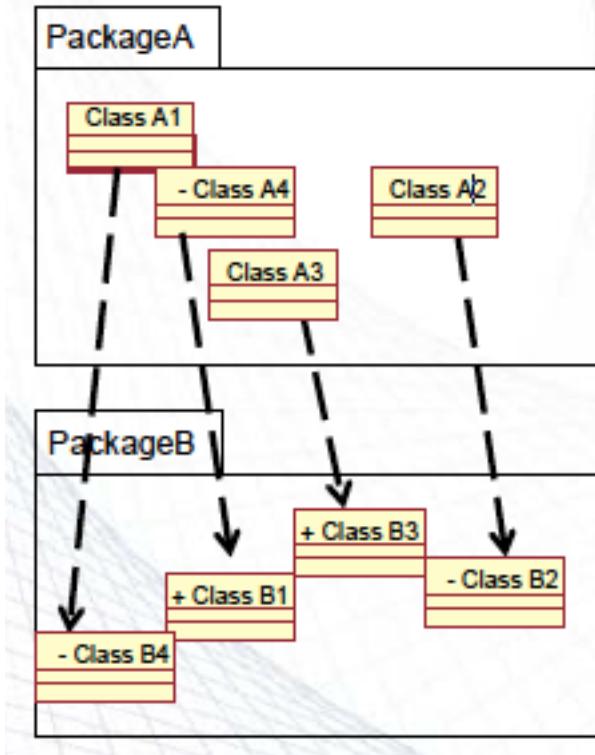
# Ejemplos de Acoplamiento

- Acoplamiento describe la fuerza con un elemento se relaciona con otro elemento.
- El objetivo es lograr “bajo acoplamiento”.
- Ejemplos de alto acoplamiento en el diseño OO:
  - Paquetes no deben tener dependencias entre sí.
  - Paquetes en capas más bajas no deben depender de paquetes que están en capas más altas.
  - Por lo general, las dependencias no deben saltar capas (al menos que se haya especificado en la arquitectura del sistema).

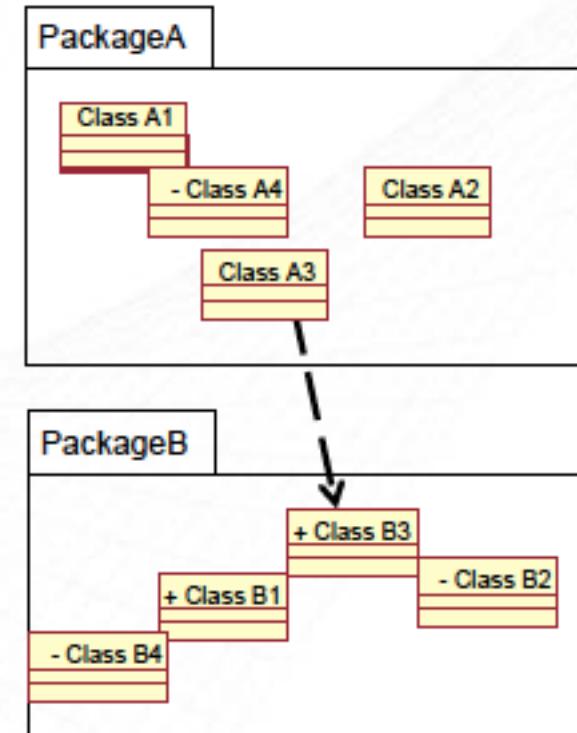


# Ejemplos de Acoplamiento

- Intente alcanzar el acoplamiento más bajo.



Alto acoplamiento



Bajo acoplamiento



# Método de Definición

## Validación Teórica

### Aproximaciones basadas en la Teoría de la Medición

- Aparato matemático más complejo
- Su objetivo es obtener la escala matemática a la que pertenece una métrica, y por tanto sus transformaciones admisibles, estadísticos y tests estadísticos aplicables.
- Se logra extraer más información al validar una métrica en un marco formal de este tipo.
- Los marcos formales más conocidos son los propuestos por Zuse (1998) o Whitmire (1997).

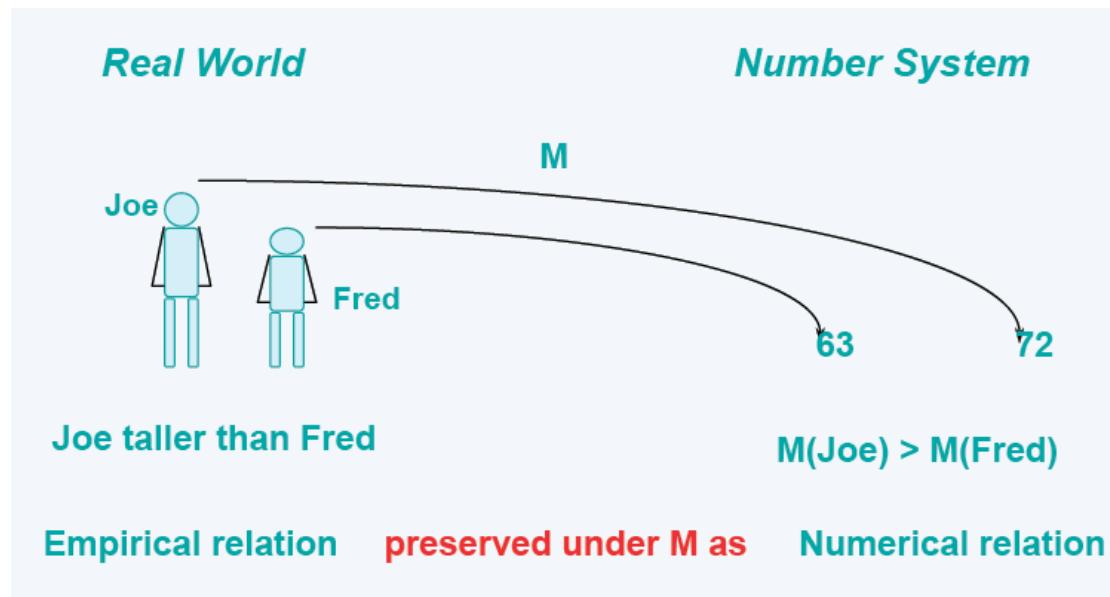


# Método de Definición

## Validación Teórica

### Aproximaciones basadas en la Teoría de la Medición

- Fundamentalmente, en la teoría de la medición se identifica tres componentes principales: un sistema relacional empírico, un sistema relacional numérico y la correspondencia (mapping) entre ambos sistemas relationales.



# Método de Definición

## Validación Teórica

### ■ Marco de Zuse (1998)

- En el área de la ingeniería del software existen muchas posibilidades de concatenar los objetos de interés (módulos de programas, clases del modelo de objetos, etc.).
- Tales **operaciones de concatenación** permiten definir diversas estructuras de medición.
- Zuse se basa en las estructuras extensivas, a continuación introducimos este tipo de estructuras.



# Método de Definición

## Validación Teórica

### ■ Marco de Zuse (1998)

#### Estructura extensiva modificada

Sea  $A$  un conjunto no vacío,  $\bullet \geq$  una relación binaria en  $A$  y  $\circ$  una operación binaria cerrada sobre  $A$ . El sistema relacional  $(A, \bullet \geq, \circ)$  es una estructura modificada extensiva si y sólo si los siguientes axiomas se cumplen para todos  $A_1, \dots, A_4 \in A$ .

- Axioma1:  $(A, \bullet \geq)$  (orden débil)
- Axioma2:  $A_1 \circ A_2 \bullet \geq A_1$  (positividad)
- Axioma3:  $A_1 \circ (A_2 \circ A_3) \approx (A_1 \circ A_2) \circ A_3$  (asociatividad débil)
- Axioma4:  $A_1 \circ A_2 \approx A_2 \circ A_1$  (comutatividad débil)
- Axioma5:  $A_1 \bullet \geq A_2 \Rightarrow A_1 \circ A \bullet \geq A_2 \circ A$  (monotonicidad débil)
- Axioma6: Si  $A_3 \bullet > A_4$  entonces para cualquier  $A_1, A_2$ , existe un número natural  $n$ , tal que  $A_1 \circ nA_3 \bullet > A_2 \circ nA_4$  (axioma arquimedeano)



# Método de Definición

## Validación Teórica

### Reglas de combinación

Si una medida  $u$  está en una escala ordinal  $((A, \bullet\geq), (\Re, \geq), u)$  entonces, una regla de combinación  $f$  se define como:  $\mu(A_1 \circ A_2) = f(\mu(A_1), \mu(A_2))$ , donde  $A_1, A_2 \in A$  es un conjunto de objetos,  $\circ$  es una operación binaria (operación de concatenación) y  $u$  es una medida.

### condiciones de independencia

$$C1: A_1 \approx A_2 \Rightarrow A_1 \circ A_3 \approx A_2 \circ A_3 \text{ y } A_1 \approx A_2 \Rightarrow A_3 \circ A_1 \approx A_3 \circ A_2$$

$$C2: A_1 \approx A_2 \Leftrightarrow A_1 \circ A_3 \approx A_2 \circ A_3 \text{ y } A_1 \approx A_2 \Leftrightarrow A_3 \circ A_1 \approx A_3 \circ A_2$$

$$C3: A_1 \bullet\geq A_2 \Rightarrow A_1 \circ A_3 \bullet\geq A_2 \circ A_3, \text{ y } A_1 \bullet\geq A_2 \Rightarrow A_3 \circ A_1 \bullet\geq A_3 \circ A_2$$

$$C4: A_1 \bullet\geq A_2 \Leftrightarrow A_1 \circ A_3 \bullet\geq A_2 \circ A_3, \text{ y } A_1 \bullet\geq A_2 \Leftrightarrow A_3 \circ A_1 \bullet\geq A_3 \circ A_2$$

# Método de Definición

## Validación Teórica

### Estructura modificada de creencia

MRB1:  $\forall A, B \in \mathfrak{J}: A \bullet \geq B \text{ o } B \bullet \geq A$  (compleción)

MRB2:  $\forall A, B, C \in \mathfrak{J}: A \bullet \geq B \text{ y } B \bullet \geq C \Rightarrow A \bullet \geq C$   
(transitividad)

MRB3:  $\forall A \subseteq B \Rightarrow A \bullet \geq B$  (axioma de dominancia)

MRB4:  $\forall (A \supset B, A \cap C = \emptyset) \Rightarrow (A \bullet \geq B \Rightarrow A \cup C \bullet > B \cup C)$   
(monotonidad parcial)

MRB5:  $\forall A \in \mathfrak{J}: A \bullet \geq 0$  (positividad)



# Método de Definición

## Validación Teórica

- Si una métrica cumple el *orden débil*, puede ser clasificada en la **escala ordinal**.
- Si una métrica cumple la *estructura modificada extensiva* y además cumple *las condiciones de independencia* entonces podrá ser clasificada dentro de la **escala de ratio**.
- Si una métrica *no satisface la estructura extensiva* pero si *las condiciones de independencia* puede ser clasificada dentro de la **escala ordinal**.
- Si una métrica cumple la *estructura modificada de creencia*, podrá ser caracterizada **por encima de la escala ordinal** pero sin llegar a la escala de ratio.



# Validación Empírica

- Métodos Experimentales
- Ejemplo de Validación Empírica de Métricas



UNIVERSIDAD  
POLITECNICA  
DE VALENCIA

# Método de Definición Validación Empírica

- Se utiliza para obtener información objetiva sobre la utilidad de las métricas propuestas
- El estudio empírico resulta necesario para comprobar y entender las implicaciones de las medidas de nuestros productos
- Esto se consigue a través de hipótesis en el mundo real que habrá que comprobar con datos empíricos



# Método de Definición Validación Empírica

Kish (1959) divide las investigaciones empíricas en:

- **Experimentos:** son las investigaciones en las que las posibles variables perturbadoras han sido aleatorizadas.
- **Casos de estudio:** son aquellos en los que no hay aleatoriedad de variables perturbadoras ni representatividad de los sujetos que componen la muestra de estudio
- **Encuestas (Surveys):** son investigaciones en las que los sujetos del estudio son una muestra representativa de la población a la que pertenecen.

