A Brief Survey of Software Architecture Concepts and Service Oriented Architecture

Mohammad Hadi Valipour, Bavar Amirzafari, Khashayar Niki Maleki and Negin Daneshpour Department of Electrical and Computer Engineering

Shahid Rajaee University Tehran 1678815811, Iran

Email: {m.h.valipour, b.amirzafari, kh.niki}@sru.ac.ir, [daneshpour@aut.ac.ir](mailto:daneshpour@aut.ac.ir)

**Resumen-Un tema crítico en el diseño y construcción de cualquier sistema de software complejo es su arquitectura. La arquitectura de software como columna importante del proceso de desarrollo de software tiene varios métodos y planes de trabajo que todos ellos tienen algunos principios comunes. Se han promovido los enfoques basados en la arquitectura como un medio para controlar la complejidad de la construcción y evolución de los sistemas. En este paper intentamos describir los fundamentos y la estructura principal de la arquitectura de software con una visión conceptual de este tema. En primer lugar, se presenta la definición de arquitectura. Finalmente, la arquitectura orientada a servicios (SOA) como una de las opciones útiles para que la arquitectura de software desarrolle software y sistemas web se glosa en un estudio.**

**Términos del índice-Arquitectura de software, arquitectura orientada a servicios, Servicios Web.**

1. INTRODUCCIÓN

El software se ha vuelto cada vez más complejo, durante la historia de la informática. Se han propuesto varios enfoques para abordar la complejidad, a diferentes niveles, como la "programación estructurada"[1], y la idea de "integridad conceptual"[2] de Fred Brooks. La fase de diseño del ciclo de vida del software se ha dividido a menudo en diseño de alto nivel y diseño detallado. Muchos conceptos en el curso ordinario (construcción) señalan que la arquitectura será útil para describir el software, que dio origen al término "arquitectura de software". El concepto de arquitectura de software ha surgido como diseño de una solución a un alto nivel de los problemas de complejidad. Brooks escribió en los años setenta sobre la importancia de la arquitectura pero lo que es dibujar la interfaz de usuario hoy en día, pero con un poco de la noción de arquitectura de software de hoy en día[2]. A finales de 1994, Denning y Dargan propusieron una "arquitectura de software" para una nueva disciplina de software[3], sin embargo, la descripción era similar a la de un método de desarrollo, no a la de un deﬁnición. No se llegó a un consenso sobre el término hasta la mitad de la década de los noventa. Garlan y Shaw dijeron en 1996 que "la atención explícita a la arquitectura como un nivel separado de diseño de software es relativamente reciente" y por lo tanto su libro está subtitulado "perspectivas sobre una disciplina emergente"[4]. La arquitectura orientada a servicios (SOA) es una arquitectura que modulariza los servicios.

A continuación, se puede coordinar el cobro de estos servicios a los procesos empresariales a la vida. En una SOA exitosa, se puede recomendar estos servicios en varias formas para la implementación de procesos de negocio nuevos o mejorados. SOA es un descendiente de la evolución lógica de las técnicas de modularización de software que se remontan a más de 50 años atrás con la introducción de la programación estructurada. La novedad de SOA es que le da más ﬂexibilidad en la elección de la implementación de tecnologías y ubicaciones para proveedores de servicios y consumidores. Las interfaces de servicios abstractos también permiten a los proveedores y a los consumidores evolucionar de forma independiente, siempre que las interfaces permanezcan estables[5]. El beneficio de SOA resultó principalmente de una sola característica: la estabilidad del servicio de interfaz. Esta estabilidad, comparada con el porcentaje total de cambios en los sistemas, aísla el servicio a los consumidores en el desarrollo de los servicios de implementación. Este aislamiento limita el alcance del cambio y el coste de las modificaciones posteriores. La reutilización evita el coste de la reimplementación o modiﬁcación de la funcionalidad encapsulada en el servicio[5].

