Contratos REST robustos e leves

uma abordagem em Design-by-Contract com NeoIDL

Lucas F. Lima[1]
Orientadores: Rodrigo Bonifácio[2], Edna Canedo[3]

[1]Departamento de Engenharia Elétrica – Universidade de Brasília – UnB

[2]Departamento de Ciência da Computação – Universidade de Brasília – UnB

[3]Faculdade UnB Gama – Universidade de Brasília – UnB
Brasília – DF – Brasil

WTDSoft 2015

UnB

O problema

O uso do padrão REST para construção de WebServices é crescente no contexto do desenvolvimento de soluções basedas em serviço.

REST não dispõe de uma linguagem padrão para especificação de contratos. Frente a isso, a NeoIDL (DSL) foi desenvolvida para ser uma alternativa, caracterizada pela coesão e simplicidade em se compreender.

Por outro lado, a NeoIDL suporta apenas contratos fracos (*weak contracts), sem suporte a construções como pré e pós condições.*

UnB

O problema

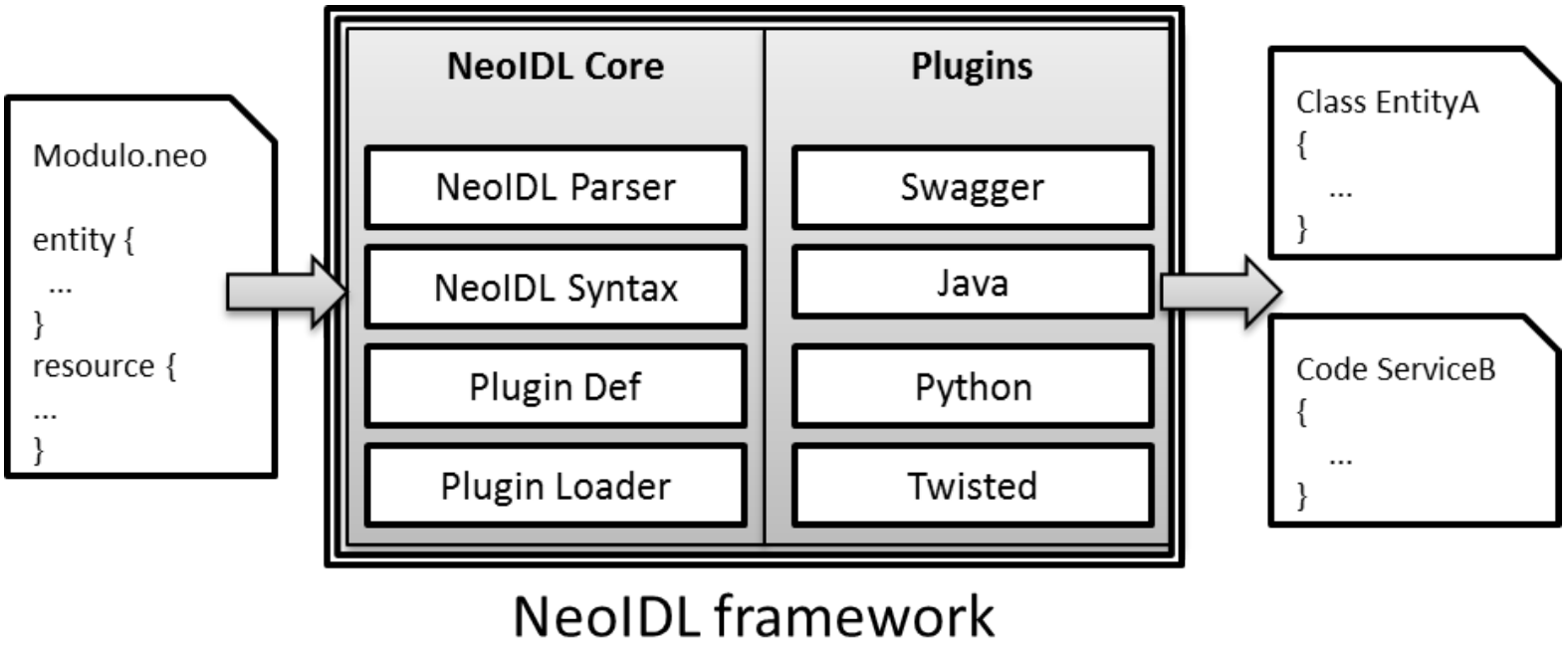
O uso do padrão REST para construção de WebServices é crescente no contexto do desenvolvimento de soluções basedas em serviço.

