Universidad de Oviedo

ES
Español

Escuela de Ingeniería Informática

# Atributos de calidad




Curso 2022/2023

Jose Emilio Labra Gayo

# Atributos de calidad

También conocidos como:

- Características de la arquitectura
- Requisitos no-funcionales

# Tipos de requisitos

Los requisitos pueden clasificarse en:

- **Requisitos funcionales**: indican lo que debe hacer el sistema, cómo debe comportarse o reaccionar a un estímulo en tiempo de ejecución
- **Requisitos de atributos de calidad**. Anotan (cualifican) requisitos funcionales
  - Ejemplos: Disponibilidad, modificabilidad, usabilidad, seguridad,...
  - También llamados: requisitos no-funcionales
- **Restricciones**. Decisiones de diseño que ya han sido tomadas

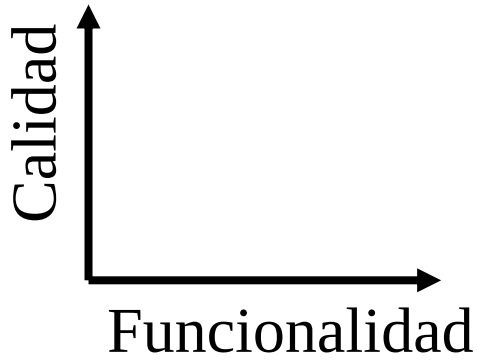
# Requisitos funcionales

Funcionalidad = capacidad del Sistema para realizar un trabajo que se pretende que haga

La funcionalidad tiene una relación extraña con arquitectura:

- No determina la arquitectura
  - Si el único requisito fuese la funcionalidad, el Sistema podría existir como un único modulo monolítico sin ninguna estructura interna

La funcionalidad y los atributos de calidad son ortogonales



# Atributos de calidad (AC)

Los atributos de calidad cualifican la funcionalidad

Si un requisito funcional fuese "*cuando el usuario pulse el botón verde, un diálogo con opciones aparece*" entonces:

- Un AC de rendimiento describiría la velocidad a la que debe aparecer
- Un AC de disponibilidad describiría cuándo podría fallar esta opción o con qué frecuencia sería reparada
- Un AC de usabilidad describiría cómo de fácil es aprender esta función

Los atributos de calidad **sí** determinan la arquitectura

# ¿Qué es la calidad?

Grado en que un Sistema satisfice las necesidades declaradas o implícitas de las personas interesadas propocionando valor

- Grado (no Booleano)
- Calidad = Encaje en un propósito (necesidades de *stakeholders*)
- Satisfacer requisitos (declarados o implícitos)
- Proporcionar valor

Algunas definiciones de calidad

`https://en.wikipedia.org/wiki/Software_quality`

# Atributos Calidad y soluciones de compromiso

Todos los ACs son buenos

...pero el valor depende del Proyecto y *stakeholder*

*"Mejor calidad"...para qué?, para quién?*

ACs no son independientes

Algunos ACs pueden entrar en conflicto

Qué es lo más importante?

Ejemplo: Sistema muy seguro puede ser menos usable

Siempre hay un precio

¡No hay sistema o arquitectura perfecto!

# Especificando Atributos Calidad

- 2 consideraciones:
  - Los ACs por sí mismos no son suficientes
    - No son operacionales
      - Ejemplo: No aporta mucho decir que un Sistema debe ser modificable, o tener alta disponibilidad
  - El vocabulario para describir ACs es muy variado
    - Es necesario describir cada atributo por separado
- Escenarios Calidad: Forma común de especificar requisitos de AC

