CBSoft 2012
III Congresso Brasileiro de Software: Teoria e Prática
24 a 28/09/2012 | Natal-RN

DIMAp UFRN

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Composição de Linguagens de Modelagem Específicas de Domínio:
Um Estudo Exploratório

Edmilson Campos[1, 2], Adorilson Bezerra[1, 2], Marília Freire[1, 2], Uirá Kulesza[1] e Eduardo Aranha[1]

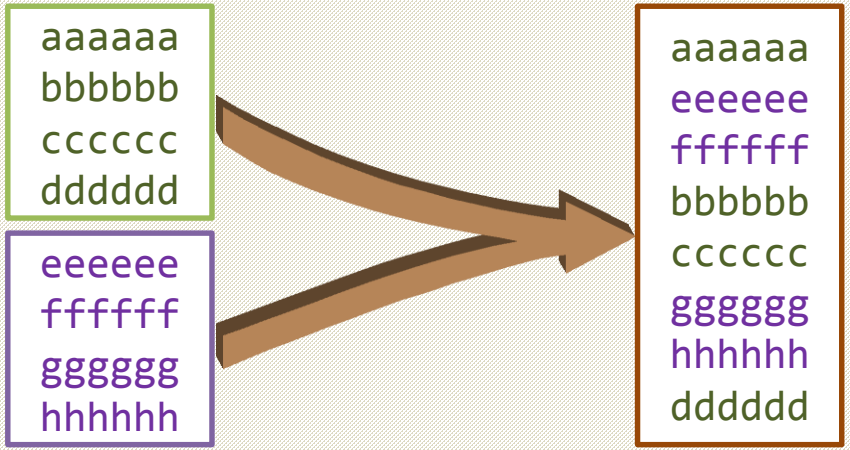
[1]Departamento de Informática e Matemática Aplicada – PPgSC/CCET/UFRN
[2]Instituto Federal de Educação, Ciência e Tecnologia do RN
{edmilson.campos, adorilson.bezerra, marilia.freire}@ifrn.edu.br, {uira, eduardoaranha}@dimap.ufrn.br

Motivação

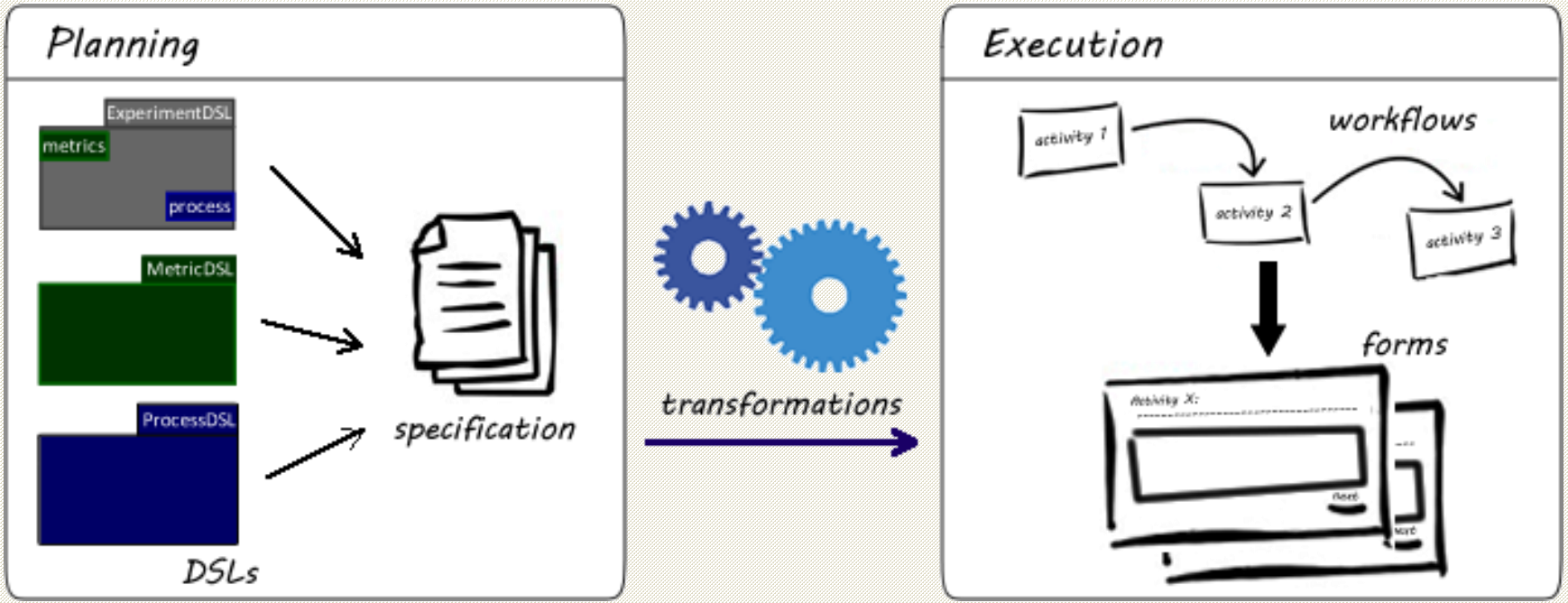
- O sucesso em um projeto de desenvolvimento requer coerente conceituação do domínio do problema
- *Domain-specific languages* (*DSLs*) têm sido usadas para elevar o nível de abstração no desenvolvimento
- Em projetos complexos, muitas vezes, uma única DSL é insuficiente para tratar diferentes visões e perspectivas de modelagem

