Arquitecturas de software actuales

Por: Arturo Cortés Sánchez

Índice:

1. Definición de Arquitecturas del software

Podríamos definir una arquitectura del software de un programa o sistema informático como la estructura o estructuras del sistema, que comprenden los elementos de software, las propiedades visibles externamente de esos elementos y las relaciones entre ellos.

Es importante separar el diseño de arquitecturas de software de otro tipo de diseños: La arquitectura se ocupa de la selección de los elementos arquitectónicos, de su interacción y de sus limitaciones. sobre esos elementos y sus interacciones. El diseño trata la modularización e interfaces detalladas de los elementos de diseño, sus algoritmos y procedimientos, y los tipos de datos necesarios para soportar la arquitectura.

Una versión mas general y simplificada sería que una arquitectura es un conjunto de componentes software y las relaciones entre ellos.

2. Arquitecturas dirigidas por eventos

2.1 Definición de evento

Un evento puede definirse como "un cambio significativo de estado" que provoca posteriormente la transmisión de un mensaje o notificación del cambio. Por ejemplo, cuando un consumidor compra un coche, el estado del coche pasa de "en venta" a "vendido". La arquitectura del sistema de un concesionario de automóviles puede tratar este cambio de estado como un evento cuya ocurrencia puede darse a conocer a otras aplicaciones dentro de la arquitectura.

2.2 EDA

Una arquitectura dirigida por eventos (EDA por sus siglas en inglés) es un patrón de arquitectura de software que se basa en la producción, detección, consumo y reacción a eventos. Abarcan mecanismos para coordinar a los clientes y proveedores de servicio, productores y consumidores de datos, sensores de eventos de software.

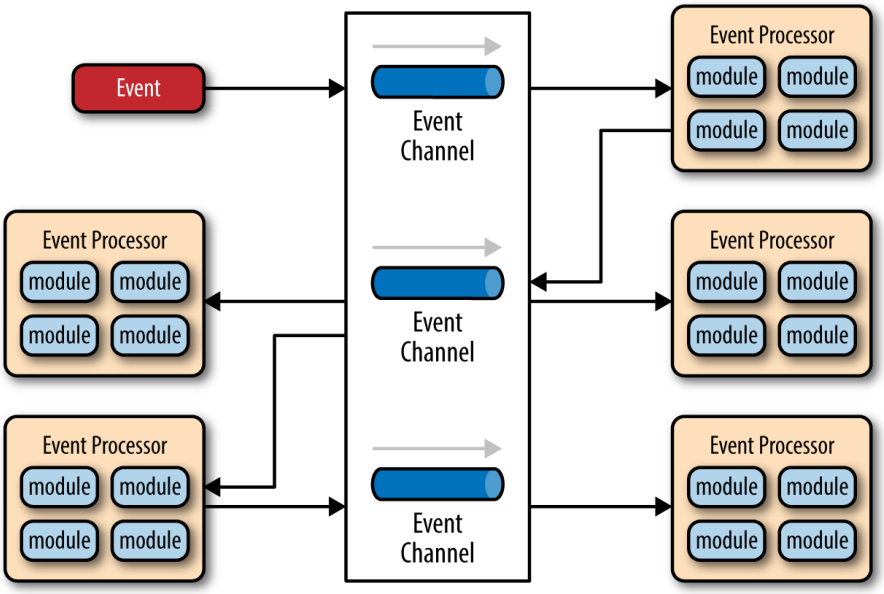
La creación de sistemas en torno a una arquitectura basada en eventos simplifica la escalabilidad horizontal de los modelos de computación distribuidos y los hace más resistentes a los fallos. Esto se debe a que el estado de la aplicación puede ser copiado a través de múltiples instantáneas paralelas para

una alta disponibilidad. Los nuevos eventos pueden iniciarse en cualquier lugar, pero lo más importante es que se propagan a través de la red de almacenes de datos actualizando cada uno a medida que llegan. Agregar nodos adicionales también se vuelve trivial, simplemente puede tomar una copia del estado de la aplicación, alimentarla con un flujo de eventos y ejecutarla.

