**UNIVERSIDAD DE TARAPACÁ**



**FACULTAD DE INGENIERÍA**

Departamento de Ingeniería en Computación e Informática



**Lectura 1:**

**Estilos de arquitectura**

**Autor**: José Vásquez Gutiérrez

Gonzalo Vega Mujica

**Curso**: Sistemas Distribuidos

**Profesor**: Diego Aracena Pizarro

ARICA, 09 de Septiembre 2019

**¿Está de acuerdo con la definición de “arquitectura de Software” y cómo lo relaciona con el tema “estilos arquitecturales”?**

Para comenzar, no estamos de acuerdo con la definición, porque en primer lugar no hay una definición clara ni objetivo de lo que es la arquitectura de Software, por lo tanto, lo que podemos decir es que si estamos de acuerdo con que es una parte importante el desarrollo de los sistemas distribuidos, ya que estos indican la estructura, funcionamiento e interacción entre los componentes del software.

**Arquitectura basada en niveles**

En esta arquitectura los componentes están organizadas en niveles, donde un componente de un nivel puede hacer un downcall a un componente de un nivel inferior y solamente en casos excepcionales un componente puede hacer un upcall a un componente de un nivel superior.

La comunicación entre niveles se basa en protocolos, uno conocido es el protocolo de comunicación de stack, para que los componentes se puedan comunicar tienen que realizar una conexión bajo unas reglas que se establecen, para así poder llegar al intercambio de información.

Para las arquitecturas de 2 o más niveles, es necesario establecer un buen protocolo de comunicación entre los componentes, ya que al ser de varios niveles, la comunicación entre componente de diferentes niveles será de mayor uso.

**Arquitectura basadas en Objetos**

En esta arquitectura a diferencia de las demás se basa como dice su título, en objetos. Estos objetos son considerados como componentes los cuales estos mismos componentes están conectados entre sí por mecanismo de llamada de procedimientos. Lo que se debe saber es que existen interfaces lo cuales son entregados por los objetos, estos ocultan detalles de la implementación, de esa forma se considera un objeto independiente de su ambiente. Entonces la diferencia entre interfaces y objeto es que en la primera nos permite poner interfaces en una máquina mientras que por otro lado, los objetos residen en otra máquina.

Con relación al Cliente-Servidor, cuando un cliente establece conexión con un objeto distribuido es llamado proxy y esta es cargada dentro del espacio de dirección del cliente, entonces para llevar a cabo ese proceso, lo único que hace es invocar el método en mensajes y mensajes de respuesta para devolver el resultado de la invocación del método al cliente.

Siguiendo, una particularidad que tiene esta arquitectura a pesar de ser objetos distribuidos estos no poseen un estado distribuido, es decir, que es estado en sí puede estar físicamente distribuido en varias máquinas, pero esta distribución también está oculta para los clientes detrás de las interfaces del objeto. Esto se debe a que solo las interfaces implementadas por los objetos están disponibles en otras máquinas, esto se conoce como objetos remotos.

Por último, la encapsulación es otro aspecto muy importante de esta arquitectura, esta herramienta separa varios servicios para que puedan operar de manera independiente, además el servicio en su conjunto se realiza como una entidad autónoma, aunque posiblemente puede hacer uso de otros servicios.

**Arquitecturas centradas en los recursos**

Si se ve un sistema distribuido como una gran colección de recursos de los cuales los componentes administran individualmente, se podría aprovechar mucho esta ventaja, ya que dichos recursos se podrían modificar, añadir, eliminar, entre otras acciones, y esta ventaja se conoce como REST.

Algunas características de la arquitectura REST es que cada recurso es identificado con un único esquema de nombre, que los mensajes enviados desde o hacia el servidor son auto descritos y por último, después de ejecutar una operación de un servicio, ese componente olvida todo sobre al que lo llama.

Existen 4 esenciales combinados con los cuales trabaja REST, los cuales son:

PUT: Crea un nuevo recurso

GET: Recupera el estado de un recurso en una representación.

DELETE: Elimina un recurso

POST: Modifica un recurso entregando un nuevo estado

Continuando, otro aspecto de la arquitectura son los llamados, buckets, los cuales en simples palabras son directorios y para crear estos, la aplicación solo manda un PUT y el protocolo HTTP, el cual es el protocolo que se usa en este tipo de arquitectura, le retorna un error en caso de que ya esté creado el bucket.

Para finalizar con esta arquitectura, la simplicidad de la arquitectura REST puede facilitar soluciones a esquemas de comunicación, por ejemplo, en transacciones distributivas, que generalmente requieren que los servicios realicen un seguimiento de la ejecución; por otro lado, hay muchos ejemplos en los que las arquitecturas REST coinciden perfectamente con un esquema simple de integración de servicios, pero donde las interfaces dificultan las cosas.

**Arquitecturas basadas en eventos**

La combinación de sistemas referencialmente desacoplados y temporalmente acoplados forman el grupo de modelos por coordinación basado en eventos. En sistemas referencialmente desacoplados los procesos no se conocen explícitamente, lo único que pueden hacer los procesos es publicar una notificación que describe la ocurrencia de un evento, los procesos pueden suscribirse a un tipo específico de notificación, así que una notificación publicada será llegada a aquellos procesos que estén suscritos.

El trabajo a futuro para la arquitectura basada en eventos es la implementación eficiente y escalable de suscripciones coincidentes con las notificaciones, aun así esta arquitectura promete un gran potencial para construir sistemas distribuidos a gran escala, aunque planificar implementaciones escalables sin perder la independencia de esta arquitectura, no es algo trivial que se pueda trabajar.

Obs: Buen trabajo, errores de escritura 6.0