Arquitectura de Software 75.73

FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

(Last updated: 2021.2)

Atributos de Calidad #1

Christian Calónico
75.73 Arquitectura de Software

FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Atributos de Calidad #1

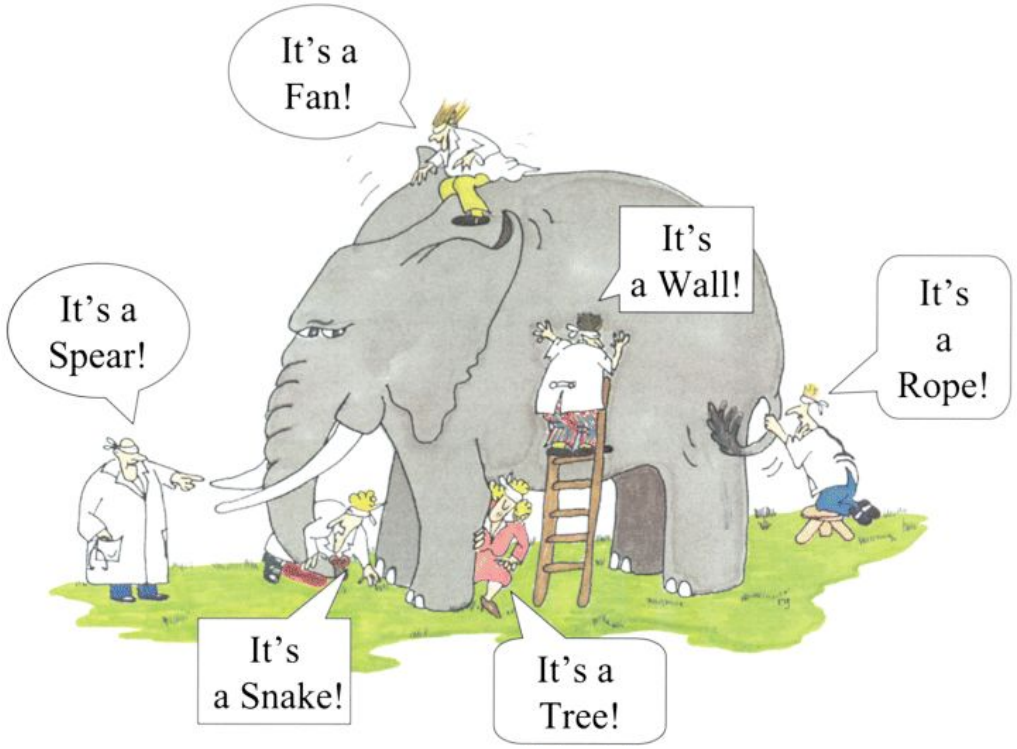
Hoja de Ruta

1. Introducción a los QA
2. Availability (disponibilidad)
3. Performance
   a. Network Performance
   b. User-Perceived Performance
   c. Efficiency
4. Scalability (escalabilidad)
5. Algunas Tácticas
   a. Availability
   b. Performance

Requerimientos del Sistema

- Requerimientos funcionales: definen lo que el sistema *debe hacer* (comportamientos)
- Atributos de calidad (QA): definen *cualidades* de los requerimientos funcionales.
  - También conocidos como no-funcionales
- Restricciones (*constraints*): decisiones de diseño ya tomadas, que no pueden cambiar.

Atributos de Calidad

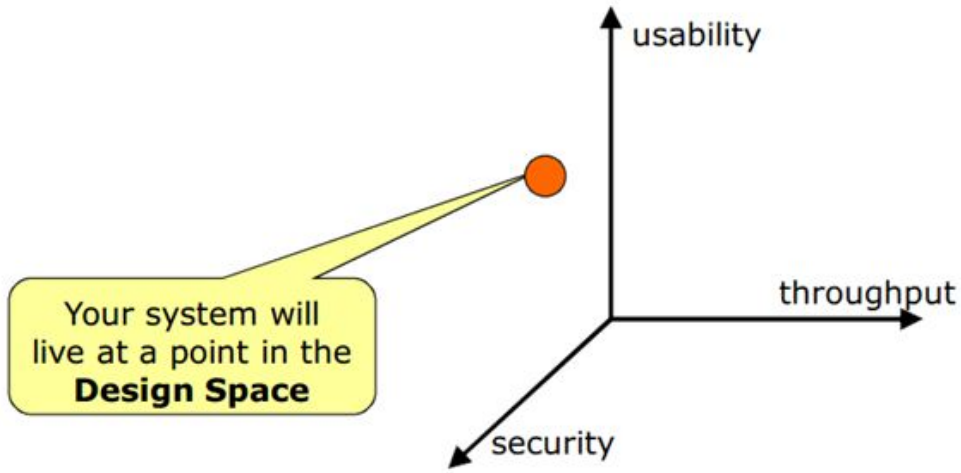


Es una propiedad/cualidad ***mensurable*** del sistema, usada para indicar cuán bien el sistema satisface las ***necesidades*** de sus ***stakeholders***

Atributos de Calidad

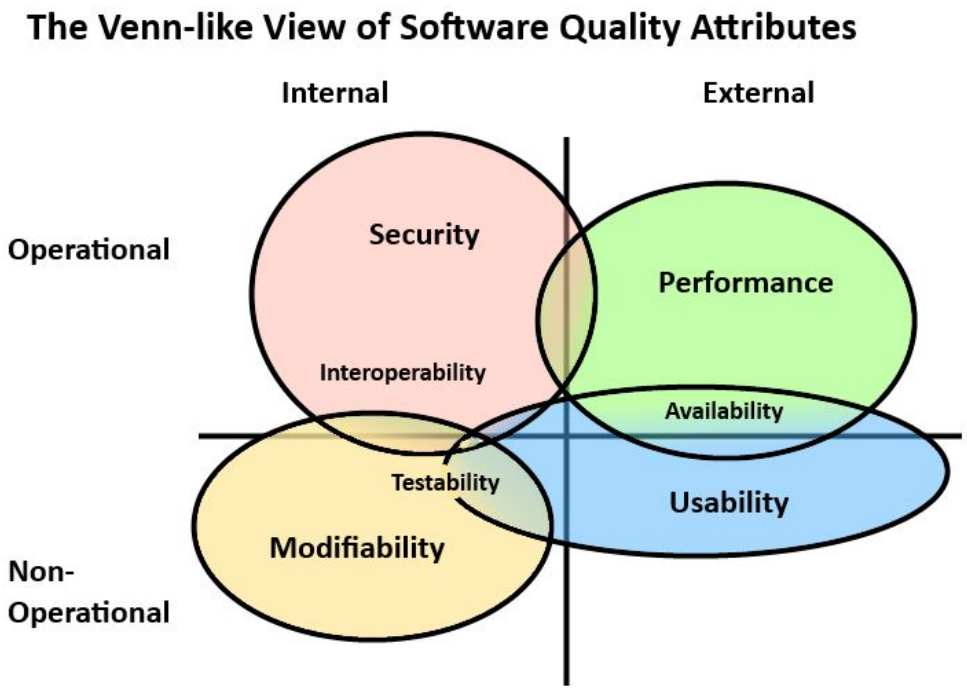
- “Es una dimensión de una cualidad, usada para evaluar un sistema”

- El desafío es optimizar múltiples QAs.
  - En general, favorecer un QA implica desfavorecer otro (trade-off)
  - Priorizar es fundamental al decidir la arquitectura



Atributos de Calidad

- ¿Qué piden los *external stakeholders*?
- ¿Actitud del *business stakeholder*?
- ¿Qué ocurre con los QA internos y no-operativos?
- ¿Cómo impactaría el concepto de trade-off en este escenario?



