Princípios do Design de Serviços

Douglas Gabriel Gouveia[1], **Emanuel Batista da Silva Filho**[2]

[1]Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - Campus Cajazeiras (IFPB)
Cajazeiras – PB – Brazil

douglasxgabriel@gmail.com, emanuelbatista2011@gmail.com

Abstract. *This article describes the concept of services in computing, along exposing the definition of service-oriented architecture (SOA), but showing the advantages and disadvantages and the interaction between the layers of this architecture. And in order to have a better understanding of this architecture the paper reports a brief history of its emergence as a solution. Finally, explains the principles of service design advocated by Thomas Erl.*

Resumo. *Este artigo descreve o conceito sobre serviços na área da computação, juntamente expondo a definição da arquitetura orientada a serviços (SOA), contudo mostrando as vantagens e desvantagens e a interação entre as camadas dessa arquitetura. E para que se tenha um melhor entendimento sobre essa arquitetura o artigo relata um breve histórico do seu surgimento como solução. E por fim, explica os princípios de design de serviços defendidos por Thomas Erl.*

1. Serviços

Um serviço, basicamente, trata-se de um componente que disponibiliza uma funcionalidade de forma independente, ou seja, não dependendo do estado de outros, podendo ser acessado, na maioria das vezes, através de um Web Service de forma que, diferentes aplicações, independentemente de tecnologia, consigam interagir com o mesmo. Para isto, deve disponibilizar uma interface que defina as possíveis operações suportadas, tornando viável esta interação. Web Services são, portanto, aplicações que disponibilizam um serviço ou um conjunto deles, conectados através de um barramento de serviços, acessados através de suas respectivas interfaces. Algumas funcionalidades que podem ser citadas como exemplo do uso deste tipo de software são a disponibilidade de dados presentes em um banco de dados que podem ser consultadas por outras aplicações a fim de preencher um formulário, consultar o extrato bancário de uma conta, e até mesmo reservar um bilhete de uma viagem de avião. Apesar de serviços serem softwares e, por isso, escritos em determinadas linguagens de programação, a interação com outras aplicações se dá através de um formato de dados universal, tais como XML, JSon, CSV e entre outros, que são de fácil processamento para a maioria das linguagens de programação. Então, é possível visualizar que as principais vantagens da adoção de softwares orientados a serviços é o fato da facilidade de comunicação entre aplicações distribuídas e a interoperabilidade entre sistemas de linguagem distinta. A seguir, abordamos como se dá a construção de tais sistemas.

2. Arquitetura Orientada a Serviços (SOA)

SOA é a sigla para Service-Oriented Architecture, ou seja, Arquitetura Orientada a Serviço e trata-se de um padrão arquitetural projetado para aplicações que possuem

componentes que disponibilizam serviços para outros componentes. A arquitetura orientada a serviço é uma evolução da computação distribuída baseada no paradigma de design requisição e resposta para aplicações síncronas e assíncronas. A lógica de negócio ou funções individuais da aplicação são modularizados e apresentados como serviços para aplicações consumidor/cliente. O ponto chave nessa arquitetura é que os serviços tem natureza fracamente acoplada, ou seja, a interface de serviço é independente da sua execução. Assim, os desenvolvedores de aplicativos podem construir aplicações, compondo um ou mais serviços sem saber a implementação subjacente dos mesmos.

2.1. Vantagens e Desvantagens

Essa arquitetura possui algumas vantagens, pois a mesma fornece uma maneira natural de modularizar sistemas complexos, integrando serviços de diferentes fornecedores independentes de plataforma e tecnologia, promove o acoplamento fraco, ajudando na interface com sistemas legados, aumentando a eficiência, reduzindo custo e tempo de desenvolvimento, melhorando a flexibilidade e escalabilidade já que vários serviços podem ser facilmente desenvolvidos a partir da integração das aplicações existentes, permite, ainda, uma redução dos custos associados à manutenção, interoperabilidade baseada em padrões entre os sistemas, fornece independência de localização para acessar os dados através de qualquer canal, tais como smartphones, tablets ou laptops e permite adotar uma abordagem incremental, o que ajuda a atender às demandas dos clientes mais rápido adicionando novos serviços em resposta a necessidades específicas de negócios. Porém a grande desvantagem em utilizar essa arquitetura é que a mesma é difícil de implementar a comunicação assíncrona entre as aplicações, é um desafio para implementar as respostas em tempo real ou alta transferências de dados, porque XML traz robustez, mas não velocidade (embora existam outras alternativas, como o formato JSON), tem inúmeras vulnerabilidades de segurança devido a aplicações e sistemas de compartilhamento de processo e envolve o gerenciamento de transação complexa em interações entre os sistemas logicamente separados.

2.2. As camadas do SOA

Para um melhor entendimento da arquitetura SOA, Tomas Erl, no livro SOA Design and Patterns, cria uma abstração dividindo a arquitetura em 4 tipos de perspectivas, como podemos acompanhar na figura 1.

Código para figura

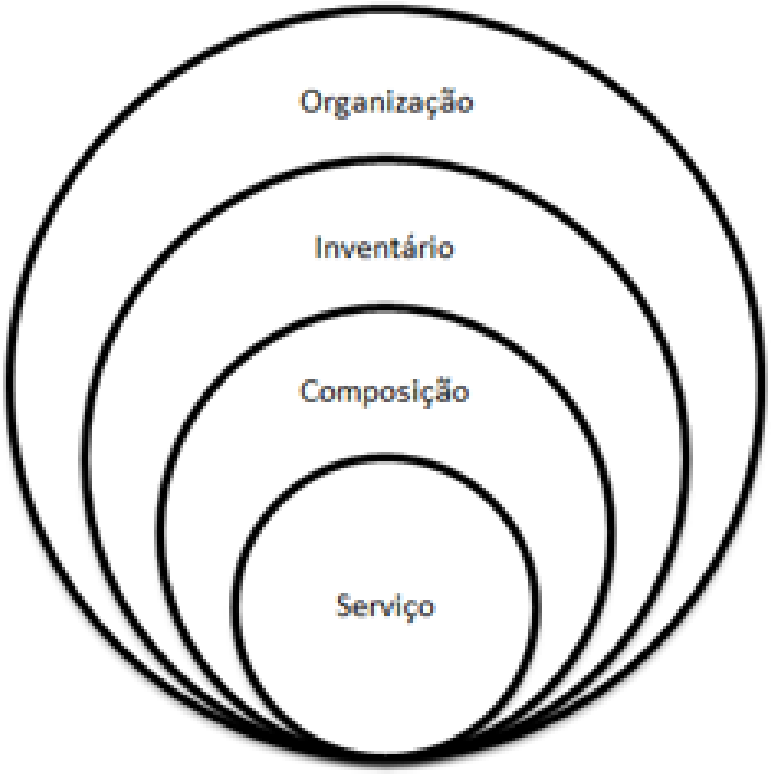

Figura 1. Camadas do SOA

Onde:

- A organização refere-se aos problemas e necessidades organizacionais reais aos quais o serviço propõe-se a resolver, isto é, assim como é válido para qualquer tipo de software desenvolvido, é necessário primeiramente procurar entender o cenário no qual o mesmo será inserido e quais problemas e necessidades serão atendidas com o seu desenvolvimento;
- A camada de inventário aborda a organização dos serviços que deverão ser desenvolvidos, mantidos e gerenciados pelo software que será desenvolvido, traçando um paralelo com softwares de outros paradigmas, o inventário seria a lista dos requisitos atendidos pelos componentes do sistema;
- Em serviços mais complexos é necessário ainda visualizar a possibilidade de o serviço atualmente em desenvolvimento utilizar-se de outros serviços desenvolvidos por terceiros, a esta perspectiva o autor dá o nome de composição;
- Na última camada está contido o próprio serviço, onde o mesmo é disponibilizado para que os softwares que possuam autorização, possam utilizar-se do recurso provido.

