Análisis y Diseño de Sistemas 2

Arquitectura de software

Material de apoyo del curso Análisis y Diseño de Sistemas 2 de la USAC

Ing. Ricardo Morales

Primer semestre 2016

Análisis y Diseño de Sistemas 2

Arquitectura de software

Tabla de contenido

Objetivos 2
Elementos arquitectura de software 2
Particionamiento, conocimiento y abstracciones 2
Tres ejemplos de arquitectura de software 3
Reflexiones 3
Arquitectos "arquitectando" arquitecturas 4
Arquitectura de software dirigida por riesgo 4
Arquitectura de software 5
Definición (SEI, 2010) 5
Definición 5
¿Qué es arquitectura de software? 5
Arquitectura y diseño detallado 6
¿Qué cuenta como arquitectura? 6
Intencionalidad 6
¿Por qué es importante la arquitectura? 6
La arquitectura actúa como un esqueleto del sistema 6
La arquitectura influencia los atributos de calidad 7
La arquitectura (generalmente) es ortogonal a la funcionalidad 7
La arquitectura restringe a los sistemas 7
¿Cuándo es importante la arquitectura? – casos con alto riesgo arquitectura 8
Espacio de solución pequeño 8
Alto riesgo de falla 8
Atributos de calidad difíciles 8
Dominio nuevo 8
Líneas de producto 8
Arquitectura presuntas (supuestas) 8
Principios 9

OBJETIVOS

- Comprender y discutir el concepto de arquitectura de software
- Identificar porque es importante la arquitectura de software
- Comprender el enfoque de un modelo dirigido por riesgo para arquitectura de software

ELEMENTOS ARQUITECTURA DE SOFTWARE

Década tras década los sistemas de software han observado incrementos en su tamaño y complejidad.

En la batalla contra la complejidad y la escala, ha ayudado:

- Avances en ingeniería de software: lenguajes de alto nivel, programación estructurada, objetos, reutilización, librerías y frameworks
- Armas como IDES, lenguajes de programación, abstracciones mentales

Particionamiento, conocimiento y abstracciones

Un desarrollador *particiona* un problema, para que sus partes sean mas pequeñas y tratables, aplica su *conocimiento* de problemas similares y usa *abstracciones* que le ayudan a razonar.

Particionamiento

Es una estrategia efectiva para combatir la complejidad y escala cuando 2 condiciones son verdaderas:

- Las partes divididas deben ser suficientemente pequeñas para que ahora puedan ser resueltas por una persona
- Debe ser posible razonar como las partes se ensamblan en un todo

Conocimiento

Puede ser específico, como que componentes funcionan bien con otros, o general, como técnicas para optimizar tablas en bases de datos.

Viene en muchas formas, incluyendo libros, conferencias, patrones, código fuente, documentos de diseño o esquemas en un pizarrón.

Abstracción

Combate efectivamente la complejidad y escala porque reduce problemas, y es mas fácil razonar acerca de un problema pequeño.

La arquitectura de software provee estos 3 elementos

Tres ejemplos de arquitectura de software

Rackspace es una compañía que administra hosting de servidores de correo. Para dar soporte, se debe buscar los archivos de log que indican que sucedió durante el procesamiento de correos. Se hicieron 3 generaciones de sistemas para manejar esto.

Versión 1: archivos locales de log

Se escribió un script que usa ssh para conectarse a cada máquina y ejecutar un query con grep en los archivos de log.

Inicialmente funcionó bien, pero con el tiempo el número de búsquedas aumento y el tiempo de corrida de los scripts fue significativo.

Además se requería un ingeniero, en vez de un técnico de soporte, para ejecutar la búsqueda.

Versión 2: base de datos central

La segunda versión movió los datos de log de los servidores de correo hacia una base de datos relacional.

Fue posible hacer búsquedas y se manejaban logs de cientos de servidores de correo. El reto era tener los archivos en la base de datos tan rápida y eficientemente como fuera posible.