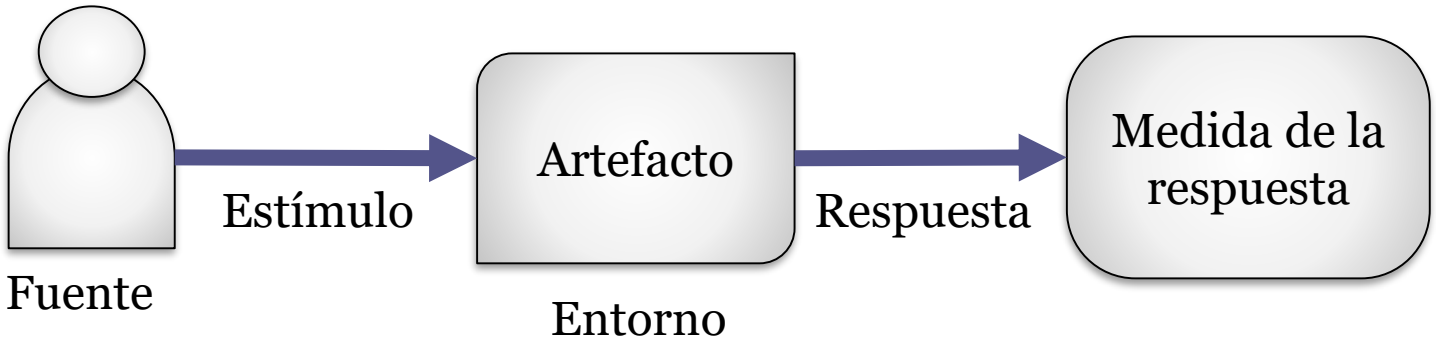
# Escenario de Calidad

Describe un estímulo de un Sistema y una respuesta medible a dicho estímulo

- Estímulo = evento iniciado por una persona o sistema
- El estímulo genera una respuesta

Debe ser medible

- La respuesta debe ser visible externamente y medible



# Componentes de escenario calidad

**Fuente**: Persona o Sistema que inicia el estímulo

**Estímulo**: Evento que requiere que responda el sistema

**Artefacto**: Parte del sistema o el sistema completo

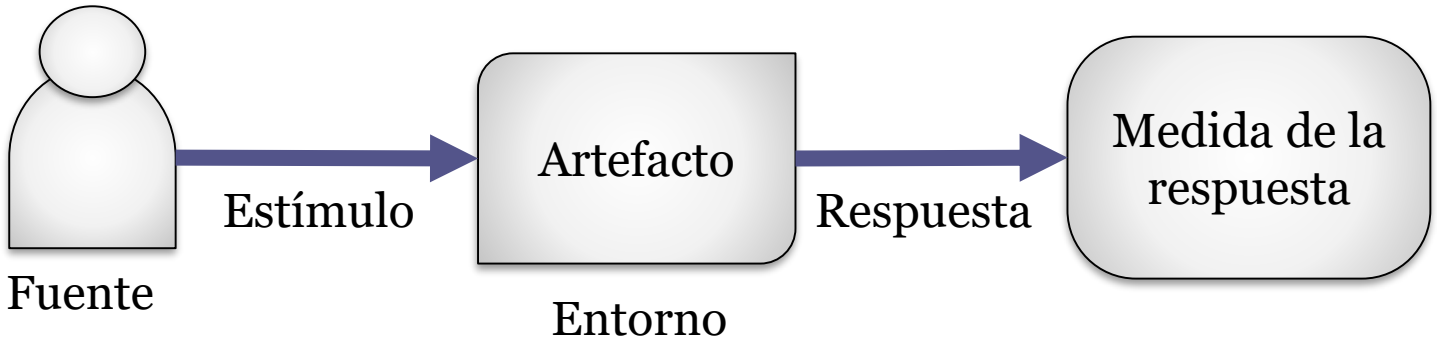
**Respuesta**: Acción visible externamente

