INTEROPERABILIDADE DO AMBIENTE SMARTYMODELING COM FERRAMENTAS DE MODELAGEM DE FEATURES.

José Rafael Silva Hermoso (PIBIC/CNPq/FA/UEM), Edson A. Oliveira Junior (Orientador), e-mail: edson@din.uem.br

UEM/CTC/DIN

Área: Ciências exatas e da Terra,
Subárea: Ciência da Computação

Palavras-chave: Interoperabilidade, Features, SMartyModeling**.**

Resumo

O presente trabalho aborda a interoperabilidade entre dois sistemas utilizados para modelagem de Linha de Produto de Software (LPS) por meio de diagramas de características (*Features*) criados utilizando as ferramentas SMartyModeling, do nosso grupo de pesquisa da UEM, e a FeatureIDE, ferramenta integrada ao ambiente de desenvolvimento Eclipse. Foi observado que ambas as ferramentas estudadas importam e exportam seus diagramas de características em arquivos de texto, sendo eles no formato ".smty" para SMartyModeling e ".xml" para FeatureIDE. Com isso, uma possível tradução de um formato para o outro poderia ser feita, fazendo com que SMartyModeling aceite diagramas feitos no FeatureIDE. A tradução foi realizada utilizando algoritmos para a leitura do arquivo de texto contendo o diagrama exportado do FeatureIDE e montagem de um novo arquivo no formato aceito pelo SMartyModeling.

Introdução

Inicialmente será apresentada a definição de interoperabilidade definida pela IEEE que servirá como base para a implementação da interoperabilidade projetada. Em sequência serão analisados os padrões utilizados pelas duas ferramentas estudadas para salvar seus diagramas de características, finalizando com a construção de um algoritmo que realiza a tradução do padrão utilizado pelo FeatureIDE para o SMartyModeling.

Materiais e métodos

Foram utilizados artigos sobre interoperabilidade, o ambiente de desenvolvimento Eclipse e as ferramentas de modelagem FeatureIDE, junto com sua documentação para auxiliar no aprendizado sobre a mesma, e o SMartyModeling

Após estudar os diagramas em formato de texto de ambas as ferramentas, foi notado que os dois continham as mesmas informações porém escritas com padrões diferentes, bastando ler essas informações e transcrevê-las no padrão desejado, o padrão do SMartyModeling.

Resultados e Discussão

Primeiramente foram discutidos os padrões utilizados pelas ferramentas para salvar seus diagramas e foi notado que ambas as ferramentas salvam suas informações em arquivos de texto utilizando tags xml. Com isso, a hipótese de fazer um algoritmo que leia um arquivo e faça a tradução de um padrão para o outro foi feita.

```
<or name="Feature">
    <feature name="Feature3"/>
    <feature name="Feature4"/>
</or>
<or name="Feature2">
    <feature name="Feature5"/>
    <feature name="Feature6"/>
</or>
```

Imagem 1. Diagrama representado por tags xml no formato de texto da ferramenta FeatureIDE.

```
<combination id="COMBINATION#11" source="VARIABILITY#10" target="FEATURE#5" root="true"></combination>
<combination id="COMBINATION#12" source="VARIABILITY#10" target="FEATURE#7" root="false"></combination>
<combination id="COMBINATION#13" source="VARIABILITY#10" target="FEATURE#6" root="false">
<variability id="VARIABILITY#14" name="" category="Exclusive" variationPoint="FEATURE#2"></combination>
     mandatory="false" x="310.0" y="260.0" globalX="0" globalY="0" height="50" width="50">
  <variant id="FEATURE#4"/>
  <variant id="FEATURE#3"/>
</variability>
```

Imagem 2. Diagrama representado por tags xml no formato de texto da ferramenta SMartyModeling.

Em seguida estudamos formas de montar um algoritmo que consiga ler o arquivo xml gerado pelo FeatureIDE e foi notado que as estruturas referentes aos diagramas de *features* são semelhantes a árvores, então um método recursivo de leitura foi montado, permitindo a leitura desse arquivo.

```
void readRoot()
        root = buscar no arquivo a tag struct que indica o inicio do diagrama no arquivo de texto
        se existir root:
          tags = and, or, alt, feature
          para cada tag:
            NodeList = buscarNoArquivo(expression + tag)
            for Node in NodeList:
              readNode(Node, expression + tag)


void readNode(node, expression)
        feature = new Feature(node)
        tags = and, or, alt, feature
        para cada tag
          new = expression atualizada com o nome da feature atual mais a tag
          NodeList = buscarNoArquivo(new)
          for Node in NodeList:
            readNode(Node, expression + tag)
```

Imagem 3. Pseudocódigo das funções de leitura do arquivo xml.

