UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CONTRATOS REST ROBUSTOS E LEVES: UMA ABORDAGEM EM DESIGN-BY-CONTRACT COM NEOIDL

LUCAS FERREIRA DE LIMA

ORIENTADOR: RODRIGO BONIFÁCIO DE ALMEIDA

DISSERTAÇÃO DE MESTRADO EM
ENGENHARIA ELÉTRICA

PUBLICAÇÃO: MTARH.DM - 017 A/99

BRASÍLIA/DF: JULHO - 2016.

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CONTRATOS REST ROBUSTOS E LEVES: UMA ABORDAGEM EM DESIGN-BY-CONTRACT COM NEOIDL

LUCAS FERREIRA DE LIMA

DISSERTAÇÃO DE MESTRADO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA.

APROVADA POR:

Profa. Rodrigo Bonifácio de Almeida, DSc. (ENE-UnB)
(Orientador)

Prof. XXX
(Examinador Interno)

YYY
(Examinador Externo)

BRASÍLIA/DF, 01 DE JULHO DE 2016.

DEDICATÓRIA

Este trabalho é dedicado a ...
continuação .....

AGRADECIMENTOS

(Página opcional) Agradeço .....
continuação .....

RESUMO

CONTRATOS REST ROBUSTOS E LEVES: UMA ABORDAGEM EM DESIGN-BY-CONTRACT COM NEOIDL

Autor: Lucas Ferreira de Lima
Orientador: Rodigo Bonifácio de Almeida
Programa de Pós-Graduação em Engenharia Elétrica
Brasília, julho de 2016

A adoção do paradigma arquiteturação baseado em serviços... O presente trabalho foi desenvolvido, ..., tendo como objetivo geral ..... Como objetivos específicos, o trabalho procurou .....

A proposta partiu...

Durante a realização...

Os resultados sugerem...

ABSTRACT

CONTRATOS REST ROBUSTOS E LEVES: UMA ABORDAGEM EM DESIGN-BY-CONTRACT COM NEOIDL

Autor: Lucas Ferreira de Lima
Orientador: Rodigo Bonifácio de Almeida
Programa de Pós-Graduação em Engenharia Elétrica
Brasília, julho de 2016

..

..

..

..

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIAÇÕES

SOC: Software Oriented Computing, modelo arquitetural baseado em serviços.

DbC: Design by Contract, mecanismos de garantias com condições na chamadas a métodos, funções, serviços, etc.

1 INTRODUÇÃO

A computação orientada a serviços (*Service-oriented computing, SOC)* tem se mostrado uma solução de *design* de *software* que favorece o alinhamento às mudanças constantes e urgentes nas instituições [1]. Nessa abordagem, os recursos de software são empacotados como serviços, os quais são módulos bem definidos e auto-contidos, provêem funcionalidades negociais e com estado e contexto independente [4].

Os benefícios de SOC estão diretamente relacionados ao baixo acoplamento dos serviços que compõem a solução, de forma que as partes (nesse caso serviços) possam ser substituídas e evoluídas facilmente, ou ainda rearranjadas em novas composições. Contudo, para que isso seja possível, é necessário que os serviços possuam contratos bem definidos e independentes da implementação.

A relação entre quem provê e quem consome o serviço se dá por meio de um contrato. O contrato de serviço é o documento que descreve os propósitos e as funcionalidades do serviço, como ocorre a troca de mensagens, condições sobre como as operações são realizadas e informações sobre as operações [2].

Nesse contexto, a qualidade da especificação do contrato é fundamental para o projeto de software baseado em SOC. Este trabalho de pesquisa aborda um aspecto importante para a melhoria da robustez de contratos de serviços: a construção de garantias mútuas por meio da especificação formal de contratos, agregando o conceito de Desing-by-Contract.