**Medida de respuesta**: Criterio de éxito para el escenario

- Debe ser específico y medible

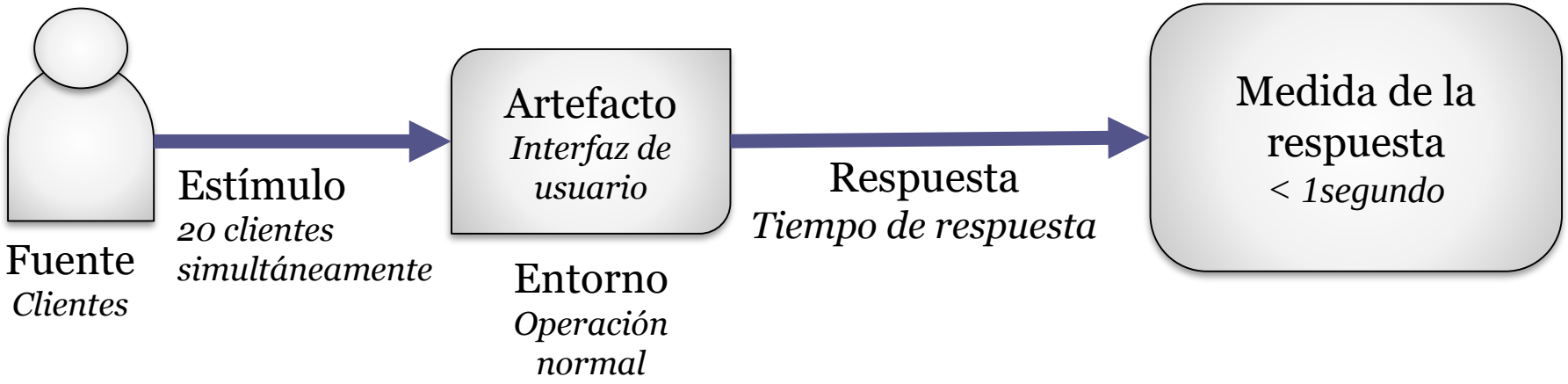
**Entorno**: Circunstancias operativas

- Siempre debe ser definido (incluso si es "*normal*")