REST não dispõe de uma linguagem padrão para especificação de contratos. Frente a isso, a NeoIDL (DSL) foi desenvolvida para ser uma alternativa, caracterizada pela coesão e simplicidade em se compreender.

Por outro lado, a NeoIDL suporta apenas contratos fracos (*weak contracts), sem suporte a construções como pré e pós condições.*

UnB

NeoIDL - Arquitetura



UnB

As contribuições

- Introduzir o conceito de Design-by-Contract no contexto da especificação de contratos REST para Computação Orientada a Serviços
- Evolução da linguagem e *framework NeoIDL para suportar novas construções*
- *Avaliar a expressividade de contratos escritos na NeoIDL em contraponto a outra linguagem*
- *Conduzir estudo empírico para avaliar:*
  - *Aptidão da NeoIDL para reuso de entidades*
  - *Comprovar necessidades e benefícios do DbC em SOC*

UnB

As contribuições

- Introduzir o conceito de Design-by-Contract no contexto da especificação de contratos REST para Computação Orientada a Serviços
- Evolução da linguagem e *framework NeoIDL para suportar novas construções*
- *Avaliar a expressividade de contratos escritos na NeoIDL em contraponto a outra linguagem*
- *Conduzir estudo empírico para avaliar:*
  - *Aptidão da NeoIDL para reuso de entidades*
  - *Comprovar necessidades e benefícios do DbC em SOC*

UnB

As contribuições

- Introduzir o conceito de Design-by-Contract no contexto da especificação de contratos REST para Computação Orientada a Serviços
- Evolução da linguagem e *framework NeoIDL para suportar novas construções*
- *Avaliar a expressividade de contratos escritos na NeoIDL em contraponto a outra linguagem*
- *Conduzir estudo empírico para avaliar:*
  - *Aptidão da NeoIDL para reuso de entidades*
  - *Comprovar necessidades e benefícios do DbC em SOC*

UnB

As contribuições

- Introduzir o conceito de Design-by-Contract no contexto da especificação de contratos REST para Computação Orientada a Serviços
- Evolução da linguagem e *framework NeoIDL para suportar novas construções*
- *Avaliar a expressividade de contratos escritos na NeoIDL em contraponto a outra linguagem*
- *Conduzir estudo empírico para avaliar:*
  - *Aptidão da NeoIDL para reuso de entidades*
  - *Comprovar necessidades e benefícios do DbC em SOC*

UnB

NeoIDL - exemplo de especificação

```
1   resource characters  {
2       path = "/v1/public/characters";
3
4        /**
5        * @ summary: Fetches lists of comic characters.
6        * @ params:
7        *        query Character.name: characters matching the specified name
8        *        query Character.story: characters matching the specified story
9        *        query limit: limit the result set
10       * @ throws
11       *        409  Limit greater than 100.
12       *        409  Limit invalid or below 1.
13       */
14       @get     [Character] getCharactersList ( string Character.name, string Character.story,
15                                                int limit);
16
17   };
```

getCharactersList - Marvel - especificado em NeoIDL

UnB

Design-by-contract - Contrato fraco e forte

Contrato fraco	
Pré condição:	True
Pós condição:	if empty(stack) then thrown emptyException else return top (stack)