1.1 PROBLEMA DE PESQUISA

As linguagens de especificação de contratos para SOC apresentam algumas limitações. Por exemplo, a linguagem WSDL (*Web-services description language*) [**?**] é considerada uma solução verbosa que desestimula a abordagem *Contract First*. Por essa razão, especificações WSDL são usualmente derivadas a partir de anotações em código fonte *Code First*. Além disso, os conceitos descritos em contratos na linguagem WSDL não são diretamente mapeados aos elementos que compõem as interfaces do estilo

arquitetural REST (*Representational State Transfer*). Outras alternativas para REST, como Swagger e RAML[1], usam linguagens de propósito geral (em particular JSON e YAML) adaptadas para especificação de contratos. Ainda que façam uso de contratos mais sucintos que WSDL, essas linguagens não se beneficiam da clareza típica das linguagens específicas para esse fim (como IDLs CORBA) e não oferecem mecanismos semânticos de extensibilidade e modularidade.

Com o objetivo de mitigar esses problemas, a linguagem NeoIDLfoi proposta para simplificar a especificação de serviços REST com mecanismos de modularização, suporte a anotações, herança em tipos de dados definidos pelo desenvolvedor, e uma sintaxe simples e concisa semelhante às *Interface Description Languages* – IDLs – presentes em *Apache Thrift*™e CORBA™. Por outro lado, a NeoIDL, da mesma forma que WSDL, Swagger e RAML não oferece construções para especificação de contratos formais de comportamento como os presentes em linguagens que suportam DBC (*Design by Contract*) [3], como JML, Spec# e Eiffel. Em outras palavras, a NeoIDLadmite apenas contratos fracos (*weak contracts*), sem suporte a construções como pré e pós condições.

1.2 OBJETIVO GERAL

O objetivo geral de trabalho é investigar o uso de construções de Design by Contract no contexto de computação orientada a serviços, verificando a viabilidade e utilidade de sua adoção na especificação de contratos e implentação de serviços REST.

1.2.1 Objetivos específicos

1. Realizar análise empírica de expressividade e reuso da especificação de contratos em NeoIDLem comparação com *Swagger*, a partir de contratos reais do Exército Brasileiro.

2. Extender a sintaxe da NeoIDLpara admitir construções de Design by Contract, com pré e pós condições para operações de serviços REST.

3. Implementar um estudo de caso de geração de código em *Python Twisted* com suporte a Design by Contracta partir de contratos especificados em NeoIDL

[1]http://raml.org/spec.html

4. Coletar a percepção de desenvolvedores sobre a aceitação da especificação de contratos REST com Design by Contractna NeoIDL

1.2.2 JUSTIFICATIVA E RELEVÂNCIA

- aumento do uso de SOC - potencial do SOC está no baixo acomplamento dos serviços. - aumento do uso de REST - contratos fracos prejudicam a qualidade/aumentam os erros - baixa qualidade prejudica o reuso -

O uso do padrão REST para construção de *Web Service*é crescente no contexto do desenvolvimento de soluções basedas em serviço e não se dipõe de uma linguagem padrão para especificação de contratos fortes. A linguagem específica de domínio NeoIDL- foi desenvolvida para ser uma alternativa para especificação REST, caracterizada pela coesão e simplicidade em se compreender, mas que não dispunha de suporte a pré e pós condições.

1.2.3 ESTRUTURA

2 REFERENCIAL TEÓRICO

2.1 COMPUTAÇÃO ORIENTADA A SERVIÇO

...

2.1.1 Motivação

...

2.1.2 Modelo arquitetural

...

2.1.3 Princípios

...

2.1.3.1 Padrões de projeto

...

2.2 Web Services

...

2.2.1 SOAP (W3C)

...

2.2.1.1 Especificação de contratos

...

2.2.2 REST (Fielding)

A tecnologia REST é caracterizada principalmente pela convenção da adoção dos métodos do protocolo HTTP para definição das operações, transferência de estado nas requisições. - Ausência de padrão para especificação de contratos REST. - Fragilidade de garantias por quem consome o serviço; - Risco de a ausência de informações prejudicarem os consumidores.

