Resumen de ARQUITECTURA

El concepto de arquitectura software tiene sus orígenes en los trabajos de investigación de Edsger Dijkstra (1968) y David Parnas (1972).

Dijkstra propuso que se establezca una estructuración correcta de los sistemas de software antes de apresurarse a programar, escribiendo código a como de lugar.

Por otra parte, Parnas, hizo énfasis en descomponer un sistema en módulos

La arquitectura de un programa o sistema software es la estructura (o estructuras del sistema), la cual constituye elementos software, las propiedades visibles de estos elementos y sus relaciones

La arquitectura es influenciada por las personas interesadas en el producto software (stakeholders) ◦ El cliente, usuarios finales, desarrolladores, administradores de proyectos, personal de mantenimiento o gente de marketing son ejemplos de lo que es un stakeholder.

La arquitectura software es el resultado de varias fuerzas influyentes como son: *técnicas, de negocio y sociales*

¿Por qué la arquitectura de software es importante?

La arquitectura software es importante para:

◦ Facilitar la comunicación entre las distintas personas

interesadas en el sistema (stakeholders).

◦ Tomar decisiones de diseño tempranas.

◦ Realizar abstracciones de un sistema que sean transferibles a otros sistemas.

Desde la perspectiva del plan, un dise˜no se compone de elementos

esenciales como son:

Declaraci´on del problema de dise˜no y objetivos

Restricciones

Descripci´on del artefacto, producto, proceso o servicio

Justificaci´on

Plan de producci´on

Descripci´on de uso

Los errores en el dise˜no de software pueden clasificarse en cuatro tipos:

Incorrecto (Incorrectness)

Inconsistente (Inconsistency)

Ambiguo (Ambiguity)

Inferior (Inferiority)

¿Qué es funcionalidad?

Es la habilidad del sistema de realizar el trabajo para el cual fue previsto.

Realizar alguna tarea en el sistema requiere que varios elementos trabajen de una manera coordinada para completarla.

¿Por qué las cualidades están por encima de la funcionalidad?

Desarrollo de una aplicación que es funcional en un entorno *nix pero ahora debe ser ejecutada en entorno Windows.

Atributos de calidad

Un atributo de calidad es un requisito no funcional de un

producto de software

Ejemplos de atributos de calidad:

- ◦ Disponibilidad (availability): se relaciona con los fallos en el sistema y sus consecuencias asociadas.
- ◦ Facilidad de modificación (modifiability): La facilidad de modificación se relaciona con el costo de los cambios, donde nos interesan dos cuestiones:

¿Qué puedo cambiar (artefacto)?

¿Cuándo se realiza el cambio y quién lo hace (entorno)?

- ◦ Desempeño (performance): El desempeño se relaciona con el tiempo. Eventos tales como interrupciones, mensajes o peticiones ocurren y el sistema debe responder a ellos. Básicamente el desempeño se relaciona con qué tanto le toma al sistema responder cuando ocurre un evento.
- ◦ Seguridad (security): es una medida del sistema para resistir a un uso no autorizado mientras éste proporciona algún servicio a sus usuarios legítimos.
- ◦ Facilidad de uso (usability): La facilidad de uso se relaciona con qué tan fácil es para el usuario completar alguna tarea.

Resumen

No basta con satisfacer los requisitos funcionales de un sistema Debemos tomar en cuenta los requisitos no funcionales del sistema (atributos de calidad)

Cada atributo de calidad satisface un propósito en particular

Ejemplos de atributos que podemos considerar en una

arquitectura:

- ◦ facilidad de modificación
- ◦ desempeño
- ◦ seguridad
- ◦ facilidad de uso

Estrategias para satisfacer atributos de calidad

disponibilidad

Las estrategias se centran en evitar que algún defecto (falta) seconvierta en un fallo en el sistema, o al menos degradar los efectos de esa falta. Estas estrategias involucran algún tipo de mecanismo de redundancia, monitoreo o algún mecanismo de recuperación cuando se detecta un fallo.

Estas estrategias se dividen en tres categorías:

- Detección:

Ping o eco: un elemento (componente) envía un mensaje (ping).

Heartbeat. Un elemento (componente) envía periódicamente un mensaje

Exceptions. Se implementa un manejador de excepciones en caso de encontrar alguna falta.

-**Recuperación**

Active redundancy. Varios elementos se disponen de manera redundante y responden de manera paralela a algún evento.

-Passive redundancy. un elemento (principal) responde a eventos e informa a otros elementos.

Checkpoint/rollback. Se refieren a contemplar puntos de restauración.

-Prevención de faltas

Estrategias para satisfacer la facilidad de modificación

tienen como objetivo controlar el tiempo y costo para implementar, probar y desplegar cambios en un sistema. Identificamos tres estrategias: -

Localizar modificaciones

-Prevenir efecto dominó

-Facilitar despliegue y cambios a quienes no son desarrolladores

Prevenir efecto dominó. Una modificación derivada del efecto dominó se refiere a la necesidad de hacer cambios a módulos que no son directamente afectados por la modificación.

Estrategias para satisfacer el desempeño

El objetivo de estas estrategias es generar una respuesta a algún evento que ingresa al sistema dentro de un lapso de tiempo específico.

Podemos entonces emplear algunas de las siguientes estrategias:

-Demanda de recursos

-Administración de recursos

-Arbitraje de recursos

Estrategias para satisfacer la seguridad

Estas estrategias pueden dividirse en aquellas enfocadas en resistir ataques, aquellas enfocadas con la detección de ataques y aquellas interesadas en la recuperación tras algún ataque. Tenemos las siguientes estrategias:

-Resistencia a ataques

-Detección de ataques

-Recuperación tras ataques

Estrategias para satisfacer la facilidad de uso

, la facilidad de uso se relaciona con qué tan fácil es para el usuario completar una tarea así como el tipo de ayuda que el sistema proporciona al usuario. Dos estrategias para satisfacer la facilidad de uso las podemos identificar como

estrategias en tiempo de ejecución y en tiempo de diseño.

Revisa la información del sitio de MS antes descrita, particular revisa la Tabla 2.

Table 2. System-quality trade-off points

	Availability	Efficiency	Flexibility	Integrity	Interoperability	Maintainabi
Availability						
Efficiency			-		-	-
Flexibility		-		-		+
Integrity		-			-	
Interoperability		-	+	-		
Maintainability	+	-	+			
Portability		-	+		+	-
Reliability	+	-	+			+

Estilos Arquitectónicos

En un estilo se define un vocabulario de componentes y de tipos de conectores. En un estilo también se define un conjunto de restricciones sobre cómo pueden combinarse los componentes y conectores.

Algunos ejemplos de estilos en el área de software son:

◦ Cliente-Servidor

- ◦ Arquitectura basada en componentes
- ◦ Diseño dirigido por dominio
- ◦ Capas
- ◦ Bus de comunicaciones
- ◦ N-Capas o tres capas
- ◦ Orientado a objetos
- ◦ Arquitectura orientada a servicios

Estilo arquitectónico REST (arquitecturas orientadas a micro-servicios)

REST es un tipo de arquitectura software en el que se describen interfaces entre diferentes sistemas que utilizan HTTP para comunicarse.