Atributos de Calidad #1

Hoja de Ruta

1. Introducción a los QA
2. Availability (disponibilidad)
3. Performance
   a. Network Performance
   b. User-Perceived Performance
   c. Efficiency
4. Scalability (escalabilidad)
5. Algunas Tácticas
   a. Availability
   b. Performance

Availability

- Una falla (*failure*) ocurre cuando el sistema deja de brindar un servicio consistente con su especificación.
  - Es observable por los usuarios del sistema, sean humanos u otros sistemas
- Una falta (*fault*) se convierte en falla si no es corregida o enmascarada
  - Pasa a ser observable por los usuarios.

Habilidad del sistema para *reparar o enmascarar fallas* de manera tal que el período que el sistema está fuera de servicio no exceda un determinado valor durante un lapso de tiempo especificado

Availability

- Tipos de fallas
  - Omission – el sistema no responde a un pedido
  - Crash – el sistema sufre omisiones repetitivas
  - Timming – el sistema responde, pero fuera de tiempo
  - Response Failure – el sistema responde, pero un valor incorrecto

- El sistema requiere disponer de, por ejemplo:
  - Procesador
  - Canal de comunicación
  - Proceso correcto
  - Almacenamiento

Availability

- Objetivo: minimizar el período fuera de servicio (outage time) a través de mitigación de fallas.
  - ¿Cada cuánto se genera una falla (MTBF)?
  - ¿Cómo se manifiesta la falla? ¿Qué consecuencias tiene?
  - ¿Cuánto tiempo se necesita para restablecer el servicio (MTTR)? ¿Cómo?
  - ¿Cuánto tiempo se acepta que deje de estar operativo el sistema?

- Posibles acciones ante una falla:
  - Log de la falla
  - Notificar usuarios seleccionados u otros sistemas
  - Pasar a un modo de degradación (menos capacidad o menos funciones)
  - Pasar a un modo de no disponibilidad (hasta que se repare la falla)

Availability

Table 5.1. System Availability Requirements

Availability	Downtime/90 Days	Downtime/Year
99.0%	21 hours, 36 minutes	3 days, 15.6 hours
99.9%	2 hours, 10 minutes	8 hours, 0 minutes, 46 seconds
99.99%	12 minutes, 58 seconds	52 minutes, 34 seconds
99.999%	1 minute, 18 seconds	5 minutes, 15 seconds
99.9999%	8 seconds	32 seconds

Se suele decir que 99.999% es “cinco nueves”

No se computa el tiempo de inactividad programada

La disponibilidad de un sistema es la probabilidad de que esté operativo cuando se lo precisa

$$\alpha = \frac{\text{mean time to failure}}{\text{mean time to failure} + \text{mean time to repair}}$$

Suele formar parte del SLA (*service level agreement*), acordado entre el proveedor de servicio y el cliente

Atributos de Calidad #1

Hoja de Ruta

1. Introducción a los QA
2. Availability (disponibilidad)
3. Performance
   a. Network Performance
   b. User-Perceived Performance
   c. Efficiency
4. Scalability (escalabilidad)
5. Algunas Tácticas
   a. Availability
   b. Performance

Performance

- Eventos
  - Interrupciones
  - Mensajes
  - Pedidos del usuario
  - Pedidos de otro sistema
  - Paso del tiempo

- Naturaleza de los eventos
  - Periódicos (ej. cada 10 ms)
    - Ej. sistemas de tiempo real
  - Estocásticos (ej. 100 reqs/s)
    - Ej. servidor web
  - Esporádicos (ej. alarma)

Habilidad del sistema para reaccionar ante ciertos *eventos* en un determinado tiempo

Ejemplos:

Cantidad de transacciones que pueden ser procesadas por minuto

Tiempo que demora el sistema entre un evento y una eventual reacción

Performance

- La interacción entre elementos determina [Fielding]:
  - La velocidad y eficiencia en el uso de la red
  - La performance percibida por el usuario

- La arquitectura determina la naturaleza de dicha interacción
  - Importante su elección

- La performance suele estar acotada por:
  - Requerimientos del sistema, que implican un costo básico (datos en A, uso en B)
  - Estilo de interacción seleccionado (N interacciones para llegar desde A hasta B)
  - Materialización de la arquitectura
  - Implementación de cada componente (ej. qué tan rápido se generan/consumen datos)

Performance

¿Cuándo atacar problemas de performance?

1. Sobre aplicación “Up & Running”
   - Instrumentación para realizar mediciones.
   - Optimizar en base a dichas mediciones.

2. Optimización temprana
   - En cada decisión de arquitectura.

Nota: algunas decisiones de arquitectura no son fáciles de revertir/mejorar con optimizaciones tardías.

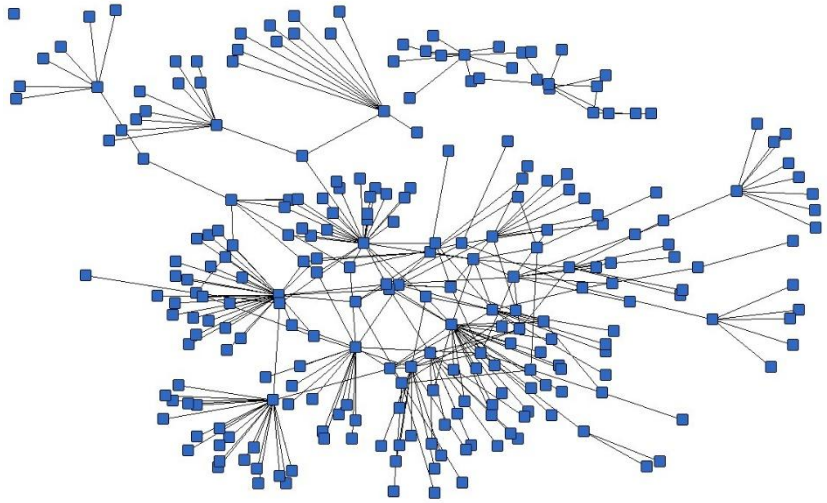
Atributos de Calidad #1

Hoja de Ruta

1. Introducción a los QA
2. Availability (disponibilidad)
3. Performance
   a. Network Performance
   b. User-Perceived Performance
   c. Efficiency
4. Scalability (escalabilidad)
5. Algunas Tácticas
   a. Availability
   b. Performance