2.2.2.1 Especificação de contratos

...

2.2.2.2 Outros padrões

...

2.3 Design-by-Contract

...

2.3.1 Origem

- Bertrand Meyer (eifel)

2.3.2 Implementações de DbC

- Eifel - JML - Spec#

2.4 NeoIDL

...

2.4.1 Objetivos

2.4.2 Arquiterura

2.4.3 Histórico

3 PROPOSTA

3.1 Serviços com Desing-by-Contract

Os benefícios esperados pela adoção da arquitetura orientada a serviços somente serão auferidos com a concepção adequada de cada serviço. Por essa razão, é necessário planejar o projeto dos serviços criteriosamente antes de lançar mão do desenvolvimento, com preocupação especial em garantir um nível aceitável de estabilidade aos consumidores de cada serviço. Nessa etapa do projeto de desenho da solução, a especificação do contrato do serviço (Web API) exerce uma função fundamental.

Na sociedade civil, contratos são meios de se formalizar acordo entre partes a fim de definir os direitos e deveres de cada parte e buscar atingir o objetivo esperado dentro de determinadas regras. Cada parte espera que as outras cumpram com suas obrigações. Por outro lado, sabe-se que o descumprimeto das obrigações costuma implicar de penalizações até o desfazimento do contrato.

Contratos entre serviços Web seguem em uma linha análoga. O desenho das capacidades (operações) e dos dados das mensagens correspondem aos termos do contrato no sentido do que o consumidor deve esperar do serviço provedor. Porém identificou-se, após ampla pesquisa realizada sobre o tema, que as linguagens disponíves para especificação de contratos atingem apenas esse nível de garantias. No contexto de web-services em REST, conforme descrito na seção 2.2.2, há ainda a ausência de padrão para especificação contratos, tal como ocorre com o WSDL adotado em SOAP.

A proposta deste trabalho é extender os níveis de garantias, de modo a promover um patamar adicional com obrigações mutuas entre os serviços (consumidor e provedor). Isso se dá para adoção do conceito de Design-by-Contract (debatido na seção 2.3) em que a execução da capacidade do serviço garantirá a execução, desde que satisfeitas as condições prévias. O detalhamento do processo é exposto nas seções que se seguem.

3.1.1 Modelo de operação

As garantias para execução dos serviços são estabelecidas em duas etapas: pré- e pós-condições. Nas pré-condições o provedor do serviço estabelece os requisitos para que

o serviço possa ser executado. A etapa de pós-condições tem o papel de validar se a mensagem de retorno do serviço possui resultados válidos.

O diagrama da Figura 3.1 descreve como ocorre a operação das pré- e pós-condições. O processo se inicia com a chamada à capacidade do serviço e a identificação da existência de uma pré-condição. Caso tenham sido estabelecidas pré-condições, essas são avaliadas. Caso alguma delas não tenham sido satisfeitas, o serviço principal não é processado e o provedor do serviço retornar o código de falha definido no contrato correspondente.