3. Histórico

O paradigma orientado à serviços passou a receber uma maior atenção em meados de 2005, onde diversos benefícios eram apontados em sua aplicação, tais como aumento do retorno no investimento em aplicações, uma maior agilidade e interoperabilidade organizacional, além de um melhor alinhamento entre negócio e T.I. (Tecnologia da

Informação). Diferentes fornecedores de plataformas SOA criaram guias expondo as definições e as boas práticas na hora de implementar aplicações orientadas à serviços. Um dos primeiros a elaborar um destes guias foi Don Box, da Microsoft, onde nele eram expostos quatro princípios que deveriam ser aplicados em soluções deste paradigma, são eles:

- Os limites são explícitos;
- Os serviços são autônomos;
- Os serviços compartilham esquemas e contratos, não classes;
- A compatibilidade dos serviços são baseados em políticas.

Posteriormente a IBM passou a publicar artigos reforçando os benefícios desta nova abordagem e a sua importância no mundo dos negócios, o que tornou o paradigma ainda mais relevante. Em seguida, Paul Allen publicou um livro sobre o assunto, onde tratava a orientação a serviços como um paradigma, de fato. Além disto, ainda apontou três componentes que julgou ser os principais destas aplicações, são eles:

- Arquitetura de negócio;
- SOA;
- Gestão orientada à software;

Além disto, elaborou também uma lista contendo sete princípios fundamentais para aplicações orientadas à serviços, que eram:

- Transparência;
- Ajuste ao cliente;
- Conectividade ao parceiro;
- Adaptação;
- Disponibilidade de multi-canais;
- Otimização;
- Disponibilizar vários serviços pertinentes ao cliente.

Sendo assim, é possível comprovar que no início os diferentes autores pensavam em serviços de forma um tanto quanto diferente, porém todos sabiam do seu potencial nas mais variadas esferas do desenvolvimento de software.

No cenário atual existem ainda algumas divergências em como os diferentes autores pensam a construção de aplicações orientadas à serviços, porém, devido à maior quantidade de material e uma maior experiência no uso destas soluções, hoje, vê-se um maior consenso entre autores, provedores de serviços e desenvolvedores tanto no conceito do paradigma, quanto aos guias de boas práticas.

4. Princípios do Design de Serviços

Como citado anteriormente, ainda há uma divergência no que diz respeito aos princípios que devem ser contemplados na elaboração de serviços, sendo possível encontrar autores que elencam uma quantidade variável de tópicos, porém, apesar desta diferença, alguns desses autores chegam a concordar em alguns destes tópicos.

Sendo assim, serão abordados aqui os princípios defendidos pelo autor Thomas Erl, em seus livros, totalizando oito pontos:

- Contrato;
- Baixo acoplamento;
- Abstração;
- Reusabilidade;
- Autonomia;
- Ausência de estados;
- Suporte para descoberta;
- Suporte para composição;

Abaixo, cada um destes tópicos será abordado com um maior detalhamento.

4.1. Contrato

Erl aponta o contrato como sendo o artefato principal da arquitetura do serviço, é nele que devem constar informações como requisitos, restrições técnicas, além de outras informações que o provedor do serviço deseja publicar, assim, o contrato trata-se da interface do serviço já citada anteriormente.

A título de exemplo, podemos citar Web Services baseados em SOAP (Simple Object Access Protocol) que utilizam-se de documentos XML para troca de informações e que têm sua interface descrita em um documento conhecido como WSLD (Web Service Definition Language) que também utiliza-se do formato XML.

Entretanto, serviços baseados em REST nem sempre disponibilizam tais informações, uma vez que há um padrão já estabelecido para o desenvolvimento e consumo de tais serviços. Erl ainda salienta a importância de que, apesar de ser possível definir diversas interfaces para os diversos serviços do inventário, é recomendado que todos obedeçam um padrão.