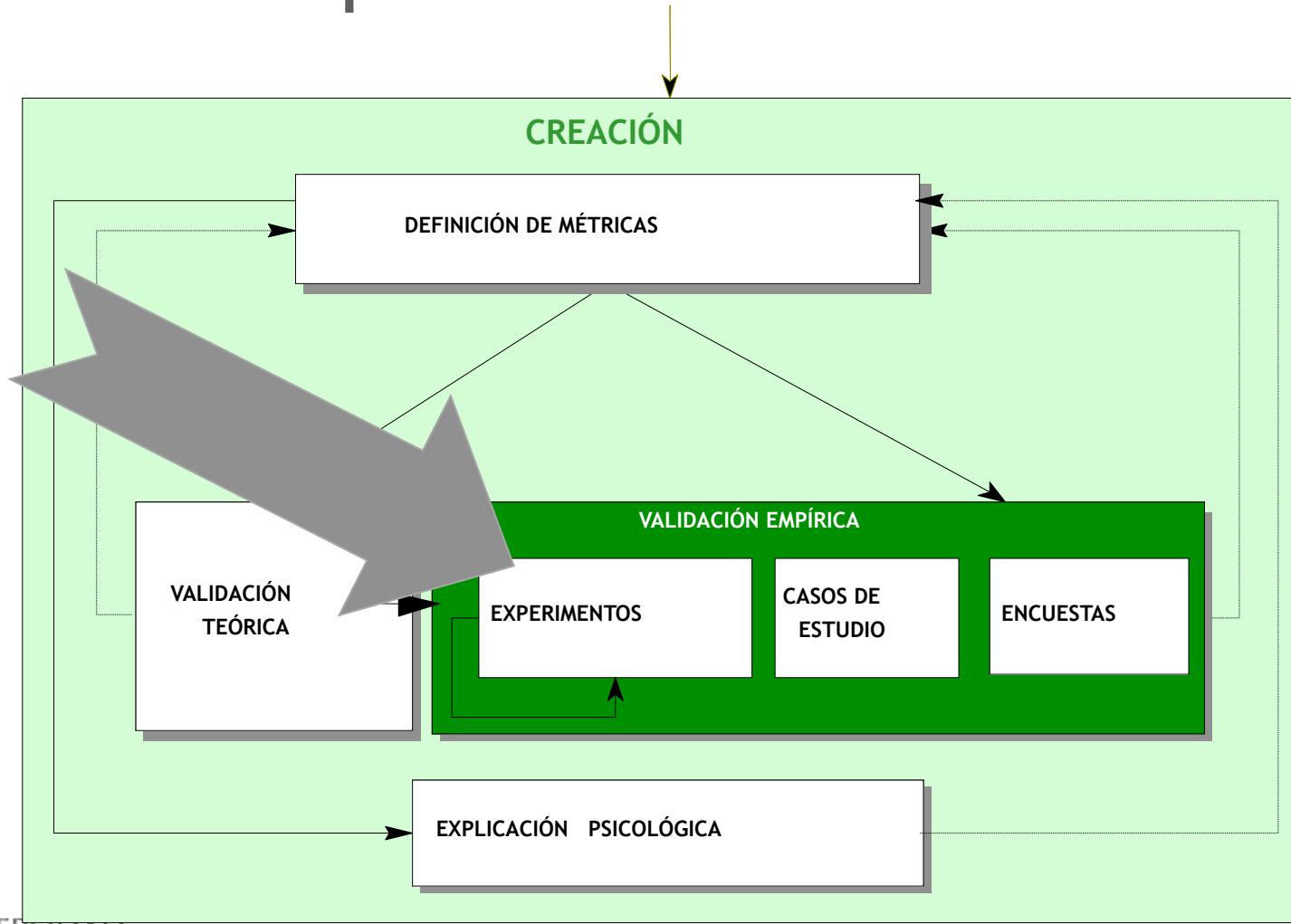
# Método de Definición Validación Empírica

## Experimentos vs. Casos de Estudio

Factor	Experimentos	Casos de estudio
Nivel de control	Alto	Bajo
Dificultad de controlar	Baja	Alta
Nivel de réplica	Alto	Bajo
Coste de replicar	Bajo	Alto



# Método de Definición Validación Empírica



UNIVERSIDAD  
POLITECNICA  
DE VALENCIA

# Método de Definición Validación Empírica

## ■ Fases fundamentales:

- Determinación del problema
- Creación de la hipótesis
- Comprobación de la hipótesis
- Análisis de resultados



UNIVERSIDAD  
POLITECNICA  
DE VALENCIA

# Método de Definición Validación Empírica

## Determinación del problema

El método experimental sólo puede ser utilizado en aquellos casos que pueden ser resolubles en términos de causalidad.

Además, la exigencia de que la situación está totalmente controlada, suele forzar al investigador a tratar con aspectos muy reducidos.

Otra característica que hay que tener en cuenta es que, tal y como indican Basili et al. (1999), las réplicas de los experimentos son necesarias.



# Método de Definición Validación Empírica

## Creación de la hipótesis

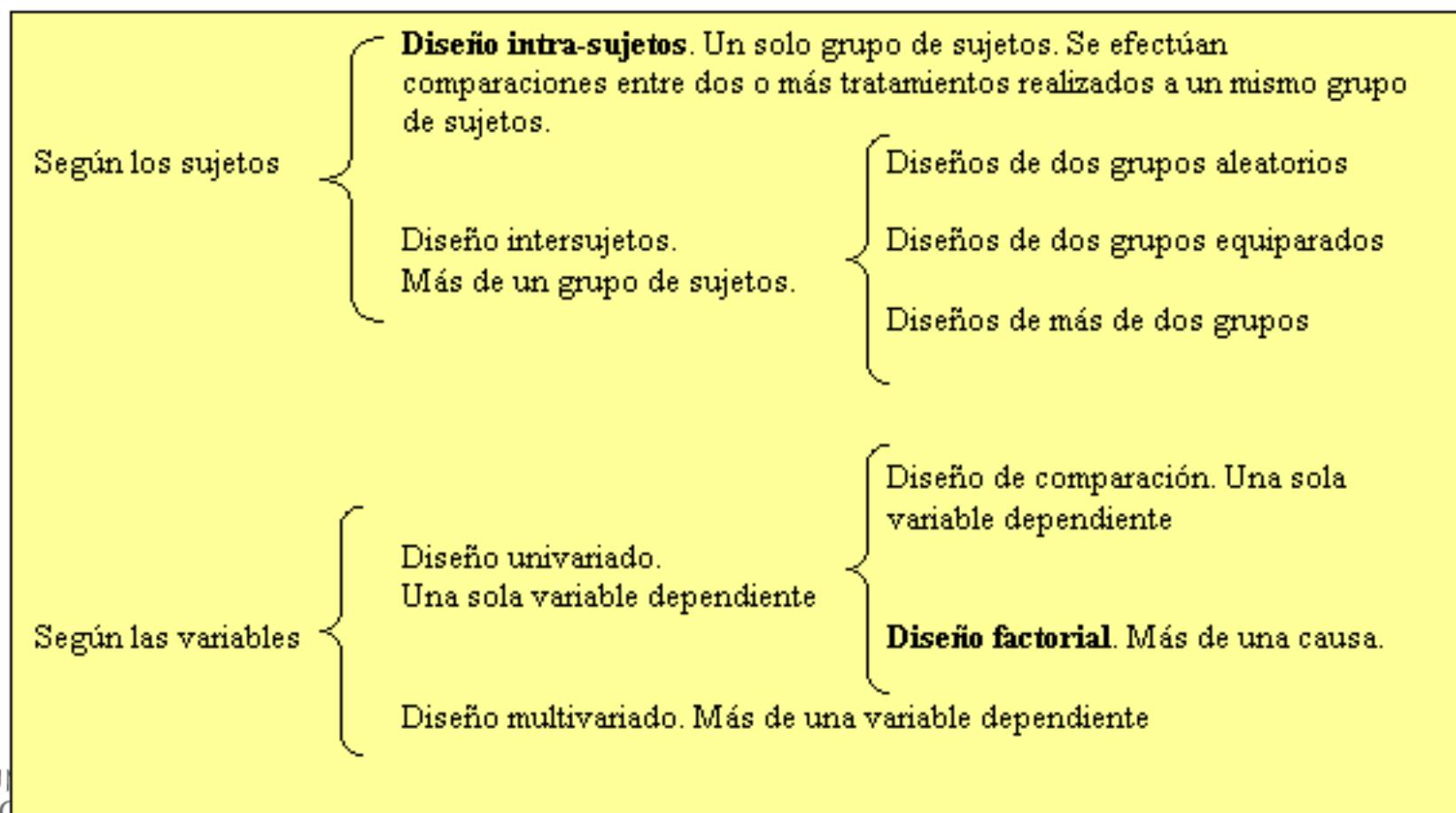
Con la definición del problema en “lenguaje natural” no se puede pasar directamente a su comprobación empírica por lo que se definen las hipótesis de trabajo.

A partir de la hipótesis de trabajo, hay que montar el experimento, especificando las condiciones concretas y controladas en las que se va a poner a prueba la hipótesis (diseño).

# Método de Definición

## Validación Empírica

### Diseño



# Método de Definición Validación Empírica

## Comprobación de la hipótesis

Ponemos en práctica la situación, es decir, hay que realizar el experimento.

Esta realización ha de ajustarse a lo previsto en el diseño.

Por eso, es conveniente planificar el experimento, siendo conveniente llevar a cabo un experimento piloto.



# Método de Definición Validación Empírica

## Análisis de resultados

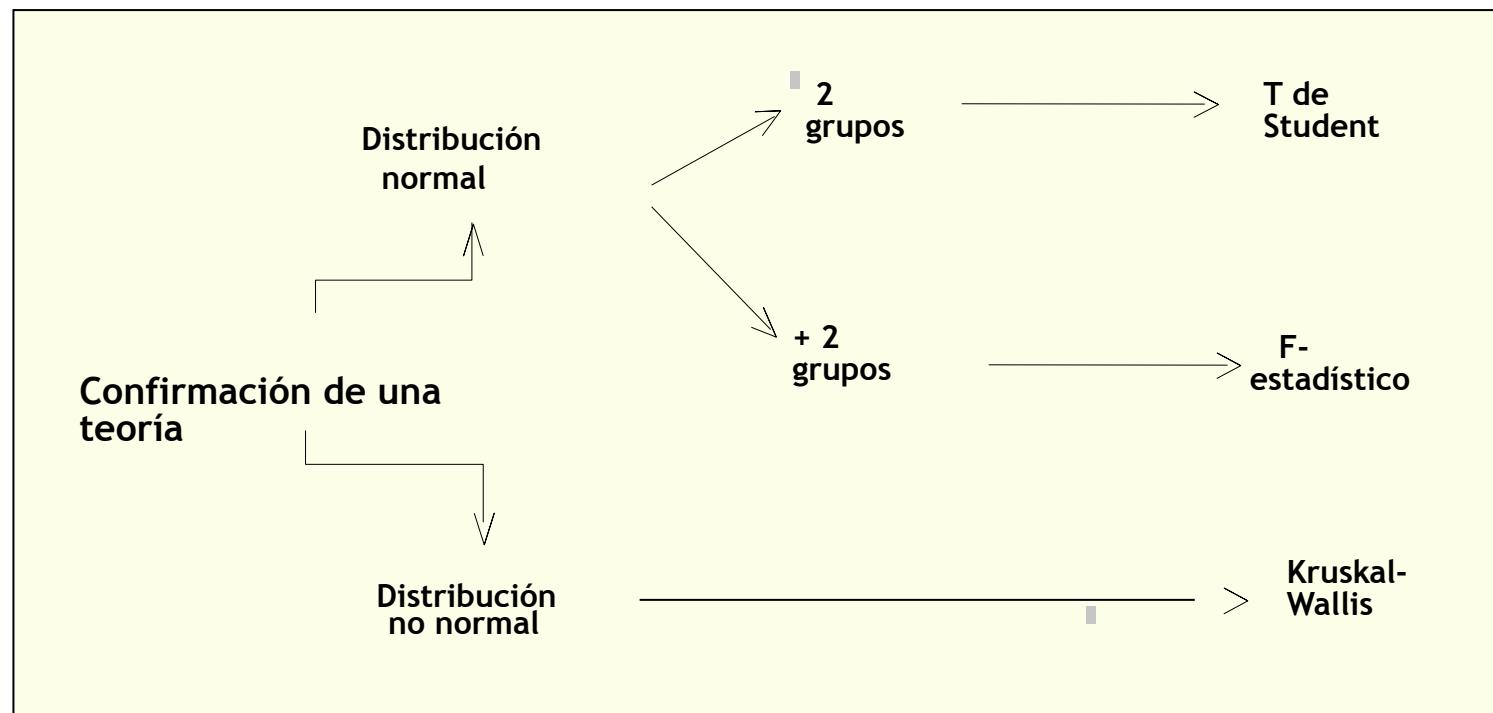
Tras realizar el experimento, obtenemos una serie de datos.

Estos datos no son directamente interpretables y deben ser sometidos a ciertas operaciones estadísticas a partir de las cuales obtenemos los resultados del experimento.