1. ARQUITECTURA DE SOFTWARE
2. *Ideas comunes y definiciones*

La arquitectura de software es un "ﬁrst-cut" para diseñar el sistema y resolver el problema o adaptar la necesidad. Lo que realmente se entiende por arquitectura de software es que es un activo de los planes que guían la selección, la construcción, modiﬁcación y el uso perfecto de la infraestructura de información de la empresa para permitir el estado futuro del negocio deseado. Utilizamos el término "arquitectura", en contraste con "diseño", para evocar nociones de codiﬁcación, de abstracción, de estándares, de formación formal (de arquitectos de software) y de estilo. La arquitectura se ocupa de la selección de los elementos arquitectónicos, sus interacciones, y las restricciones sobre esos elementos y sus interacciones, también se puede ver un dibujo ad hoc de “box-and-line” del sistema que realmente se crea y se destina a la solución de los problemas que se articulan por las especificaciones. En el dibujo mencionado:

-Las cajas definen los elementos o “partes” del sistema.

-Las líneas definen las interacciones entre las partes.

Necesario para proporcionar un framework en el cual satisfacer los requisitos y servir como base para el diseño. Ahora bien. Lo que generalmente se entiende por diseño es que el diseño se refiere a la modularización e interfaces detalladas de los elementos de diseño, sus algoritmos y procedimientos, y los tipos de datos necesarios para soportar la arquitectura y satisfacer los requisitos. Entre sus valores se puede afirmar:

- Visión actual y futura del negocio

-Calidad en la toma de decisiones incluyendo las decisiones de inversión.

-Uso de software de una manera rentable alineado con las operaciones de negocio

- Reducir la redundancia

- Reutilizar información y componentes de software

- Aprovechar las nuevas soluciones tecnológicas

- Compartir sistemas y datos

- Integración en toda la empresa

- Normas comunes

- Reducción del número de interfaces de aplicación

- Identiﬁcación y la planificación de los datos que faltan

1. *Los roles de la arquitectura de software*

Aunque existen numerosas deﬁniciones de arquitectura de software, en el centro de todas ellas se encuentra la noción de que la arquitectura de un sistema describe su estructura bruta. Esta estructura ilumina el diseño de alto nivel, incluyendo cosas como la forma en que el sistema está compuesto de partes interactivas, donde se encuentran las principales vías de interacción y cuáles son las propiedades clave de las partes. Además, una descripción arquitectónica incluye información suficiente para permitir un análisis de alto nivel y una valoración crítica. La arquitectura de software suele desempeñar un papel clave como puente entre los requisitos y la implementación (véase la figura 1). Al proporcionar una descripción abstracta de un sistema, la arquitectura expone ciertas propiedades, mientras que oculta otras. La arquitectura de software puede jugar un papel importante en al menos seis aspectos del desarrollo de software[7].

Requirements

Code

Software Architecture

Fig. 1. Software Architecture as a Bridge

- Entendimiento: La arquitectura de software simpliﬁca nuestra capacidad para comprender grandes sistemas presentándolos a un nivel de abstracción en el que el diseño de alto nivel de un sistema puede ser fácilmente comprendido.

- Reutilizar: Las descripciones arquitectónicas permiten la reutilización en múltiples niveles. El trabajo actual sobre la reutilización se centra generalmente en las librerias de componentes. El diseño arquitectónico soporta, además, tanto la reutilización de grandes componentes como de estructuras en las que se pueden integrar componentes.

- Construcción: Una descripción arquitectónica proporciona un plan parcial para el desarrollo indicando los principales componentes y dependencias entre ellos.

- Evolución: La arquitectura de software puede exponer las dimensiones a lo largo de las cuales se espera que evolucione un sistema. Al hacer explicito los "muros de carga" de un sistema, los responsables del sistema pueden comprender mejor las ramificaciones de los cambios y, por lo tanto, estimar con mayor precisión los costes de modiﬁcaciones.

- Análisis: Las descripciones arquitectónicas proporcionan nuevas oportunidades para el análisis, incluyendo la comprobación de la consistencia del sistema[8],[9], la conformidad con las restricciones impuestas por un estilo arquitectónico[10], la conformidad con los atributos de calidad[11], el análisis de dependencia[12], y el análisis de domain-specific para arquitecturas construidas en estilos de específicos[13].