Con el tiempo, el servidor de bd fue llevado a sus límites con cargas altas de CPU y disco. Se prohibieron las búsquedas con wildcards debido a la carga en el servidor.

Versión 3: cluster de indexación

Esta versión solucionó los problemas de la anterior salvando los datos de log en un filesystem distribuido y paralelizando la indexación de los datos del log.

En vez de usar una máquina, usa 10 servidores y el filesystem distribuido Hadoop, que ejecuta jobs para indexar los datos (se ejecutan cada 10 minutos y toman 5 en ejecutarse).

El sistema es capaz de indexar 14Gb por día y se habían ejecutado 150 mil jobs de indexación, con 6Tb de datos.

Reflexiones

La primera cosa a notar de estos 3 sistemas es que tienen la misma funcionalidad, aunque tienen diferentes arquitecturas. *Su arquitectura fue una elección diferente de su funcionalidad.*

Atributos de calidad

Los 3 sistemas difieren en su modificabilidad, escalabilidad y latencia.

En las primeras 2 versiones las consultas se pueden crear en segundos, mientras que en la 3ra toma mas tiempo (modificabilidad).

Promover una cualidad puede inhibir otra. El 3er sistema es mas escalable, pero se reduce la habilidad de hacer querys y el tiempo de espera por los resultados es mayor.

Usualmente los atributos de calidad se afectan, maximizar uno de ellos significa tener menos de otros.

Modelo conceptual

Al hacer un análisis desde el punto de vista de arquitectura se habrían distinguido entre grupos de código (módulos), grupos de corrida (componentes) y grupos de hardware.

Se puede identificar patrones y saber cual es mejor para alcanzar que atributo de calidad (cliente servidor tiene menos latencia que map-reduce).

Abstracciones y restricciones

En el caso del 3er sistema, un job es una abstracción que obedece mas restricciones que un conjunto de código.

Arquitectos "arquitectando" arquitecturas

El rol de trabajo: arquitecto

Un título posible de un trabajo o rol es arquitecto de software.

Algunos se sientan en una esquina y hacen pronunciamientos desconectados de la realidad, mientras otros están íntimamente involucrados en la construcción del software

Todos los desarrolladores de software, no solo los arquitectos, deben entender su arquitectura de software.

El proceso: "arquitectando"

Para obtener el software funcionando, el equipo realiza actividades para construir el sistema. Algunos hacen un gran diseño previo y otros mientras se construye.

El proceso que un equipo sigue es separable del diseño que emerge.

El artefacto de ingeniería: la arquitectura

Si se ve un sistema de software terminado, se pueden distinguir varios diseños; por ejemplo, nodos punto a punto colaborando en una red de voz sobre IP, múltiples capas en sistemas TI, etc.

ARQUITECTURA DE SOFTWARE DIRIGIDA POR RIESGO

Diferentes desarrolladores han tenido éxito con diferentes procesos. Algunos tuvieron éxito con procesos ágiles y otros con mucho diseño previo. Una forma de elegir cuanta arquitectura se debe hacer es con base en el riesgo

La arquitectura de software es relativamente nueva tecnología e incluye muchas técnicas para modelar y analizar sistemas. Sin embargo, cada una de estas técnicas toma tiempo que de otra forma sería usado construyendo el sistema.

Entonces, el esfuerzo debe ser conmensurado con el riesgo de fallar

Por ejemplo, un sistema puede tener requerimientos demandantes para la escalabilidad porque corre un web service popular. Es mejor asegurarse que el sistema manejará el número esperado de usuarios, antes de invertir mucha energía en su diseño.

Cada proyecto enfrente diferentes riesgos, de manera que no hay una manera correcta de hacer la arquitectura de software: debe evaluarse los riesgos de cada proyecto.

La idea de trabajar consistentemente para reducir riesgos de ingeniería hace eco del modelo de espiral de Barry Boehm para desarrollo de software.