Las EDA engendran una red que escucha miles de eventos entrantes de diferentes fuentes, en donde el procesamiento de eventos resulta en la respuesta prevista del sistema, además soportan flujos dinámicos, paralelos y asíncronos de mensajes y, por lo tanto, reaccionan a las entradas externas que pueden ser impredecibles por naturaleza.

Por ejemplo, el enfoque de una EDA puede permitir que una persona invoque a un proveedor de bloques de construcción de servicios utilizando eventos sin saber quién lo proporciona o qué dirección utiliza el proveedor, que proveedor se ha elegido, al tiempo que se equilibra la carga entre ellos. Mientras tanto, el mismo evento de software que representa la solicitud o los eventos generados por el servicio en respuesta pueden ser de interés para otros bloques de construcción de servicios en la red, abriendo un canal de valor añadido para el cliente y el negocio.

Una EDA puede coordinar de forma síncrona o asíncrona entre puntos finales de software y, posiblemente, proporcionar acceso síncrono y asíncrono entre los mismos participantes. En una EDA también, se pueden ejecutar flujos simultáneos de ejecución independientes entre sí para cumplir con una solicitud del cliente o responsabilidad del sistema.



Una arquitectura impulsada por eventos puede construirse sobre cuatro capas lógicas, comenzando con la detección de un evento, procediendo a la creación de su representación técnica en forma de una estructura de eventos y terminando con un conjunto no vacío de reacciones a ese evento.

Generador de evento: La primera capa lógica es el generador de eventos, que detecta un hecho y lo representa en un evento. Puesto que un evento puede ser casi cualquier cosa que se pueda detectar, también puede serlo un generador de eventos. Por ejemplo, un generador de eventos podría ser un cliente de correo electrónico, un sistema de comercio electrónico, un agente de supervisión o algún tipo de sensor físico.

Canal de evento: Un canal de eventos es un mecanismo por el cual la información de un generador de eventos se transfiere al motor de eventos o se descarta. Puede ser una conexión TCP/IP o cualquier tipo de fichero de entrada.

Motor de procesamiento del evento: El motor de procesamiento de eventos es donde se identifica el evento y se selecciona y ejecuta la reacción apropiada.

Actividad de descarga dirigida por evento: Aquí es donde se muestran las consecuencias del evento. Esto se puede hacer de muchas maneras y formas diferentes, por ejemplo, se envía un correo electrónico a alguien o una aplicación puede mostrar algún tipo de advertencia en la pantalla.

Ventajas:

1. Simplicidad
2. Evolución: se pueden reemplazar componentes suscriptores
3. Modularidad: una sola modalidad para eventos diversos
4. Puede mejorar la eficiencia, eliminando la necesidad de polling por ocurrencia de evento

Desventajas:

1. Posibilidad de desborde
2. Potencial imprevisión de escalabilidad
3. Pobre comprensibilidad: Puede ser difícil prever qué pasará en respuesta a una acción
4. No hay garantía del lado del publicador, que el suscriptor responderá al evento
5. No hay mucho soporte de recuperación en caso de falla parcial

3. Arquitectura orientada a servicios

3.1 Definición de servicio

Las aplicaciones tradicionales basadas en sistemas de silo tienden a extenderse con el tiempo, resultando en ambientes potencialmente complejos con esfuerzos intensivos requisitos de mantenimiento. En combinación con el surgimiento de arquitecturas de integración en continuo crecimiento que pueden ser incluso más difícil de mantener y evolucionar.

La computación orientada a servicios aboga por la creación de una solución agnóstica a cualquier propósito y por lo tanto útil para múltiples propósitos. Esta lógica multipropósito o reutilizable aprovecha al máximo el carácter intrínsecamente interoperable de los servicios. Los servicios agnósticos tienen un mayor potencial de reutilización que puede lograrse permitiendo que sean repetidamente ensambladas en diferentes composiciones. Cualquier servicio agnóstico puede, por lo tanto, verse obligado en numerosas ocasiones a automatizar diferentes procesos de negocio como parte de diferentes procesos orientados a servicios soluciones.

3.2 SOA

Una arquitectura orientada a servicios (SOA) es un estilo de diseño de software en el que los servicios se proporcionan a los demás componentes mediante componentes de aplicación, a través de un protocolo de comunicación a través de una red. Los principios básicos de la arquitectura orientada a servicios son independientes de productos y tecnologías. Un servicio es una unidad discreta de funcionalidad a la que se puede acceder de forma remota y actuar y actualizar de forma independiente, como la recuperación de un extracto de tarjeta de crédito en línea.

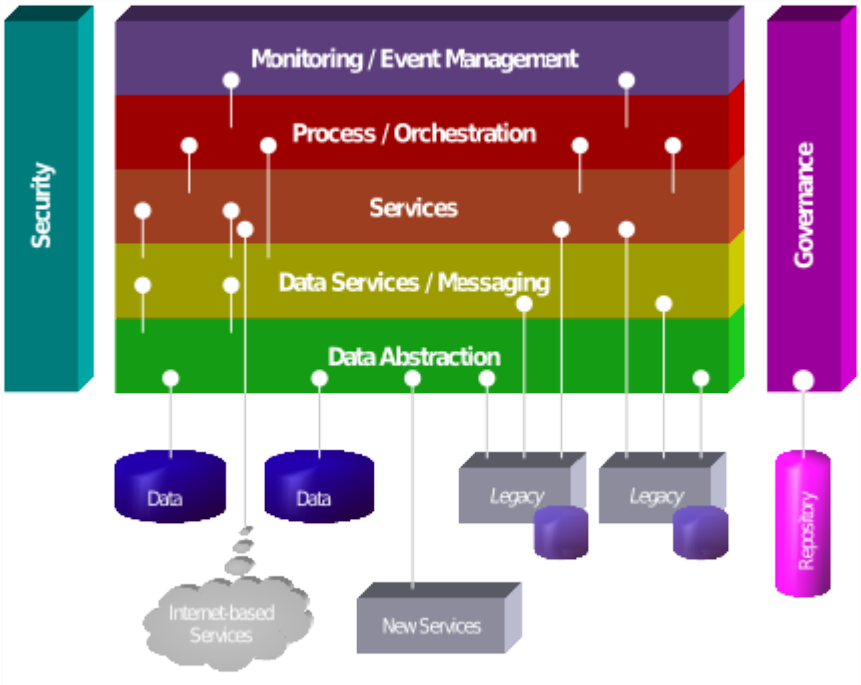
Un servicio tiene cuatro propiedades según una de las muchas definiciones de SOA:

1. Lógicamente, representa una actividad con un resultado específico.
2. Es autónomo.
3. Es una caja negra para sus consumidores.
4. Puede consistir en otros servicios subyacentes.

Se pueden utilizar diferentes servicios conjuntamente para proporcionar la funcionalidad de una gran aplicación de software. La arquitectura orientada a servicios no se trata tanto de cómo modularizar una

aplicación como de cómo componer una aplicación mediante la integración de componentes de software distribuidos, mantenidos por separado e implementados. Es posible gracias a tecnologías y estándares que hacen más fácil que los componentes se comuniquen y cooperen a través de una red, especialmente una red IP.

En SOA, los servicios utilizan protocolos que describen cómo pasan y analizan los mensajes utilizando metadatos. Estos metadatos describen tanto las características funcionales del servicio como las características de calidad del servicio. La arquitectura orientada a servicios tiene como objetivo permitir a los usuarios combinar grandes cantidades de funcionalidad para formar aplicaciones que se construyen exclusivamente a partir de servicios existentes y combinarlos. Un servicio presenta una interfaz simple para el solicitante que abstrae la complejidad subyacente actuando como una caja negra. Otros usuarios también pueden acceder a estos servicios independientes sin tener conocimiento de su implementación interna.



Una SOA establece un modelo arquitectónico que tiene como objetivo mejorar la eficiencia, agilidad y productividad de una empresa mediante el posicionamiento los servicios como el recurso principal. Básicamente, un sistema orientado a servicios gira en torno al paradigma de diseño orientado a servicios y su relación con la arquitectura orientada a servicios. De hecho, el término "arquitectura

orientada a servicios" y sus acrónimos asociados han sido utilizados de manera tan amplia por los medios de comunicación y dentro del marketing de los vendedores que casi se ha convertido en sinónimo de computación orientada a servicios. Por lo tanto, es muy es importante hacer una clara distinción entre lo que realmente es SOA y cómo se relaciona con otros elementos de la computación orientada a servicios. Como una forma de arquitectura tecnológica, una implementación SOA puede consistir en de una combinación de tecnologías, productos, APIs, que soportan extensiones de infraestructura, y varias otras partes. La realidad de una arquitectura orientada a servicios es que es única para cada empresa, sin embargo se caracteriza por la introducción de nuevas tecnologías y plataformas que apoyan específicamente la creación, ejecución y evolución de las soluciones orientadas al servicio.

La arquitectura orientada a servicios puede ser implementada con servicios Web. Esto se hace para que los módulos funcionales sean accesibles a través de protocolos estándar de Internet que son independientes de plataformas y lenguajes de programación. Estos servicios pueden representar tanto nuevas aplicaciones como simples envoltorios de los sistemas heredados existentes para hacerlos compatibles con la red.

Los implementadores comúnmente construyen SOAs usando estándares de servicios web. Un ejemplo es SOAP, que ha ganado una amplia aceptación en la industria tras la recomendación de la Versión 1.2 del World Wide Web Consortium. Estos estándares (también conocidos como especificaciones de servicios web) también proporcionan una mayor interoperabilidad y cierta protección contra el bloqueo y el software propietario del proveedor. Sin embargo, también se puede implementar SOA utilizando cualquier otra tecnología basada en servicios, como Jini, CORBA o REST.

Beneficios de usar una SOA:

Una SOA podría considerarse como una evolución arquitectónica más que como una revolución. Captura muchas de las mejores prácticas de arquitecturas de software anteriores. En los sistemas de comunicaciones, por ejemplo, se han desarrollado muy pocas soluciones que utilicen enlaces verdaderamente estáticos para hablar con otros equipos de la red. Al adoptar un enfoque SOA, estos sistemas pueden posicionarse para enfatizar la importancia de interfaces bien definidas y altamente interoperables.

Las SOA prometen simplificar las pruebas indirectamente. Los servicios son autónomos, sin estado, con interfaces totalmente documentadas y separados de las preocupaciones transversales de la implementación. Si una organización posee datos de prueba apropiadamente definidos, entonces se construye un comprobante correspondiente que reacciona a los datos de prueba cuando se está construyendo un servicio. También se captura para el servicio un conjunto completo de pruebas de regresión, scripts, datos y respuestas. El servicio puede probarse como una "caja negra" utilizando los

comprobantes existentes correspondientes a los servicios que llama. A medida que cada interfaz se documenta completamente con su propio conjunto completo de documentación de pruebas de regresión, resulta sencillo identificar problemas en los servicios de pruebas. Las pruebas evolucionan para validar simplemente que el servicio de pruebas funciona de acuerdo con su documentación, y encuentra lagunas en la documentación y los casos de prueba de todos los servicios del entorno. La gestión del estado de los datos de los servicios idempotentes es la única complejidad.

Problemas asociados a las SOA:

SOA se ha combinado con servicios Web, sin embargo, los servicios Web son sólo una opción para implementar los patrones que conforman el estilo SOA. En ausencia de formas nativas o binarias de llamada de procedimiento remoto (RPC), las aplicaciones podrían ejecutarse más lentamente y requerir más potencia de procesamiento, aumentando los costos. La mayoría de las implementaciones incurren en estos gastos generales, pero SOA puede implementarse utilizando tecnologías que no dependen de las llamadas de procedimientos remotos o la traducción a través de XML. Al mismo tiempo, las nuevas tecnologías de análisis de XML de código abierto y varios formatos binarios compatibles con XML prometen mejorar significativamente el rendimiento de SOA. Los servicios implementados usando JSON en lugar de XML no sufren de este problema de rendimiento.

Los servicios estatales requieren que tanto el consumidor como el proveedor compartan el mismo contexto específico del consumidor, que está incluido en los mensajes intercambiados entre el proveedor y el consumidor o al que se hace referencia en ellos. Esta limitación tiene el inconveniente de que podría reducir la escalabilidad general del proveedor de servicios si éste necesita mantener el contexto compartido para cada consumidor. También aumenta el acoplamiento entre un proveedor de servicios y un consumidor y dificulta el cambio de proveedores de servicios. En última instancia, algunos críticos consideran que los servicios SOA están aún demasiado limitados por las aplicaciones que representan.

4. SOA dirigida por eventos

La SOA basada en eventos es una forma de arquitectura orientada a servicios (SOA), que combina la inteligencia y proactividad de la arquitectura basada en eventos con las capacidades organizativas que se encuentran en las ofertas de servicios. Antes de la SOA basada en eventos, la plataforma típica de SOA orquestaba los servicios de forma centralizada, a través de procesos de negocio predefinidos. Este antiguo enfoque (a veces llamado SOA 1.0) no tiene en cuenta los eventos que ocurren a través o fuera de procesos de negocio específicos. Por lo tanto, en la arquitectura SOA tradicional no se tienen en

cuenta los eventos complejos, en los que un patrón de actividades debería desencadenar un conjunto de servicios.

La programación de SOA 2.0 está estructurada alrededor del concepto de relaciones desacopladas entre productores y consumidores de eventos: a un consumidor de eventos no le importa dónde o por qué ocurre un evento, más bien, le preocupa que sea invocado cuando el evento ha ocurrido. Los sistemas y aplicaciones que separan a los productores de eventos de los consumidores de eventos suelen depender de un despachador o canal de eventos. Este canal contiene una cola de eventos que actúa como intermediario entre los productores de eventos y los organizadores de eventos.

El paradigma prototípico de SOA 2.0 contiene cuatro elementos esenciales:

1. Múltiples eventos del sistema de bajo nivel que, por separado, no parecen tener ninguna relación, pero a través de la detección de patrones comparando estos muchos eventos se hace evidente una correlación inusual o menos obvia.

2. Una cierta cantidad de enriquecimiento de datos mediante la introducción de información relacionada con cada evento para ilustrar más claramente cómo se relacionan los muchos eventos.

3. Una condición desencadenante que, cuando no se cumple, no se crea el evento a nivel empresarial, pero cuando se cumple la condición desencadenante, se crea el evento superior.

4. Algún proceso humano o automatizado que se invoque cuando se alcanza el evento de disparo.

Los Servicios Web de SOA 2.0 pueden ser compuestos de dos maneras: orquestación y coreografía. En la orquestación, un proceso central toma el control de los servicios web involucrados y coordina la ejecución de diferentes operaciones en los servicios web involucrados en la operación. Los servicios SOA 2.0 involucrados no saben (y no necesitan saber) que forman parte de una composición o de un proceso de negocio superior.

Sólo el coordinador central de la orquestación lo sabe, por lo que la orquestación está centralizada con definiciones explícitas de operaciones y el orden de invocación de los servicios SOA 2.0.

Por otro lado, la coreografía no cuenta con un coordinador central. Más bien, cada servicio SOA 2.0 involucrado en la coreografía sabe exactamente cuándo ejecutar sus operaciones (basándose en criterios de activación definidos) y con quién interactuar. La coreografía es un esfuerzo de colaboración centrado en el intercambio de mensajes. Todos los participantes de la coreografía deben ser conscientes del proceso de negocio, de las operaciones a ejecutar, de los mensajes a intercambiar, y de la sincronización de los intercambios de mensajes.

5. Orquestación

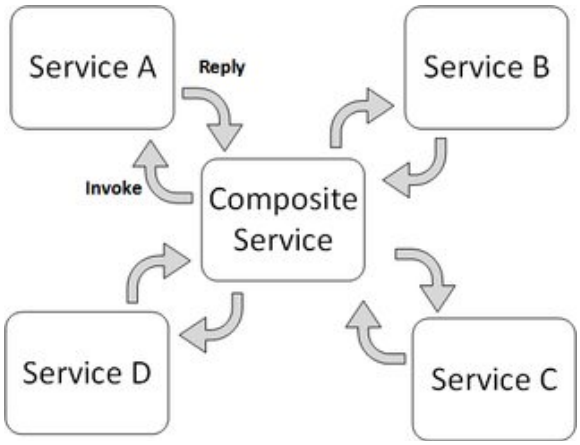
La orquestación es un montaje automatizado que lleva la coordinación y la gestión de sistemas informáticos, middleware y servicios.

A menudo se discute que la orquestación tiene una inteligencia inherente o incluso un control implícitamente autónomo, pero en gran medida se trata de aspiraciones más que de descripciones técnicas. En realidad, la orquestación es en gran parte el efecto de la automatización.

Este uso de la orquestación se discute a menudo en el contexto de la arquitectura orientada a servicios, la virtualización, la infraestructura convergente y los temas de centros de datos dinámicos. En este sentido, la orquestación consiste en alinear la solicitud de negocio con las aplicaciones, los datos y la infraestructura. Define las políticas y los niveles de servicio a través de flujos de trabajo automatizados, aprovisionamiento y gestión del cambio. Esto crea una infraestructura alineada con la aplicación que puede ampliarse o reducirse en función de las necesidades de cada aplicación.

La orquestación también proporciona una gestión centralizada del conjunto de recursos, incluyendo la facturación, la medición y la devolución de cargo por consumo. Por ejemplo, la orquestación reduce el tiempo y el esfuerzo para desplegar múltiples instancias de una sola aplicación. Y a medida que se dispara el requerimiento de más recursos o una nueva aplicación, las herramientas automatizadas ahora pueden realizar tareas que antes sólo podían ser realizadas por múltiples administradores que operaban en sus piezas individuales de la pila física.

Un uso algo diferente se relaciona con el proceso de coordinación de un intercambio de información a través de las interacciones de los servicios web. Las aplicaciones que desacoplan la capa de orquestación de la capa de servicio a veces se denominan aplicaciones ágiles.



En el cloud computing se entiende por "orquestador" la entidad que gestiona procesos entre dominios (sistema, empresa, cortafuegos) y gestiona excepciones. Dado que un orquestador es valioso en los procesos de cumplimiento, aseguramiento y facturación, las encarnaciones conscientes del servicio de un orquestador deben ser capaces de realizar ajustes basados en la retroalimentación de las herramientas de monitorización. En el nivel más básico, un orquestador es un humano.

La principal diferencia entre una "automatización" de flujos de trabajo y una "orquestación" (en el contexto de la computación en nube) es que los flujos de trabajo se procesan y completan como procesos dentro de un único dominio con fines de automatización, mientras que la orquestación incluye un flujo de trabajo y proporciona una acción dirigida hacia metas y objetivos más amplios.

En este contexto, y con el objetivo general de alcanzar metas y objetivos específicos (descritos a través de parámetros de calidad de servicio), por ejemplo, alcanzar las metas de rendimiento de las aplicaciones utilizando un coste mínimo y maximizar el rendimiento de las aplicaciones dentro de las limitaciones presupuestarias.

La orquestación de servicios en la nube consiste en estos elementos:

1. Composición de la arquitectura, las herramientas y los procesos utilizados por los seres humanos para prestar un servicio definido.
2. Cosido de componentes de software y hardware para ofrecer un servicio definido.

3. Conectando y Automatizando los flujos de trabajo cuando sea aplicable para entregar un Servicio definido.

La orquestación es crítica en la entrega de servicios cloud por estas razones:

1. Los servicios en la nube están destinados a ampliarse de forma arbitraria dinámica, sin necesidad de intervención humana directa.
2. La entrega de servicios en la nube incluye la garantía de cumplimiento y la facturación.
3. La entrega de servicios cloud implica flujos de trabajo en varios dominios técnicos y de negocio.

6. Coreografía

La coreografía de servicios es una forma de composición del servicio en la que el protocolo de interacción entre varios servicios asociados se define desde una perspectiva global.

En tiempo de ejecución cada participante en una coreografía de servicio ejecuta su parte de ella (es decir, su papel) de acuerdo con el comportamiento de los otros participantes. El rol de una coreografía especifica el comportamiento de mensajería esperado de los participantes que lo interpretarán en términos de la secuencia y tiempo de los mensajes que pueden consumir y producir.

La coreografía describe la secuencia y las condiciones en las que los datos intercambiados entre dos o más participantes con el fin de cumplir un propósito útil.

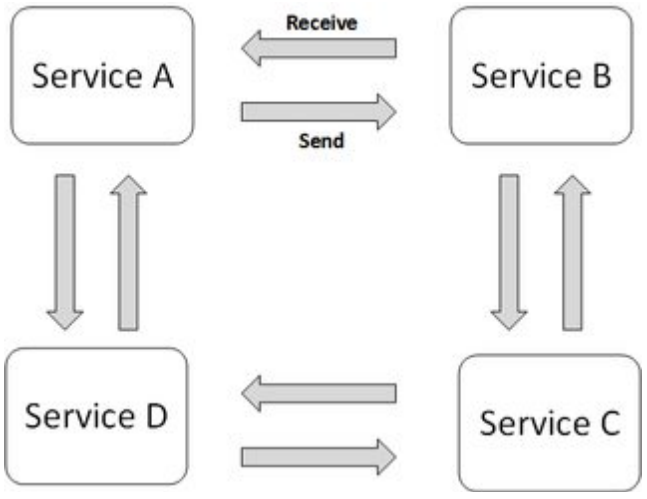
La coreografía de serviciso se entiende mejor a través de la comparación con otro paradigma de composición de servicio: la orquestación de servicios. Por un lado, en las coreografías de servicios la lógica de las interacciones basadas en mensajes entre los participantes se especifica desde una perspectiva global. En la orquestación de servicio, por otro lado, la lógica se especifica desde el punto de vista local de un solo participante, llamado el orquestador. En el lenguaje de orquestación de servicios BPEL, por ejemplo, la especificación de la orquestación de servicios puede desplegarse en la infraestructura de servicios. En el despliegue de la especificación de orquestación del servicio se crea el servicio compuesto.

En cierto sentido, las coreografías y orquestaciones de servicio son dos tiradas de la misma moneda. Por un lado, los roles de una coreografía de servicio pueden ser extraídos como orquestaciones de

servicio a través de un proceso llamado proyección. A través de la proyección es posible realizar esqueletos, es decir, orquestaciones de servicio incompletas que pueden ser utilizadas como líneas de base para realizar los servicios web que participan en la coreografía del servicio. Por otro lado, las orquestaciones de servicio ya existentes pueden ser compuestas en coreografías de servicio.

Las coreografías de servicio no se ejecutan: se representan. Una coreografía de servicio se representa cuando sus participantes ejecutan sus roles. Es decir, a diferencia de la orquestación de servicio, las coreografías de servicio no son ejecutadas por algún motor en la infraestructura de servicio, sino que "suceden" cuando se ejecutan sus roles. Esto se debe a que la lógica de la coreografía de servicio se especifica desde un punto de vista global, y por lo tanto no es realizada por un solo servicio como en la orquestación de servicio.

La pregunta clave que gran parte de la investigación coreográfica intenta responder es ésta: Supongamos que se construye una coreografía global que describe las posibles interacciones entre los participantes en una colaboración. ¿Qué condiciones debe cumplir la coreografía para garantizar el éxito de la colaboración? Aquí, éxito significa que el comportamiento emergente que resulta cuando se realiza la colaboración, con cada participante actuando independientemente según su propio esqueleto, sigue exactamente la coreografía a partir de la cual los esqueletos fueron proyectados originalmente. Cuando éste es el caso, se dice que la coreografía es realizable. En general, determinar la realización de una coreografía es una cuestión no trivial, particularmente cuando la colaboración utiliza mensajería asíncrona y es posible que diferentes participantes envíen mensajes simultáneamente.