- Gestión: La experiencia ha demostrado que los proyectos exitosos ven el logro de una arquitectura de software viable como un hito clave en un proceso de desarrollo de software industrial. La evaluación crítica de la arquitectura normalmente conduce a una comprensión mucho más clara de los requisitos, las estrategias de implementación y los riesgos potenciales[14].

1. *Impedimentos comunes para lograr el éxito arquitectónico*

Entre los diversos factores que pueden llevar a arquitecturas fracasadas está la falta de aptitud suficiente y/o experiencia arquitectónica, en el tiempo adecuado dedicado a la señalización y análisis arquitectónico, la incapacidad de documentar y comunicar la arquitectura apropiadamente, y de darse cuenta de que "los estándares no son un sustituto de una arquitectura de software" para comprender plenamente la relación directa entre arquitectura e implementación, para actualizar la documentación a medida que la arquitectura evoluciona, etc.[15].

III. ARQUITECTURA ORIENTADA A SERVICIOS

SOA es un diseño para vincular recursos empresariales y computacionales (principalmente organizaciones, aplicaciones y datos) bajo demanda para lograr los resultados deseados para los consumidores de servicios (que pueden ser usuarios finales u otros servicios). OASIS[16] deﬁne SOA como el siguiente:

*Un paradigma para organizar y utilizar capacidades distribuidas que pueden estar bajo el control de diferentes dominios de propiedad. Proporciona un medio uniforme para ofrecer, descubrir, interactuar y utilizar las capacidades para producir los efectos deseados de acuerdo con condiciones previas y expectativas mensurables.*

La Arquitectura Orientada a Servicios (SOA) es una evolución de la Arquitectura Basada en Componentes, Diseño Basado en Interfaces (Orientado a Objetos) y Sistemas

Distribuidos de los años 90, como DCOM[17], CORBA3[18], J2EE4[19] e Internet en general. SOA no significa necesariamente Servicios Web como .NET, J2EE, CORBA o ebXML. En su lugar, se trata de implementaciones

SOA especializadas que incorporan los aspectos centrales de un enfoque de arquitectura orientado a servicios. Cada una de estas implementaciones extiende el Modelo de Referencia SOA básico.

Mientras que los servicios Web proporcionan soporte para muchos de los conceptos de SOA, no los implementan todos. Actualmente no apoyan el concepto de contrato de arrendamiento. Además, no hay una especificación oficial que proporcione niveles de QoS para un servicio. Una organización no puede implementar una arquitectura completa orientada a servicios dadas estas limitaciones con los servicios Web. Además, los consumidores de servicios pueden ejecutar servicios Web directamente si conocen la dirección y el contrato del servicio. No tienen que ir al registro para obtener esta información. Hoy en día, de hecho, la mayoría de las organizaciones implementan servicios Web sin un registro. En consecuencia, la medida en que una organización implementa una SOA con servicio Web varía enormemente.

1. *Los beneficios de la arquitectura orientada a servicios.*

Los principales impulsores de las arquitecturas basadas en SOA son facilitar el crecimiento manejable de los sistemas empresariales a gran escala, facilitar el aprovisionamiento y uso de servicios a escala de Internet y reducir los costes de la cooperación entre organizaciones. El valor de SOA es que proporciona un paradigma simple y escalable para organizar grandes redes de sistemas que requieren interoperabilidad para realizar el valor inherente a los componentes individuales. De hecho, SOA es escalable porque hace el menor número posible de suposiciones sobre la red y también minimiza cualquier suposición de confianza que a menudo se hace implícitamente en sistemas de menor escala. Un arquitecto que utiliza los principios de SOA está mejor equipado, por lo tanto, para desarrollar sistemas que sean escalables, evolutivos y manejables. Debería ser más fácil decidir cómo integrar la funcionalidad a través de las fronteras de la propiedad. Por ejemplo, una gran empresa que adquiere una empresa más pequeña debe determinar cómo integrar la infraestructura de TI adquirida en su cartera general de TI. A través de esta habilidad inherente para escalar y evolucionar, SOA permite un portafolio de TI que también es adaptable a las variadas necesidades del dominio de problemas o arquitectura de procesos específicos. La infraestructura que SOA fomenta es también más ágil y sensible que una construida sobre un número exponencial de interfaces de pares. Por lo tanto, SOA también puede proporcionar una base sólida para la agilidad y adaptabilidad del negocio.