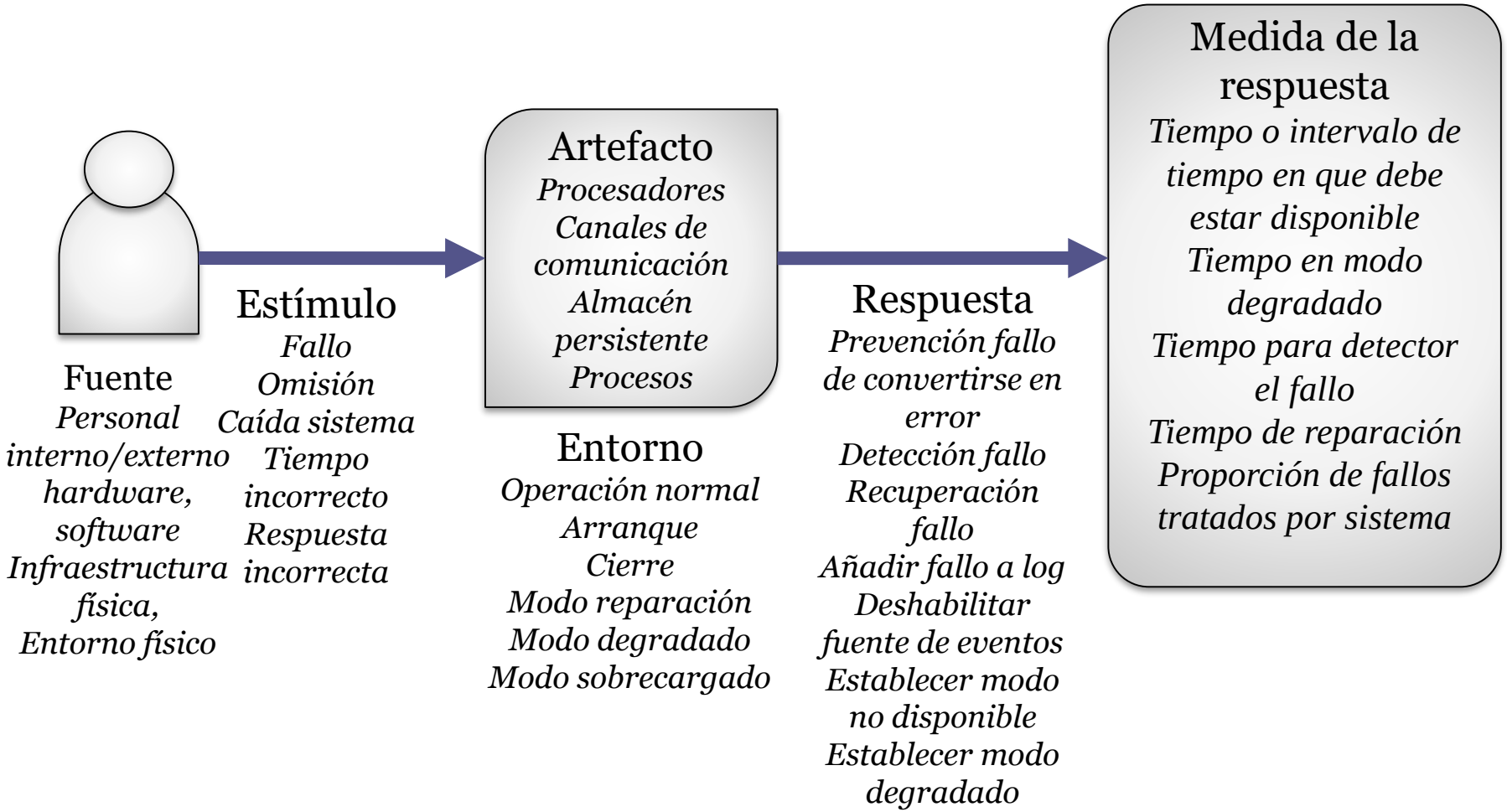


# Escenario de calidad, ejemplo 1

***Rendimiento**: Si hay 20 clientes simultáneamente, el tiempo de respuesta debería ser menos que 1 segundo en circunstancias normales*



# Escenario calidad para disponilibidad



# Tipos de escenarios de calidad

- Escenarios de uso
  - El Sistema es utilizado (cualquier caso de uso o función del Sistema que se ejecuta)
    - Describe cómo se comporta el Sistema en dichos casos
      - Tiempo de ejecución, velocidad de respuesta, consume de memoria, *throughput*,...
- Escenarios de cambio (o modificación)
  - Cambio en cualquier componente dentro del Sistema, en entorno o la infraestructura operacional
- Escenarios de fallo:
  - Alguna parte del Sistema, infraestructura o entorno falla

# Priorizando escenarios de calidad

Los escenarios deben ser priorizados

- 2 valores (Low/Medium/High)
  - Cómo es de importante para el éxito (cliente)
  - Cómo de difícil es de alcanzar (arquitecto)

| Ref | AQ | Escenario | Prioridad |
|---|---|---|---|
| 1 | Disponibilidad | Cuando la base de datos no responda, el Sistema debería registrar el fallo en el log y responder con datos anteriores durante 3 seg. | High, High |
| 2 | Disponibilidad | Un usuario que busca elementos de tipo X recibe una lista de Xs el 99% del tiempo como media a la largo del año | High, Medium |
| 3 | Escalabilidad | Nuevos servidores pueden ser añadidos durante la ventana de mantenimiento (en menos de 7 horas) | Low, Low |
| 4 | Rendimiento | Un usuario puede ver los resultados de búsqueda en menos de 5 segundos cuando el Sistema tiene una carga de 2 búsquedas por seg | High, High |
| 5 | Fiabilidad | Las actualizaciones a elementos externos de tipo X deberían reflejarse en la aplicación dentro de 24 horas del cambio | Low, Medium |
| | . . . | . . . | . . . |

# Identificando Atributos de calidad

- ¿Cómo encontrar los Atributos de calidad?
  - La mayoría de las veces no están explícitos
    - Pueden ser mencionados de pasada en req. funcionales
    - Normalmente implícitos o mencionados sin mucha consideración
  - Arquitecto del software debe tratar de identificarlos
- Talleres de atributos de calidad
  - Reuniones con *stakeholders* para identificar y priorizar los atributos y escenarios de calidad
- Listas formales
  - ISO25010
  - Wikipedia: https://en.wikipedia.org/wiki/List_of_system_quality_attributes

# Atributos de calidad habituales

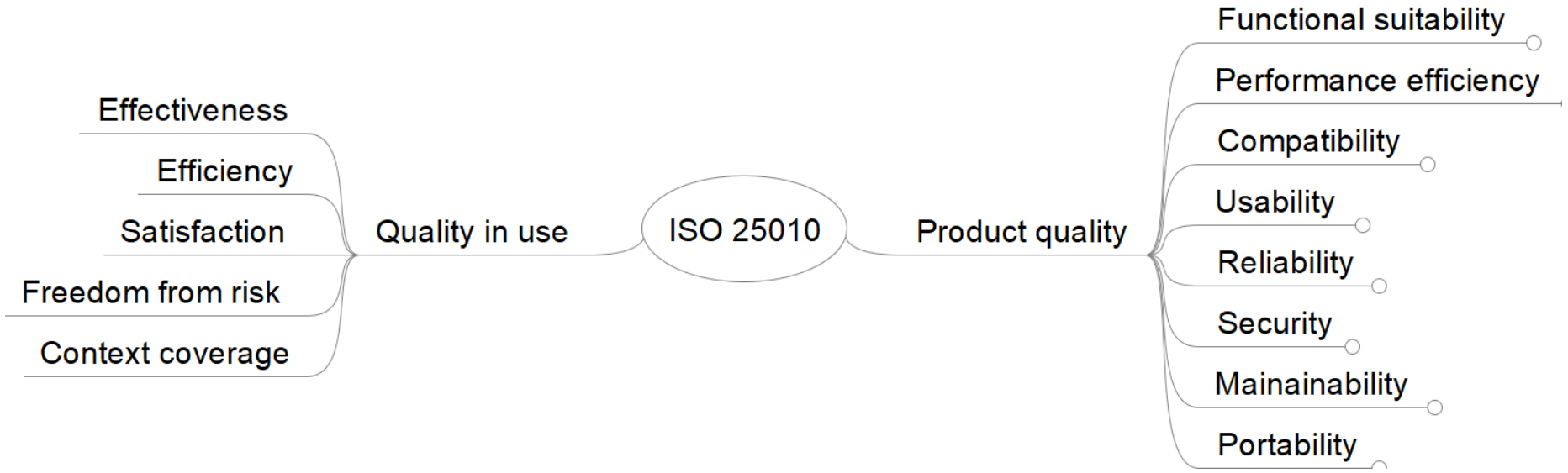
Disponibilidad
Modificabilidad
Rendimiento
Seguridad
Testabilidad
Mantenibilidad
Usabilidad
Escalabilidad
Interoperabilidad
Portabilidad
Cambiabilidad
. . .

# ISO-25010 Software Quality Model

*Systems and software Quality Requirements and Evaluation (SQuaRE)*

2 partes:

- *Quality in-use*
- *Product quality*

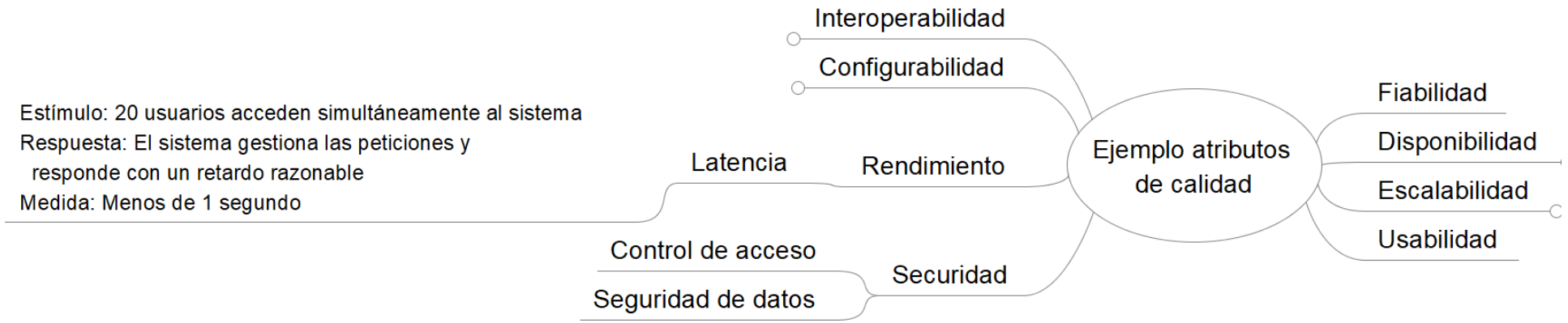



https://arquisoft.github.io/Iso25010QualityAttributes.html

# Árbol de Atributos de calidad

Representaciones Mindmap pueden ser útiles para visualizar Atributos de calidad

Permiten seleccionar y ampliar atributos bajo demanda



# Midiendo atributos de calidad

Los escenarios de AC requieren medir respuestas

Problemas habituales:

- Muchos AC tienen significados vagos
  - Ejemplo: desplegabilidad, escalabilidad
- Definiciones muy diversas
  - No hay definiciones aceptadas universalmente
- Demasiado compuestos
  - Normalmente, un AC está compuesto de varias características

# Métricas de software

Intentar medir de forma objetiva aspectos de la calidad

Numerosas métricas para diferentes aspectos

Más información: https://en.wikipedia.org/wiki/Software_metric

# Medidas operacionales

Muchas métricas dependiendo del atributo

- Valores absolutos
  - Ejemplos: Nº de usuarios, Nº de errores
- Valores y modelos estadísticos
  - Ejemplo

$$availability = \frac{MTBF}{MTBF + MTTR}$$ donde $MTBF = mean\ time\ between\ failures\ (uptime)$, $MTTR = mean\ time\ to\ recover\ (downtime)$

| Availability | Downtime/90días | Downtime/año |
|---|---|---|
| 99,0% | 21 horas, 36 min | 3 días, 15.6 horas |
| 99,99% | 12 min, 58secs | 52min, 34 segs |
| 99,9999% | 8 segs | 32 segs |

# Medidas estructurales

Medir estructura de los módulos

Ejemplo: Complejidad ciclomática (CC) de McCabe, 1976

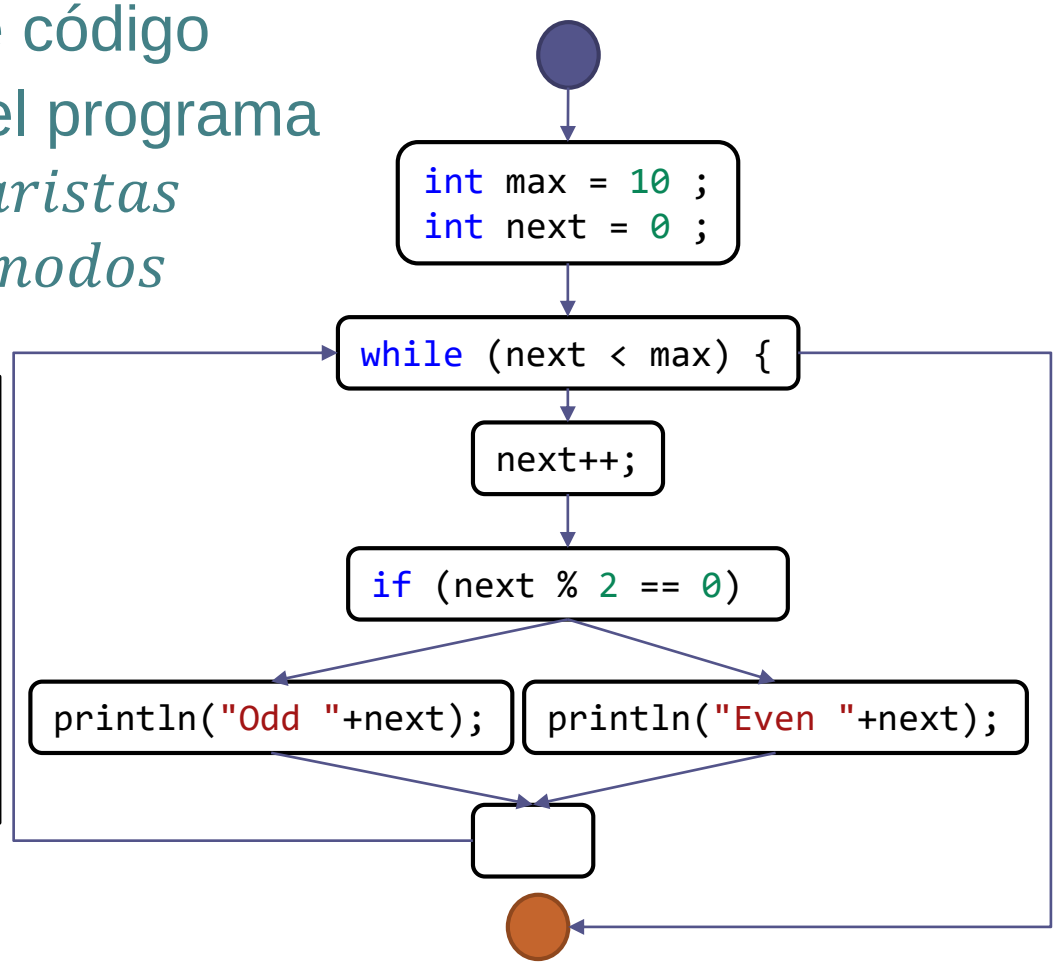
Medida objetiva de complejidad de código

