Arquitectura de Software

Sesión 1: Presentación y descripción del Curso

Perla Inés Velasco Elizondo

Email: [pvelasco@uaz.edu.mx](mailto:pvelasco@uaz.edu.mx)

Proceso de desarrollo de la arquitectura de software

Anticipación más de 24 para justificar horas de entrega tardía

Sesión 2: Software Architecture

Divide y vencerás: Atacar un problema en partes, el modularizar un sistema complejo nos permite, asignar el desarrollo de cada parte a un equipo específico, de esta manera:

* Se reduce el tiempo de desarrollo
* Puedo conocer las propiedades individuales de cada parte
* El diseño detallado de cada una de las partes se pueda realizar después y de manera separada.

Arquitectura de software: Esta envuelta con un alto nivel de descripción de las estructuras e interacciones de un sistema.

Está relacionado con la toma de decisiones acerca del esqueleto de un sistema, tomando en cuenta aspectos técnicos, y atributos de calidad (Desplegabilidad , Disponibilidad, Escalabilidad, Interoperabilidad, Modificabilidad, Rendimiento, Seguridad, Testeabilidad , Usabilidad.)

**La arquitectura de un sistema de software es el conjunto de estructuras necesarias para razonar sobre un sistema las cuales comprenden elementos de software, relaciones entre ellos y propiedades entre ambos.**

Considera elemento de hardware (no donde corre) si no que usan sensores, actuadores, alarmas.

Estructuras y perspectivas

**Una vista es una manera de representar un elemento de una arquitectura. FISICA, ESTATICA Y DINAMICA.**

**Vista física o de localización:** Representa donde están instalados los elementos del sistema.

**Vista estática, modulo, lógica:** Muestra los elementos de esta estructura y relaciones. **COMO EL SISTEMA ESTA PARTICIONADO.**

Las relaciones son las líneas en el diagrama, (las conexiones entre los elementos de hardware y software).

**Vista dinámica:** Muestra el comportamiento de los elementos de la estructura. Por ejemplo, un ***diagrama secuencia.***

**UN SISTEMA PUEDE SER REPRESENTADO POR UNA SOLA VISTA ESTATICA, PERO PUEDE TENER VARIAS VISTAS DINAMICAS**

Objetivos de negocios

La arquitectura debe de estar ligada a los objetivos de negocio, los objetivos de negocio son las metas del sistema.

Ejemplo de reglas de negocios:

* Beneficios financieros
* Reducir tiempos
* Posicionamiento estratégico/competitivo

Una buena regla de negocio debe de poder ser medible.

Sesión 3: Architecture Driven Development

Ciclo de desarrollo de arquitectura de software:

Requerimiento de software: Es algo que orienta o influencia una decisión de diseño.

Requerimientos de usuario, atributos de calidad, restricciones. Son los más importantes para arquitectura de software.

Requerimientos de negocios, interfaces externas.

Vimos Ciclo de desarrollo de arquitectura y entramos a requerimientos.

**Casos de uso (no son diagramas):** Son descripciones en lenguaje natural en un idioma, siempre hay una interacción explícita entre el sistema y el usuario. Dos columnas izquierda de actor y derecha del sistema.

* Describe como un actor usa un sistema
* Como un actor interactúa con el sistema

Actores siempre son sustantivos.

**Historias de usuario:** Las tres WWW (Who, What, Why),

As a do I want to order food online so I dont neet to rely on people anymore.

Las historias de usuario son cortas y generales.