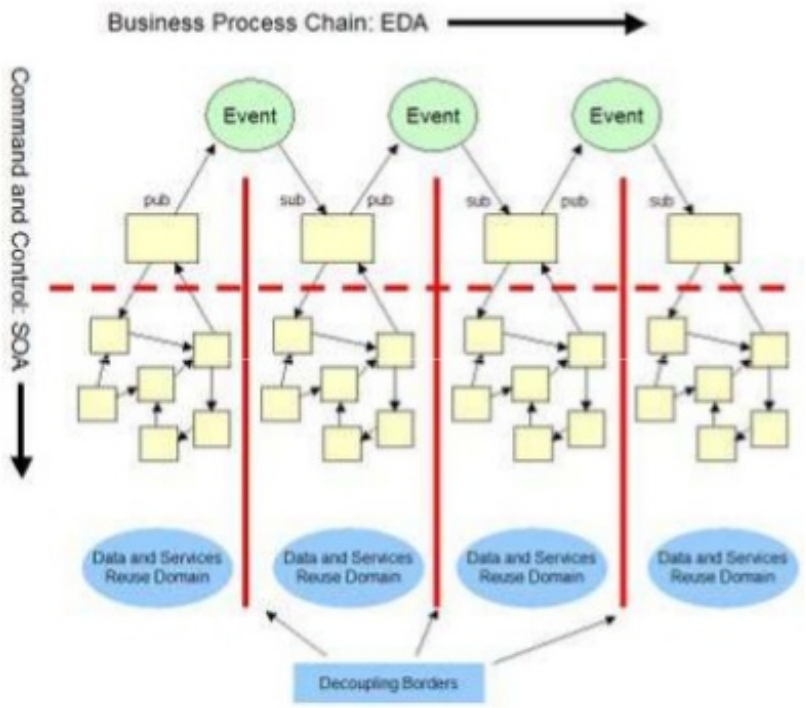


7. Conclusiones

Tras la realización del trabajo podemos concluir que SOA y EDA no son arquitecturas competidoras sino que se complementan muy bien. EDA a menudo proporciona una buena descripción del mundo real, y los eventos a menudo pueden ser fácilmente leídos de los modelos de proceso. EDA es más eficiente que SOA si hay varios consumidores para ciertos datos, ya que no es necesario programar muchas llamadas de servicio. Esto se debe a que la generación única de un evento en el sistema fuente es suficiente.

Las SOA permiten una forma de modularizar los sistemas y aplicaciones en componentes de negocio que pueden combinarse y recombinarse con interfases bien definidas para responder a las necesidades de la empresa. Los servicios Web encajan en este tipo de arquitectura ya que éstos empaquetan una función que puede ser reutilizada permitiendo interoperabilidad y transparencia de localización, es decir, que el cliente de un servicio es esencialmente independiente de la construcción del mismo. Tanto la orquestación como la coreografía en la composición de los Servicios Web son elementos claves para lograr una arquitectura orientada al servicio.

Finalmente tenemos la unión de ambas arquitecturas, las SOA dirigida por eventos, esta, al ser también una evolución de sus antecesoras presenta facilidades para el mundo actual.



8. Referencias

https://www.researchgate.net/publication/221553844_Architecture_Design_Implementation

https://www.researchgate.net/publication/3248354_The_Past_Present_and_Future_for_Software_Architecture

http://www.omg.org/news/meetings/workshops/MDA-SOA-WS_Manual/04-A4_Radhakrishnan-Sriraman.pdf

https://en.wikipedia.org/wiki/Service-oriented_architecture

https://en.wikipedia.org/wiki/Event-driven_architecture

https://en.wikipedia.org/wiki/Event-driven_SOA

https://en.wikipedia.org/wiki/Orchestration_(computing)

https://en.wikipedia.org/wiki/Service_choreography

http://www.oracle.com/technetwork/articles/soa/ind-soa-events-2080401.html

http://comunidad.ingenet.com.mx/anacareaga/2010/03/25/bailando-en-la-web-coreografia-y-orquestacion-de-los-servicios-web/

Thomas Erl. SOA: principles of service design. Prentice-Hall (2008)

Bass, L., Clements, P., and Kazman, R. (2003). Software Architecture In Practice