1. *Detalles de SOA*

Antes de analizar los detalles de SOA, es importante que primero se explore el concepto de arquitectura de software, que consiste en las estructuras de “granularidad gruesa” del software. La arquitectura de software describe los componentes del sistema y la forma en que interactúan a un alto nivel. Estos componentes no son necesariamente “beans” de entidad u objetos distribuidos. Son módulos abstractos de software desplegados como una unidad en un servidor con otros componentes. Las interacciones entre los componentes se denominan conectores. La configuración de componentes y conectores describe la forma en que un sistema está estructurado y se comporta, como se muestra en la Figura 2.

|  |  |  |
| --- | --- | --- |
| Component | Connector | Component |
|  |

Fig. 2. Software architecture describes a system’s components and connec- tors.

La arquitectura orientada a servicios es un tipo especial de arquitectura de software que tiene varias características únicas. Es importante que los diseñadores y desarrolladores de servicios comprendan los conceptos de SOA, para que puedan hacer el uso más efectivo de los servicios Web en su entorno. SOA es un término relativamente nuevo, pero el término "servicio" en lo que se refiere a un servicio de software ha existido al menos desde principios de la década de 1990, cuando se utilizó en Tuxedo para describir "servicios" y "procesos de servicio"[20]. Recientemente ha habido más interés en la comunidad de desarrollo de software acerca de los conceptos detrás de SOA. La Figura 3 muestra que se pueden utilizar otras tecnologías para implementar una arquitectura orientada a servicios.

Jini

Web Services

Service-Oriented Architecture

Software Architecture

Fig. 3. Web services are one set of technologies for implementing service oriented architecture.

Los servicios web son simplemente un conjunto de tecnologías que pueden utilizarse para implementarlos con éxito.

El aspecto más importante de la arquitectura orientada a servicios es que separa la implementación del servicio de su interfaz. En otras palabras, separa el "qué" del "cómo". Los consumidores de servicios ven un servicio simplemente como un punto final que soporta un formato de solicitud o contrato en particular. No se preocupan por la forma en que el servicio ejecuta sus solicitudes; sólo esperan el resultado.

1. *Características de SOA*

La arquitectura de software de cada sistema reﬂeja los diferentes principios y el conjunto de compromisos utilizados por los diseñadores. La arquitectura de software orientada a servicios tiene las siguientes características:

* 1. *Descubrible y vinculado dinámicamente:* SOA admite el concepto de descubrimiento de servicios. Un consumidor de servicios que necesita un servicio descubre qué servicio usar en función de un conjunto de criterios en tiempo de ejecución. El consumidor del servicio solicita a un registro un servicio que satisfaga su necesidad.
  2. *Autonomo y Modular:* Los servicios son autónomos y modulares. Uno de los aspectos más importantes de SOA es el concepto de modularidad. Un servicio soporta un conjunto de interfaces. Estas interfaces deben ser cohesivas, refiriendose a que todas deben relacionarse entre sí en el contexto de un módulo. Los principios de modularidad deben respetarse al diseñar los servicios que soportan una aplicación para que los servicios se puedan agregar a una aplicación con algunas dependencias “bien conocidas”. Ya que este es un concepto tan importante cuando se estan creando servicios, se explicaran algunos de los principios de modularidad y en particular, como aplicarlos a la creación de servicios. Bertrand Meyer [22] describió los siguientes cinco criterios para determinar si un componente es lo suficientemente modular. Estos criterios se aplican igualmente para determinar si un servicio es suficientemente modular.

- Descomponibilidad modular: la descomponibilidad modular de un servicio se refiere a la división de una aplicación en muchos módulos más pequeños. Cada módulo es responsable de una función única y distinta dentro de una aplicación. Esto a veces se denomina "diseño de arriba hacia abajo", en el que los problemas más grandes se descomponen iterativamente en problemas más pequeños. El objetivo principal de la descomposición es la reutilización. El objetivo del diseño del servicio es identificar la unidad más pequeña de software que se puede reutilizar en diferentes contextos.