Com o algoritmo de leitura pronto, restaram apenas dois problemas: a montagem de um novo arquivo utilizando o padrão do SMartyModeling e o posicionamento das features na área de edição de diagrama do SMartyModeling.

Foi montado um algoritmo que utiliza as informações já lidas do arquivo xml do FeatureIDE para realizar a tradução para o padrão utilizado no SMartyModeling e já deixa configurado as posições das *features*, evitando sobreposições na representação gráfica do diagrama.

```
public String criarVariabilityOr(ArrayList<String> aux1, Feature i) {
    String tagaux = "<variability id=\"VARIABILITY#subid\" name=\"\" category=\"Exclusive\" "
            + "variationPoint=\"FEATURE#"+i.getId()+"\" "
            + "mandatory=\"false\" x=\""+(i.getX()+50)+"\" y=\""+(i.getY()+60)+"\" "
                    + "globalX=\"0\" globalY=\"0\" height=\"50\" width=\"50\">\n";

    aux1.add(String.valueOf(i.getId()));
    for(Feature j : features.values()) {
        if(j.getPai()==(i.getName())) {
            tagaux+= "      <variant id=\"FEATURE#"+j.getId()+"\"/>\n" ;
            aux1.add(String.valueOf(j.getId()));
        }
    }
    tagaux+= "    </variability>\n" ;
    return tagaux;
}
```

Imagem 4. Exemplo de uma das funções utilizadas para tradução de informações do FeatureIDE para o padrão do SMartyModeling .

Ao final, temos como resultado a análise e entendimento dos padrões utilizados nas ferramentas para armazenar seus diagramas de *features* e um algoritmo que recebe arquivos exportados pelo FeatureIDE e o traduz para o formato aceito pelo FeatureIDE.

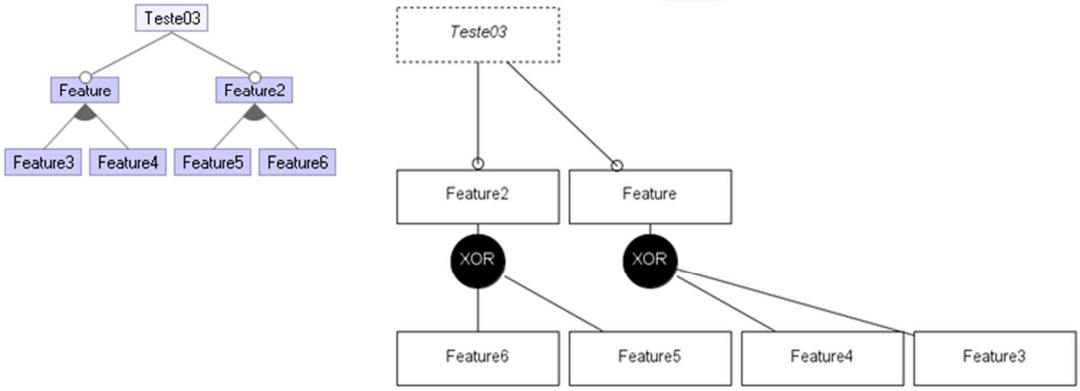

Imagem 5. Exemplo de diagrama montado no FeatureIDE à esquerda e sua tradução para o SMartyModeling à direita**.**

Conclusões

O projeto de iniciação científica permitiu o estudo da interoperabilidade entre sistemas de uma forma prática, onde foi desenvolvida uma forma de comunicação eficiente entre duas ferramentas de modelagem de features, fazendo com que o SMartyModeling possa aceitar diagramas montados da ferramenta FeatureIDE.

Agradecimentos

Agradeço ao meu orientador Prof. Dr. Edson A. Oliveira Junior e Ms. Leandro Flores da Silva, pelo suporte oferecido por eles no desenvolvimento do projeto, a Universidade Estadual de Maringá(UEM), o CNPq e a Fundação Araucária pelo incentivo à pesquisa científica e apoio financeiro.

Referências

ABUKWAIK, H.; ROMBACH, D. Software interoperability analysis in practice - A survey. **ACM International Conference Proceeding Series**, v. Part F1286, p. 12–20, 2017.

DA SILVA SERAPIÃO LEAL, G.; GUÉDRIA, W.; PANETTO, H. Interoperability assessment: A systematic literature review. **Computers in Industry**, v. 106, p. 111–132, 2019.

REZAEI, R. et al. Interoperability evaluation models: A systematic review. **Computers in Industry**, v. 65, n. 1, p. 1–23, 2014.