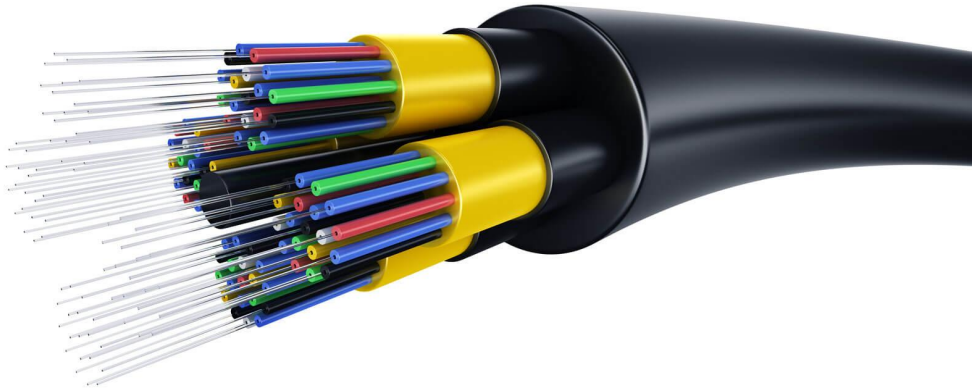
Network Performance

- Throughput
  - Velocidad de transmisión de información entre componentes (ej. bytes/seg).
  - Incluye datos de aplicación (*payload*) + *overhead.*
  - El overhead puede ser por *setup inicial* o por *interacción.*
- Capacity
  - Máximo throughput o carga máxima soportada por el sistema.
- El throughput puede ser *cualquier-cosa/tiempo*
  - Es común hablar de transacciones por segundo (tx/s, tps)
  - Suele medirse con el Load Test
  - Se usan transacciones representativas (ej. frecuentes, típicas, etc.)
  - Importante cumplir con la capacidad requerida por el SLA

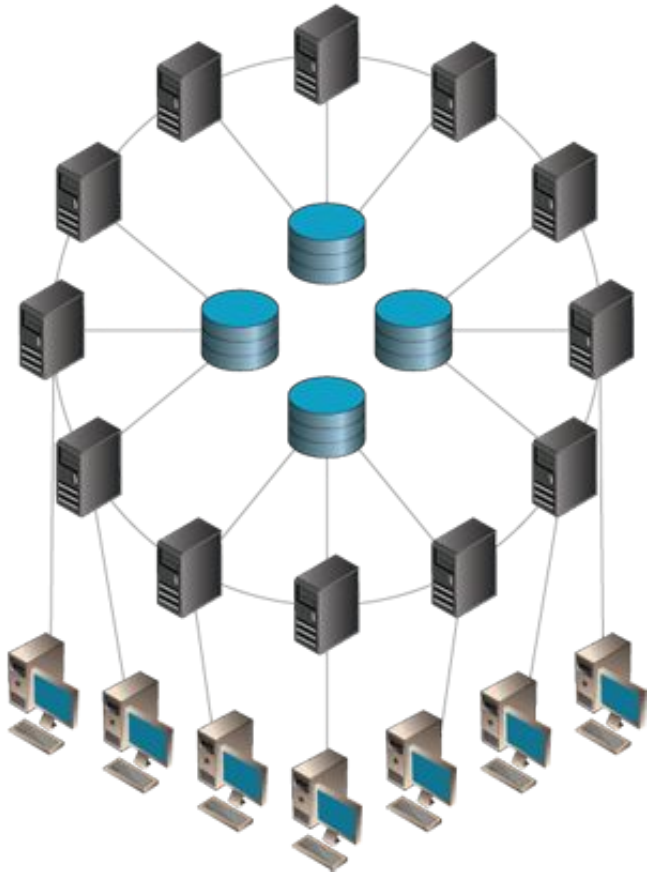


Network Performance

- Bandwidth
  - Máximo throughput disponible en un cierto canal de comunicación (ancho de banda)
- Usable bandwidth
  - Porción del ancho de banda disponible para la aplicación en cuestión



Network Performance



- Los estilos definen [Fielding]
  - La cantidad de interacciones por request
  - La granularidad de los datos
- Mejor performance, no utilizar la red
  - Caso extremo
- Estilos más eficientes, minimizan accesos
  - Eliminando la necesidad (mover los datos cerca)
  - Reduciendo la frecuencia
  - Reusando interacciones pasadas

Atributos de Calidad #1

Hoja de Ruta

1. Introducción a los QA
2. Availability (disponibilidad)
3. Performance
   a. Network Performance
   b. User-Perceived Performance
   c. Efficiency
4. Scalability (escalabilidad)
5. Algunas Tácticas
   a. Availability
   b. Performance

User-Perceived Performance

Mediciones desde el punto de vista del *usuario*.

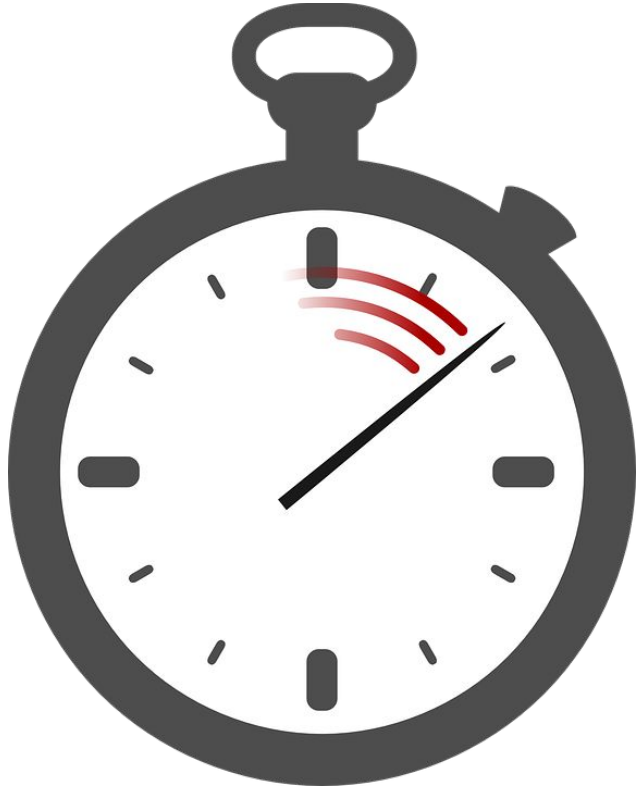
1. Completion Time (Response Time)

   - "Tiempo requerido por el sistema para completar un request"

   Medido desde afuera (difícil por tiempos pequeños, variables, dependientes de parámetros)

2. Latency

   - "Tiempo mínimo que necesita el sistema para ejecutar cualquier request, incluyendo el 'no hacer nada'" (Fowler)
   - "Tiempo que transcurre entre el estímulo inicial y la primera indicación de respuesta" (Fielding)



User-Perceived Performance

Responsiveness

- Tiempo que requiere un sistema para “aceptar” (*acknowledge*) un request

- Ejemplos:

  - Una progress bar mejora responsiveness, pero no mejora el completion time

  - Un web-browser renderiza una imagen grande a medida que la va recibiendo (vs uno que no lo hace)
    - Igual network performance
    - Mejor user-perceived performance