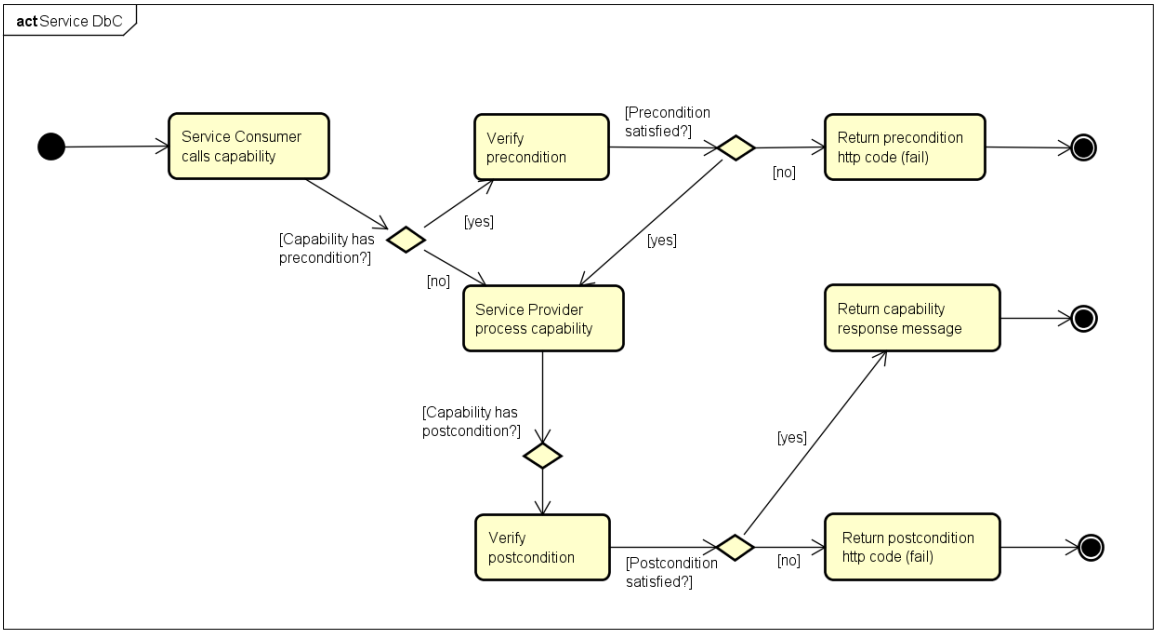

Figura 3.1: Digrama de atividades com verificação de pré e pós condições

Caso tenham sido definidas pós-condições, essas são acionadas após o processamento da capacidade, porém antes do retorno ao consumidor do serviço. Assim, conforme Figura 3.1, visando não entregar ao cliente uma mensagem ou situação incoerente, as pós-condições são validadas. Caso todas as pós-condições tenham sido satisfeitas, a mensagem de retorno é encaminhada ao cliente. Caso contrário, será retornado o código de falha.

3.1.1.1 Verificação das pré-condições

As pré-condições podem ser do tipo baseado nos parâmetros da requisição ou do tipo baseado na chamada a outro serviço. Denominamos, para o contexto desta dissertação, de básica a pré-condição baseada apenas nos parâmetros da requisição (atributos da chamada ao serviço). Nessa validação é direta, comparando os valores passados com os valores admitidos.

No caso das pré-condições baseadas em serviços, é realizada chamada a outro serviço para verificar se uma determinada condição é satisfeita. Este modo de funcionamento, que se assemelha a uma composição de serviço, é mais versátil, pois permite validações de condições complexas sem que a lógica associada seja conhecida pelo cliente. Assim, os contratos que estabelecem esse tipo de pré-condição se mantem simples.

A Figura 3.2 detalha as etapas de verificação de cada pré-condição. Nota-se que a saída para as situações de desatendimento às pré-condições, independentemente do tipo, é o mesmo. O objetivo desta abordagem é simplificar o tratametno de exceção no consumidor.