4.2. Baixo Acoplamento

O baixo acoplamento em serviço é associado ao uso de contrato, pois o uso do mesmo isola a implementação do serviço e com isso consequentemente oculta a lógica de negócio utilizada para o desenvolvimento do serviço, as tecnologias utilizadas na sua criação e a sua própria implementação do serviço.

Erl enfatiza que o baixo acoplamento em serviço é garantido pelo isolamento da implementação de um serviço através de seu contrato.

4.3. Abstração

Assim como é importante divulgar informações úteis sobre os serviços que auxiliam no consumo dos mesmos, Erl ainda aborda a importância da não divulgação de informações, sobre o serviço, que não sejam necessárias para quem deseja consumir o serviço.

Tal recomendação se dá pelo fato de que, quanto mais informações divulgadas, o risco de causar um acoplamento indesejado aumenta.

4.4. reusabilidade

Uma das grandes vantagens da implementação de serviços é a possibilidade de reutilizar suas funcionalidades em mais de um contexto, porém, para isto, é necessário que o mesmo seja devidamente projetado e desenvolvido.

Segundo Erl, este ponto torna-se ainda mais importante pois priorizar a reutilização do serviço, implica em um cuidado maior com os demais pontos. Um serviço torna-se mais reutilizável a medida que tiver um menor acoplamento, maior abstração, autonomia e suporte a composição.

4.5. Autonomia

O uso de autonomia em serviço faz com que ele tenha a capacidade de funcionar de forma independente de outros serviços ou de outras implementações, ou seja, ele não possui a necessidade de aprovação ou acompanhamento realizada por terceiros. Erl afirma que quanto maior for a autonomia de um serviços em seu ambiente, mas fácil será a manutenção e a evolução. Assim, quanto for planejar um serviço, deve-se pensar em alternativas que diminuam suas dependências externas.

4.6. Suporte e Descoberta

Todo serviço deve possuir o suporte a descoberta, pois não teria nenhum sentido ter um serviço que ninguém soubesse que o mesmo está disponível para uso.

Segundo Erl, suporte a descoberta resulta na capacidade de determinar quais requisitos de automação devem-se ser atendidos para que se possa utilizar um serviço que exista em um inventário. O uso de suporte a descoberta trás a vantagem de um serviço ser utilizado sem que seja necessário o envolvimento direto entre os times de desenvolvimento.

Garantir que um serviço possa ser "descoberto" evidencia o compromisso com a reusabilidade. Portanto, quando for planejar um serviço, garanta que ele possa ser descoberto.

4.7. Suporte Para Composição

Como citado anteriormente, a preocupação com a capacidade de utilização de serviços de terceiros para construir aplicações mais robustas é uma das visões da arquitetura SOA, assim, é um dos princípios da construção de serviços que os mesmo sejam projetados de forma que os possíveis consumidores possam, de forma facilitada, combinar o serviço desenvolvido com quaisquer outros a fim de resolver problemas mais complexos.

5. Conclusão

Atualmente a discussão sobre o desenvolvimento de softwares baseados no paradigma de serviços encontra-se bastante aquecido, sendo possível encontrar opiniões que defendem uma adoção tão ampla quanto a que encontramos no paradigma orientado a objetos, porém, faz-se necessário o estudo das melhores formas de projeto destes softwares para que possam atender as expectativas desejadas. Hoje, com a adoção desta solução em diversas aplicações há uma maior base de princípios e boas práticas bastantes difundidas, facilitando a vida de desenvolvedores que são novos neste paradigma. Além disto, já é possível ver com mais clareza os diversos benefícios desta abordagem, sendo que os mais notáveis são a interoperabilidade de aplicações de diferentes tecnologias e a facilidade na implementação de sistemas distribuídos.

Referências

[Erl 2005] Erl, T. (2005). *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*. Prentice Hall.

[Erl 2007] Erl, T. (2007). *SOA: Principles of Service Design*. Prentice Hall, 1st edition.

[JR. 2011] JR., E. (2011). Quatro tipos de soa. Accessada Jan. 2015.