User-Perceived Performance

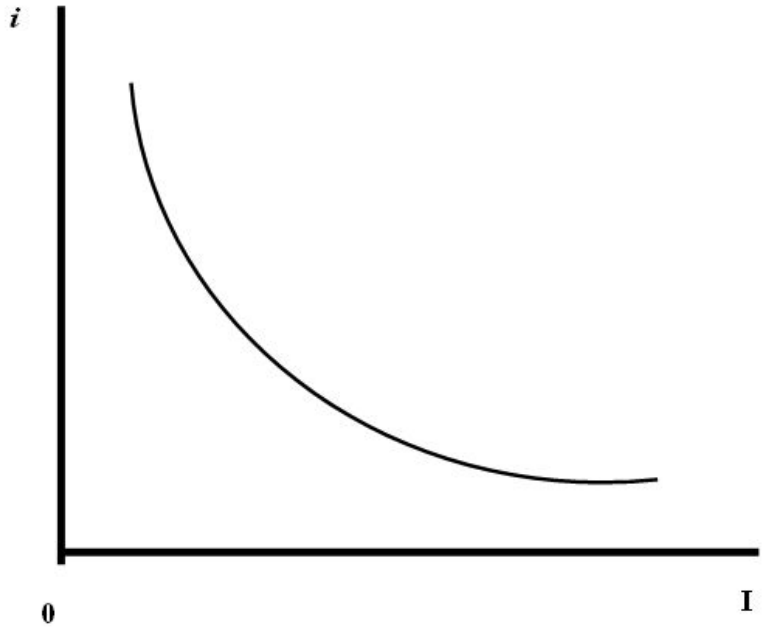
- Optimizar responsiveness suele afectar negativamente el completion time (y viceversa)
    - Ej.: la tasa de compresión de comunicación mejora conforme aumenta la cantidad de datos
    - Acumular datos antes de comprimir ayudaría al completion time, pero reduciría responsiveness
- Muchas veces, hay que pensar qué conviene
    - Ej.: se busca optimizar el funcionamiento de un web-server, sin saber si del otro lado tenemos:
        - Un web browser (prefiere responsiveness)
        - Un web spider (prefiere completion time)

Atributos de Calidad #1

Hoja de Ruta

1. Introducción a los QA
2. Availability (disponibilidad)
3. Performance
   a. Network Performance
   b. User-Perceived Performance
   c. Efficiency
4. Scalability (escalabilidad)
5. Algunas Tácticas
   a. Availability
   b. Performance

Efficiency



- La eficiencia es la performance por recurso consumido
  - Cumplir un pedido consume recursos
  - Es posible que el sistema esté sirviendo varios pedidos al mismo tiempo
  - Ej. “20 tps, utilizando 2 procesadores”
- Carga (load): cantidad de stress en un sistema
  - Ej. “el response time es de 1s con una carga de 20 usuarios, pero degrada a 3s cuando la carga sube a 25 usuarios”
  - Importante el Load Test
- Load Sensitivity: variación del response time en función de la carga (degradación de la performance)

Atributos de Calidad #1

Hoja de Ruta

1. Introducción a los QA
2. Availability (disponibilidad)
3. Performance
   a. Network Performance
   b. User-Perceived Performance
   c. Efficiency
4. Scalability (escalabilidad)
5. Algunas Tácticas
   a. Availability
   b. Performance

Scalability

- Habilidad de la arquitectura para soportar un número grande de componentes, o interacciones entre componentes [Fielding]

- Capacidad de agrandarse al creciente número de usuarios, transacciones, picos, etc.

Habilidad del sistema para *incorporar* recursos adicionales de manera efectiva

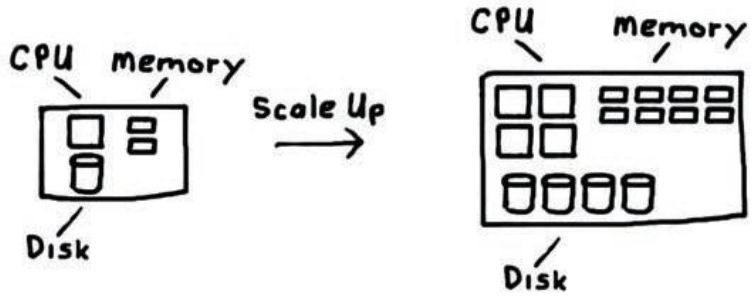
Considerando efectividad como:

- Mejora mensurable de algún atributo de calidad
- No requerir un esfuerzo inapropiado
- No interrumpir el funcionamiento del sistema

Scalability

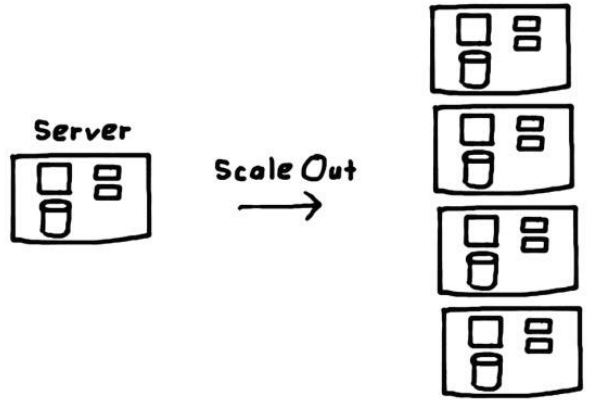
Escalabilidad vertical (*scaling up*)

- Agregar más recursos a unidades físicas
- "Crecer en hardware" (ej. agregar memoria)
- Esfuerzo bajo, crecimiento limitado



Escalabilidad horizontal (*scaling out*)

- Agregar más recursos a unidades lógicas
- "Más servidores (nodos) trabajando como un todo" (ej. agregar server a cluster)
- Más potente, más complicado



Scalability

Elasticidad (*elasticity*)

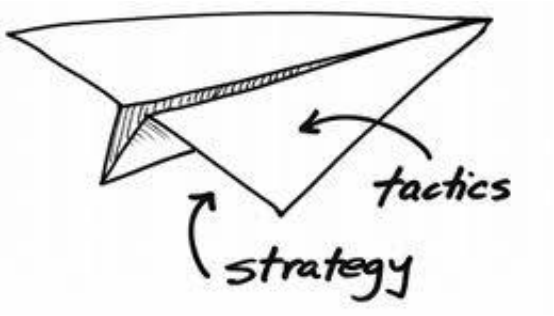
- Propiedad del sistema para crecer/decrecer en la cantidad de recursos, a fin de adaptarse a los cambios de demanda
- Ej. escalabilidad horizontal en la nube

Atributos de Calidad #1

Hoja de Ruta

1. Introducción a los QA
2. Availability (disponibilidad)
3. Performance
   a. Network Performance
   b. User-Perceived Performance
   c. Efficiency
4. Scalability (escalabilidad)
5. Algunas Tácticas
   a. Availability
   b. Performance