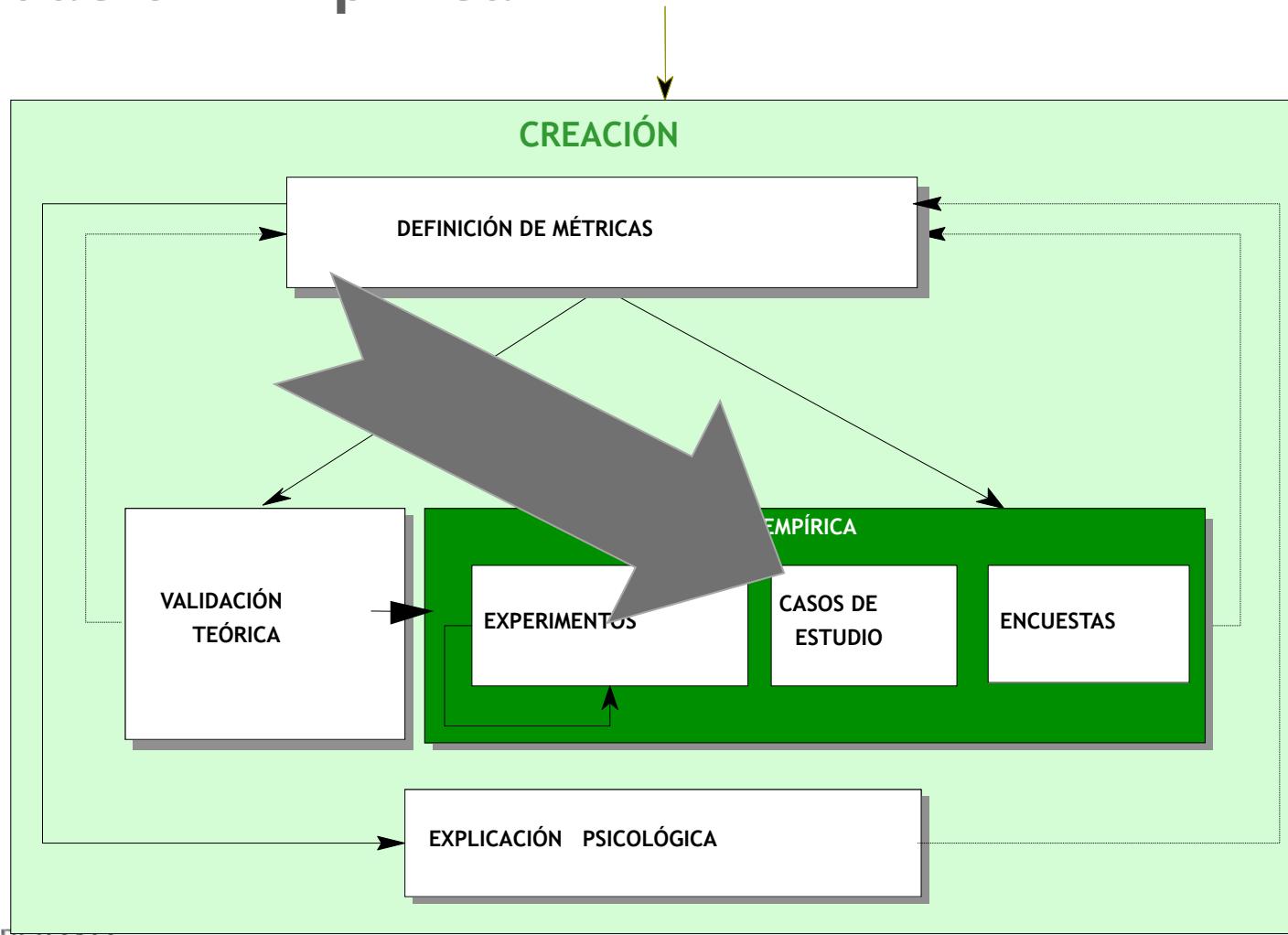
A esta labor estadística se le denomina análisis de los datos.

# Método de Definición Validación Empírica

Pfleeger (1995)



# Método de Definición Validación Empírica

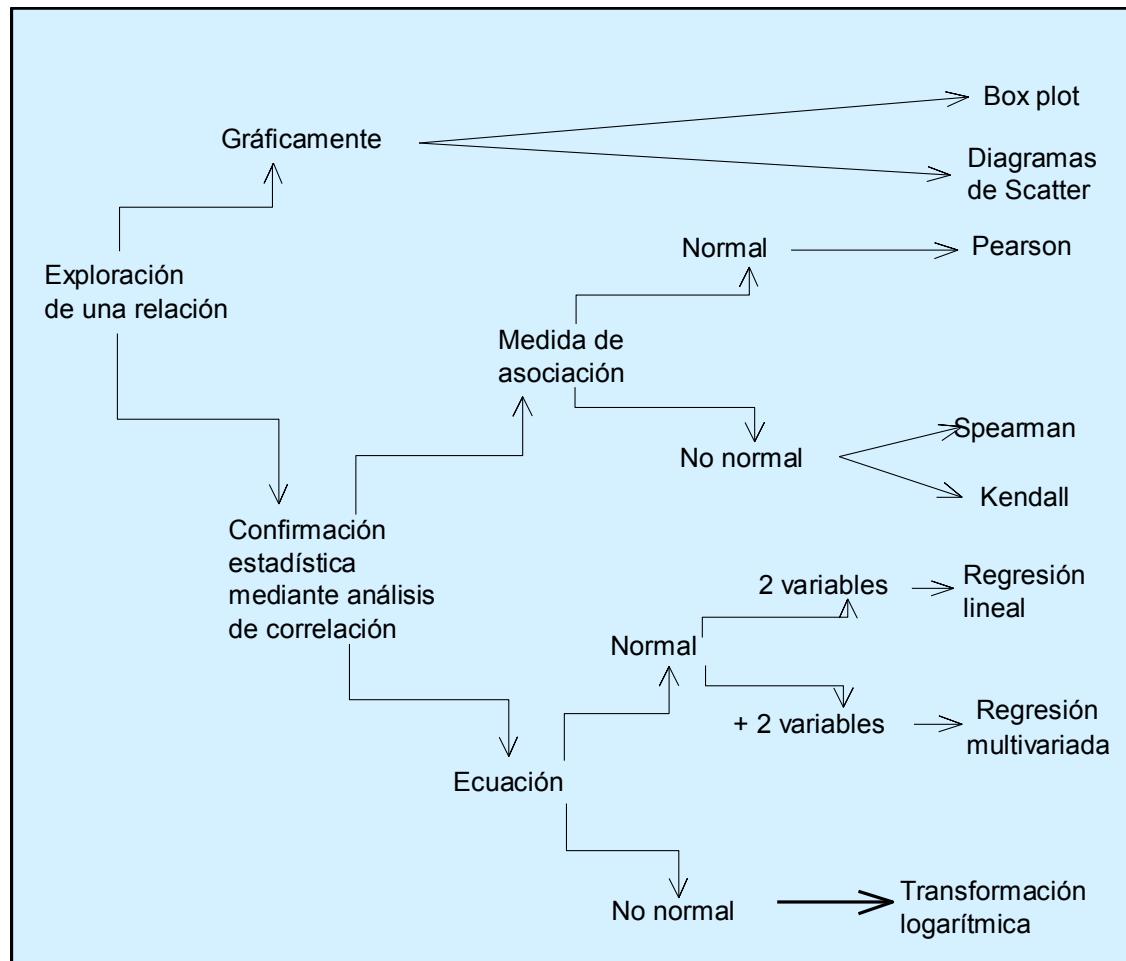


UNIVERSIDAD  
POLITECNICA  
DE VALENCIA

# Método de Definición Validación Empírica

- Hay ocasiones en las que el investigador se dedica a observar o registrar lo que sucede en una situación natural.
- No introduce ninguna variable para comprobar si ejerce efectos sobre la conducta de los sujetos ni asigna aleatoriamente los sujetos del estudio a grupos distintos. Simplemente observa.
- El caso de estudio complementa al experimento.

# Método de Definición Validación Empírica



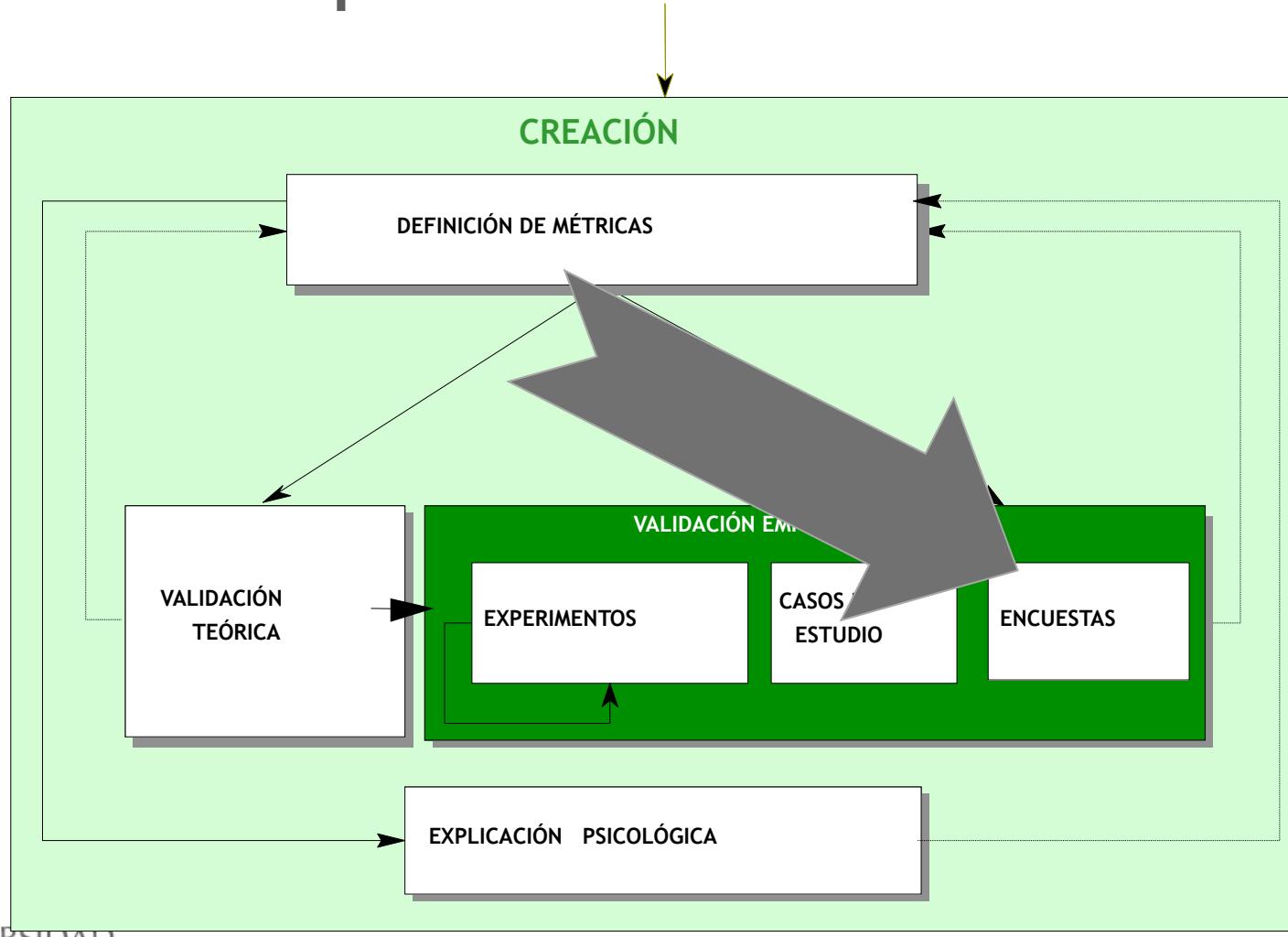
# Método de Definición Validación Empírica

- Técnicas estadísticas tradicionales
- Técnicas avanzadas de análisis de datos:
  - Aprendizaje automático
    - RoC
    - C4.5



UNIVERSIDAD  
POLITECNICA  
DE VALENCIA

# Método de Definición Validación Empírica



UNIVERSIDAD  
POLITECNICA  
DE VALENCIA

# Método de Definición Validación Empírica

- Una encuesta es un sistema completo para:
  - recoger información,
  - describir,
  - comparar
  - o explicar
- conocimiento, actitudes y comportamiento
- El primer paso antes de comenzar cualquier encuesta es fijar los objetivos con los resultados que se esperan de la encuesta.
- Una vez hecho esto, el siguiente paso a realizar consiste en diseñar el cuestionario.



# Método de Definición Validación Empírica

- El diseño ha de ser adecuado para conseguir nuestros objetivos:
  - Debe evitar parcialidad: Los resultados de la encuesta reflejen la realidad de la situación.
  - Debe ser apropiada: debe tener sentido dentro del contexto de la población.
  - Debe tener un coste efectivo: se quiere un diseño cuya distribución y análisis estén dentro de los recursos destinados a la encuesta.



# Método de Definición Validación Empírica

- Para construir un instrumento de encuesta hay dos opciones:
  - Buscar en la literatura
  - Construir un instrumento
- Cuando los investigadores utilizan instrumentos existentes tienen las siguientes ventajas:
  - Se ha evaluado la validez y la fiabilidad
  - Se pueden comparar los nuevos resultados con los resultados de otros estudios.
- Pero, en ocasiones ocurre que no es posible utilizar de forma directa un instrumento existente, por lo que se ha de modificar



# Método de Definición Validación Empírica

- La siguiente tarea a realizar consiste en la evaluación del cuestionario. Para ello, se ha de tener en cuenta:
  - Cómo motivar a las personas que responden
  - Cómo evitar o disminuir la parcialidad de la persona que responde.
  - Cómo evaluar los cuestionarios y los instrumentos de encuesta.



# Método de Definición

## Validación Empírica

- También es necesario determinar la muestra.
- Para obtener una muestra, se debe comenzar por definir una población destinataria.
- Los métodos de muestreo son: probabilísticos ( Muestra aleatoria simple, Muestra aleatoria estratificada, Muestreo sistemático, Muestreo basado en grupos) o no probabilísticos (Muestreo por conveniencia, Muestreo que aumenta progresivamente, Muestreo de cuota)

# Método de Definición Validación Empírica

- Por regla general, se realiza la encuesta a la muestra anteriormente obtenida, ya que las respuestas de un grupo pequeño deben representar lo que hubiese ocurrido si se hubiese entrevistado a todo el grupo entero.