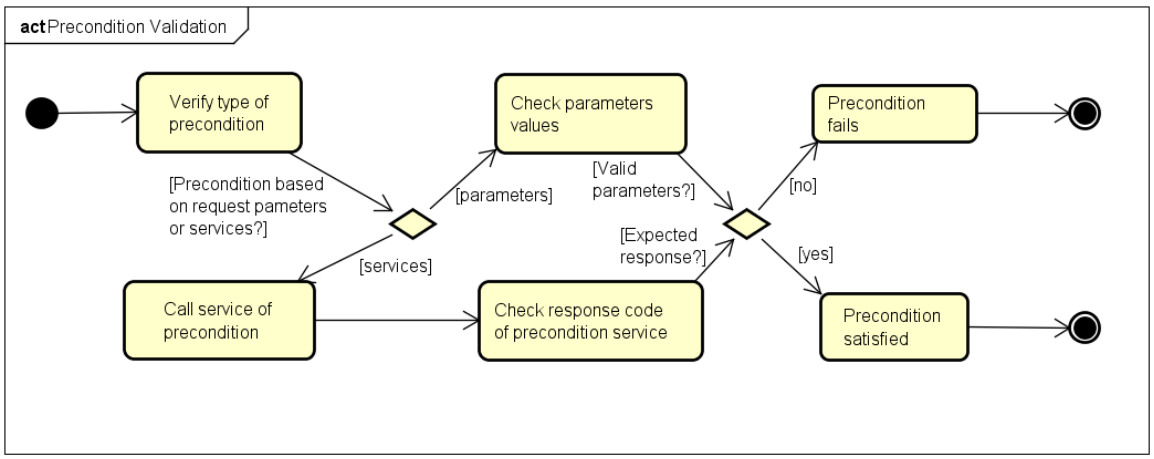

Figura 3.2: Diagrama de atividades do processamento da pré-condição

3.1.1.2 Verificação das pós-condições

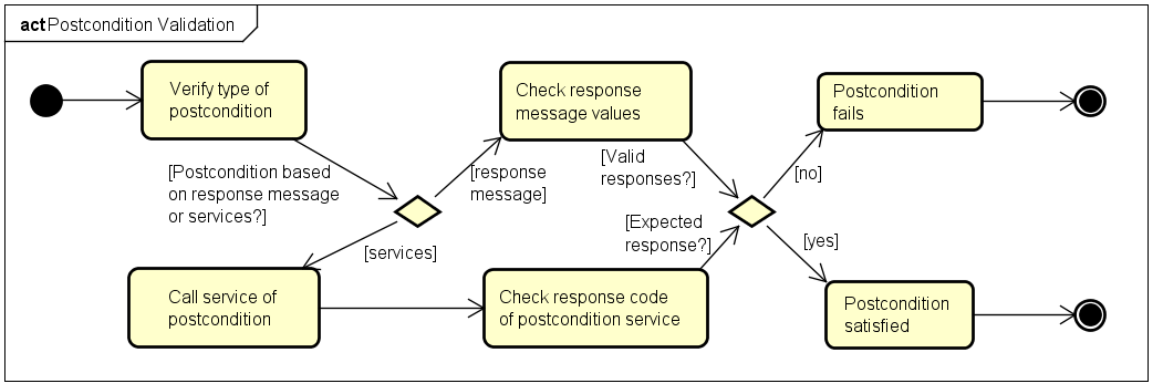

Figura 3.3: Diagrama de atividades do processamento da pós-condição

A verificação das pós-condições acontece de modo muito similar a das pré-condições. Há também os dois tipos, baseado em valores e em chamadas a outros serviços. O

diferencial está em que a validação dos valores passa a ocorrer a partir dos valores contidos na mensagem de retorno. A Figura 3.3 descreve as etapas necessárias para validação de cada pré-condição.

3.2 Implementação de Design-by-Contract na NeoIDL

3.2.1 Pré-condição básida

[language=NeoIDL,firstnumber=1]DBCsimple.neo

Figura 3.4: Exemplo da notação DBC básica na NeoIDL

4 ESTUDO DE CASO E AVALIAÇÃO

4.1 Geração de código com garantias de DbC

Nesta seção tratar da implementação de contratos com DbC e do pluggin para Twisted ...

4.2 Avaliação da NeoIDL com DbC

...