Contrato forte	
Pré condição:	not empty(stack)
Pós condição:	return top (stack)

Operação top()

UnB

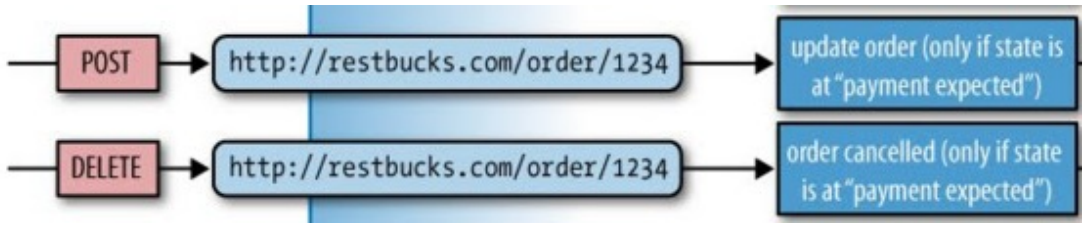
NeoIDL - exemplo de especificação com uso de DbC

```
1  resource characters  {
2       path = "/v1/public/characters";
3
4        /**
5        * @ summary: Fetches lists of comic characters.
6        * @ params:
7        *        query Character.name: characters matching the specified name
8        *        query Character.story: characters matching the specified story
9        *        query limit: limit the result set
10       * @ throws
11       *        409  Limit greater than 100.
12       *        409  Limit invalid or below 1.
13       */
14       @get     [Character] getCharactersList ( string Character.name, string Character.story,
15                                                int limit)
16            require ((limit > 0 and limit <= 100)),
17            ensure (RespCode = 200),
18            otherwise 409;
19
20   };
```