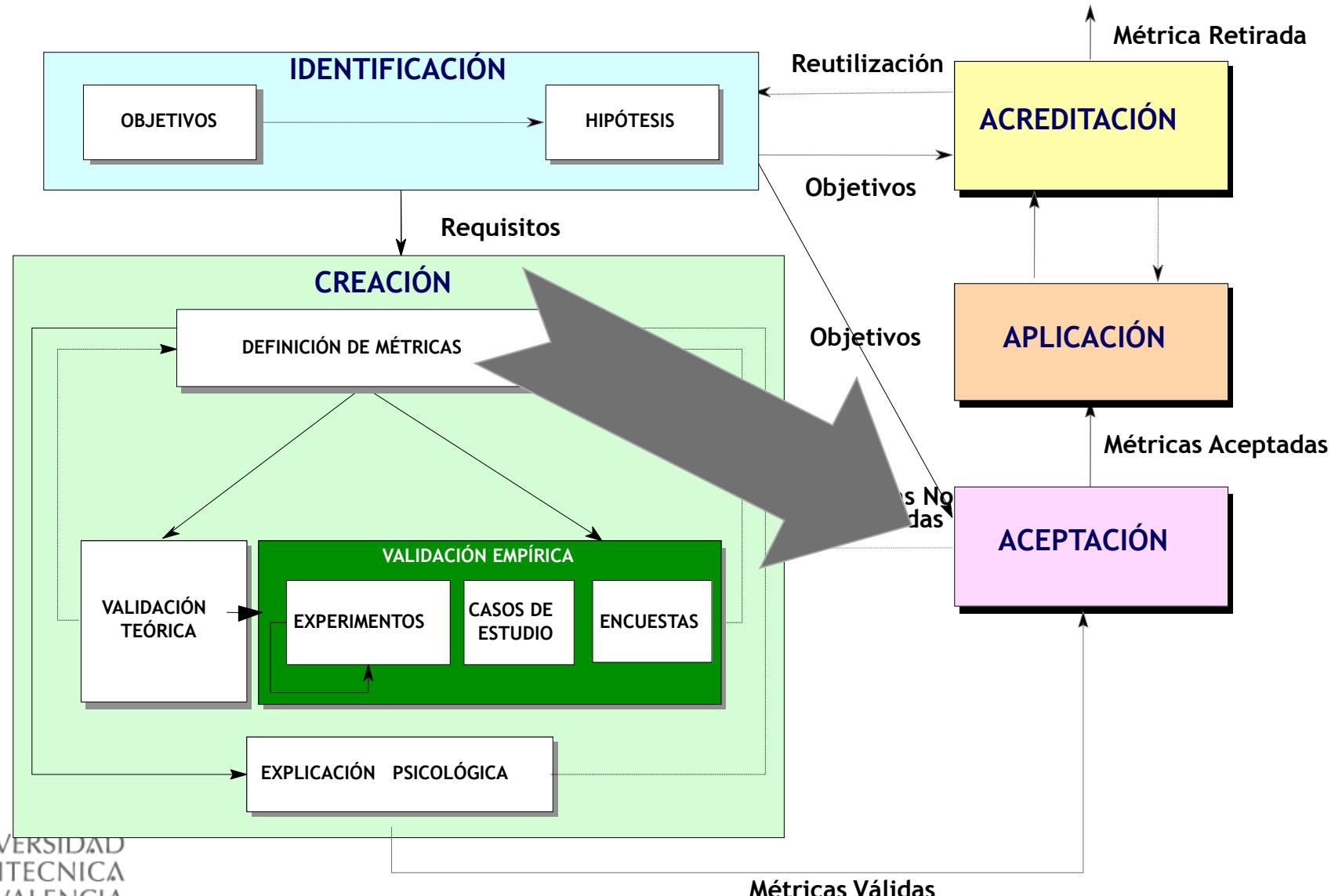
# Método de Definición Validación Empírica

- A partir de ahí se aplica el cuestionario y
- Se analizan los resultados para ver si se cumplen las hipótesis de partida



UNIVERSIDAD  
POLITECNICA  
DE VALENCIA

# Método de Definición de Métricas



# Método de Definición Aceptación

- Una vez obtenida una métrica válida, suele ser necesario pasar por una etapa de aceptación de la métrica en la que se harán pruebas en entornos reales, de manera que podamos comprobar si la métrica cumple los objetivos deseados dentro del campo de aplicación real.



# Método de Definición Aplicación y Acreditación

- **Aplicación.** Una vez que tengamos una métrica aceptada, la utilizaremos dentro del campo de aplicación para la que fue definida.
- **Acreditación.** Es la última etapa del método, que discurre en paralelo con la fase de aplicación y tiene como objetivo el **mantenimiento de la métrica**, de manera que se pueda adaptar al entorno cambiante. Como consecuencia de esta etapa, puede que una métrica sea retirada, porque ya no sea útil en el entorno en el que se aplica, o que sea reutilizada para iniciar el proceso de nuevo.



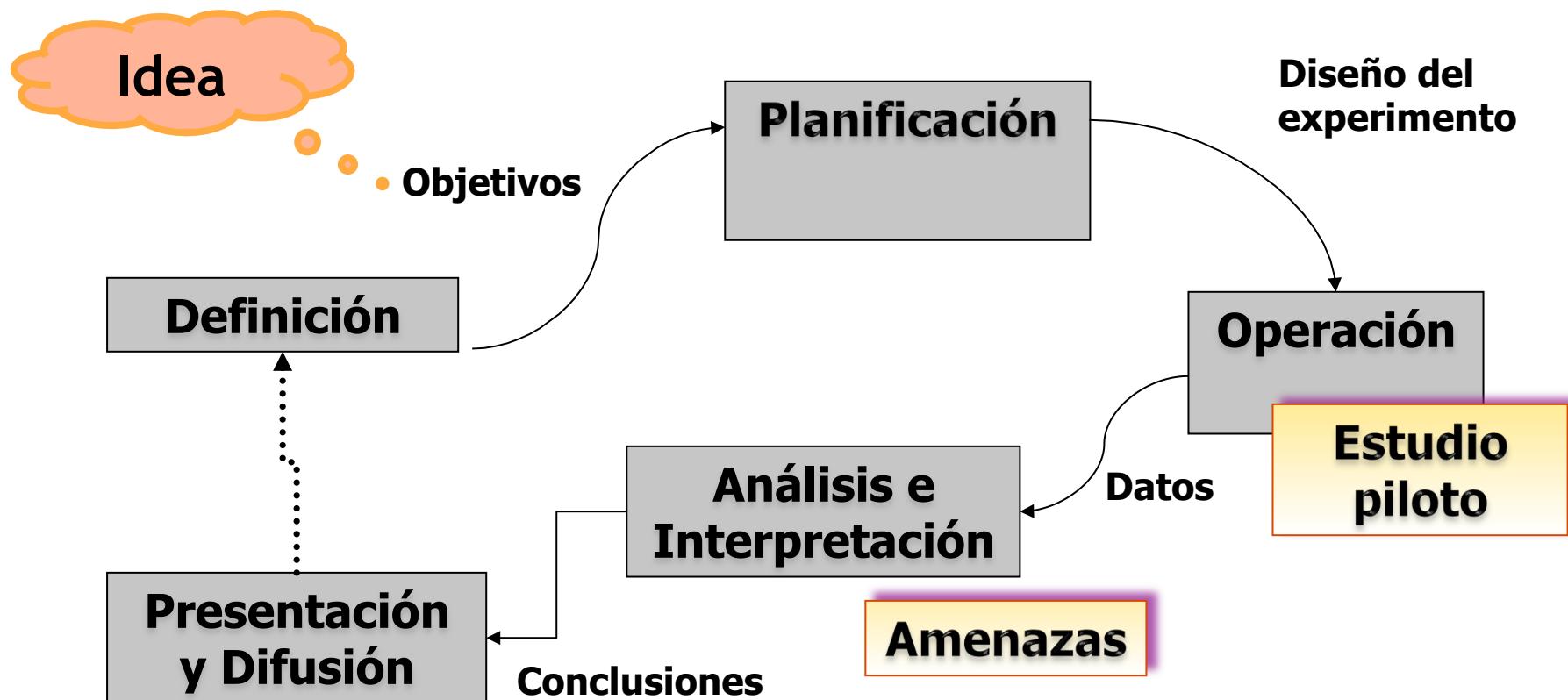
# Ejemplo Validación Empírica Métricas

- El **objetivo** del estudio empírico es:

Determinar si dos métricas, ***NFK (Number of Foreign Keys)*** y ***DRT (Depth of Referential Tree)*** pueden ser utilizadas como mecanismos para controlar la mantenibilidad de las BD relacionales.



# Proceso Experimental



UNIVERSIDAD  
POLITECNICA  
DE VALENCIA

Wohlin C., Runeson P., Höst M., Ohlson M., Regnell B. y Wesslén A.  
(2000) Experimentation in Software Engineering: An Introduction,  
Kluwer Academic Publishers.

# Ejemplo

## Definición

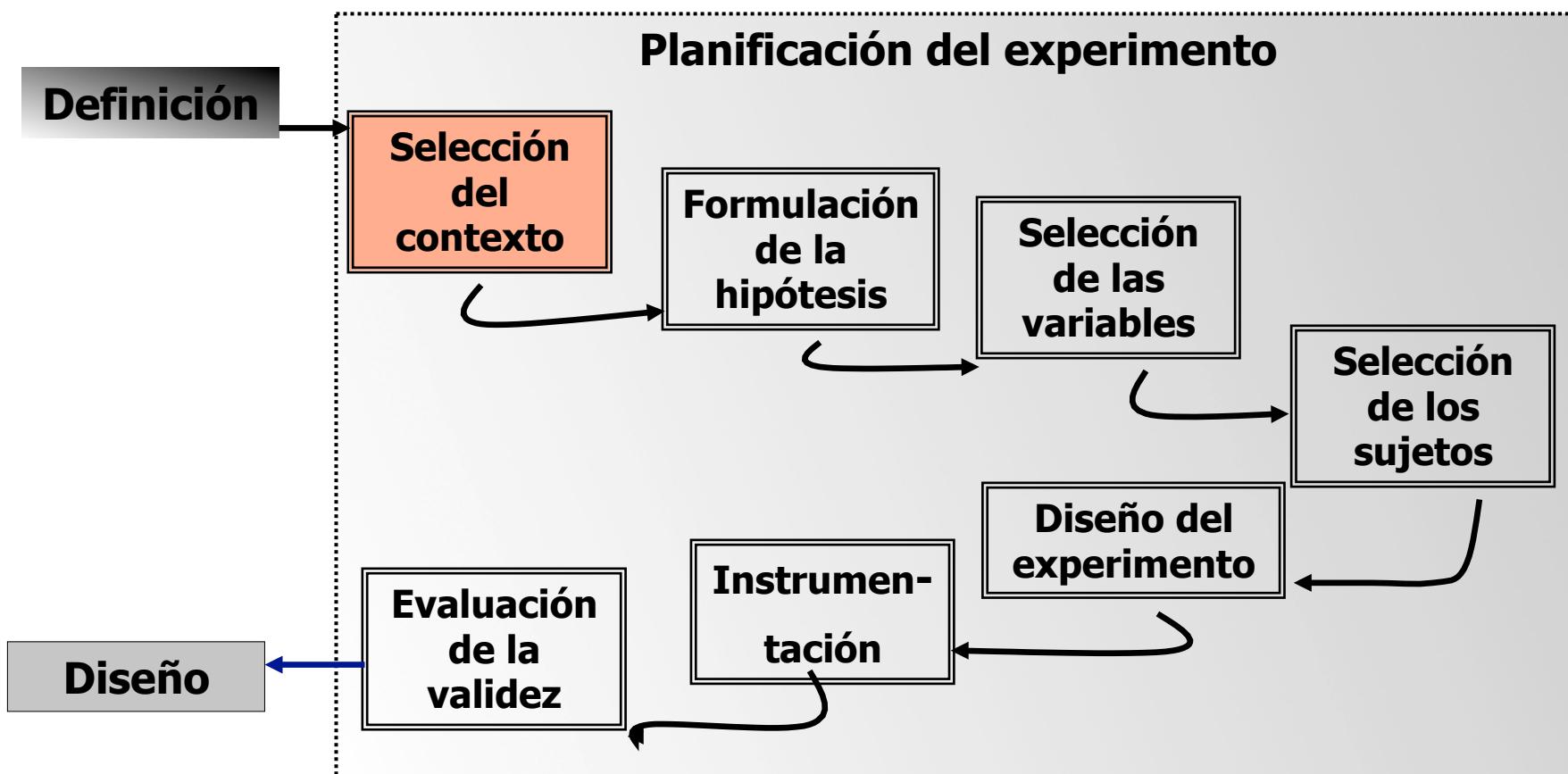
- La definición del **objetivo del experimento** puede ser definido como:

**Analizar** las métricas para bases de datos relacionales con el **propósito** de evaluar si pueden ser utilizados como mecanismos útiles  
**respecto a** la analizabilidad de las BD relacionales desde el **punto de vista** de los investigadores  
**en el contexto de** profesionales con experiencia en bases de datos



# Ejemplo

## Planificación del Experimento



# Ejemplo

## Planificación del Experimento

### Selección del contexto

- Caracterización: **más de un objeto** y **más de un sujeto**
- El contexto del experimento es un grupo de profesionales y se lleva cabo **“off-line”**
- Los sujetos son **profesionales**
- El experimento aborda un **problema real**
- El experimento es **específico**



# Ejemplo

## Planificación del Experimento

### Formulación de la hipótesis

- Las hipótesis del experimento son:
  - **Hipótesis Nula:** Valores diferentes de DRT y NFK no afectan a la analizabilidad del esquema de la base de datos.
  - **Hipótesis alternativa H1:** El valor de DRT afecta a la analizabilidad del esquema de la base de datos.
  - **Hipótesis alternativa H2:** El valor de NFK afecta a la analizabilidad del esquema de la base de datos.
  - **Hipótesis alternativa H3:** La interacción de DRT y NFK afecta a la analizabilidad del esquema de la base de datos.