Tácticas de Disponibilidad



Objetivo – Mantener al sistema, a pesar de las fallas, dentro de su especificación

Detección

- Ping/Echo
- Heartbeat/Watchdog
- Voting (*)

Recuperación

- Active Redundancy (Hot restart)
- Passive Redundancy (Warm restart)
- Spare (Cold restart)

Prevención

- Removal from Service
- Transactions
- Process Monitor

Ping/Echo (táctica de detección)

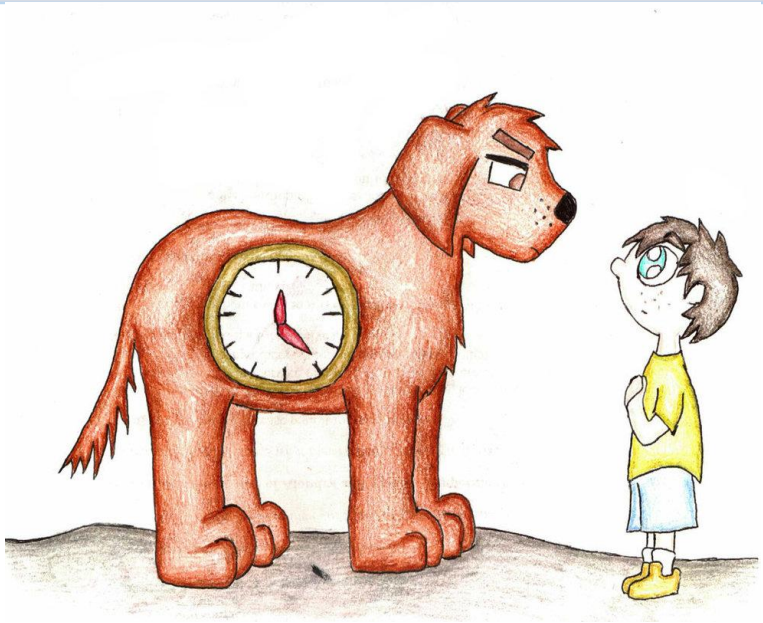
- Ejemplos de uso:
  - Dos servidores, responsables de una misma tarea
  - Clientes, para asegurarse que el servidor (y el canal de comunicación) estén activos y con cierta velocidad de respuesta

- Variante jerárquica:
  - Un detector hace ping a un proceso con quien comparte procesador
  - Otros (externos), de jerarquía mayor, controlan a los de menor rango
  - Se reduce el flujo de mensajes

Un componente hace un *ping* y espera que otro le devuelva un *echo*, en un cierto tiempo predefinido

Heartbeat/Watchdog (táctica de detección)

Un componente emite *latidos* periódicos, que otro escucha; si no llega el latido, se asume falla del primero, notificando al componente corrector



Variante:

- El latido puede también contener información (ej. log de la última transacción realizada)

Heartbeat vs Watchdog:

- El heartbeat cambia algo, que implica vida
- El watchdog espera que algo cambie en cierto tiempo; si no ocurre, actúa.

Voting (táctica de detección/recuperación)

- Varios procesos (diferentes procesadores) reciben los mismos *inputs*, y producen un *output*, que envían al componente *voter*
- Si no todos coinciden, el *voter* detecta anomalía
  - Majority rules
  - Preferred component
- La diversidad es costosa
  - Mismo algoritmo
  - Distintos softwares –equipos dev–, ejecutados en distintas plataformas

- Ej. Sistema de control aéreo
- ¿Single Point of Failure (SPOF)?

Verifica que componentes replicados estén produciendo el mismo resultado

Redundancia

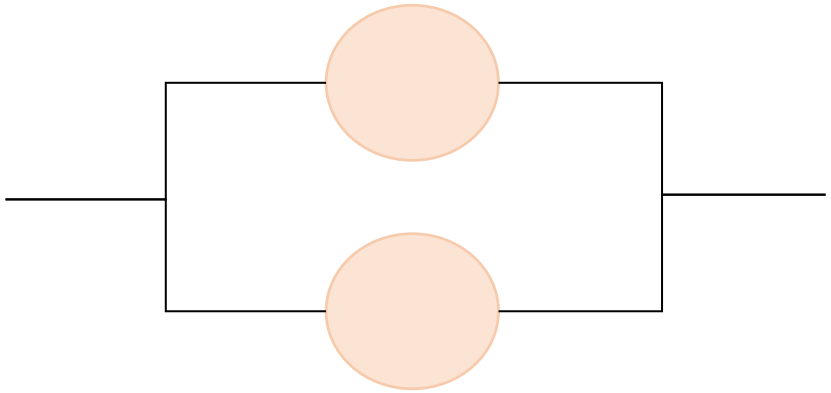
Se pueden evitar los SPOF utilizando redundancia

- Redundancia de Información
  - Información redundante en una transmisión o almacenamiento.
  - Útil para detectar/corregir errores (ej. checksum).

- Redundancia de Tiempo
  - Tiempo extra de procesamiento, a fin de volver a realizar acciones.
  - Útil para fallas transitorias o intermitentes.

- Redundancia Física
  - Hardware y software extra para que el sistema soporte el mal funcionamiento de uno de sus componentes

Redundancia Activa (táctica de recuperación)



Existen componentes redundantes continuamente en funcionamiento, listos para ser utilizados.

- Cuando ocurre una falla, el *downtime* es muy corto (ms)
- Usualmente usada en cliente/servidor
- En sistemas distribuidos, se busca redundancia hasta en los enlaces de red
- Se debe garantizar que los mensajes lleguen siempre a todos los componentes

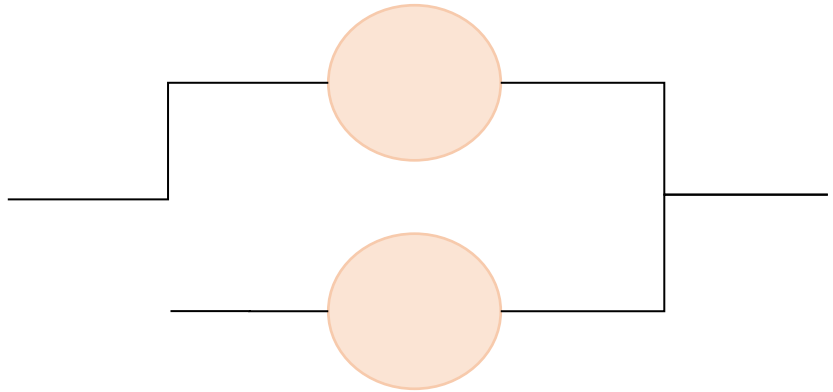
- Los componentes reciben y procesan el mismo *input* en paralelo.
- Se mantienen todos sincronizados.
- Se utiliza la respuesta del primero en responder (las otras se descartan)

Redundancia Pasiva (táctica de recuperación)

Existe al menos un componente redundante, que tomará el control (*takeover*) solo en caso que el principal se desconecte por alguna razón.