ARQUITECTURA DE SOFTWARE

La arquitectura de software es acerca del diseño del sistema y el impacto que tiene en las cualidades del sistema, cualidades como rendimiento, seguridad y modificabilidad.

La elección de arquitectura de software es importante la arquitectura actúa como el esqueleto del sistema, influencia sus atributos de calidad y restringe el sistema.

En la mayoría de casos es ortogonal a la funcionalidad, de manera que se puede mezclar arquitectura y funcionalidad, aunque algunas combinaciones funcionan mejor que otras.

Definición (SEI, 2010)

La arquitectura de software de un sistema de computación es el conjunto de estructuras necesarias para razonar acerca del sistema, que incluye elementos de software, relaciones entre ellos y las propiedades de ambos.

Definición

La arquitectura de un sistema es el conjunto de conceptos o propiedades fundamentales del sistema en su ambiente, incorporadas en sus elementos, relaciones y los principios de su diseño y evolución.

Elementos y relaciones:

- Estructuras estáticas
- Estructuras dinámicas

Propiedades fundamentales de sistemas:

- Comportamiento visible desde el exterior
- Propiedades de calidad, una propiedad visible externamente, no funcional

¿Qué es arquitectura de software?

Arquitectura y diseño detallado

El diseño de un sistema de software consiste de las decisiones e intenciones que están en las cabezas de los desarrolladores. El diseño puede dividirse en arquitectura de software y diseño detallado.

Los temas estructurales incluyen, que son el nivel de diseño de arquitectura de software son:

- La organización del sistema como una composición de componentes
- Estructuras de control global
- Protocolos de comunicación, sincronización y acceso a datos
- Asignación de funcionalidad a elementos de diseño
- Distribución física, escalabilidad y rendimiento
- Selección entre alternativas de diseño

¿Qué cuenta como arquitectura?

Definir arquitectura como el conjunto de estructuras necesarias para razonar acerca de un sistema es útil en que enfoca atención en el propósito de la arquitectura (razonar).

Sería mas simple y mas claro decir que la arquitectura son las partes macroscópicas del diseño, tales como módulos y como se conectan, dejando que el diseño detallado cubra lo demás.

En algunos casos, esto puede ser no satisfactorio, ya que se hacen decisiones de bajo nivel que afectan la arquitectura.

El arquitecto incluye detalles en la arquitectura si contribuyen a la calidad total del edificio (software).

Intencionalidad

Para distinguir detalles de arquitectura de otros detalles, la intencionalidad es la clave: hay una cadena de intencionalidad desde unas pocas intenciones o decisiones de alto nivel del arquitecto, que llegan hasta algunos detalles de bajo nivel.

La mayoría de los detalles se dejan abiertos a cualquier implementación razonable, pero algunos están restringidos a través de las decisiones de alto nivel del diseñador.

¿Por qué es importante la arquitectura?

La arquitectura actúa como un esqueleto del sistema

Un esqueleto provee la estructura para un animal e influencia lo que puede hacer. Las aves son buenas volando y los canguros saltando por su esqueleto.

No se puede decir que un esqueleto es mejor que otro, a menos que dependa de su función.

El software es similar, una arquitectura de 3 capas permite a un sistema localizar los cambios y manejar cargas transaccionales. Una arquitectura de procesos cooperativos es mejor para un sistema operativo porque aísla fallas.

La metáfora del esqueleto es imperfecta porque hay partes invisibles que son importantes. Por ejemplo, la política de bloqueos, administración de memoria, etc.

La arquitectura influencia los atributos de calidad

Los desarrolladores deben poner atención a lo que el software hace, esto es, funcionalidad.

Un software de contabilidad que no contabiliza o un software de animación que no anima, no es útil.

Los desarrolladores deben poner atención a los requerimientos de atributos de calidad también, ya que un software de contabilidad que permita robar las cuentas o un software de animación lento, tampoco es útil.

Además de soportar la funcionalidad requerida, la arquitectura de un sistema habilita o inhibe cualidades como seguridad o rendimiento.

