INTEROPERABILIDADE DO AMBIENTE SMARTYMODELING COM FERRAMENTAS DE MODELAGEM DE FEATURES - FASE 2.

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:