- Un componente (*primary*) responde los eventos e informa a los otros componentes (*standbys*) las actualizaciones periódicas del *estado.*
- La sincronización es responsabilidad del *primary.*

- ¿Tiempo de *switch*?
- ¿Es una alternativa más o menos costosa que la redundancia activa?



Spare (táctica de recuperación)

Una plataforma completa es configurada para reemplazar diferentes componentes que podrían fallar

- El reemplazo implica reiniciar el *spare* con la configuración de software correcta y actualizar su *estado* interno en caso de ser necesario.
- Útil hacer *checkpoints* periódicos facilita la sincronización
- El *downtime* es del orden de minutos
- Ej. reemplazo de una estación de trabajo

Tácticas de Prevención

- Removal from Service
  - Retirar el componente periódicamente para anticiparse a las fallas
  - Puede ser automático o manual
  - Ej. *reboots* periódicos para evitar fallas por *memory leaks*

- Transactions
  - Pasos secuenciales que se realizan sobre un componente
  - Se realizan todos, o ninguno
  - Útil para evitar problemas por errores de un paso, o colisión de threads

- Process Monitor
  - Cuando se detecta una falla en un componente, un proceso de monitoreo puede detenerlo, relanzarlo, recuperar su estado (state syncronization), etc.

Atributos de Calidad #1

Hoja de Ruta

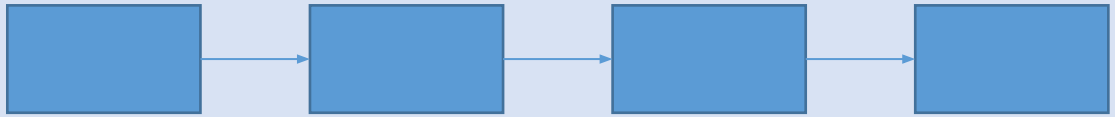
1. Introducción a los QA
2. Availability (disponibilidad)
3. Performance
   a. Network Performance
   b. User-Perceived Performance
   c. Efficiency
4. Scalability (escalabilidad)
5. Algunas Tácticas
   a. Availability
   b. Performance

Tácticas de Performance

Objetivo – Reducir el tiempo de respuesta a un evento (latencia)

RECURSOS – ¿Qué serían?

- Hardware (CPU, storage, bandwidth, memory, etc)
- Software (buffers, critical sections, etc.)



- Cada parte agrega latencia

¿Cuál es el cuello de botella?

BLOQUEOS – Un proceso puede bloquearse por:

- Disputa de recursos – un proceso espera porque otro proceso está accediendo a un recurso compartido (más latencia).
- Disponibilidad de recursos – un proceso debe esperar por un recurso no disponible (ej. recurso en offline, en falla, etc.)
  Identificar posibles fuentes
- Dependencias de resultados – un proceso espera porque requiere inputs de otro proceso que aún no terminó
  Inputs en paralelo vs serie

Tácticas de Performance

Demanda de Recursos

- Reduce Events Number
- Reduce Resources Consumption

- ¿Cada cuanto llegan?

ej. sampling reduction

no siempre se puede

- ¿Cuántos recursos consumen?

ej. eficiencia de algoritmos

Gestión de Recursos

- Concurrency
- Multiple Copies
- More Resources Availability

- Concurrencia: reducen block time. Ej. threads (eventos, subprocesos)
- Múltiples copias (Ej. cache, múltiples clientes, etc.
- Más recursos: CPU más rápida, más memoria, mejores redes, etc.

Cuidar costo!

Arbitraje de Recursos

- Scheduling Policy

Siempre existe arbitraje.

Entender opciones, seleccionar mejor estrategia.

- FIFO – todos tienen igual derecho a los recursos
- FIXED-PRIORITY – se asignan prioridades (valora importancia, podría causar altas latencias)
- DYNAMIC-PRIORITY – ej. round robin, o deadline más urgente.

Atributos de Calidad #1

Hoja de Ruta

1. Introducción a los QA
2. Availability (disponibilidad)
3. Performance
   a. Network Performance
   b. User-Perceived Performance
   c. Efficiency
4. Scalability (escalabilidad)
5. Algunas Tácticas
   a. Availability
   b. Performance

Próximas clases...

Consultas?

Feedback

https://goo.gl/forms/NvrORS12kuuBitpE3

Christian Calónico
calonico@gmail.com

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Arquitectura de Software 75.73

FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

(Last updated: 2021.2)

Atributos de Calidad #2

Christian Calónico
75.73 Arquitectura de Software

FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Atributos de Calidad #2

Hoja de Ruta

1. Visibility (visibilidad)
2. Testability (testeabilidad)
3. Portability (portabilidad)
4. Interoperability (interoperabilidad)
5. Usability (usabilidad)
6. Manageability (manejabilidad)
7. Reliability (fiabilidad)
8. Security (seguridad)
9. Simplicity (simplicidad)
10. Modifiability (modificabilidad)

Visibility

- Ej. Network Firewalls para mejorar la seguridad
  - Filtrado
- Ej. Cache compartido de interacciones
  - Performance
- Ej. Métricas, detección de fallas
  - Monitoreo

Habilidad de un componente de monitorear o mediar la interacción entre otros dos componentes

Atributos de Calidad #2

Hoja de Ruta

1. Visibility (visibilidad)
2. Testability (testeabilidad)
3. Portability (portabilidad)
4. Interoperability (interoperabilidad)
5. Usability (usabilidad)
6. Manageability (manejabilidad)
7. Reliability (fiabilidad)
8. Security (seguridad)
9. Simplicity (simplicidad)
10. Modifiability (modificabilidad)

Testability

Relacionado usualmente al esfuerzo necesario para llevar adelante las tareas de testing

Grado en que un componente/sistema de software permite ser probado para exhibir sus fallas

- Componentes aislados, desacoplados
  - Mocks
- Suite de test completa
- Automatización
  - Record/playback
- TDD (testeable por definición)
- Built-in monitors

Atributos de Calidad #2

Hoja de Ruta

1. Visibility (visibilidad)
2. Testability (testeabilidad)
3. Portability (portabilidad)
4. Interoperability (interoperabilidad)
5. Usability (usabilidad)
6. Manageability (manejabilidad)
7. Reliability (fiabilidad)
8. Security (seguridad)
9. Simplicity (simplicidad)
10. Modifiability (modificabilidad)

Portability

- Arquitecturas de hardware
- Sistemas operativos
- Plataformas (ej. Cloud)

Capacidad de un software de ser utilizado en distintos ambientes (*environments*)