# Ejemplo

## Planificación del Experimento

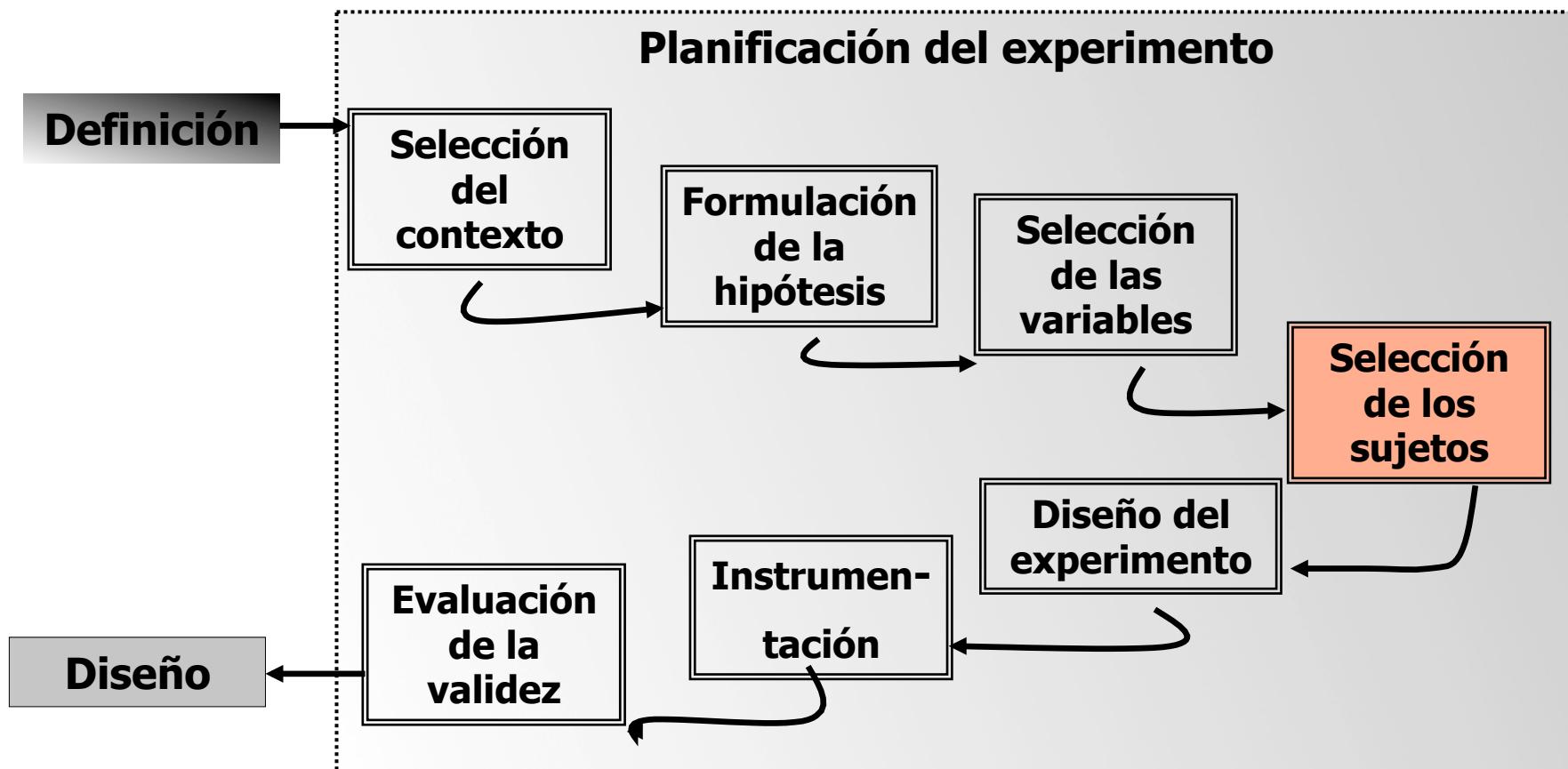
### Selección de las variables

- **Variables Independientes.** Se corresponden con las dos métricas bajo estudio: NFK y DRT. Cada una de estas métricas (factores) pueden tomar dos valores diferentes (niveles): 5 y 8 para la métrica NFK y 2 y 5 para la DRT.
- **Variables Dependientes.** La analizabilidad se midió como el tiempo que cada sujeto utilizó para realizar todas las tareas de cada test experimental y la corrección con la que completaban las tareas incluidas en cada test (*Delete*, *Update* y *Insert*).
  - Es importante puntualizar que este tiempo incluía el tiempo utilizado en el análisis del esquema y el requerido para completar las tres preguntas sobre el mismo.



# Ejemplo

## Planificación del Experimento



# Ejemplo

## Planificación del Experimento

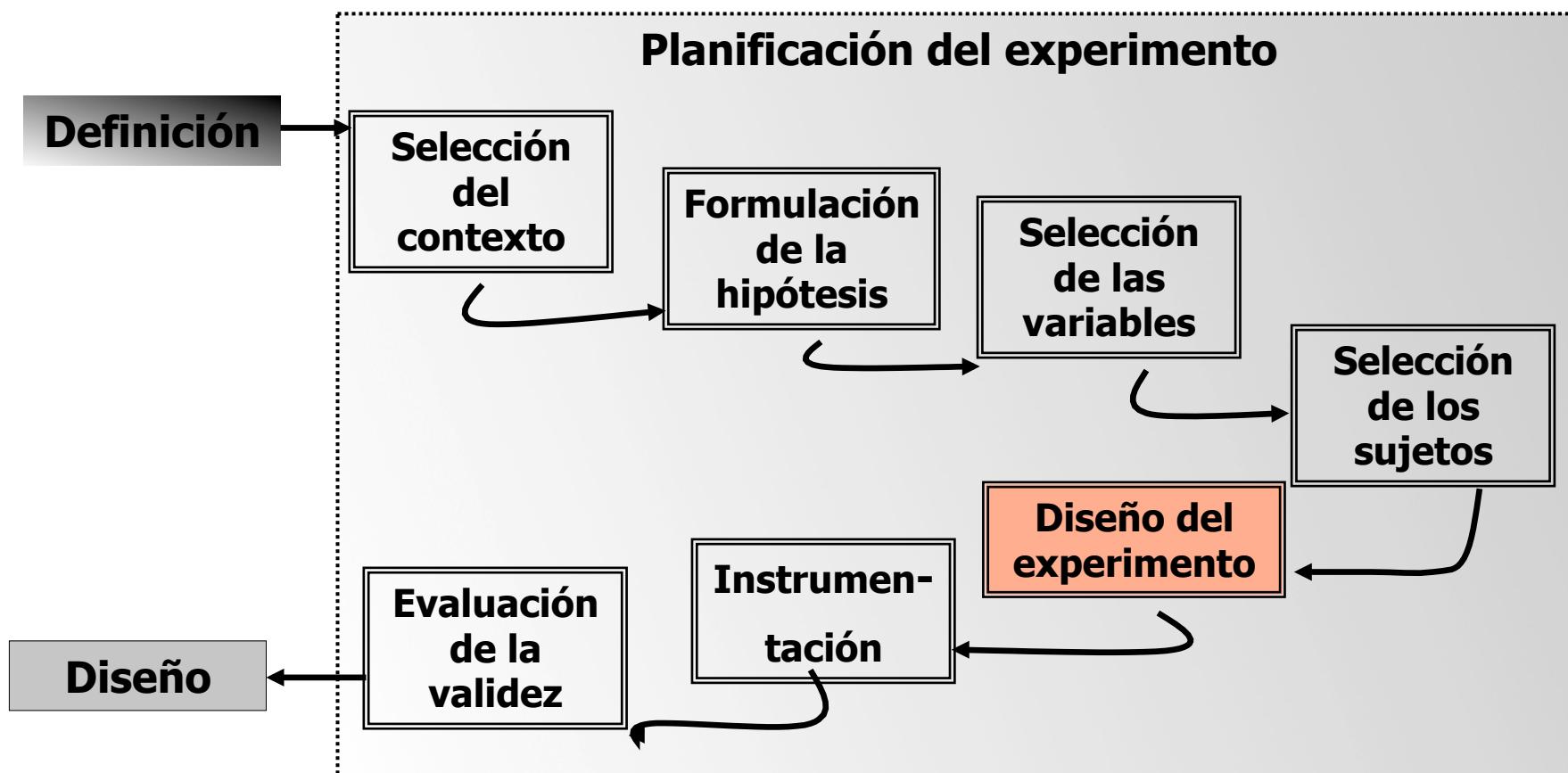
### Selección de los sujetos

- 11 **profesionales** de una empresa realizaron el experimento
- Todos ellos tenían una **experiencia** media de 3 años trabajando con BD relacionales.



# Ejemplo

## Planificación del Experimento



# Ejemplo

## Planificación del Experimento

### Diseño del experimento

- El experimento trabaja con **dos factores**: DRT y NFK, por lo que tenemos un modelo factorial.
- Ambas métricas pueden tomar **dos valores (tratamientos)**, 2 y 5 para la métrica DRT y 5 y 8 para la métrica NFK.

Diseño factorial

$2 \times 2$

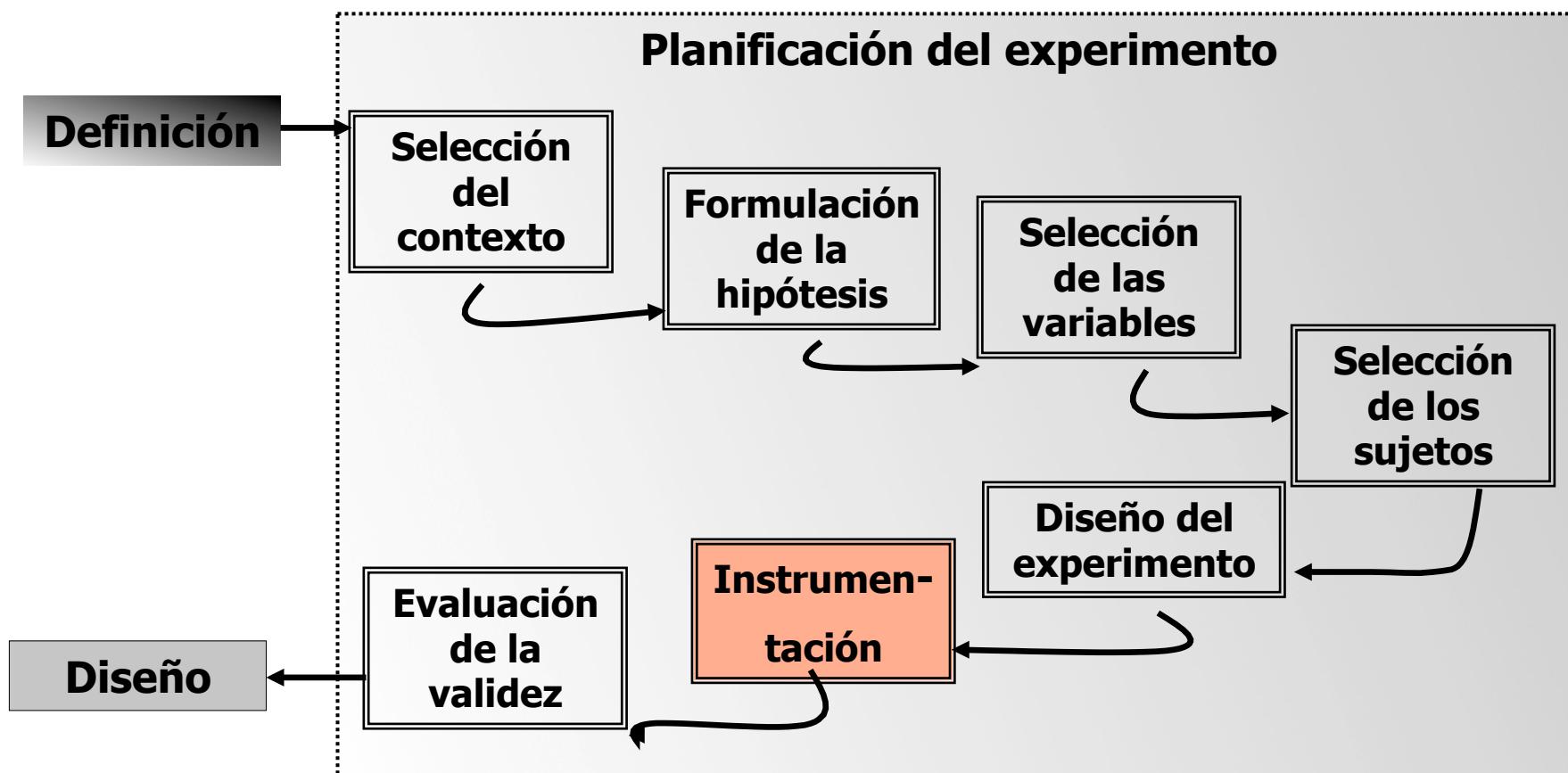
Intra-sujetos



UNIVERSIDAD  
POLITECNICA  
DE VALENCIA

# Ejemplo

## Planificación del Experimento



# Ejemplo

## Planificación del Experimento

### Instrumentación

- Para el experimento se utilizaron **cuatro** esquemas de bases de datos relacional.
- Para cada diseño la documentación constaba de unas **7 páginas** que incluían además de los **esquemas** de las bases de datos, una **descripción general** y un **documento de requisitos**.
- Para conseguir que los **diseños** fueran comparables, tratamos de que fueran lo más **similares** posible.
- Para cada diseño había que **realizar 3 operaciones** (*Insert*, *Delete* y *Update*) y los sujetos debían anotar cuantas columnas de cada tabla iban a resultar afectadas como resultado de cada una de estas operaciones.



