Padrões de Design de Serviços

João Marcos Ferreira[1], **Kleonte Gomes**[2], **Rafael Tavares Rufino**[3]

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

joaomarccos.ads@gmail.com, kleontesousa@gmail.com, rafaeltavares.rofino198@gmail.com

Abstract. *The objective of this paper is to present the main concepts of a Service-Oriented Architecture (SOA) and its principles, well as Foundational Service Patterns presented by Thomas Erl in his book [Erl 2008], which are applied in solving common problems encountered in the development of systems based on SOA. The union of principles and design patterns, form a core set of good practices to any project.*

Resumo. *O objetivo desse artigo é apresentar os principais conceitos de uma Arquitetura orientada a serviços (SOA) e seus seus princípios, bem como padrões de serviços fundamentais apresentados por Thomas Erl em seu livro [Erl 2008], que são aplicados na solução de problemas comuns encontrados no desenvolvimento de sistemas baseados em SOA. A união de princípios e padrões de design, formam um conjunto de boas práticas essencial a qualquer projeto.*

1. O Que é SOA?

Arquitetura orientada a serviços ou SOA (Service-Oriented Architecture) é descrita de forma diferente por vários autores, alguns o vêem como um estilo técnico de arquitetura que fornece os meios para integrar sistemas distintos e expor as funções de negócios reutilizáveis. Outros autores, no entanto, tem uma visão muito mais ampla:

SOA é uma abordagem arquitetural corporativa que permite a criação de serviços de negócio interoperáveis que podem facilmente ser reutilizados e compartilhados entre aplicações e empresas.

A arquitetura orientada a serviços é um estilo de design que orienta todos os aspectos da criação e utilização de serviços de negócios em todo o seu ciclo de vida (desde a concepção à aposentadoria). [Newcomer and Lomow 2005]

Arquitetura Orientada a Serviços (SOA) é um paradigma para organização e utilização de capacidades distribuídas que podem estar sob controle de diferentes domínios proprietários. [OASIS 2006]

De maneira simples, SOA é uma abordagem de negócios para criar sistemas de TI (Tecnologia de Informação) que permitem alavancar recursos existentes, criar novos recursos e, principalmente, estar preparado para inevitáveis alterações exigidas pelo mercado, obtendo mais produtividade e lucro para a empresa.

2. Princípios de Design

Os Princípios de design são uma representação do paradigma de design orientado a serviços, eles estipulam boas praticas para construção de um sistema baseado em SOA

bem implementado. A seguir estão os princípios de design de serviços listados por [Erl 2009]:

- Service Abstraction
- Service Autonomy
- Service Composability
- Service Discoverability
- Service Loose Coupling
- Service Reusability
- Service Statelessness
- Service-Orientation and Interoperability
- Standardized Service Contract

Especificamente, a potencial relação entre princípios e padrões de projeto orientados a serviço pode ser resumido da seguinte forma:

Princípios de design são aplicados em conjunto para uma solução lógica, a fim de moldá-la de tal maneira que ela promova características chaves de design que suportem os objetivos estratégicos associados à computação orientada a serviços. Os padrões de design fornecem soluções para problemas comuns encontrados ao aplicar os princípios de design e ao estabelecer um ambiente adequado para a execução lógica concebida de acordo com os princípios de orientação a serviços.

3. Padrões de Design

Em [Erl 2008], são apresentados diversos padrões, contudo serão descritos nesse artigo apenas os chamados padrões de serviços fundamentais, pois esses constituem um conjunto de padrões básicos de design que ajudam a estabelecer as características fundamentais do projeto de serviço através de uma sequência de aplicação sugerida. Coletivamente, esses padrões formam a aplicação mais básica de orientação a serviço.

3.1. Agnostic Capability

Funções de serviços derivadas de interesses específicos podem não ser uteis para múltiplos consumidores, assim reduzem a potencial reusabilidade do serviço.

Para resolver este problema, o *Agnostic Capability* recomenda particionar a lógica do serviço em um conjunto de capacidades bem definidas que que abordam preocupações comuns não específicos de um problema.