- Componibilidad modular: la componibilidad modular de un servicio se refiere a la producción de servicios de software que pueden combinarse libremente en su conjunto con otros servicios para producir nuevos sistemas. Los diseñadores de servicios deben crear servicios lo suficientemente independientes como para reutilizarlos en aplicaciones completamente diferentes de las destinadas originalmente. Esto a veces se denomina diseño de abajo hacia arriba.

- Comprensibilidad modular: la comprensibilidad modular de un servicio es la capacidad de una persona para comprender la función del servicio sin tener ningún conocimiento de otros servicios. Por ejemplo, si una aplicación bancaria implementa un servicio de cuenta corriente que no implementa una función de depósito, sino que depende del cliente para usar un servicio de depósito separado; esto restaría valor a la comprensibilidad modular del servicio.

- Continuidad modular: la continuidad modular de un servicio se refiere al impacto de un cambio en un servicio que requiere un cambio en otros servicios o en los consumidores del servicio. Una interfaz que no oculta suficientemente los detalles de implementación del servicio crea un efecto dominó cuando se necesitan cambios. Requerirá cambios a otros servicios y aplicaciones que usan el servicio cuando la implementación interna del servicio cambia. Cada servicio debe ocultar información sobre su diseño interno. Un servicio que expone esta información limitará su continuidad modular, porque una decisión de diseño interno se expone a través de la interfaz.

- Protección modular: la protección modular de un servicio es suficiente si una condición anormal en el servicio no afecta a otros servicios o consumidores. Por ejemplo, si un error en el servicio de cuenta corriente hace que se almacenen datos no válidos en una base de datos, esto podría afectar el funcionamiento de otros servicios que usan las mismas tablas para sus datos. Las fallas en la operación de un servicio no deben afectar la operación de un cliente u otro servicio o el estado de sus datos internos o de otra manera romper el contrato con los consumidores del servicio.

* 1. *Interoperabilidad:* la arquitectura orientada a servicios enfatiza la interoperabilidad, la capacidad de los sistemas que utilizan diferentes plataformas e idiomas para comunicarse entre sí. Cada servicio prove una interfaz que puede ser ser invocada a traves de un tipo de conector. Un conector interoperable consiste en un protocolo y un formato de datos que cada uno de los clientes potenciales del servicio entiende. La interoperabildad se logra al admitir el protocolo y los formatos de datos de los clientes actuales y potenciales del servicio. Las técnicas para admitir protocolos estándar y formatos de datos consisten en asignar las características y el lenguaje de cada plataforma a una especificación de mediación. La especificación mediadora asigna entre los formatos del formato de datos interoperables a los formatos de datos específicos de la plataforma. A veces, esto requiere conjuntos de caracteres de mapeo como ASCII a EBCDIC, así como tipos de datos de mapeo. Por ejemplo, un servicio web es una especificación de mediación para la comunicación entre sistemas. JAX-RPC y JAXM asignan tipos de datos Java a SOAP. Otras plataformas que admiten servicios web median entre las especificaciones del servicio web y sus propias especificaciones internas para conjuntos de caracteres y tipos de datos.
  2. *Perdida de acomplamiento:* el acoplamiento se refiere al número de dependencias entre módulos. Hay dos tipos de acoplamiento: libres y estrechos. Los módulos acoplados libremente tienen algunas dependencias bien conocidas. Los módulos estrechamente acoplados tienen muchas dependencias desconocidas. Cada arquitectura de software se esfuerza por lograr un acoplamiento flexible entre los módulos. La arquitectura orientada a servicios promueve un acoplamiento flexible entre los consumidores de servicios y los proveedores de servicios y la idea de algunas dependencias bien conocidas entre consumidores y proveedores.
  3. *Transparencia de ubicación:* la transparencia de ubicación es una característica clave de la arquitectura orientada a servicios. Los consumidores de un servicio no conocen la ubicación de un servicio hasta que lo encuentran en el registro. La búsqueda y el enlace dinámico a un servicio en tiempo de ejecución permite que la implementación del servicio se mueva de una ubicación a otra sin el conocimiento del cliente. La capacidad de mover servicios mejora la disponibilidad y el rendimiento del servicio. Al emplear un equilibrador de carga que reenvía las solicitudes a varias instancias de servicio sin el conocimiento del cliente del servicio, podemos lograr una mayor disponibilidad y rendimiento.
  4. Componibilidad: la componibilidad de un servicio está relacionada con su estructura modular. La estructura modular permite que los servicios se ensamblen en aplicaciones que el desarrollador no tenía noción al diseñar el servicio. El uso de servicios probados y preexistentes mejora en gran medida la calidad de un sistema y su retorno de la inversión debido a la facilidad de reutilización. Un servicio puede estar compuesto de tres maneras: composición de la aplicación, federaciones de servicios y orquestación de servicios.