# Ejemplo

## Planificación del Experimento

### Instrumentación

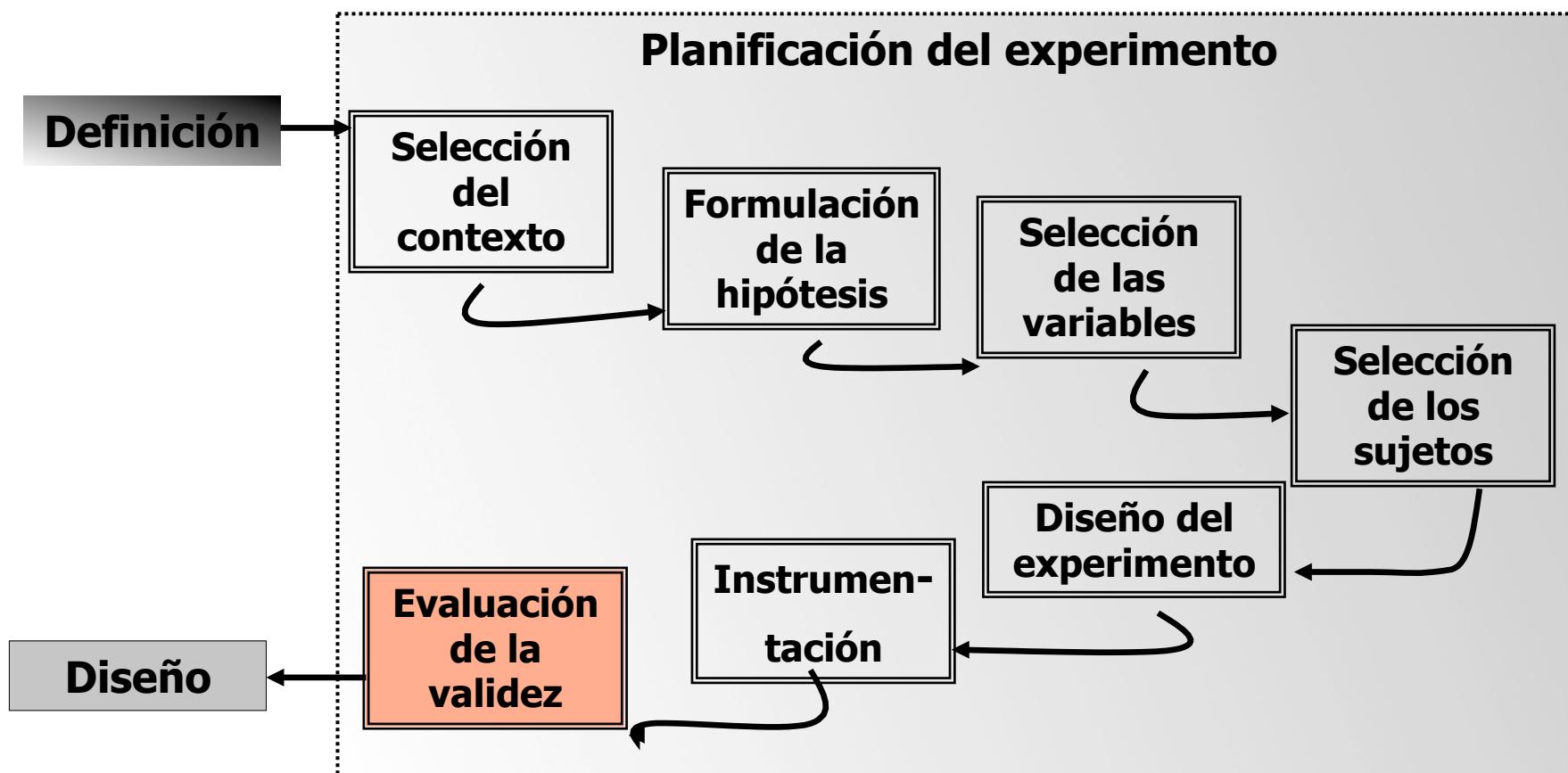
- Con los resultados obtenidos debían llenar un cuestionario

START TIME:					
1. What tables and how many rows in each table are affected if we delete in the Table 5 the row with code1=210?					
Table 1	Table 2	Table 3	Table 4	Table 5	Table 6
2. What tables and how many rows in each table are affected if we update the column X of the row with code2=11 in the table 3?					
Table 1	Table 2	Table 3	Table 4	Table 5	Table 6
3. What tables and how many rows and columns are necessary to add if we want add a new row in the table 4? (Suppose that all the necessary data are news in the database)					
Table 1	Table 2	Table 3	Table 4	Table 5	Table 6
END TIME:					



# Ejemplo

## Planificación del Experimento



# Ejemplo

## Planificación del Experimento

### Evaluación de la validez

#### Validez de constructo

- Proponemos el **tiempo** para determinar el estado final de la base de datos (tras realizar las tareas) como medida razonable de la analizabilidad.
- Para asegurar la validez de constructo sería necesario realizar **más experimentos**, variando las operaciones a desarrollar.



# Ejemplo

## Planificación del Experimento

### Evaluación de la validez

#### Validez Interna

- **Diferencias entre sujetos.** Los experimentos intra-sujetos reducen la variabilidad entre sujetos. En los experimentos todos los sujetos tenían, aproximadamente, la misma experiencia trabajando con bases de datos relacionales.
- **Diferencias entre esquemas.** Todos los esquemas fueron diseñados con seis tablas relacionadas con más o menos claves ajena dependiendo de los valores de las métricas. Sin embargo, el dominio de los esquemas eran diferentes lo que pudo influenciar, de alguna forma, en los resultados obtenidos.



# Ejemplo

## Planificación del Experimento

### Evaluación de la validez

- **Precisión de los valores del tiempo.** Los sujetos fueron los encargados de apuntar los tiempos de comienzo y fin de cada test. Pensamos que este método es más efectivo que tener un supervisor encargado de anotar los tiempos de cada sujeto. Sin embargo, somos conscientes de que el sujeto puede introducir alguna imprecisión.
- **Efectos de aprendizaje.** Los tests fueron colocados en distinto orden para los diferentes sujetos. De esta forma, cada sujeto debía contestar los tests en el orden que se le entregaban.
- **Efectos de fatiga.** El tiempo medio para completar el experimento fue una hora por lo que los efectos de fatiga son prácticamente inexistentes. Además, el orden diferente de los tests ayudaron a evitar este tipo de efectos.



# Ejemplo

## Planificación del Experimento

### Evaluación de la validez

- **Efectos de persistencia.** Este tipo de efectos no se dieron ya que era la primera vez que los sujetos realizaban un experimento de estas características.
- **Motivación de los sujetos.** Se les indicó a los profesionales que los tests eran anónimos y que en ningún caso la información obtenida sería comentada entre los trabajadores ni entre los jefes.
- **Otros factores.** El plagio y la influencia entre los sujetos fueron controlados. Los sujetos fueron informados sobre la imposibilidad de hablar con otros sujetos y compartir respuestas. Sin embargo, no hubo ningún tipo de vigilancia por lo que no podemos asegurar que no aparecieran efectos del tipo comentado, aunque no parece posible.



# Ejemplo

## Planificación del Experimento

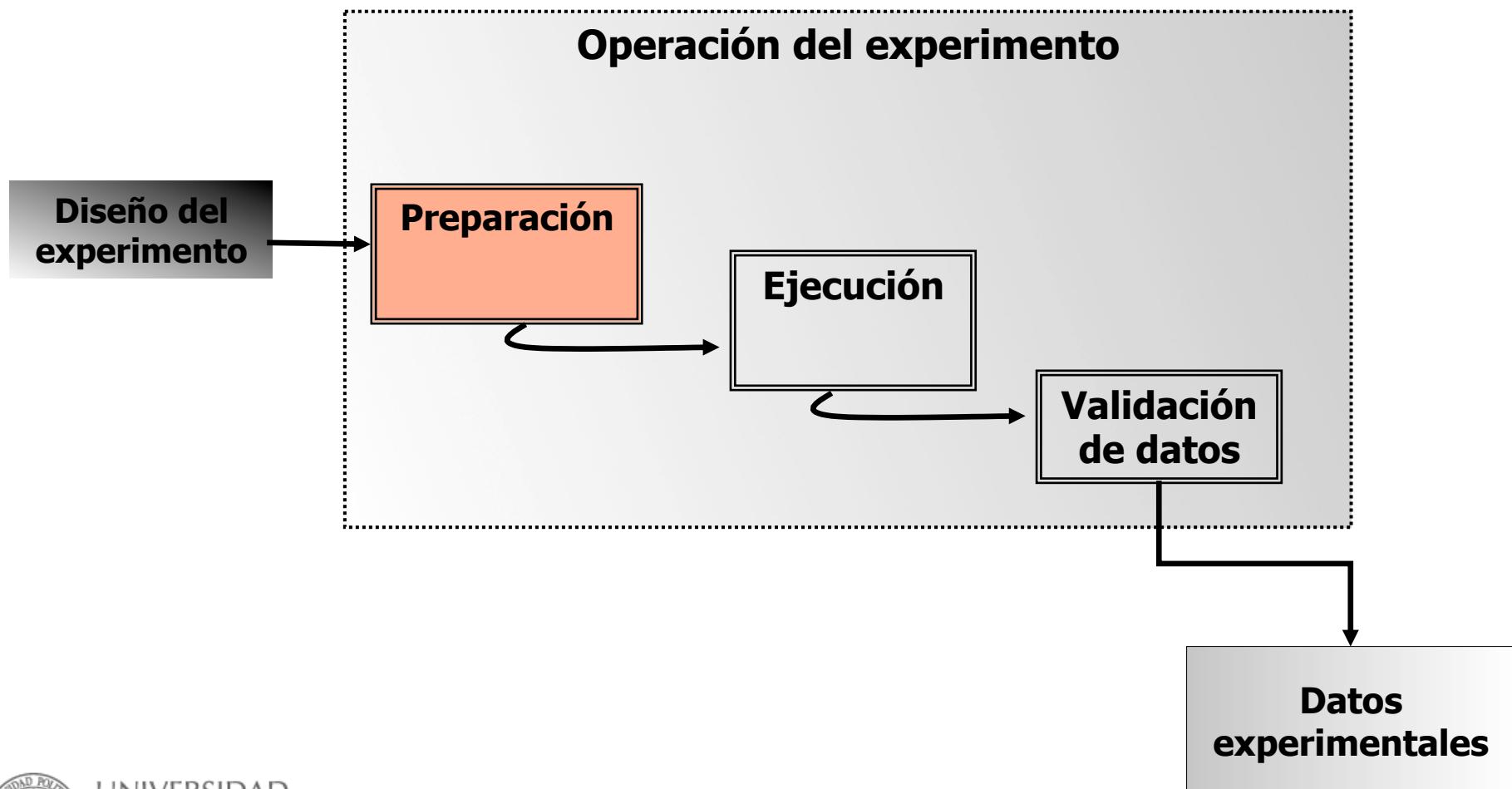
### Evaluación de la validez

#### Validez externa

- **Materiales y tareas utilizados**
  - Se ha utilizado en el experimento esquemas y operaciones que sean representativos de casos reales.
  - Son necesarios más experimentos con BD relacionales de mayor tamaño y complejidad.
- **Sujetos**
  - Son necesarios nuevos experimentos con un mayor número de sujetos (estudiantes y profesionales) y con una mayor diferencia entre los valores de las métricas.