Através da aplicação deste padrão, a lógica de serviço são agrupadas dentro de um contexto determinado serviço, disponibilizado como um conjunto de recursos bem definidos e complementares. Veja a figura 1.

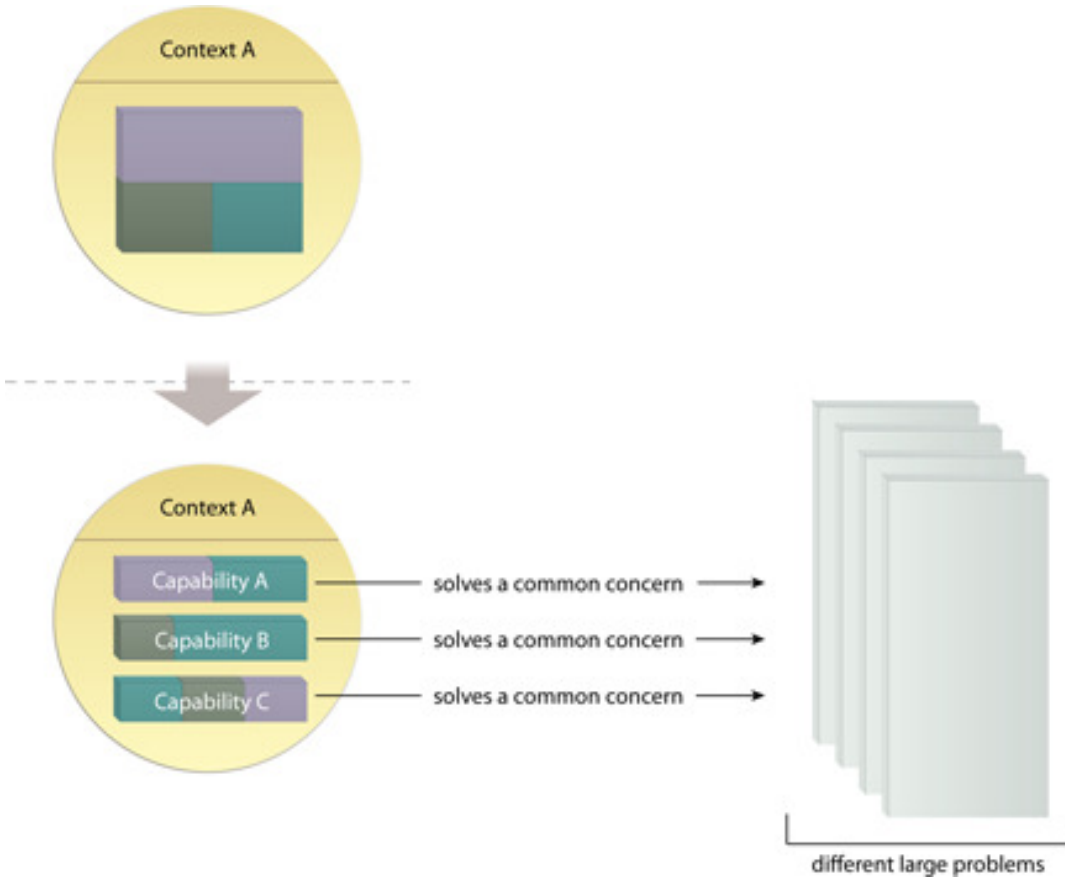

Figura 1. Agnostic Capability

3.2. Agnostic Context

Muita lógica agrupada com um único propósito resultam em serviços com pouco ou nenhum potencial de reutilização, acabando por gerar redundância dentro de uma empresa.

Para este tipo de problema, aqui é sugerido que a lógica que não é específica para uma finalidade seja isolada em serviços separados com contextos diferentes. Isto torna a solução reutilizável em um nível empresarial. Para entendermos melhor, vejamos a figura 2.