getCharactersList - Marvel - especificado em NeoIDL

UnB

DbC - outro exemplo da literatura

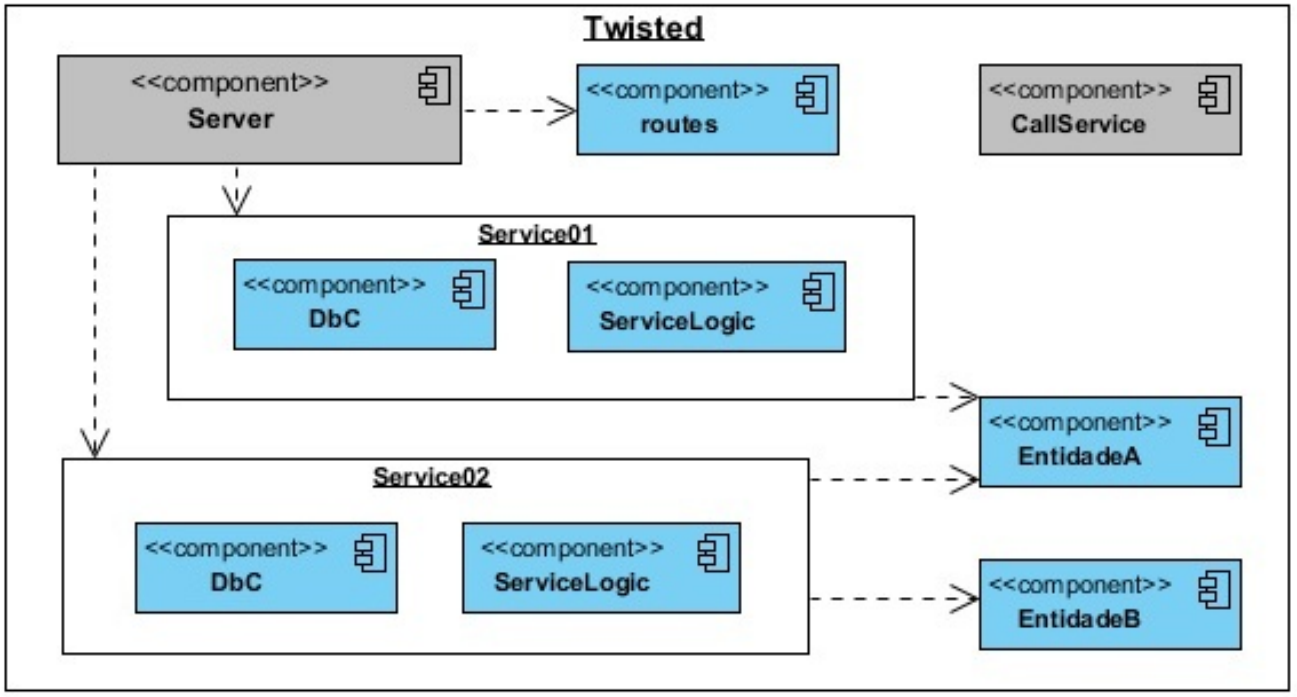


Cafeteria[1]

```
1  resource order  {
2       path = "/order/{orderId}";
3        /**
4        * @ summary: delete order.
5        * @ params:
6        *        path orderId: Id of order
7        * @ throws
8        *        404 Order not found.
9        *        401 Payment already processed.
10       */
11       @delete null deleteOrder ( int orderId)
12            require (call store.getPaymentStatus(orderId, 'expected') == 200),
13            otherwise 401;
14 };
```