4.2.1 Utilidade da NeoIDL com DbC

* Questionário montado para avaliar a utilidade de DbC com NeoIDL

* Inicialmente motivado pelo estudo do Alessandro Garcia

* GQM e Avaliação TAM

* Montagem do questionário

1. Perfil técnico-profissional do respondente 1.1 Para qual órgão ou empresa você presta serviços atualmente? 1.2 A quanto tempo você trabalha com desenvolvimento Web 1.3 A quanto tempo você desenvolve com uso de APIs Web (Web Service) 1.4 Qual o seu nível de experiência com especificação de API REST 1.5 Qual o seu nível de experiência com especificação de contratos com Swagger

3 Questões sobre especificação e implementação de APIs Web 3.1 A especificação do contrato formalmente, seja em Swagger ou NeoIDL, em relação a descrição textual, aumentará meu nível de acerto na implementação (efetividade). 3.2 Identificar e compreender as operações e atributos na especificação Swagger é simples para mim. 3.3 Identificar e compreender as operações e atributos na especificação NeoIDL é simples para mim.

4. DbC 4.1 Conhecer previamente e explicitamente as pré-condições será útil para mim. (Useful) 4.2 Aprender a identificar as pré-condições na NeoIDL parece ser simples pra mim (Easy to learn) 4.3 Parece ser fácil para mim declarar uma pré-condição na NeoIDL (Clear and understandable) 4.4 Me lembrar da sintaxe da pré-condição na NeoIDL é fácil (Remember)

5. Geração de código 5.1 É claro e compreensível para mim o efeito da pré-condição sobre o código gerado (Controllable) 5.2 A geração do código de pré e pós-condições aumentará minha produtividade na implementação do serviço (Job performance) 5.3 Assumindo ter a disposição a NeoIDL no meu trabalho, para especificação de contratos e geração de código, eu presumo que a utilizarei regularmente no futuro. 5.4 Nesse mesmo contexto, eu vou preferir utilizar contratos escritos em NeoIDL do que descritos de outra forma

* Distribuição do questionário

* Avaliação dos resultados

* Ameaças - não foi fornecido nenhum material sobre a NeoIDL, apresentando somente o uma descrição de serviço - O questionário foi aplicado uma única vez, sem melhorias a partir do primeiro conjunto de respostas

* Questionários futuros

A principal questão de pesquisa a ser avaliada com o uso do questionário é a utilidade em se agregar ao design das especificações de serviços REST as garantias de pré e pós-condições. Em segundo momento, pressupondo a utilidade, avaliar se a NeoIDL cumpre satisfatoriamente com este propósito, agregando à sintaxe da linguagem a possibilidade de se expressar pré e pós-condições.

* Separar os respondentes em faixas de experiência. Verificar se as respostas dos menos experientes precisam ser descartadas pela pouca capacidade crítica. Separar a análise entre os respondentes que conhecem e os que não conhecem Swagger.

* Perspectivas de comparação a) Experiência com desenvolvimento com uso de REST b) Experiência com Swagger c) Utilidade da especificação formal de contratos d) Percepção da NeoIDL sem DbC

4.2.2 Outras avaliações

4.2.2.1 Expressividade

Nesta subseção falar da avaliação feita nos contratos do exército.

4.2.2.2 Potencial de reuso

Complementar a avaliação de expressividade com o potencial de reuso.

5 CONCLUSÕES E TRABALHOS RELACIONADOS

5.1 CONCLUSÕES GERAIS

A boa definição dos contratos ...

5.2 TRABALHOS RELACIONADOS E PESQUISAS FUTURAS

...

- Trabalhos relacionados com hipermedia

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Chen, H.-M. Towards service engineering: service orientation and business-it alignment. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*, pages 114–114. IEEE, 2008.

[2] Erl, T., Karmarkar, A., Walmsley, P., Haas, H., Yalcinalp, L. U., Liu, K., Orchard, D., Tost, A., e Pasley, J. *Web service contract design and versioning for SOA*. Prentice Hall, 2009.

[3] Meyer, B. Applying'design by contract'. *Computer*, 25(10):40–51, 1992.

[4] Papazoglou, M. P., Traverso, P., Dustdar, S., e Leymann, F. Service-oriented computing: State of the art and research challenges. *Computer*, (11):38–45, 2007.

APÊNDICES

A CONTRATO NEOIDL COM DBC

...