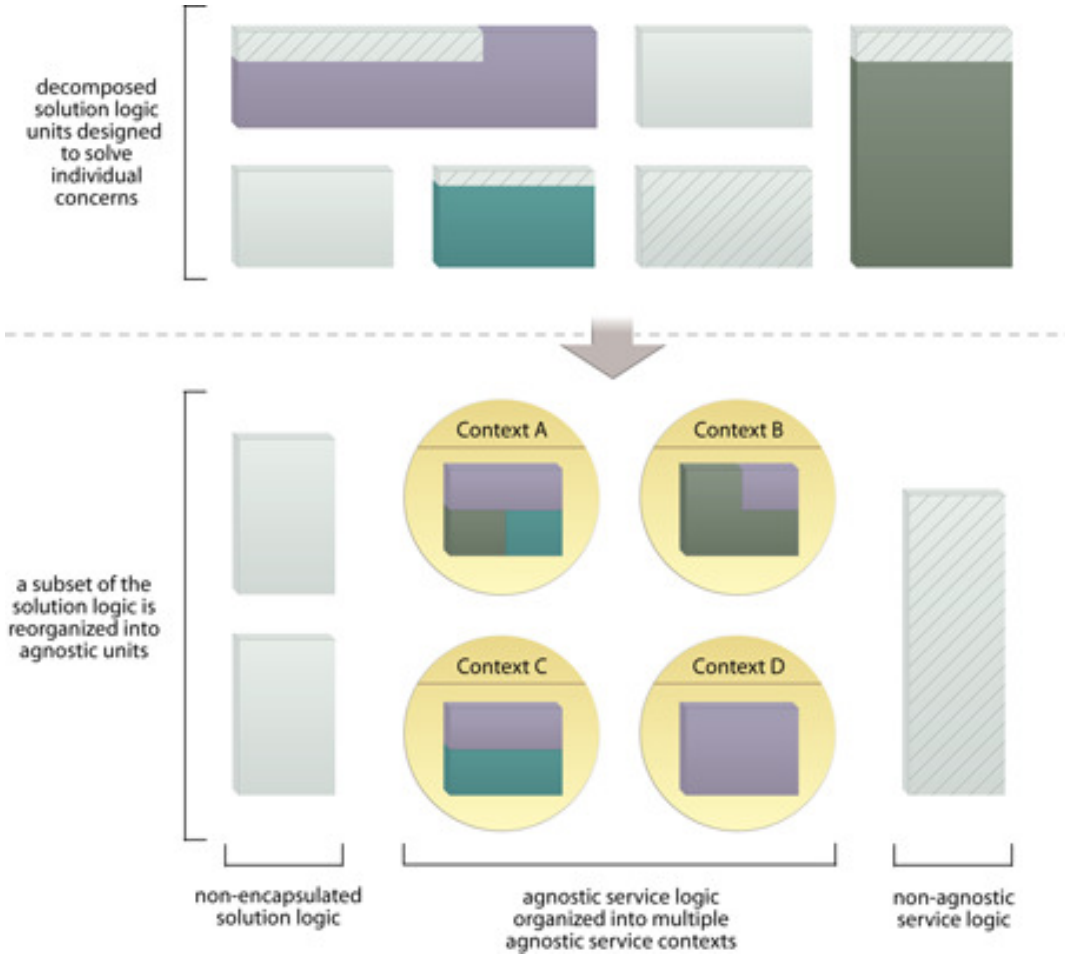

Figura 2. Agnostic Context

A aplicação desse padrão resulta em um subconjunto da solução lógica e em seguida são distribuídos em serviços com contextos específicos.

3.3. Functional Decomposition

Para resolver grandes e complexos problemas uma lógica também complexa deve ser criada, isto resulta em um serviço autônomo com restrições quanto sua reutilização.

Para evitar este tipo de problema o padrão *Funcional Decomposition* sugere que quebremos este grande problema em um conjunto de problemas menores. Isto permite que as soluções também seja decompostas em um conjunto de soluções mais simples.