Problema

- Composição de DSLs aumenta o risco de perda de consistência entre elementos dos modelos
- Essa problema requer a adoção de métodos e ferramentas de suporte



Contexto


Planning
ExperimentDSL
metrics
process
MetricDSL
ProcessDSL
DSLs
specification
transformations
Execution
workflows
activity 1
activity 2
activity 3
forms
Activity X:

Limitação dos trabalhos atuais

- Alguns trabalhos têm abordado a consistência de modelos, porém não relacionada à composição de DSLs
- Hessellund *et al* foram um dos únicos a propor um método para realizar composição de DSLs
  - O método foi aplicado, porém, apenas à DSLs baseadas em XML

Objetivo

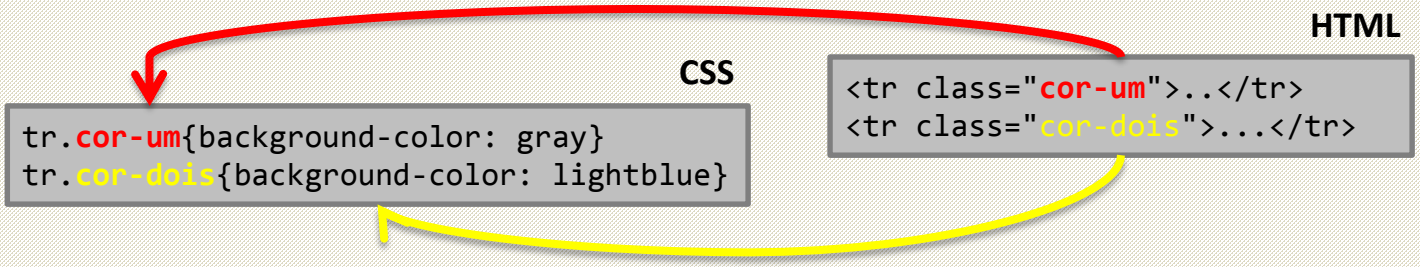
Investigar a aplicação do método de Hessellund *et al* para composição de DSLs baseadas no metamodelo *Ecore*, usando *xText*

Agenda

- Problemas de composição de DSLs
- Método para composição de DSLs
- Estudo exploratório
  - Definição das DSLs
  - Composição das DSLs
    - Modelagem do experimento
    - Aplicação do método
  - Avaliação preliminar
- Conclusão e trabalhos futuros

