



Ingeniería Industrial
ISSN: 0258-5960
revistaii@ind.cujae.edu.cu
Instituto Superior Politécnico José
Antonio Echeverría
Cuba

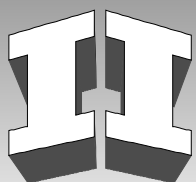
Romero, Pedro Ángel
ARQUITECTURA DE SOFTWARE, ESQUEMAS Y SERVICIOS
Ingeniería Industrial, vol. XXVII, núm. 1, 2006, pp. 19-21
Instituto Superior Politécnico José Antonio Echeverría
La Habana, Cuba

Disponible en: <http://www.redalyc.org/articulo.oa?id=360433560001>

- Cómo citar el artículo
- Número completo
- Más información del artículo
- Página de la revista en redalyc.org

redalyc.org

Sistema de Información Científica
Red de Revistas Científicas de América Latina, el Caribe, España y Portugal
Proyecto académico sin fines de lucro, desarrollado bajo la iniciativa de acceso abierto



ARQUITECTURA DE SOFTWARE, ESQUEMAS Y SERVICIOS

Resumen / Abstract

Cuando en la industria de software los productos tienen requerimientos cada vez más complejos y dinámicos, y los tiempos para desarrollarlos son cada vez menores; la reutilización y el bajo acoplamiento entre los componentes cobran vital importancia. En este trabajo, partiendo de las definiciones de arquitectura de software y esquema se tratan las características del paradigma de arquitectura orientada a servicios y se exponen algunos elementos significativos que muestran cómo los servicios son la evolución natural de los componentes de software. También se comentan algunas cuestiones a tener en cuenta a la hora de diseñar orientado a servicios.

When software products have requirements every day more complex and dynamic, and development times are shorter; reutilization and low coupling among components take special importance. In this paper, departing from definitions of Software Architecture and Framework are exposed the characteristics of Service Oriented Architecture paradigm and several elements that show how services are the natural evolution for software components. There are a few questions to take into account when designing oriented to services.

Palabras clave / Key words

Arquitectura de software, esquemas, servicios de software, servicios Web

Software architecture, frameworks, software services, Web-services

INTRODUCCIÓN

Cada vez es más frecuente que las aplicaciones requeridas por clientes empresariales deban responder a necesidades que hace unos pocos años eran muy raras o no existían: deben ser independientes de los sistemas operativos y gestores de bases de datos; ser accedidas desde lugares geográficamente distantes; interactuar con otros sistemas ya en funcionamiento; atender grandes volúmenes de interacciones por unidad de tiempo, originadas por usuarios que utilizan la aplicación simultáneamente entre otras necesidades.

Así, las aplicaciones desarrolladas han transitado de un enfoque centralizado y monolítico hacia entornos distribuidos y heterogéneos;¹ lo que ha conllevado un aumento de la complejidad de dichas aplicaciones desde el punto de vista estructural.

Así, dentro de la ingeniería de software, ha tenido un auge significativo la arquitectura de software como rama emergente. Al tiempo que el rol del arquitecto ha cobrado especial importancia.

Tómese como arquitectura de software el conjunto de decisiones de diseño que definen la estructura fundamental de un sistema de cómputo. Contemplando las partes componentes del sistema: sus interfaces, comportamiento, las relaciones entre ellas, las relaciones de ellas con el medio y la evolución de cada componente junto al sistema.

ARQUITECTURA Y ESQUEMAS

La mayoría de las aplicaciones contienen funcionalidades comunes, se pudieran citar entre otras: almacenamiento en memoria estática, comunicaciones, tratamiento de la interfaz usuario, errores, transacciones, mantenimiento de la seguridad.

La solución de duplicar el código encargado de estas aplicaciones se vuelve impensable para aplicaciones de gran envergadura: aumentando su tamaño, y disminuyendo la productividad en el desarrollo, perdiéndose la consistencia y si se llegaran a desarrollar y desplegar la aplicación, se haría inmanejable su mantenimiento.

Se hace necesario tener unidades que implementen las diversas funcionalidades que son necesidades comunes a diferentes aplicaciones o a diferentes partes de una misma aplicación. Surge así el término **esquema**, que para este trabajo ha de entenderse como un conjunto de clases relacionadas que brindan, de manera unificada, las funcionalidades necesarias para implementar un servicio específico. Dicho servicio puede ser utilizado o adaptado para la implementación de otras aplicaciones o servicios.

La idea es que cada aplicación que lo requiera utilice los **esquemas** que implementen cada una de las funcionalidades comunes.

Muchas veces se torna más factible comprar u obtener un esquema desarrollado por un tercero que construirlo. Además, existen estándares para algunas funcionalidades y es probable que estén disponibles **esquemas** que los implementan, de utilizarse un esquema así se obtendrían los beneficios de la estandarización. Es importante señalar que desarrollar un **esquema** de fortaleza industrial puede tomar años de esfuerzo y además hay asuntos sutiles que requieren de experticia especial.

Si el caso es que las funciones comunes que se necesitan son parte de aplicaciones nuevas a desarrollar desde cero, entonces la solución se muestra fácil. Sin embargo, cuando son parte de sistemas existentes y en explotación al arquitecto le quedan dos caminos: modificar los sistemas viejos en aras de reutilizar el código existente; o por otra parte reimplementar la funcionalidad existente.

Según la referencia 2, la segunda es la más fácil y segura, lo que hace que sea la escogida en la generalidad de los casos trayendo consigo:

- Funcionalidad replicada.
- Dificultad de migración de los sistemas internos. Al haber múltiples conexiones desde sistemas que dependen de estos para su funcionamiento.
- Al no haber una estrategia de integración de aplicaciones, se generan múltiples puntos de falla, que pueden detener la operación de todos los sistemas muy fácilmente.
- Un modelo así, por lo general, no escala muy bien.
- El inconveniente final es una pobre respuesta al cambio. Las aplicaciones siguen siendo concebidas desde un principio como islas independientes.

Las inconveniencias de esta solución pueden llegar a tener una influencia crítica en ambientes muy competitivos; cuando se

busque desarrollar aplicaciones complejas con alto valor agregado, contando con pocos recursos: especialmente tiempo de desarrollo.

ARQUITECTURA ORIENTADA A SERVICIOS

El término arquitectura orientada a servicios ha sido descrito en la industria de tecnologías de la información de varias maneras a través de los años. Sin embargo, desde el 2001, los estándares ampliamente adoptados de la Edición Empresarial de Java y los Servicios Web han habilitado un nuevo nivel de interoperabilidad y arquitecturas orientadas a servicios.³

Una estrategia de aplicaciones empresariales debe facilitar su integración. Además de motivar la construcción de servicios, más que de aplicaciones.²

Así, puede llegarse a un enfoque donde los diversos procesos a automatizar han de ser vistos como servicios autónomos que interactúan entre sí a través de mensajes. Cada servicio además de poseer lógica y datos propios, expone claramente la interfaz basada en mensajes para accederlo; constituyendo una aplicación autónoma e independiente que permite la comunicación con otras aplicaciones.

Cada aplicación conoce, en este enfoque, cuáles servicios son necesarios y cómo deben ser utilizados para ofrecer los resultados esperados a los usuarios finales.

La comunicación hacia y desde el servicio, es realizada utilizando mensajes y no llamadas a métodos. Estos mensajes deben contener o referenciar toda la información necesaria para entenderlo. La idea es que haya el mínimo posible de llamadas entre el cliente y el servicio.

Un servicio es la evolución en complejidad de un componente distribuido, y se diferencian en:

- Mucho menos acoplados con sus aplicaciones cliente que los componentes.
- Menor granularidad que los componentes.
- No son diseñados e implementados necesariamente como parte de una aplicación *end-to-end*.
- Son controlados y administrados de manera independiente.
- Expone su funcionalidad a través de protocolos abiertos e independientes de plataforma, incluso arriesgando el rendimiento y consumo de recursos.
- Son transparentes de su localización en la red, de esta manera garantizan escalabilidad y tolerancia a fallos.
- Tienen sus propias políticas de escalabilidad, seguridad, tolerancia a fallos, manejo de excepciones, configuración.²

Después de observar todas estas características se puede llegar a una comparación entre las aplicaciones tradicionales y las que poseen una arquitectura orientada a servicios.

El desarrollo tradicional de aplicaciones favorece una arquitectura y estructura monolíticas. Estas aplicaciones estaban dirigidas a la automatización de funciones y destinadas a durar. La reducción de costos se mantenía como el principal motivador de este estilo de desarrollo de aplicaciones. El desarrollo

tradicional lleva a largos ciclos de desarrollo y orientación al código. Parcialmente esto estaba influenciado por la dificultad o imposibilidad de reutilizar las funcionalidades altamente acopladas propias de este paradigma.

Las arquitecturas orientadas a servicios son aplicaciones y procesos de negocios que empaquetan funcionalidades de negocios. De ahí que las arquitecturas orientadas a servicios presenten una infraestructura y arquitectura de software ágiles, adaptativas y de bajo acoplamiento.³

DISEÑO DENTRO DE LA ARQUITECTURA ORIENTADA A SERVICIOS

La implementación ideal de un servicio exige resolver algunos inconvenientes técnicos inherentes a su modelo:

Los tiempos de llamado no son despreciables, gracias a la comunicación de la red, tamaño de los mensajes entre otros factores con influencia. Esto necesariamente implica la utilización de mensajería confiable.

La respuesta del servicio es afectada directamente por aspectos externos como problemas en la red y configuración por citar algunos. Estos deben ser tenidos en cuenta en el diseño, desarrollándose los mecanismos de contingencia que eviten la parálisis de las aplicaciones y servicios que dependen de él. Debe manejar comunicaciones no confiables, mensajes impredecibles, reintentos, mensajes fuera de secuencia.²

Según lo anterior, se puede ver que la construcción de un servicio es una tarea mucho más complicada que la de un simple componente distribuido.

Cuando se usan múltiples servicios para implementar un sistema, es muy fácil que la comunicación entre estos se salga de control. Por ejemplo, si se tiene un servicio que llama a otros seis servicios, alguno de los cuales llama a otros servicios, por lo que muy fácilmente el sistema se vuelve inmanejable. De esta manera, un sistema grande puede terminar con múltiples dependencias. Detectar un problema de rendimiento o funcionalidad se puede volver muy complicado.


Una solución lógica a este problema es extraer los aspectos de procedimiento de varios servicios dentro de uno dedicado, llamado servicio de negocio. Un servicio de negocio controla las acciones paso a paso en la ejecución de algún trabajo, moviendo el sistema de un estado a otro. En cada paso, este llamara una operación de negocio provista por un servicio.

Así, un servicio de negocio centraliza la definición del proceso, en vez de tener piezas del proceso entre todos los servicios. Esto disminuye las dependencias entre servicios y las aplicaciones clientes; ayudando a facilitar la administración del sistema.

CONCLUSIONES

Después de transitar por los conceptos de arquitectura de software, esquema y las características del paradigma de

arquitectura orientada a servicios, se puede llegar a las siguientes conclusiones:

- En la implementación de la arquitectura de software, los esquemas tienen especial importancia cuando se trata de aumentar la reutilización y potenciar la productividad en los ciclos de desarrollo.
- El desarrollo bajo el paradigma de arquitectura orientada a servicios disminuye el acoplamiento entre los componentes de una aplicación, facilitando la adaptación de las aplicaciones desarrolladas e independizando la colaboración entre componentes de su distribución geográfica o plataforma de base. 

REFERENCIAS

1. **CUESTA QUINTERO, C. E.:** *Arquitectura de software dinámica basada en reflexión*. ETS, Informática. Valladolid, Universidad de Valladolid, 2002.
2. **PARRA, JOSÉ DAVID:** *Hacia una arquitectura empresarial basada en servicios*, MSDN, 2004.
3. **JBOSS Inc. 2005.** *JEMS: The Open Source Platform for SOA*. <http://www.jboss.com/elqNow/elqRedir.htm?ref=/pdf/JEMS-PlatformForSOA.pdf> (2005-12-20).