[1] Livro: REST in Practice - Hypermedia and System Architecture

Prova de conceito - Twisted



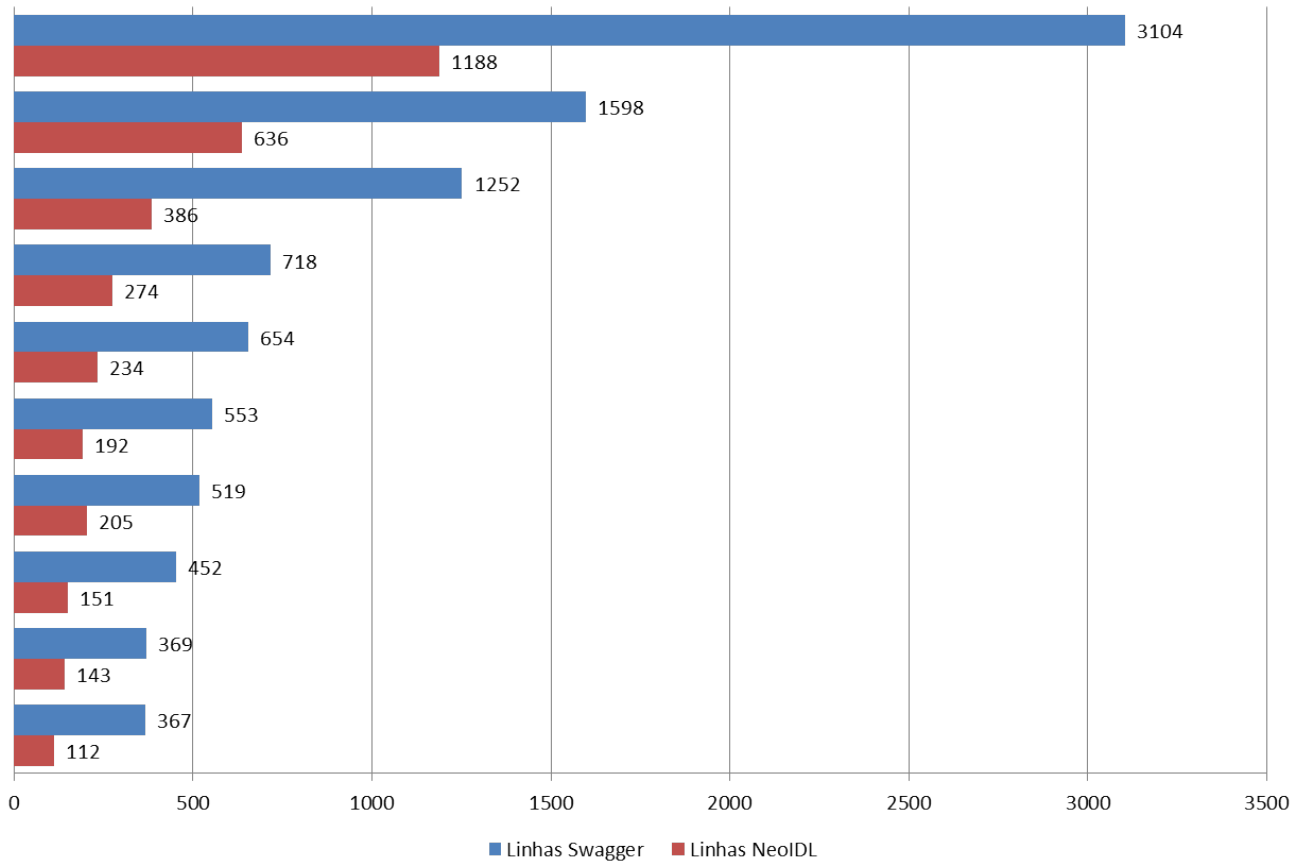
Avaliação de resultados

- Expressividade (em andamento)
- Reuso de entidades (em andamento)
- Benefícios de Design-by-Contract (inicial)