Se crea grafo de flujo de control del programa

$CC = E - N + 2$ donde $E = nº\ de\ aristas$, $N = nº\ de\ nodos$

Ejemplo:

```
public static void main(String[] args) {
    int max = 10 ;
    int next = 0 ;
    while (next < max) {
      next++;
      if (next % 2 == 0)
       System.out.println("Even " + next);
      else
       System.out.println("Odd " + next);
     }
   }
```




$$CC = 10 - 9 + 2 = 3$$

# Medidas de procesos

Medir aspectos del proceso como agilidad, testabilidad, mantenibilidad, etc.

Ejemplo:

- Testabilidad: Cobertura de código mediante prueba
  - Porcentaje de líneas de código que han sido ejecutados durante las pruebas
- Desplegabilidad:
  - % de despliegues con éxito/fallidos
  - Tiempo de despliegue
  - Incidencias/errores en despliegue

# 4 métricas clave (métricas DORA)

Origen: Informe State of DevOps de DORA (equipo DevOps Research and Assessment)

Miden rendimiento de la entrega de software

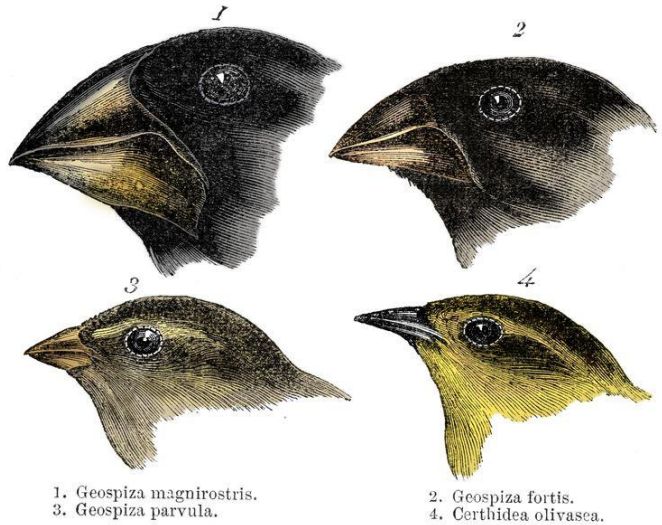
1. Lead time to change (tiempo de espera para cambios)
   Tiempo desde que se hace commit hasta que código va a producción
2. Frecuencia de despliegue
   Frecuencia con la que se despliega código en producción
3. Tiempo medio de restauración (MTTR)
   Cuánto tiempo se tarda en restaurar un servicio tras un incidente
4. Tasa de fallos tras cambios
   Porcentaje de cambios en producción que resultan en fallos

https://www.devops-research.com/quickcheck.html

# Gestionando atributos de calidad

## Arquitecturas evolutivas

- Función de encaje (fitness): mecanismo que proporciona una valoración de la integridad objetiva alguna combinación de atributos de calidad
- Medir valores de AC y evolucionar la arquitectura



# Fin