La arquitectura (generalmente) es ortogonal a la funcionalidad

Es importante reconocer que se pueden mezclar arquitectura y funcionalidad.

Esto es, se puede cambiar la arquitectura y mantener la funcionalidad o reutilizar la misma arquitectura en un sistema con diferente funcionalidad, aunque algunas combinaciones funcionan mejor que otras.

Aunque la arquitectura del sistema es una elección separada de su funcionalidad, una pobre elección de arquitectura puede hacer que sea difícil alcanzar los requerimientos de funcionalidad y atributos de calidad.

La arquitectura restringe a los sistemas

Algunas elecciones restringen el sistema con la intención de guiarlo a cierto destino. Estas restricciones actúan como guías y son esenciales en la construcción de un sistema, en la habilidad para ejecutar su trabajo y la habilidad para mantenerlo en el tiempo.

Los ingenieros usan restricciones para asegurar que los sistemas que diseñan hagan lo que es su intención. Algunos beneficios de esto son:

- Definen un juicio. Son una forma de transmitir conocimiento o entendimiento de un desarrollador a otro. A través de restricciones en el diseño, pueden guiar a otros ingenieros a soluciones aceptables sin transferir todo su conocimiento.
- Promueven la integridad conceptual. La integridad conceptual de un sistema es un objetivo importante del diseño de un sistema y una buena idea simple consistentemente aplicada es mejor que varias ideas brillantes dispersas en el sistema.
- Reducen la complejidad. Las restricciones pueden factorizar complejidad, dando un sistema con principios fundamentales evidentes. Por ejemplo si los datos solo pueden registrarse en una bd, se sabe dónde buscarlos.
- Entender comportamiento en tiempo de corrida.

¿CUÁNDO ES IMPORTANTE LA ARQUITECTURA? – CASOS CON ALTO RIESGO ARQUITECTURA

Espacio de solución pequeño

La arquitectura es importante cuando el espacio de solución es pequeño o es difícil diseñar una solución aceptable.

Considerar la dificultad de crear un aeroplano movido por un humano comparado a crear un carro mas veloz.

Alto riesgo de falla

Cada vez que los riesgos de una falla son altos, probablemente es importante tener la arquitectura correcta.

Gente puede morir si el sistema en un hospital falla.

Atributos de calidad difíciles

La arquitectura influencia la habilidad para satisfacer atributos de calidad, mientras hacer otro sistema de correo electrónico puede ser fácil, hacer uno con alto rendimiento que soporte millones de usuarios puede ser difícil.

Dominio nuevo

Se debe poner atención cuando el dominio es nuevo, al menos para el desarrollador. NO es lo mismo hacer la decima aplicación de escritorio que hacer la primera.

Líneas de producto

Algunos conjuntos de productos comparten la misma arquitectura, el hacer variantes de un producto puede ser fácil o difícil dependiendo de la misma.

ARQUITECTURA PRESUNTAS (SUPUESTAS)

Una arquitectura presunta es una familia de arquitecturas que es dominante en un dominio particular

En vez de justificar la elección de usarla, los desarrolladores en ese dominio podrían tener que justificar una elección diferente a esa. Son similares a una arquitectura de referencia.

Una arquitectura de referencia es una familia de arquitecturas que describe una solución de arquitectura a un problema y que usualmente está escrita como una especificación

Ejemplos:

- Uso de sistema con capas (acceso compartido datos, reglas negocio cambiantes)
- Uso de procesos cooperativos en un sistema operativo

PRINCIPIOS

- Cada sistema tiene una arquitectura, sea o no documentada y entendida.
- Las arquitecturas son creadas solamente para satisfacer las necesidades de los stakeholders.
- Una buena arquitectura es aquella que satisface los intereses de los stakeholders y cuando dichos intereses están en conflicto, los balancea de forma aceptable para ellos.
- Una buena descripción de arquitectura es aquella que comunica efectiva y consistentemente los aspectos clave de la arquitectura para los stakeholders apropiados.