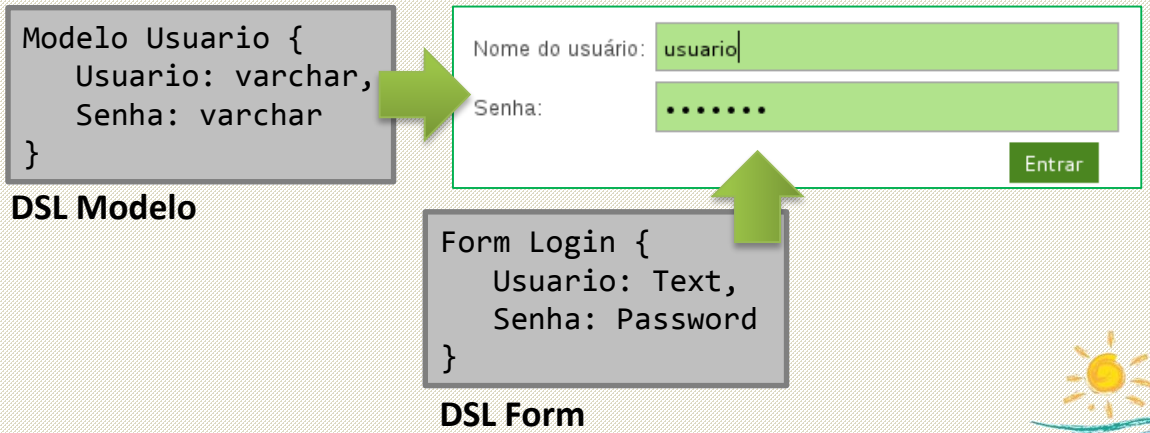
Problemas de composição de DSLs

- Principais tipos de restrições à consistência exploradas:
  - **Integridade entre artefatos**
    - Exemplo: Classes CSS referenciadas no HTML



Problemas de composição de DSLs

- Principais tipos de restrições à consistência exploradas:
  - Integridade entre artefatos
  - **Referências com restrições adicionais**
    - Exemplo: Caixa de texto com ocultação de caracteres para senha



Método para composição de DSLs

- Método proposto por Hessellund *et al*, divide-se em três passos:

PASSO 01: Identificação

- Objetivo de identificar pontos de referências entres as DSLs
- Identificação pode ser manual ou automática
  - Classificação quanto ao tipo de referências

PASSO 02: Especificação

- Implementação das referências do passo anterior
- Pode ser parcial ou completa
  - Varia de acordo com o tipo de referência

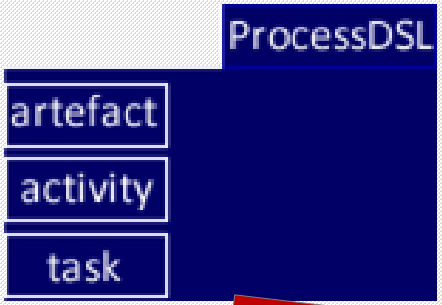
PASSO 03: Aplicação

- Aplica a especificação
- Ações divididas em três áreas:
  - Navegação
  - Verificação de consistência
  - Apresentação de guias

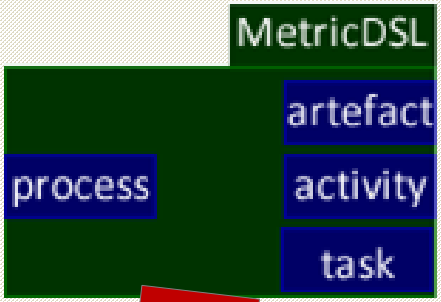
ESTUDO EXPLORATÓRIO

Etapas do estudo

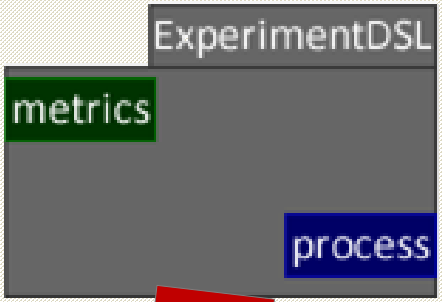
Definição das DSLs



Utilizada para definir os tratamentos em um experimento ou um desenvolvimento



Permite especificar métricas relacionadas a elementos de um processo



Definir os tratamentos e variáveis controladas do experimento

Composição das DSLs

- Essa etapa subdivide-se em:
  - Modelagem do experimento controlado
  - Aplicação do método