# Ejemplo Operación



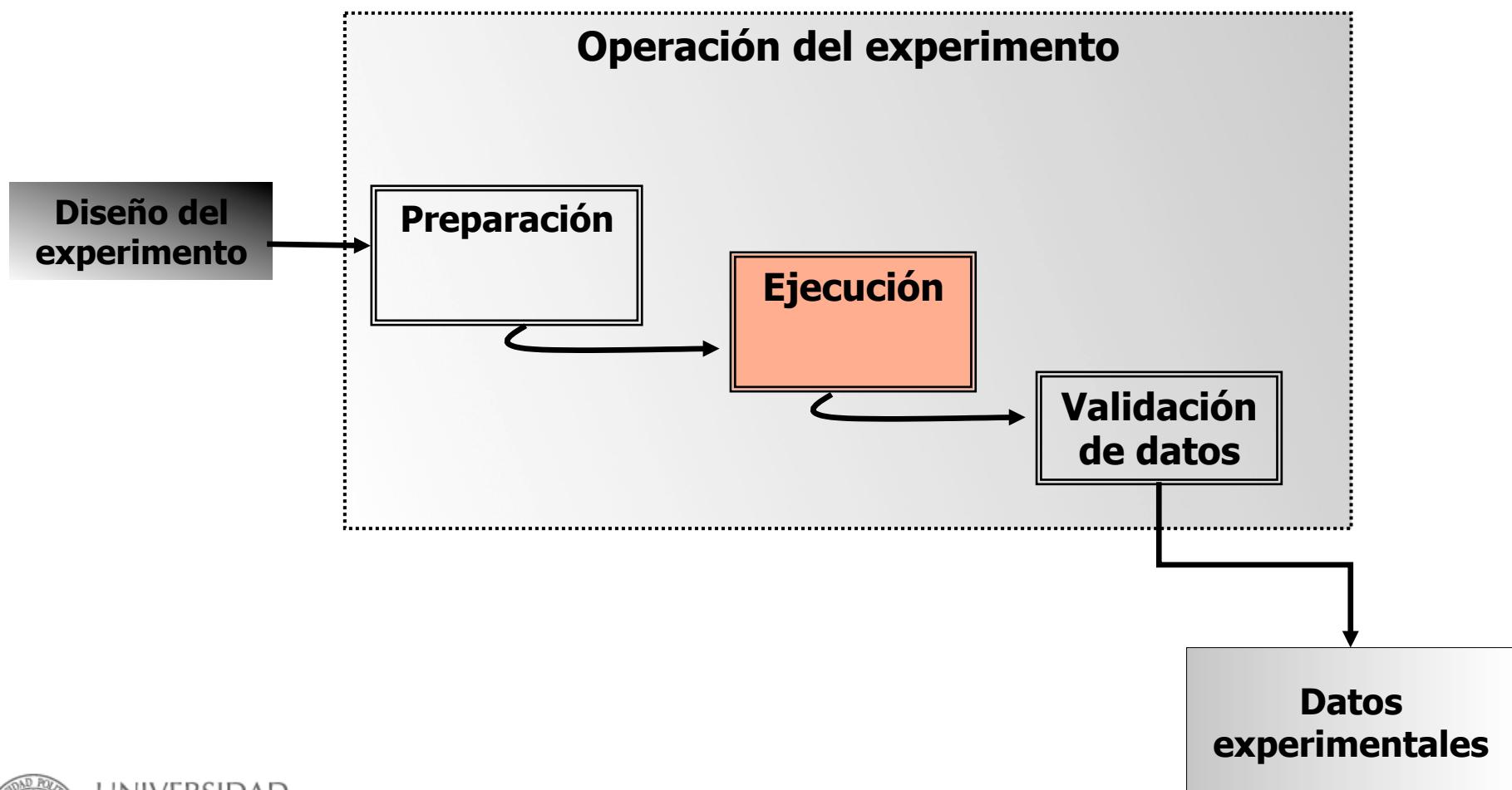
# Ejemplo Operación

## Preparación

- Antes de que los sujetos realizaran el experimento, este fue realizado por un **grupo reducido** para mejorarla y asegurar que tanto el experimento como la documentación estaban bien diseñados.
- Como **resultado** de este pre-ejecución el único cambio necesario fue en el cuestionario que fue cambiado para hacerlo más fácil de utilizar (transformándolo en tabla).



# Ejemplo Operación



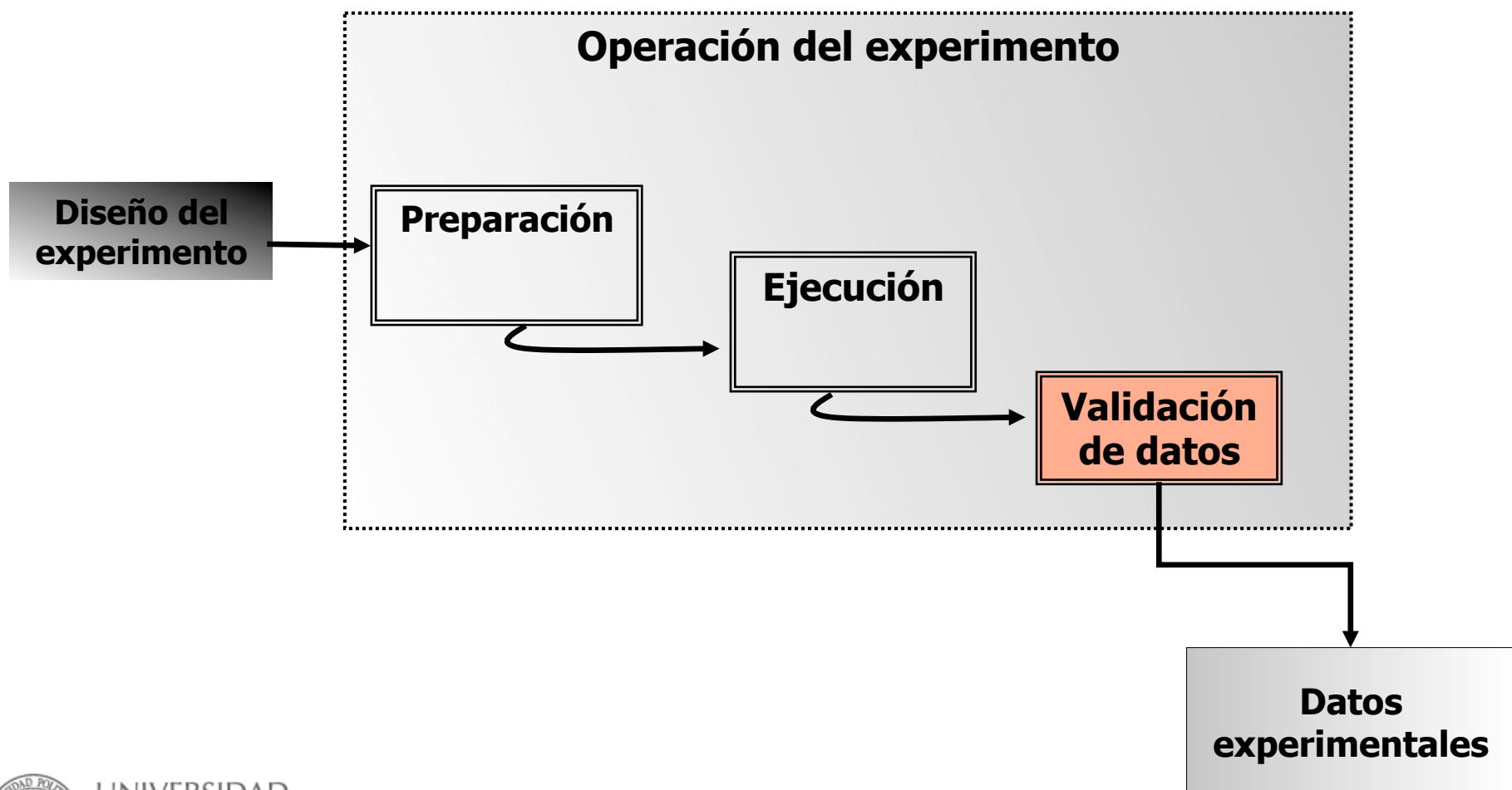
# Ejemplo Operación

## Ejecución

- Los tests se realizaron a lo largo de **una hora**. Se le explicó a los sujetos que tipo de ejercicios realizarían, el material que podrían usar, el tipo de respuestas que debían proporcionar y como debían anotar el tiempo utilizado en resolver los ejercicios.
- Antes de cada test, los sujetos debían anotar el **tiempo de comienzo** y, al terminar los ejercicios, el **tiempo de fin**.
- Los tests fueron realizados en **diferente orden** por los distintos sujetos para evitar los efectos de aprendizaje.



# Ejemplo Operación



# Ejemplo Operación

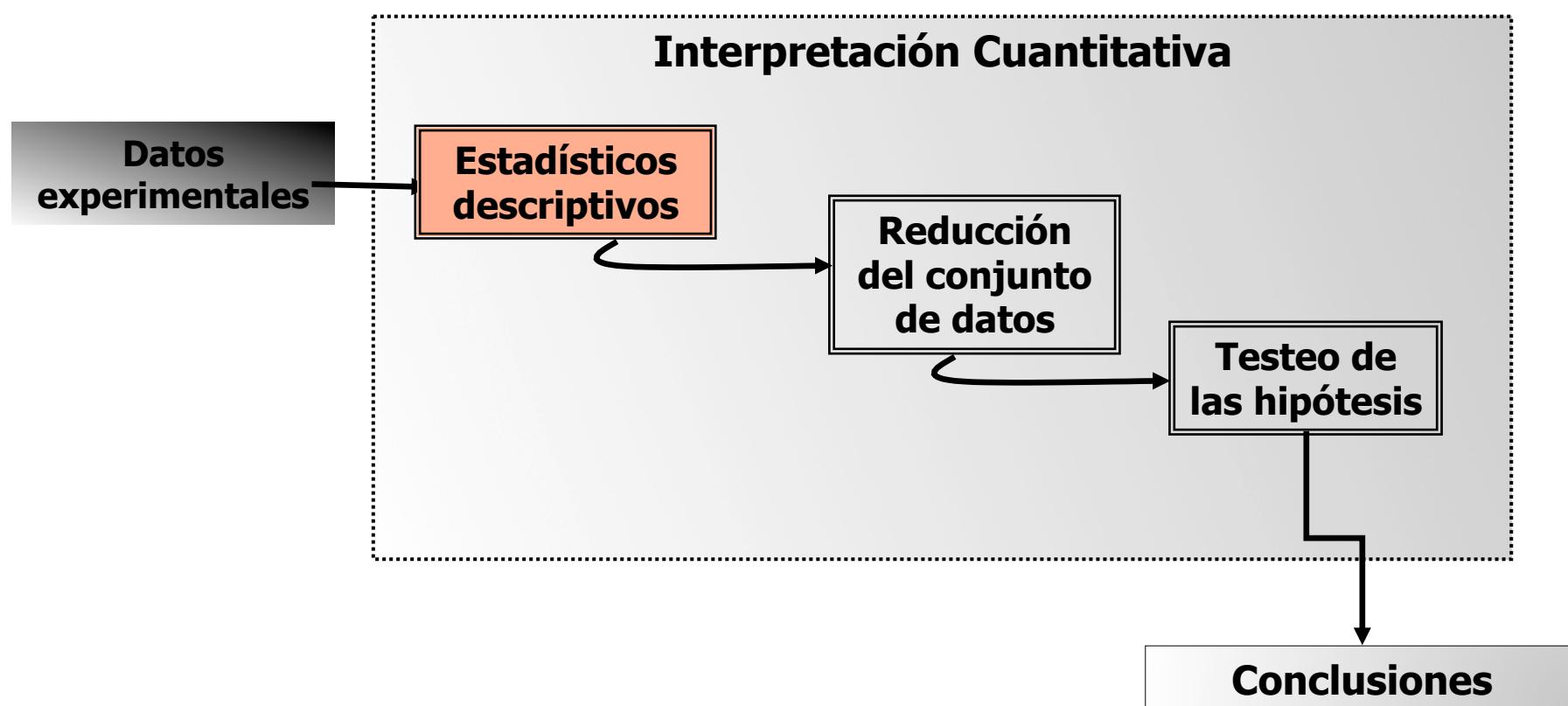
## Validación de datos

- Como todos los tests fueron **contestados correctamente** pudimos utilizar el número de minutos utilizados por cada sujeto para la obtención de los resultados del experimento.
- Asimismo, por ser correctos todos los tests, contamos con **11 conjuntos de datos** para cada test, cada uno de los cuales corresponde a la respuesta de cada sujeto.