- Una aplicación: generalmente es un conjunto de servicios, componentes y lógica de aplicación que une estas funciones para un propósito específico.

- Federaciones de servicios: son colecciones de servicios administrados juntos en un dominio de servicio más grande. Por ejemplo, un servicio de cuenta corriente, un servicio de cuenta de ahorro y un servicio al cliente pueden integrarse en un servicio de cuenta bancaria más grande.

-Orquestación de servicios: es la ejecución de una sola transacción que impacta uno o más servicios en una organización. A veces se le llama un proceso comercial. Consiste en múltiples pasos, cada uno de los cuales es una invocación de servicio. Si alguna de las invocaciones de servicio falla, la transacción completa debe revertirse al estado que existía antes de la ejecución de la transacción.

Para que un servicio se componga en una aplicación transaccional, federación u orquestación, los métodos de servicio deben ser subtransaccionales. Es decir, no deben realizar confirmaciones de datos ellos mismos.

* 1. Autocuración: con el tamaño y la complejidad de las aplicaciones distribuidas modernas, la capacidad de un sistema para recuperarse de un error es cada vez más importante. Un sistema de autocuración es aquel que tiene la capacidad de recuperarse de errores sin intervención humana durante la ejecución.

1. Fiabilidad

La fiabilidad en realidad mide qué tan bien funciona un sistema en presencia de perturbaciones. En la arquitectura orientada a servicios, los servicios estarán arriba y abajo de vez en cuando. Esto es especialmente cierto para aplicaciones ensambladas a partir de servicios de múltiples organizaciones a través de Internet. La medida en que un sistema se autocura depende de varios factores. La confiabilidad depende de la capacidad del hardware para recuperarse de una falla. La red también debe permitir la conexión dinámica a diferentes sistemas en tiempo de ejecución. Los protocolos modernos de redes de Internet proporcionan inherentemente esta capacidad.

1. Implementaciones conocidas

Si bien la naturaleza de los servicios en sí mismos puede variar, un estándar común para declarar un servicio es deseable cuando se construye una infraestructura. Actualmente existen dos de estos estándares: W3C’s Web Services Description Language (WSDL) and ebXML’s Collaboration Protocol Proﬁle.La versión 2.0 de WSDL es impresionante en su integridad y facilidad de implementación; sin embargo, solo cubre los aspectos básicos de la descripción del servicio. ebXML es una iniciativa conjunta de SOA entre UN / CEFACT5 [23] y OASIS [16]. Además de proporcionar componentes técnicos, el perfil del protocolo de colaboración se desarrolló para satisfacer las necesidades específicas de los negocios electrónicos que implican interacciones orientadas a servicios entre empresas legales.

1. CONCLUSIÓN

La arquitectura del software está emergiendo como disciplina en los últimos diez años [7]. La arquitectura del software del sistema describe sus estructuras y propiedades de grano grueso a un alto nivel. Mientras la tecnología se encargue de estas estructuras y propiedades, la tecnología se puede utilizar para la implementación de la arquitectura. Por ejemplo, Jini es una tecnología que admite arquitectura orientada a servicios, debido a las propiedades de SOA. Es importante para la aplicación de los conceptos de arquitectura de software de una nueva tecnología aprovecharla al máximo. La arquitectura orientada a servicios se implementa mediante servicios web y otras tecnologías, pero los términos y los conceptos han ganado popularidad recientemente como resultado de los servicios web. Por ejemplo, la industria informática es el término utilizado durante dos décadas para describir varias plataformas. Algunas características de SOA son compatibles con algunas mejores que otras tecnologías.