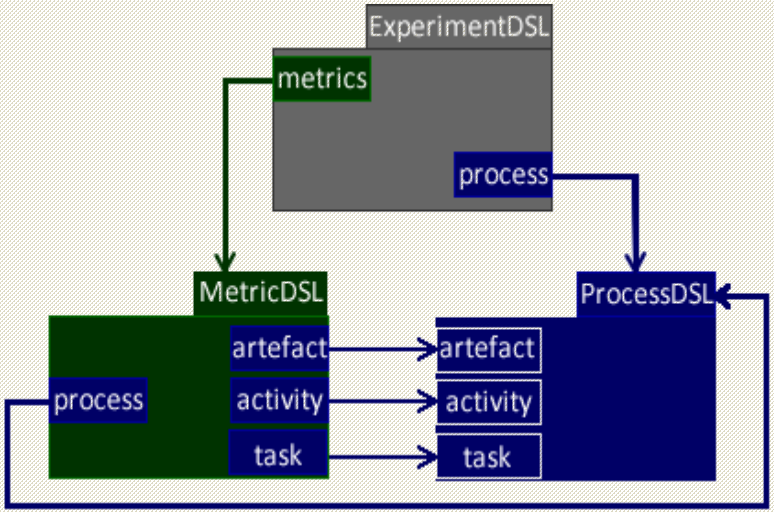
Modelagem do experimento

- Experimento controlado para comparar a produtividade das linguagens Java e C++
- Três fatores:
  - Linguagem, sistema e *subject*
- Um único processo
- Quatro métricas:
  - Tempo de projeto
  - Tempo codificação
  - Tempo testes
  - Quantidade de defeitos

```
Experiment "Comparison of Java and C++"
   Process "Simple Process"
   Metric "ProjectTime" "CodeDefect" "CodeTime" "TestTime"
   Experimental Plan Design "ComparisonPlanning"
        type LS - Latin Square
        Factor "ProgammingLanguage" isMain True
            Level "Java";
            Level "C++";
        ;
        Factor "Subject" isMain False;
        Factor "Sytem" isMain False
            Level "Phonebook";
            Level "EventManagement";
        ;
    ;
```

Aplicação do método

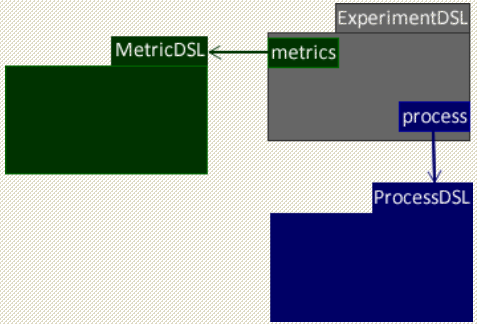
- **PASSO 1) Identificação**:
  - Identificação manual
  - Seis pontos de referências
    - Referências simples
    - Referências simples com restrições adicionais



Aplicação do método

- **PASSO 2) Especificação**:
  - Implementação de referências simples
    - Utilizado em referências explícitas por meio de recursos de importação de modelos *Ecore* no *xText*

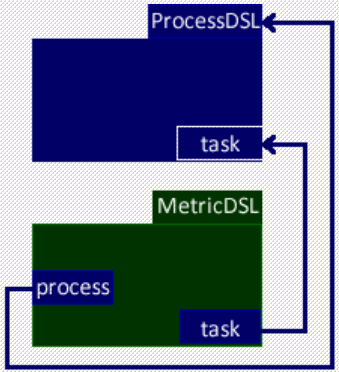
```
grammar br.ufrn.dimap.ExperimentDslLanguage.ExperimentDsl with org.eclipse.xtext.common.Terminals
import "platform:/resource/br.ufrn.dimap.ProcessDslLanguage/src-gen/br/ufrn/dimap/ProcessDsl.ecore" as processDsl
import "platform:/resource/br.ufrn.dimap.MetricDslLanguage/src-gen/br/ufrn/dimap/MetricDslLanguage/MetricDsl.ecore" as metricDsl
generate experimentDsl "http://www.ufrn.br/dimap/ExperimentDslLanguage/ExperimentDsl"
```




```
ExperimentElement:
    'Experiment' name=STRING
    'Process' process+= [processDsl::Process]*
    'Metric' metrics+=[metricDsl::Metrics]*
    ('Experimental Plan' experiments+=ExperimentalPlan*)?
```

Aplicação do método

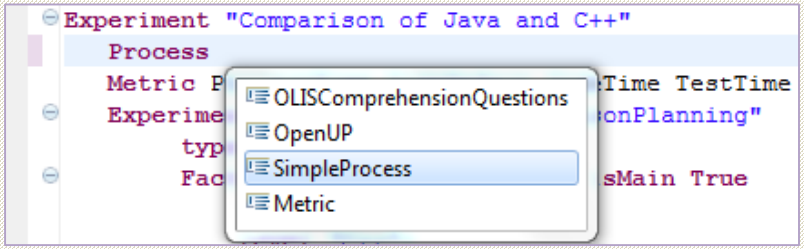
- **PASSO 2) Especificação**:
  - Implementação de referências com restrição adicional
    - Métodos de validação do *xText*

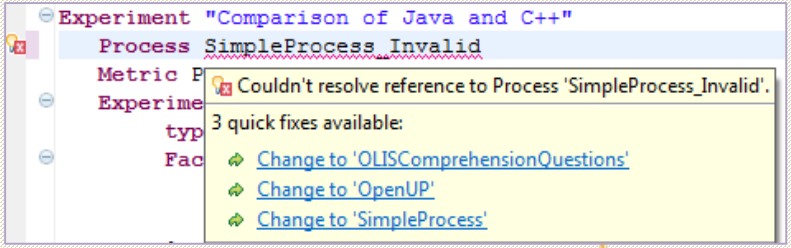



```
public class MetricDslJavaValidator extends AbstractMetricDslJavaValidator {
    @Check
    public void checkTaskIsValid(Metrics metrics) {
        boolean isValid = false;
        if (metrics.getDetails() instanceof TaskMetric)
        for (Task _task : metrics.getRelatesTo().getTasks() )
            for (Task task : ((TaskMetric)metrics.getDetails()).getTasks())
                if (task.getName().equals(_task.getName())) isValid = true;
        if (!isValid) error("Invalid task reference",
                            MetricDslPackage.Literals.METRICS__DETAILS);
    }
}
```

Aplicação do método

- **PASSO 3) Aplicação:**
  - Verificação de consistência
    - Integridade referencial
      - Recurso de auto complemento
      - Sugestões de consertos





Aplicação do método

- **PASSO 3) Aplicação:**
  - Verificação de consistência
    - Referências com restrição adicional

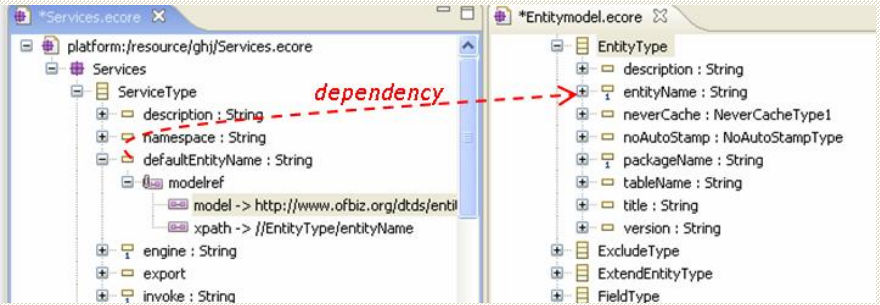
```
"CodeTime" relates SimpleProcess
                   description "Time taken to answer a question correctly"
                   type hardData
                   form continuous
                   id "CodeTime"
                   unit minutes
                   tasks "OpenUP.DevelopTechnicalVision"
```

Invalid task reference

Avaliação preliminar dos resultados

SmartEMF

- Referência implícita

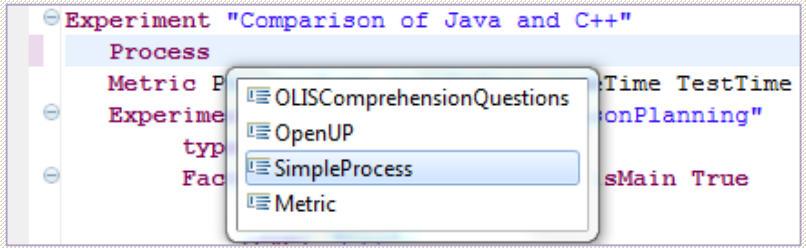



```
% name
constraint( no_dangling_modelrefs ).
% rule
no_dangling_modelrefs( Object, AnnotatedFeature ) :-
  modelref( AnnotatedFeature, DomainFeature ) ,
  attrvalue( Object, AnnotatedFeature, Value ) ,
  not( attrvalue( _ , DomainFeature, Value ) ) ) .
```

xText

- Referência explícita

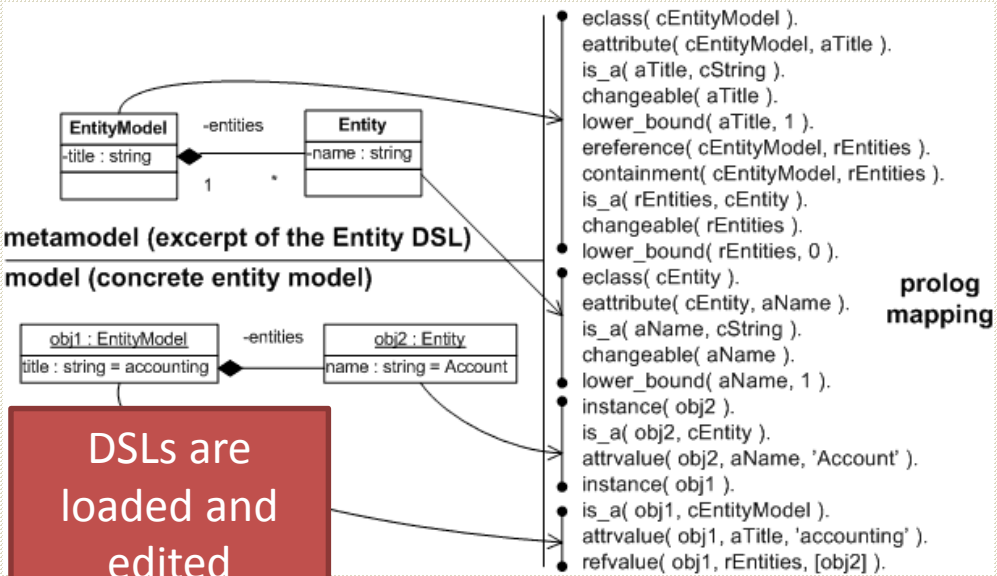
```
ExperimentElement:
    'Experiment' name=STRING
    'Process' process+= [processDsl::Process]*
    'Metric' metrics+=[metricDsl::Metrics]*
    ('Experimental Plan' experiments+=ExperimentalPlan*)?
```



Avaliação preliminar dos resultados

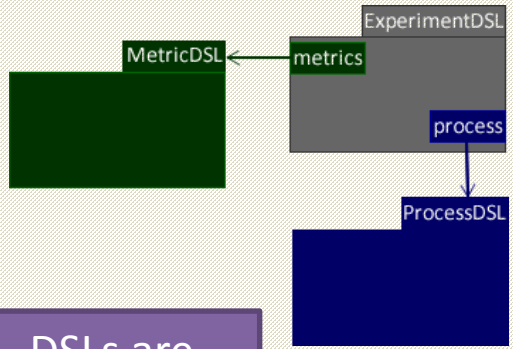
SmartEMF

- Especificação completa



xText

- Especificação parcial



Conclusão e trabalhos futuros

- Foi possível aplicar o método utilizando xText
- Principais contribuições:
  - Avaliação do método citado com DSLs construída no *xText*
  - Composição de DSLs para modelagem de experimentos controlados, processos e métricas
- Trabalhos futuros:
  - Adotar mecanismo de mapeamento de variabilidades
  - Suporte a realização de experimentos controlados reais

Questionamentos...

Obrigado pela atenção!

CBSoft 2012
III Congresso Brasileiro de Software: Teoria e Prática
24 a 28/09/2012 | Natal-RN

DIMAp UFRN

UFRN
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

Composição de Linguagens de Modelagem Específicas de Domínio:
Um Estudo Exploratório

Edmilson Campos[1, 2], Adorilson Bezerra[1, 2], Marília Freire[1, 2], Uirá Kulesza[1] e Eduardo Aranha[1]

[1]Departamento de Informática e Matemática Aplicada – PPgSC/CCET/UFRN
[2]Instituto Federal de Educação, Ciência e Tecnologia do RN
{edmilson.campos, adorilson.bezerra, marilia.freire}@ifrn.edu.br, {uira, eduardoaranha}@dimap.ufrn.br