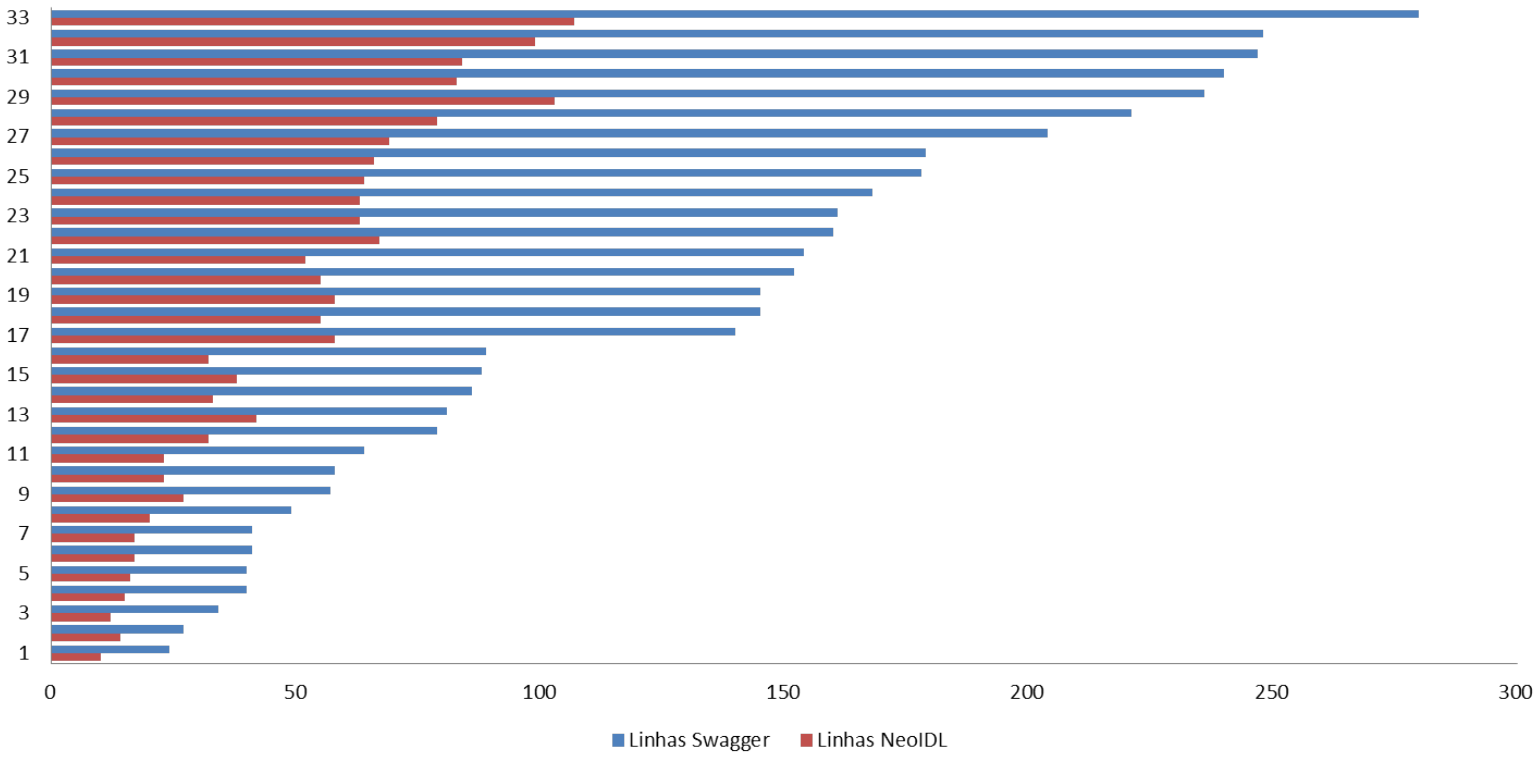
UnB

Expressividade - Swagger x NeoIDL

- 13.742 linhas em 43 contratos Swagger
- 5.117 linhas transcritos para NeoIDL
- Redução média: 62,7% (Maior redução: 69,5%)



Expressividade - Swagger x NeoIDL



Lucas Lima, et al. *NeoIDL: A Domain Specific Language for Specifying REST Contracts— Detailed Design and Extended Evaluation*. International Journal of Software Engineering and Knowledge Engineering. (**submetido em Setembro de 2015**).

UnB

Reuso

- 463 entidades diferentes declaradas nos 43 contratos
- 51 entidades com mais de uma ocorrência
- Casos críticos
  - Ponto: 12 ocorrências
  - Localizacao: 10 ocorrências
  - Area: 6 ocorrências
  - Item: 6 ocorrências

Reuso

- 463 entidades diferentes declaradas nos 43 contratos
- 51 entidades com mais de uma ocorrência
- Casos críticos
  - Ponto: 12 ocorrências
  - Localizacao: 10 ocorrências
  - Area: 6 ocorrências
  - Item: 6 ocorrências

Reuso

- 463 entidades diferentes declaradas nos 43 contratos
- 51 entidades com mais de uma ocorrência
- Casos críticos
  - Ponto: 12 ocorrências
  - Localizacao: 10 ocorrências
  - Area: 6 ocorrências
  - Item: 6 ocorrências

UnB

Reuso

```
1  entity Ponto {
2        double : longitude;    /*Required.*/
3        double : altitude;
4        double : latitude;    /*Required.*/
5  };
```

Ponto em NeoIDL

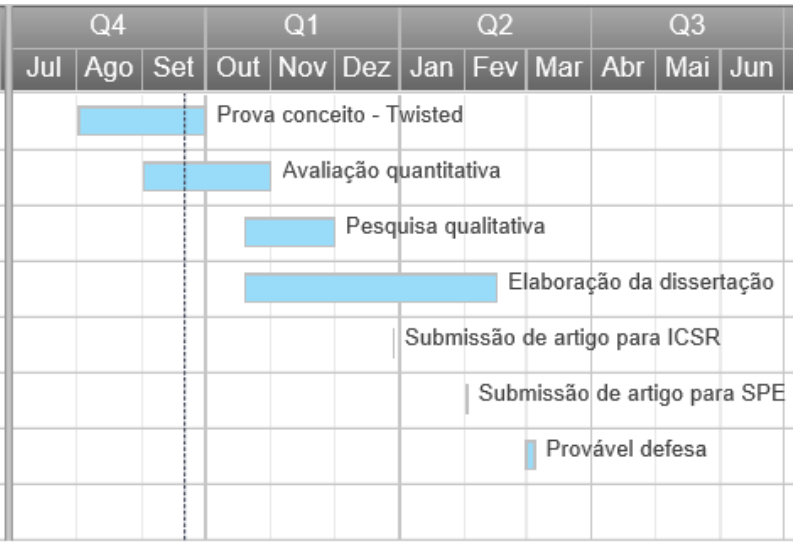
```
1  entity Area {
2        [Ponto] : pontos;    /*Required.*/
3  };
```