**Story Map:** Es una herramienta eficiente para capturar requerimientos a través de historias de usuario.

Sesión 4

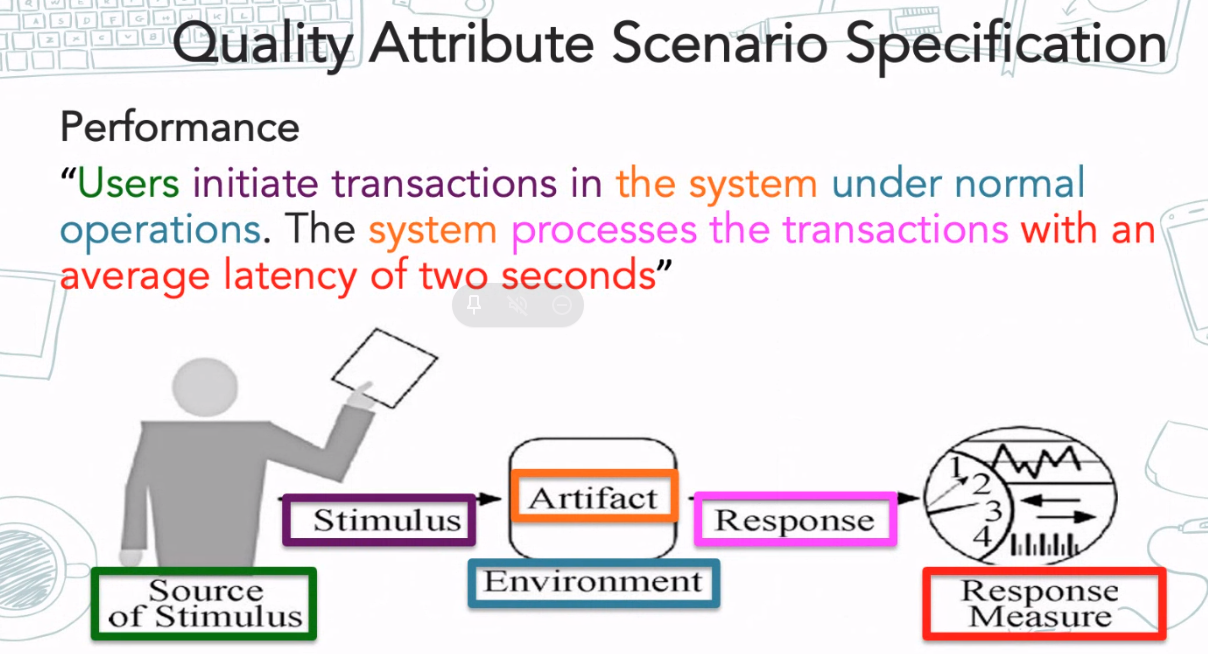
Requerimientos: Lo que puede hacer el usuario a través del sistema.

Sesión 4: Atributos de calidad

Atributos de calidad externa: Por ejemplo, el desempeño características que se pueden observar mientras se ejecuta el sistema.

Atributos de calidad interna: Por ejemplo, la modificabilidad

Estímulo, artefacto, respuesta y medida de respuesta



Sesión 4: Deuda Técnica

Deuda técnica: Es la deuda que se acumula cuando se hacen malas decisiones de diseño. Representa todo el trabajo no terminado debido a atajos que el equipo debe tomar para entregar más rápido el sistema.

La deuda es invisible para el usuario, solo es visible para los desarrolladores.

La deuda técnica se manifiesta en dificultad y aumenta su costo en dos principales caminos:

* Evolución:
* Mantenibilidad:

Prueba para saber si pensamos sistemáticamente, todos los del tercer aro debo encontrar en el tablero de history map deben de hablar en el release 1 y mvp.

LA DEUDA TECNICA MAS CARA ES LA DE ARUITECTURA.

Sesión 8: Principios de Diseño

Diseño es aquel que implementa los requerimientos, respeta las limitaciones, basado en los principios elementales.

Principio de diseño se entiende como una regla, algo universalmente aceptado.

Principios de diseño:

1. Modularidad: Componentes pequeños, fáciles de entender.
2. Cohesión: Se refiere a poner cosas relacionadas juntas. ALTA COHESIÓN ES BUENA.

Nivel más alto es FUNCIONAL.

Nivel más bajo COINCIDENCIAL.

1. Acoplamiento: Algo que depende de muchas otras cosas.

Nivel más alto CONTENIDO

Nivel más bajo DATOS

1. Ocultación de información: Principal función es de mantener un sistema más fácilmente
2. Simplicidad / KISS

Sesión 9: Conceptos de Diseño

Los que están en rojo siempre voy a decidir sobre esas categorías.

1.- Localización de responsabilidades: Identificar las responsabilidades importantes.

Ejemplo: Se decide o no guardar los fallos y en donde, se decide notificar los fallos a una persona u otro sistema, decidir si enmascarar el fallo, arreglar el fallo.

2.- Modelo de

3.- Modelo de datos: Determina como los datos principales son creados, inicializados, accesados, manipulados, traducidos y destruidos.

Se determina donde el dato será almacenado en una base de datos u otra.

4.- Administración de recursos: Identificar cuales recursos (físico o dato ejemplo boleto ticketmaster).

6.- Binding time decisions: En que tiempo hago yo la asociación atributo valor.

* Tiempo de compilación
* Tiempo de compilación
* Tiempo de carga
* Tiempo de ejecución

7.- Elección de la tecnología: d

Localización de responsabilidades: Notificar al administrador del sistema cuando no se pueda mostrar el porcentaje de llenado de un contenedor de basura.

Modelo de Coordinación:

Modelo de datos: Usar base de datos SQL para almacenar los datos de usuarios en una base de datos relacional.

DECISION DE DISEÑO: Resoluciones que se toman de manera informada por el arquitecto de software.

Patrón de capas:

Tácticas de disponibilidad: El arbolito del libro.

Sesión 10: Patrones de arquitectura

Táctica

Sesión N: Diseño o algo así

Los requerimientos arquitecturales son (técnicamente retadores).

Método add, decisiones justificadas siempre guiadas por los drivers de arquitectura.

Principios de diseño:

