


Arquitectura de Sistemas de Software

Departamento de Ciencia de la Computación
Escuela de Ingeniería – PUC
Hans Findel {hifindel@uc.cl}



Requisitos No Funcionales

Diseño de la Arquitectura de Software

Tradicionalmente: Análisis no debe mancharse con consideraciones de diseño

Plan, costo, riesgo?

1. Identificar **requisitos no funcionales** (atributos de calidad) y restricciones
2. Diseñar y/o elegir un framework / metáfora / estilo
3. Validar
4. Documentar decisiones de diseño

NFP: Non-Functional Properties



- Las propiedades no funcionales (atributos de calidad) son el resultado de las decisiones de arquitectura
- Los requisitos no funcionales, son los “valores” que se esperan de dichas propiedades

No es un capricho tecnológico: **Requisitos (No Funcionales)** y Restricciones.

Arquitectura de Software

Atributo de Calidad	Requisito Arquitectónico
Desempeño	La aplicación debe entregar tiempos de respuesta menores a 4 segundos para el 90% de las solicitudes.
Seguridad	Toda la comunicación debe ser autenticada y encriptada usando certificados.
Gestión de recursos	El servidor debe correr como un proceso no prioritario en un PC con 512MB de memoria.
Usabilidad	La interfaz de usuario debe correr en un browser de Internet para los usuarios remotos.
Disponibilidad	El sistema debe correr 24x7x365, con una disponibilidad promedio de 0.99.
Confiabilidad	No se acepta pérdida de mensajes, el resultado de la entrega de mensajes debe conocerse en 30secs.
Escalabilidad	La aplicación debe manejar un carga peak de 500 usuarios concurrentes durante el periodo de inscripción.
Flexibilidad	La arquitectura debe permitir una migración en fases del lenguaje 4GL a .NET

Arquitectura de Software

- Restricciones: Técnicas o tecnológicas, de negocio, de calidad

Restricción	Requisito Arquitectónico
Negocio	La aplicación debe correr como un plugin para MS BizTalk (queremos venderse a MS)
	Queremos aliarnos (y obtener fondos) de Xcorp, necesitamos usar su tecnología
Desarrollo	El sistema debe estar escrito en Java (la mantención y testing lo hará nuestro equipo)
Tiempo	La primera versión debe entregarse en 6 meses

Atributos de calidad

□ Performance – Desempeño

Carga de trabajo que una aplicación debe ejecutar en una unidad de tiempo, y/o deadlines que debe cumplir para una correcta operación.

Carga de trabajo: Transacciones, mensajes, lectura a la DB, actualización de una DB distribuída en 2 fases.

- **Throughput – Salida:** Carga de trabajo en una unidad de tiempo (transacciones por segundo - tps). **Peak** y promedio
- **Tiempo de respuesta:** Latencia que una aplicación exhibe al procesar una carga de trabajo. **Garantizado** (límite) y promedio
- **Deadline – Plazo:** Ventana de tiempo para la compleción de una carga de trabajo

Atributos de calidad

□ Escalabilidad

¿Qué también trabaja una solución cuando el tamaño del problema **aumenta**?

- **Request – Cantidad de Solicitudes:** Multi-thread, balance de carga (repartir las solicitudes entre varios servidores/CPU's)
- **Cantidad de Conexiones concurrentes**
- **Tamaño de datos:** Cantidad de datos que puede procesar la aplicación en cada transacción (incremento en el tamaño de los paquetes de datos)
- **Deployment – Implantación:** Cantidad de esfuerzo requerido en distribuir, actualizar, configurar una aplicación (incremento en cantidad de usuarios)

Atributos de calidad

□ Integración

Esfuerzo requerido para integrar una aplicación en un contexto mayor

- A nivel de datos (DB accesible a otras aplicaciones)
 - Flexible, simple y peligrosa
- API (Application Programming Interface)
 - Conjunto de funciones (parametrizadas) que controlan el acceso a los datos. Garantizan integridad de datos y reglas de negocio
 - Compleja, costosa y segura

Atributos de calidad

■ Flexibilidad

Esfuerzo requerido para cambiar una aplicación (**modificar** funcionalidad o arquitectura)

- Requiere identificar escenarios de cambio probables
- Cohesión dentro los componentes y subsistemas débilmente acoplados facilitan el cambio

■ Seguridad

- Autenticación, autorización, encriptación, integridad, no repudiación

■ Disponibilidad

Porcentaje de tiempo en que una aplicación puede ser usada

- Recuperación ante fallas (automática o manual – tiempo de respuesta o reacción) ... Sábados? Vacaciones?, 3am?

Factores y métricas de calidad

<i>Métrica</i>	<i>Definición</i>	<i>Factor</i>
Accuracy	Grado en el cual un programa satisface su especificación y cumple los objetivos del usuario . El producto hace lo esperado?	Trace-ability, completeness, consistency
Reliability	Grado en el un programa cumpla sus funciones con precisión. Cómo satisface los requisitos dado un periodo de tiempo?	Consistency, accuracy, error tolerance, simplicity
Efficiency	Cantidad de recursos computacionales y código requerido para que el programa cumpla sus funciones. Utiliza los recursos de hardware bien?	Execution efficiency, storage efficiency
Integrity	Grado en el cual el acceso a datos o software por personas no autorizadas puede ser controlado. Es seguro?	Access control, access unit