# Ejemplo

## Interpretación Cuantitativa



# Ejemplo

## Interpretación Cuantitativa

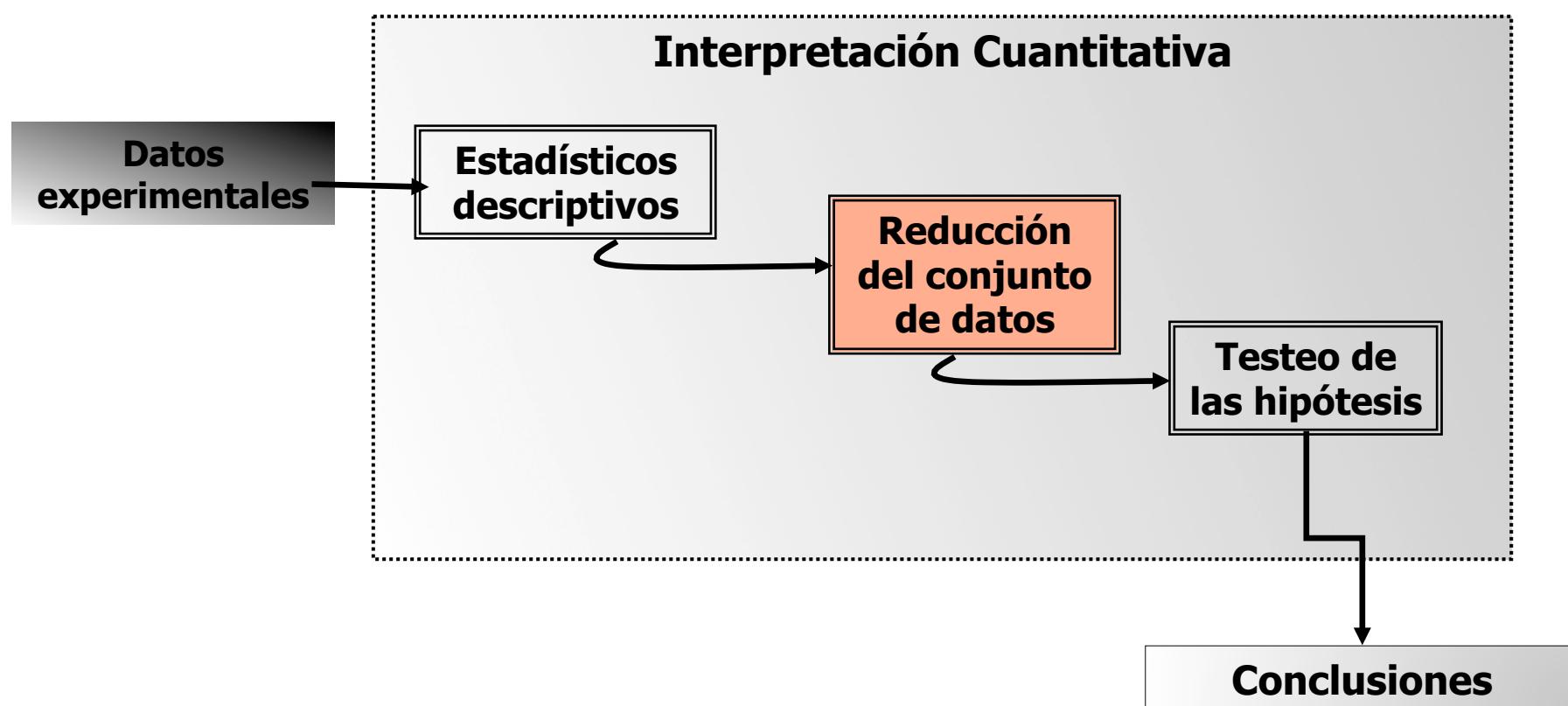
### Estadísticos Descriptivos

	Mínimo	Máximo	Media	Desviación estándar
Caso Uno	4	16	7,1818	3,6005
Caso Dos	3	8	5,4545	1,8635
Caso Tres	6	15	10,0909	2,7002
Caso Cuatro	3	15	7,0909	3,7803



# Ejemplo

## Interpretación Cuantitativa

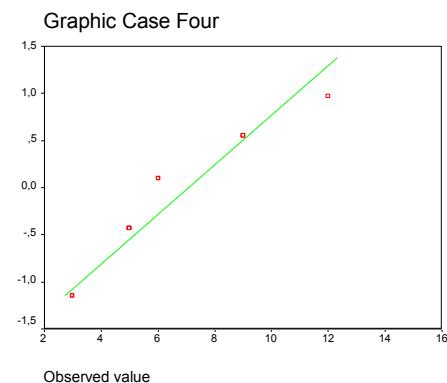
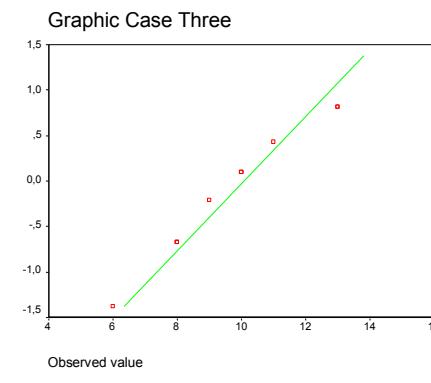
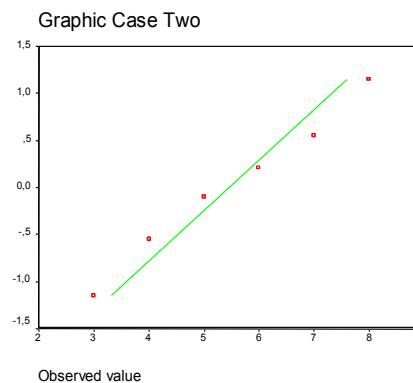
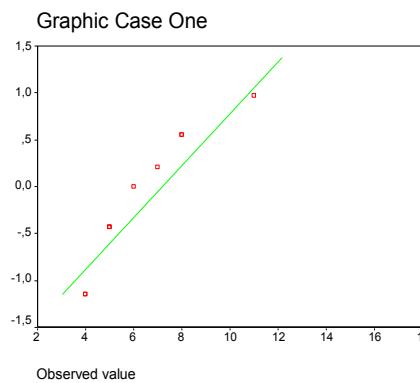


# Ejemplo

## Interpretación Cuantitativa

### Reducción del conjunto de datos

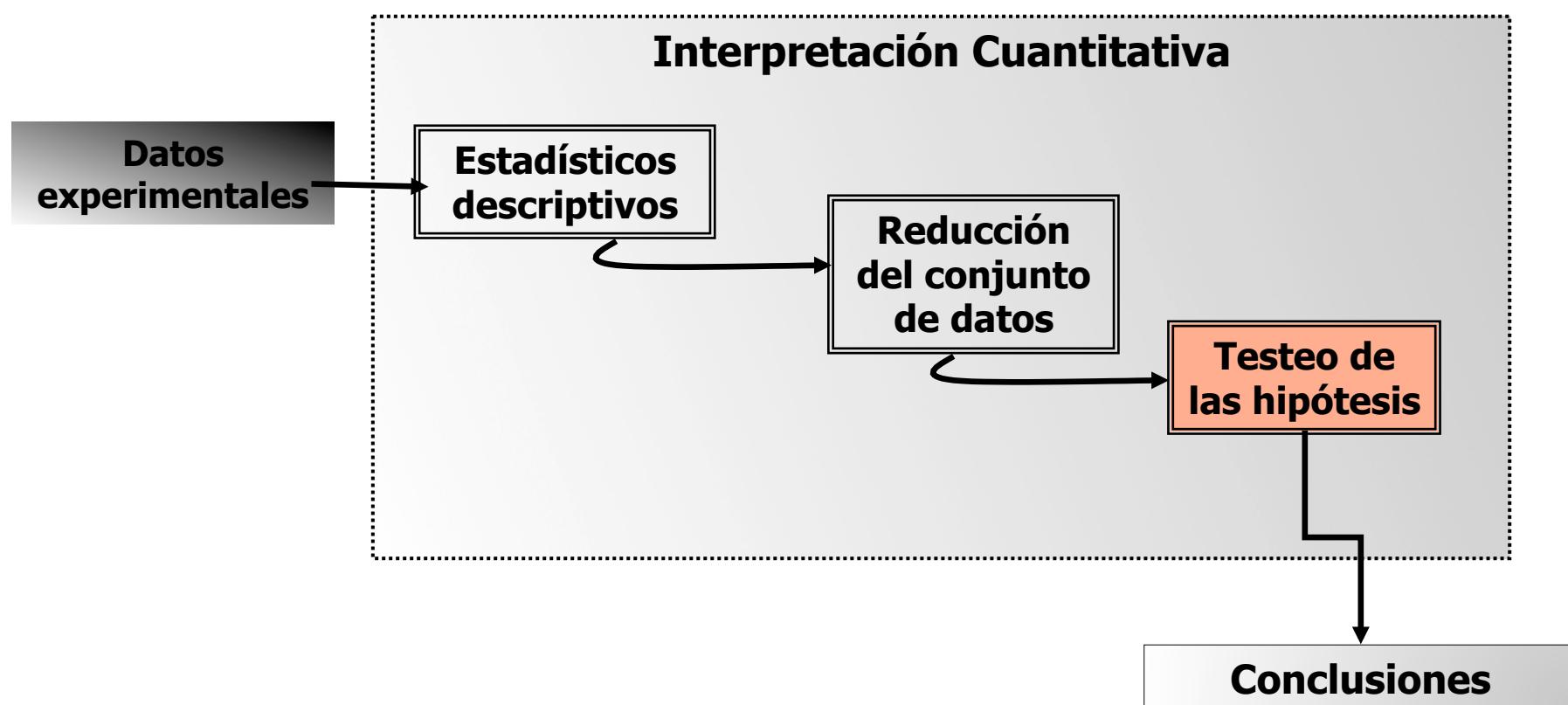
- Mediante el gráfico del test de normalidad, los puntos están cerca de la línea por lo que los datos tienen una **distribución normal** y, además, no hay *outliers*.



- También aplicamos los test de **Shapiro-Wilk** y de **Kolmogorov-Smirnov** encontrando también que los datos eran normales.

# Ejemplo

## Interpretación Cuantitativa



# Ejemplo

## Interpretación Cuantitativa

### Prueba de las hipótesis

- Como los datos siguen una distribución normal, podemos utilizar **tests paramétricos** (que requieren menor cantidad de datos).
- Antes de proceder al análisis, es necesario fijar el **nivel de significación**. Se ha decidido establecer un valor de  $\alpha = 0.1$  que significa un nivel del 90% de confianza.
- Después de todas estas consideraciones es posible seleccionar el mejor test para analizar los datos. Teniendo en cuenta las hipótesis, se ha seleccionado el test ANOVA.



# Ejemplo

## Interpretación Cuantitativa

### Prueba de las hipótesis

Fuente de variación	Grados de libertad	F-Ratio	Sig.
DRT	1	<b>9.137</b>	<b>0.013</b>
NFK	1	12.659	0.005
Interacción	1	0.631	0.445
Error	1		
Total	40		

**H1.** El valor de la métrica DRT afecta a la analizabilidad del esquema de base de datos.

Como  $9.137 > 2.84$ , la hipótesis alternativa 1 es válida porque el valor de la métrica DRT influye en los resultados obtenidos.



# Ejemplo

## Interpretación Cuantitativa

### Prueba de las hipótesis

Fuente de variación	Grados de libertad	F-Ratio	Sig.
DRT	1	9.137	0.013
NFK	1	12.659	0.005
Interacción	1	0.631	0.445
Error	1		
Total	40		

**H2. El valor de la métrica NFK afecta a la analizabilidad del esquema de base de datos.**

Como  $12.659 > 2.84$ , la hipótesis alternativa 2 es válida ya que el valor de la métrica NFK influye en los resultados obtenidos.



# Ejemplo

## Interpretación Cuantitativa

### Prueba de las hipótesis

Fuente de variación	Grados de libertad	F-Ratio	Sig.
DRT	1	9.137	0.013
NFK	1	12.659	0.005
Interacción	1	<b>0.631</b>	<b>0.445</b>
Error	1		
Total	40		

**H3. La interacción de las métricas DRT y NFK afecta a la analizabilidad del esquema de base de datos.**

Como  $0.631 < 2.84$ , la hipótesis alternativa 3 es inválida ya que los valores de las métricas DRT y NFK no infuyen en los resultados obtenidos.



# Ejemplo

## Interpretación Cuantitativa

### Prueba de las hipótesis

- Ambas métricas parecen ser **buenos indicadores de la analizabilidad** de esquemas de bases de datos relacionales.
- También se ha demostrado que **no existe interacción** entre ambas métricas.



# Ejemplo

## Interpretación Cuantitativa

### Prueba de las hipótesis

- Como resultado del proceso experimental podemos concluir:
  - **NFK** parece ser un **buen indicador** de la analizabilidad de bases de datos relacionales.
  - Sin embargo, es **difícil** extraer una conclusión sobre **DRT**
- Sería imprescindible realizar **más experimentos** con las métricas para llegar a conclusiones más sólidas.



# Conclusiones

- Sin realizar mediciones no existe una forma de determinar si el proceso/producto está mejorando.
- Las actividades de medición deben ser dirigidas por un objetivo.
- Las métricas permiten establecer objetivos significativos para la mejora.
- Las métricas permiten identificar las causas de defectos que tienen un mayor impacto en el desarrollo del software.
- Cuando las métricas son aplicadas a los productos software estas permiten identificar:
  - Que requisitos de usuarios pueden cambiar
  - Que módulos son los más propensos a errores
  - La cantidad de pruebas que deben ser planificadas para cada módulo.
- Etc.