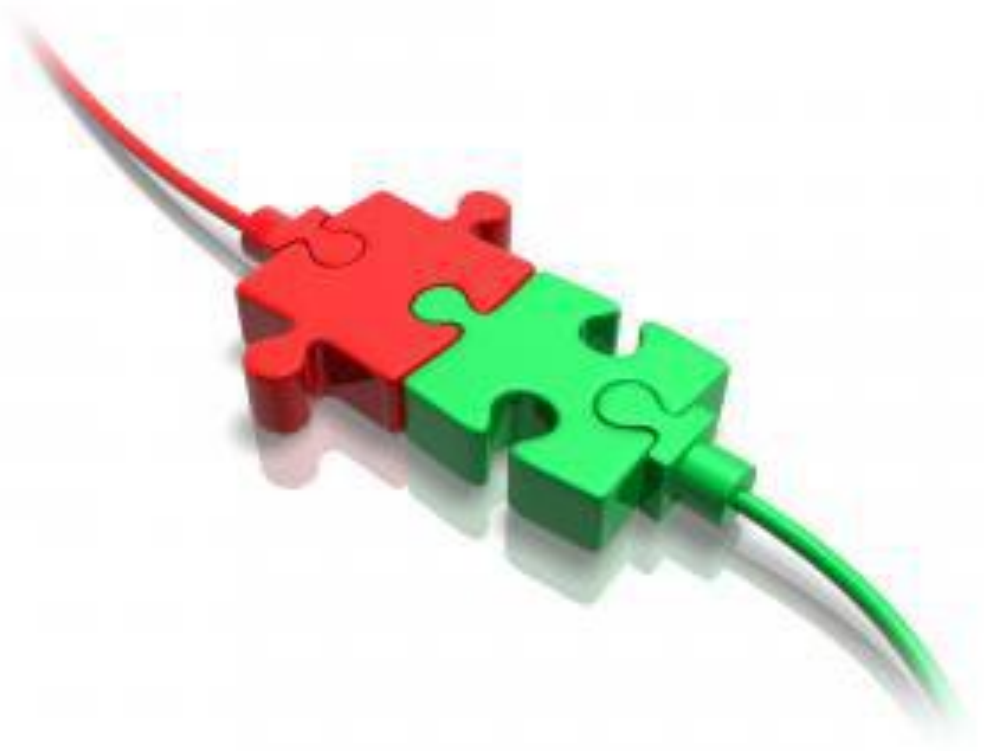
- Separar lógica de las interfaces
- Software Cross-Platform
- Portabilidad a nivel código fuente
- Virtualización

Atributos de Calidad #2

Hoja de Ruta

1. Visibility (visibilidad)
2. Testability (testeabilidad)
3. Portability (portabilidad)
4. Interoperability (interoperabilidad)
5. Usability (usabilidad)
6. Manageability (manejabilidad)
7. Reliability (fiabilidad)
8. Security (seguridad)
9. Simplicity (simplicidad)
10. Modifiability (modificabilidad)

Interoperability



Habilidad de un sistema de operar satisfactoriamente en conjunto con otros sistemas, comunicándose e intercambiando información

- Sistemas Distribuidos
- Sistemas Legacy
- Sistemas de Terceras Partes

Atributos de Calidad #2

Hoja de Ruta

1. Visibility (visibilidad)
2. Testability (testeabilidad)
3. Portability (portabilidad)
4. Interoperability (interoperabilidad)
5. Usability (usabilidad)
6. Manageability (manejabilidad)
7. Reliability (fiabilidad)
8. Security (seguridad)
9. Simplicity (simplicidad)
10. Modifiability (modificabilidad)

Usability

Define que tan bien el sistema cumple con los requerimientos del usuario, siendo *intuitivo*, *localizable/personalizable*, disponible para personas con *limitaciones físicas*, y ofreciendo una *experiencia de uso* satisfactoria

- MVC
- UX/GUI
- Encuestas/Focus Groups
- A/B Testing

Atributos de Calidad #2

Hoja de Ruta

1. Visibility (visibilidad)
2. Testability (testeabilidad)
3. Portability (portabilidad)
4. Interoperability (interoperabilidad)
5. Usability (usabilidad)
6. Manageability (manejabilidad)
7. Reliability (fiabilidad)
8. Security (seguridad)
9. Simplicity (simplicidad)
10. Modifiability (modificabilidad)

Manageability

Define la facilidad de gestión del sistema por parte de los administradores del mismo

- Ejemplos:
  - Administración general
  - Monitoreo del sistema
  - Debugging
  - Performance tunning

Atributos de Calidad #2

Hoja de Ruta

1. Visibility (visibilidad)
2. Testability (testeabilidad)
3. Portability (portabilidad)
4. Interoperability (interoperabilidad)
5. Usability (usabilidad)
6. Manageability (manejabilidad)
7. Reliability (fiabilidad)
8. Security (seguridad)
9. Simplicity (simplicidad)
10. Modifiability (modificabilidad)

Reliability

Grado en que la arquitectura es susceptible a fallas a nivel de sistema cuando se producen fallas parciales en componentes, conectores, o datos

- Evitar puntos únicos de fallo
- Redundancia
- Monitoreo
- Acciones correctivas

Atributos de Calidad #2

Hoja de Ruta

1. Visibility (visibilidad)
2. Testability (testeabilidad)
3. Portability (portabilidad)
4. Interoperability (interoperabilidad)
5. Usability (usabilidad)
6. Manageability (manejabilidad)
7. Reliability (fiabilidad)
8. Security (seguridad)
9. Simplicity (simplicidad)
10. Modifiability (modificabilidad)

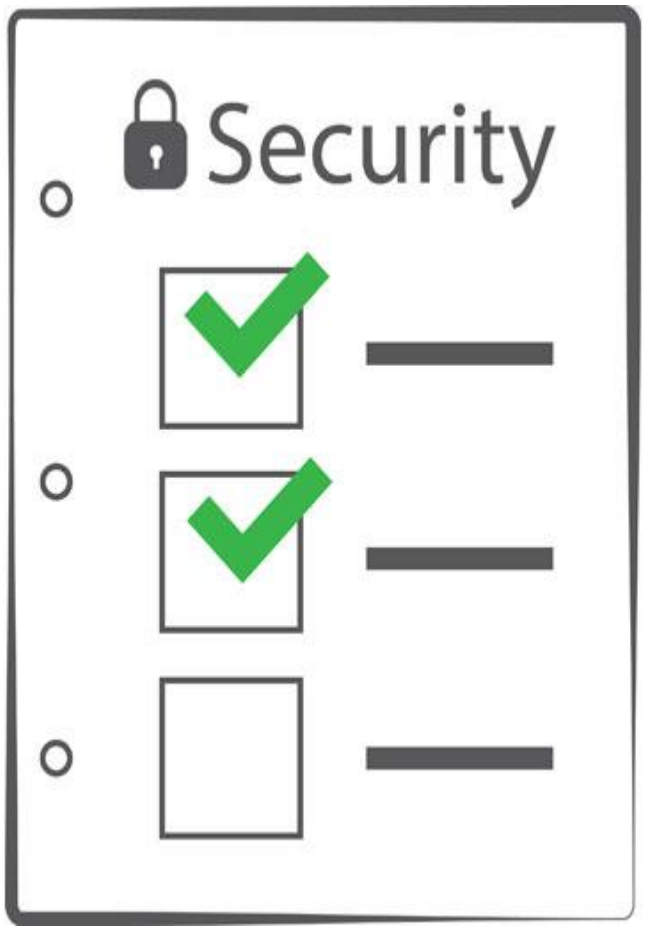
Security

Capacidad del sistema de prevenir acciones maliciosas o accidentales fuera del uso esperado

Security

- **Confidencialidad** – servicios y datos protegidos contra accesos no autorizados
  - Ej. Encryption, SSL, etc.
- **Integridad** – servicios y datos no manipulables sin autorización
- **Disponibilidad** – sistema disponible para su uso
  - Ej. DoS, Exploits, Intrussion Detection, etc.