Factores y métricas de calidad

<i>Métrica</i>	<i>Definición</i>	<i>Factor</i>
<i>Usability</i>	<i>Esfuerzo requerido para aprender, operar, ingresar datos, e interpretar la salida de un programa. Puedo manejarlo fácilmente?</i>	<i>User friendly, operability, training</i>
<i>Maintainability</i>	<i>Esfuerzo requerido en ubicar y arreglar un error. Puedo corregir una falla fácilmente?.</i>	<i>Readability and code structure, modularity, consistency, simplicity, documentation</i>
<i>Testability</i>	<i>Puedo probar el producto sin añadir costo adicional después de hacer cambios?</i>	<i>Simplicity and code structure, modularity, instrumentation, documentation</i>
<i>Interoperability</i>	<i>Esfuerzo requerido para unir dos sistemas. Puedo crear una interfaz con otros sistemas?</i>	<i>Modularity, compatibility</i>

Factores y métricas de calidad

<i>Métrica</i>	<i>Definición</i>	<i>Factor</i>
<i>Flexibility</i>	<i>Esfuerzo requerido para modificar un programa. Puedo ejecutar un cambio fácilmente?</i>	<i>Modularity, simplicity of design and code, generality, quality of documentation</i>
<i>Portability</i>	<i>Esfuerzo para transferir un programa de una configuración de hardware y/o ambiente a otro. Puedo usar el programa en otro hardware?</i>	<i>Modularity, quality of documentation, software system independence</i>
<i>Reusability</i>	<i>Grado en el cual un programa puede ser usado en otras aplicaciones. Puedo usar partes de la aplicación en otros programas?</i>	<i>Simplicity and code structure, modularity, quality and availability of documentation, software system independence, machine independence</i>

Atributos de Calidad - Interesados

Usuario Final

- Performance
- Availability
- Usability
- Security

Desarrollador

- Maintainability
- Portability
- Reusability
- Testability

Negocios - Stakeholders

- Time To Market
- Cost and Benefits
- Projected life time
- Targeted Market
- Integration with Legacy System
- Roll back Schedule

Atributos de Calidad - Producto

Revisión

- Maintainability
- Flexibility
- Testability

Transición

- Portability
- Reusability
- Interoperability

Operación

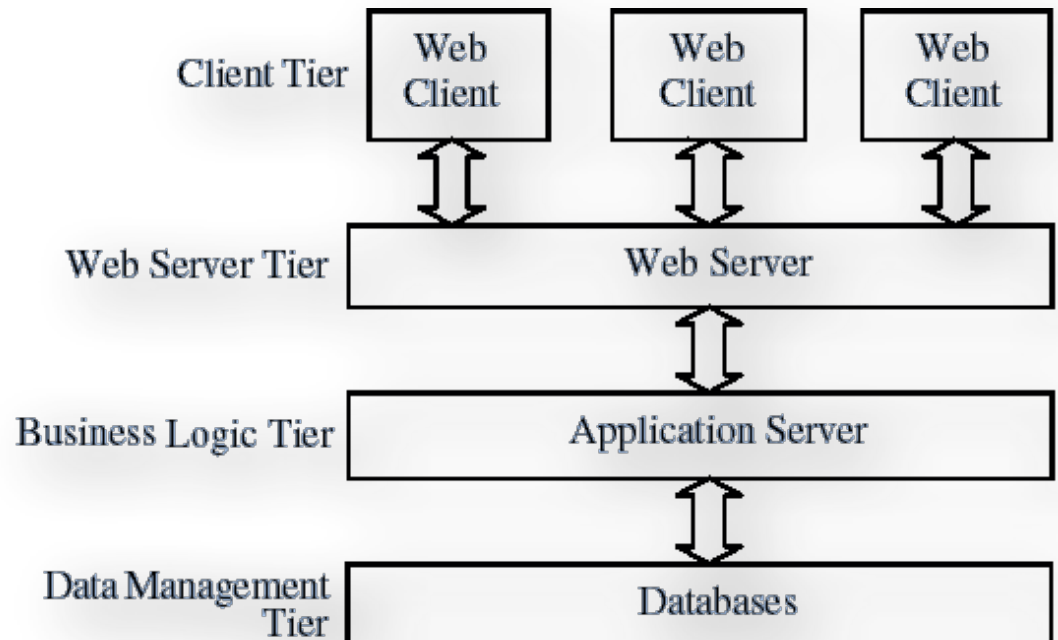
- Correctness
- Reliability
- Usability
- Integrity
- Efficiency



Ejemplo de Arquitecturas

N-Capas

- **Modularidad**: Separación de temas
- **Comunicación sincrónica entre** capas (request – wait – answer)
- **Implantación flexible**: capas independientes, distribuidas, heterogéneas, etc..
- Gran número de clientes, solicitudes concurrentes, requiere poco tiempo de proceso



Cliente-Servidor n-capas

Atributo de Calidad	Ventajas
Disponibilidad	Réplica de servidores en cada capa, si uno falla, los otros responden. La QoS disminuye hasta restaurar el servidor fallado.
Manejo de fallas	La mayoría de los servidores web y de aplicación implementan una recuperación transparente (el cliente se redirecciona a un servidor replicado para responder a la solicitud).
Flexibilidad	La modularidad facilita el cambio, debido la encapsulación (presentacion, negocios y datos). El impacto del cambio al interior de una capa debería ser mínimo en las otras.
Desempeño	Alto desempeño mediante replicación de software (threads concurrentes), velocidad de la conexión entre capas y la cantidad de datos a ser transferidos. Es recomendable minimizar las llamadas entre capas al procesar cada solicitud.
Escalabilidad	Como los servidores pueden replicarse , la arquitectura escala bien. En la práctica, la gestión de datos se convierte en el cuello de botella.