Estilos Arquitecturales

Alumno: Juan Manuel Copia

Ramo: Sistemas Distribuidos

Profesor: Diego Aracena

Introducción

La correcta elección de la arquitectura para un sistema de software es crucial para el éxito del desarrollo de grandes sistemas. Por arquitectura entendemos la organización lógica del sistema en forma de componentes. Hoy en día existen múltiples estilos arquitecturales, estos definen los componentes involucrados, la forma en que se conectan entre ellos, los datos que intercambian y como estos configuran en conjunto todo el sistema.

Arquitectura basada en niveles

Las arquitecturas basadas en niveles es un estilo arquitectural que se caracteriza por organizar el sistema en componentes situados en distintos niveles, es decir, existen distintas capas de abstracción (niveles) con responsabilidades distintas que se comunican entre ellos. Existen diversas variantes, podemos destacar 3:

- **Organización de niveles pura:** Los componentes solo realizan llamadas (requieren servicios) al nivel inferior subsiguiente sin saltear otras capas ni realizar llamadas a capas superiores, es decir, una capa inferior nunca realiza una petición de servicio a una superior.

- **Organización de niveles mixta:** Al igual que la pura, solo se realizan llamadas a niveles inferiores, pero no se restringe solo al nivel inferior subsiguiente, sino que es posible "saltear" niveles y realizar una petición a un nivel de más de un orden inferior.

- **Organización de niveles con llamadas ascendentes:** A diferencia de las otras, los niveles inferiores pueden realizar llamadas a niveles superiores.

Uno de los ejemplos más conocidos que lleva este estilo arquitectural es el protocolo **TCP (Transfer Control Protocol).**

En aplicaciones, usualmente se dividen las responsabilidades del sistema en 3 niveles:

- La interfaz de aplicación.
- El nivel de procesamiento.
- El nivel de datos.

Es muy común encontrar aplicaciones web construidas con esta arquitectura y en combinación con otras.

Arquitectura basada en objetos

Este tipo de arquitecturas es menos rígida y consiste en la definición de los distintos componentes del sistema en objetos que se conectan entre ellos a través de un mecanismo de llamadas de procedimientos. En el caso de los sistemas distribuidos, una llamada a un procedimiento puede tener lugar a través de la red y finalmente ser ejecutado en otra máquina.

Esta arquitectura provee una forma natural de encapsular tanto los datos (estado del objeto) como las operaciones que se realizan sobre estos (métodos) en una sola entidad. La interfaz de cada objeto nos abstrae de sus posibles implementaciones, lo que permite que los objetos sean independientes del entorno (siempre y cuando la interfaz este bien definida y los objetos remplazantes la respeten).

La separación entre objetos e interfaces nos permite poner la interfaz en una máquina por un lado y el objeto en otra, a este tipo de objetos se les llama "objetos distribuidos". En sistemas distribuidos se utiliza un proxy del lado del cliente (interfaz) que realiza llamadas al servidor donde reside el objeto que provee el servicio.

Arquitecturas centradas en recursos

A medida se incrementó la cantidad de servicios disponibles en la web y que el desarrollo de sistemas distribuidos través de la composición de servicios se volvió mas importante, los investigadores tuvieron que repensar las arquitecturas de la mayoría de los sistemas distribuidos web, dado que uno de los grandes problemas residida en el momento de la integración de los distintos componentes.

Como alternativa podemos ver a los sistemas distribuidos como una gran colección de recursos que son manejados individualmente por componentes. Los recursos pueden ser agregados, modificados y eliminados por aplicaciones.

Este enfoque ha sido ampliamente adoptado por la web y es conocido como **REST (Representational State Transfer)**. Y posee cuatro características importantes:

- Los recursos son identificados por un único esquema de nombres.
- Todos los servicios poseen una misma interfaz, ofreciendo de 4 operaciones:
    - **PUT** (crea un nuevo recurso).
    - **GET** (Recupera el estado de un recurso en alguna representación).
    - **DELETE** (Elimina un recurso).
    - **POST** (Modifica un recurso transfiriendo un nuevo estado).

- Los mensajes enviados desde o para un servicio son completamente descriptivos por si mismos.
- Después de ejecutar una operación en un servicio, ese componente olvida todo sobre el invocante. (ejecución sin estado).

Arquitecturas basadas en eventos (publicador/suscriptor)

A medida que los sistemas crecen y los procesos pueden entrar o salir más fácilmente creció la importancia de tener una arquitectura en que las dependencias de los procesos se vean disminuidas al mínimo.

Esta arquitectura establece una separación fuerte entre procesos y coordinación. Esta última abarca la comunicación y la cooperación entre procesos.

Existen diferentes tipos de coordinación en base al acoplamiento que tengan los procesos. Los tipos de acoplamiento pueden ser:

- **Referencialmente Acoplados:** Se dice que los procesos son referencialmente acoplados cuando se hacen referencias explicitas de comunicación entre los procesos, es decir, un proceso conoce el nombre o identificador del otro proceso con el que se quiere comunicar.
- **Referencialmente Desacoplados**: Cuando no se cumple el punto anterior, es decir cuando los procesos no poseen referencias explicitas al proceso con el que se están comunicando.
- **Temporalmente Acoplados**: Se dice que los procesos son Temporalmente acoplados cuando ambos procesos en una comunicación deben estar disponibles y en ejecución
- **Temporalmente Desacoplados:** Cuando no se requiere que ambos procesos estén en ejecución para realizar la comunicación.

Cuando los procesos son acoplados referencialmente y temporalmente, la coordinación se realiza de manera directa, la llamamos **coordinación directa**.

Cuando el acoplamiento es referencial pero no temporal utilizamos la llamada "**coordinación mailbox**", en la cual para establecer la comunicación no es necesario que el otro proceso esté en ejecución sino que los mensajes se depositan en un mailbox compartido.

La combinación de procesos referencialmente desacoplados y temporalmente acoplados utilizan una **coordinación basada en eventos** en este tipo de casos, los procesos no se conocen entre sí explícitamente, lo único que hace es publicar notificaciones que describen la ocurrencia de eventos. Como estas notificaciones pueden ser de distintas características, los procesos

se deben suscribir los distintos tipos de notificaciones que quieren recibir. Cuando un evento ocurre, el proceso encargado del manejo de ese evento realiza una notificación, y esta es entregada a todos los procesos suscriptos a este tipo de notificación. Sin embargo es necesario que los procesos suscriptores estén en ejecución para poder recibir la notificación.

La combinación de procesos referencialmente y temporalmente desacoplados nos lleva a un modelo de coordinación conocido como "**espacio de datos compartido**", en el cual los procesos se comunican en su totalidad a través de tuplas. Los procesos emisores depositan tuplas de cualquier cantidad de campos y de cualquier tipo en un espacio compartido. Los procesos receptores recuperan en dicho espacio las tuplas que coinciden con su patrón de búsqueda.

Tanto la coordinación basada en eventos como la coordinación de espacio de datos compartidos pueden ser utilizadas en la arquitectura publicador-suscriptor. Es decir, existen arquitecturas publicador-suscriptor tanto temporalmente y referencialmente desacopladas como temporalmente acopladas pero referencialmente desacopladas.