- **Autenticación** – confirmar que las partes sean quienes dicen ser
- **Autorización** – otorgar a las partes los permisos correspondientes
- **No repudio** – garantizar que el emisor/receptor no desconozcan una transacción realizada



Atributos de Calidad #2

Hoja de Ruta

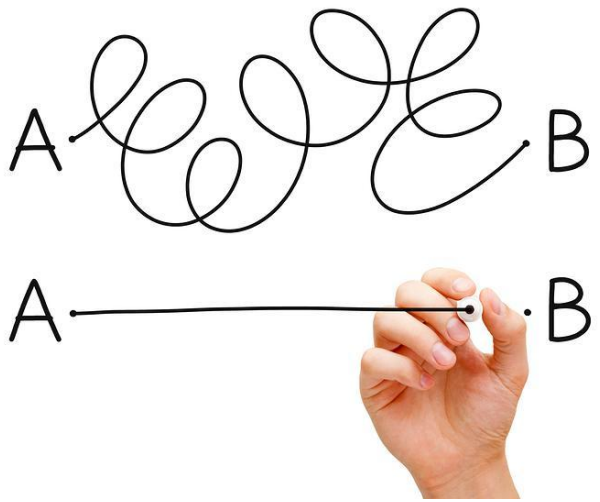
1. Visibility (visibilidad)
2. Testability (testeabilidad)
3. Portability (portabilidad)
4. Interoperability (interoperabilidad)
5. Usability (usabilidad)
6. Manageability (manejabilidad)
7. Reliability (fiabilidad)
8. Security (seguridad)
9. Simplicity (simplicidad)
10. Modifiability (modificabilidad)

Simplicity

- Componentes menos complejos, más sencillos de entender, implementar y testear

- Principios de diseño
  - Separation of Concerns (SoC)
  - Principio de generalidad
  - KISS (keep it simple, stupid!)
  - YAGNI (you aren't gonna need it) (XP)
  - No over-engineering
    - Necesitas este nivel de robustes/performance?
  - Last Responsible Moment (Lean)
    - Demorar decisiones lo máximo posible, hasta tener facts que las justifiquen (no asumir o predecir)

Facilidad con la que la arquitectura de un sistema puede ser entendida, razonada y explicada



Atributos de Calidad #2

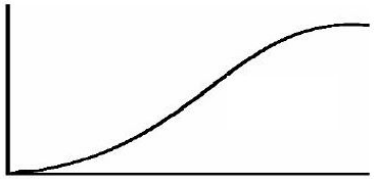
Hoja de Ruta

1. Visibility (visibilidad)
2. Testability (testeabilidad)
3. Portability (portabilidad)
4. Interoperability (interoperabilidad)
5. Usability (usabilidad)
6. Manageability (manejabilidad)
7. Reliability (fiabilidad)
8. Security (seguridad)
9. Simplicity (simplicidad)
10. Modifiability (modificabilidad)

Modifiability

Sucesos a lo largo del tiempo

- Cambia funcionalidad
- Corrección de bugs
- Mayor seguridad
- Mayor performance
- Mejor UX
- Nuevas tecnologías
- Integración de sistemas
- Etc.



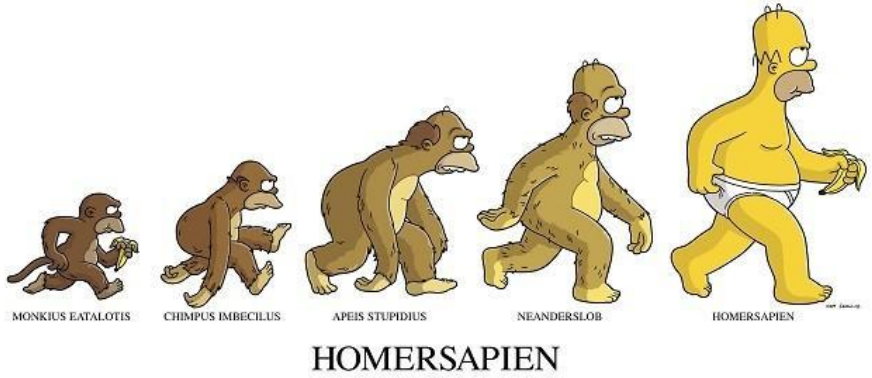
Costos de mantenimiento

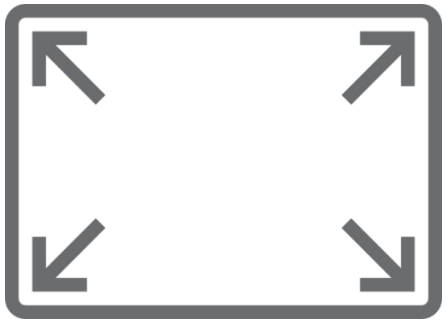
Costo de realizar cambios (tiempo y dinero) a la arquitectura del sistema

Modifiability

Evolvability (evolucionabilidad)
Habilidad para poder modificar un componente sin impactar negativamente en otros

Suele depender de lo bien que fue aplicada la abstracción impuesta por la arquitectura





Extensibility (extensibilidad)
Posibilidad de agregar más funcionalidad al sistema de forma simple

Reducir el acomplamiento suele ayudar (ej. sistema basado en eventos)

Modifiability

Customizability (personalización)
Un componente es personalizable si puede ser extendido por un cliente sin impactar en otros clientes del mismo

Configurability (configurabilidad)
Facilidad de realizar cambios (modificación o configuración) de los componentes post-deployment, de forma que puedan realizar otro tipo de servicios

Reusability (Reusabilidad) – Habilidad de una arquitectura de utilizar sus componentes en otras aplicaciones, sin modificación (bajo acoplamiento, generalidad en interfaces)

Atributos de Calidad #1

Hoja de Ruta

1. Introducción a los QA
2. Availability (disponibilidad)
3. Performance
   a. Network Performance
   b. User-Perceived Performance
   c. Efficiency
4. Scalability (escalabilidad)
5. Algunas Tácticas
   a. Availability
   b. Performance

Atributos de Calidad #2

Hoja de Ruta

1. Visibility (visibilidad)
2. Testability (testeabilidad)
3. Portability (portabilidad)
4. Interoperability (interoperabilidad)
5. Usability (usabilidad)
6. Manageability (manejabilidad)
7. Reliability (fiabilidad)
8. Security (seguridad)
9. Simplicity (simplicidad)
10. Modifiability (modificabilidad)

Consultas?

Feedback

https://goo.gl/forms/NvrORS12kuuBitpE3

Christian Calónico
calonico@gmail.com

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Arquitectura de Software 75.73

FACULTAD
DE INGENIERIA
Universidad de Buenos Aires