Assim como na computação distribuída que baseia-se em uma abordagem onde um grande problema é decomposto, este padrão resulta na decomposição do problema maior em problemas menores, como mostra figura abaixo.

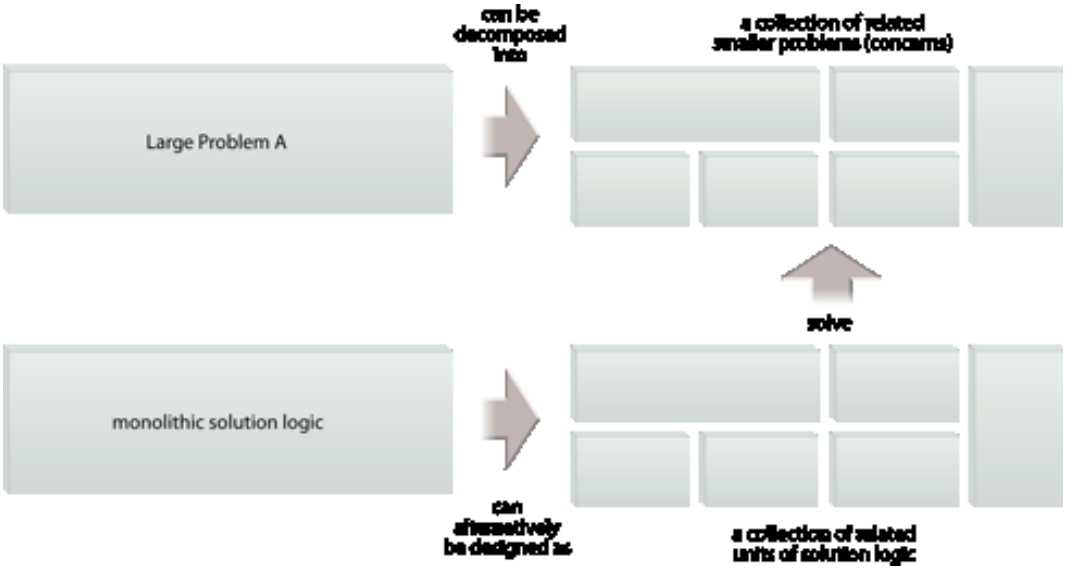

Figura 3. Functional Decomposition

3.4. Non-Agnostic Context

Quando se trata do padrão Non-Agnostic Context, ele foca na abstração e na posição da lógica de solução que seja agnóstica para tarefas e processos de negócios que resultam na lógica não-agnóstica sendo filtrada e rebaixada para encapsulação dentro dos sistemas que não fazem parte da relação dos serviços. Com isso, não se é aplicada à solução lógica não-agnóstica, a computação orientada a serviços, pois pode comprometer a qualidade das composições do serviço que a lógica pode ser responsável por controlar.

Uma forma de se contornar esse problema seria encapsular a solução lógica não-agnóstica adequada por um serviço que seja correspondente ao contexto funcional não-agnóstica. Um outro benefício seria que a lógica como o serviço está mais disponível para qualquer envolvimento inesperado nas composições do serviço.

Em combinação com o Process Abstraction, este padrão é aplicado para se estabelecer uma camada de serviço padrão, e não há uma forma exata de o padrão ser colocado antes ou depois do contexto agnóstico. De toda a forma, o resultado final é que tudo que se considera adequado para encapsulação do serviço acaba organizado em um conjunto de contextos de serviço bem definidos.

Pelo fato de precisar ser aplicado a uma solução lógica básica de um serviço não-agnóstico, a computação orientada a serviços consumirá muito mais tempo e será mais caro que se existisse em um programa paralelo à relação de serviço. Quando se estuda o contexto não-agnóstico, vale salientar que ele é aplicado de forma subsequente ao Service Encapsulation.

De forma que os tipos de serviços que mais geralmente este padrão seriam aqueles baseados nos modelos de serviço centrado em tarefas. A natureza de finalidade única da lógica encapsulada por serviços baseados em contextos não-agnósticos é associada de forma geral com a lógica da composição necessária para automatizar uma tarefa de negócio.