Reuso de Ponto em NeoIDL

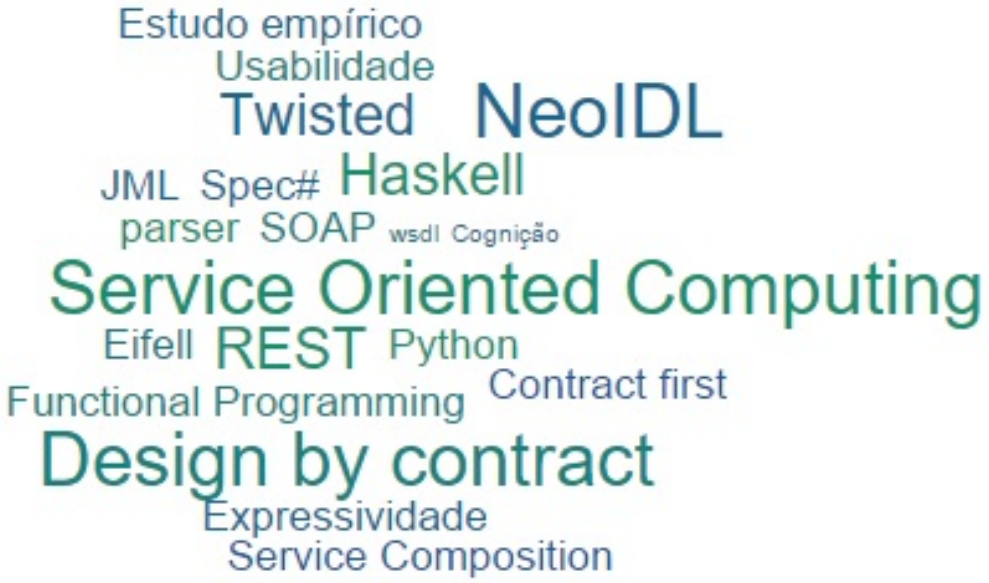
UnB

Cronograma das próximas etapas

Atividade	Início	Fim
Prova conceito - Twisted	01/08/15	30/09/15
Avaliação quantitativa	01/09/15	31/10/15
Pesquisa qualitativa	19/10/15	30/11/15
Elaboração da dissertação	19/10/15	15/02/16
Submissão de artigo para ICSR	28/12/15	
Submissão de artigo para SPE	01/02/16	
Provável defesa	29/02/16	04/03/16



UnB


Estudo empírico
Usabilidade
Twisted
NeoIDL
JML
Spec#
Haskell
parser
SOAP
wsdl
Cognição
Service Oriented Computing
Eifell
REST
Python
Functional Programming
Contract first
Design by contract
Expressividade
Service Composition

UnB

Comentários

UnB

Contratos REST robustos e leves

uma abordagem em Design-by-Contract com NeoIDL

Lucas F. Lima[1]
Orientadores: Rodrigo Bonifácio[2], Edna Canedo[3]

[1]Departamento de Engenharia Elétrica – Universidade de Brasília – UnB

[2]Departamento de Ciência da Computação – Universidade de Brasília – UnB

[3]Faculdade UnB Gama – Universidade de Brasília – UnB
Brasília – DF – Brasil

WTDSoft 2015

UnB