3.5. Service Encapsulation

Discorrendo sobre o Service Encapsulation, podemos citar que muitos sistemas desenvolvidos no passado foram construídos na forma de um grupo de sistemas relacionados representando como um corpo maior e dividido de uma solução lógica poderia continuar a existir dentro de um contorno de aplicação isolado, de forma que para que se aumentasse a escalabilidade, melhorasse a segurança, aumentasse a confiabilidade, utilizasse o reuso, a solução lógica foi dividida em unidades menores.

Vale mencionar que a solução lógica adequada para a classificação como um recurso corporativo pode ser encapsulado por e exposto como um serviço. Quanto à sua aplicação, faz-se necessário identificar e filtrar a solução lógica que seja adequada como um recurso corporativo.

Para a solução lógica encapsulada se tornar um membro efetivo de uma relação de serviço, faz-se necessário ser modelada por outros padrões e princípios para seja designada para estar de acordo com as metas estratégicas associadas com a computação orientada a serviços. Portanto é necessário um sólido conhecimento do paradigma de design orientado a serviços a fim de melhor determinar quando a lógica é ou não adequada para o encapsulamento do serviço. Com isso, como esta lógica é determinada é baseada na metodologia utilizada e na maturidade na relação do serviço existente.

Não há impacto imediato por conta da aplicação do padrão resultar na filtragem e identificação da lógica, todavia deve-se notar que sua aplicação é limitada ao processo de filtragem apenas. Portanto, a sequência desse padrão tem que ser parte de uma análise maior que abrange a modelagem da solução lógica.

Seguindo essa linha, qualquer lógica considerada adequada para encapsulamento do serviço é agrupada em serviços de única ou multifinalidades. A identificação da lógica adequada para o encapsulamento do serviço e o agrupamento e distribuição daquela lógica dentro de contextos funcionais distintos estabelece contornos fundamentais de serviço. Para definir o contorno mais adequado para um serviço, é necessário que se estabeleça o contexto funcional mais adequado.

4. Considerações Finais

Este artigo apresentou conceitos básicos e fundamentais para o universo de SOA e algumas soluções para problemas comuns encontrados na aplicação de princípios de design, vale ressaltar a importância que uma aplicação adequada com esses princípios e padrões de design tem para construção de sistemas que atendam as características relativas ao paradigma de orientação a serviço, os padrões aqui apresentados são uma pequena parte de um conjunto enorme de padrões que podem ser usados em SOA, quase em sua maioria, estes estão descritos de forma detalhada nos livros SOA Princípios de Design de Serviços e SOA Padrões de Design, ambos de Thomas Erl. Por fim é importante dizer que a visão de SOA traz um melhor entendimento de como pensar em software e como aproximar soluções tecnológicas de problemas reais.

Referências

[Baccaro 2013] Baccaro, M. (2013). 8 princípios de design para soa. Accessada JAN. 2016.

[Daigneau 2011] Daigneau, R. (2011). *Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services*. Addison-Wesley.

[Erl 2008] Erl, T. (2008). *SOA: Design Patterns*. Prentice hall.

[Erl 2009] Erl, T. (2009). *SOA: Principios de design de serviços*. Prentice hall Brasil.

[Group] Group, G. Service-oriented architecture (soa). Accessada JAN. 2016.

[Newcomer and Lomow 2005] Newcomer, E. and Lomow, G. (2005). Understanding soa with web services. Pearson Education.

[OASIS 2006] OASIS (2006). The oasis reference model for service oriented architecture 1.0. Accessada JAN. 2016.

[Oliveira] Oliveira, E. M. D. Vantagens e desvantagens de soa. Accessada JAN. 2016.

[Patterns] Patterns, S. Soa design patterns and design principles. Accessada JAN. 2016.

[Siqueira] Siqueira, B. R. Introdução a arquitetura orientada a serviços (soa). Accessada JAN. 2016.

[Wikipédia] Wikipédia. Service-oriented architecture. Accessada JAN. 2016.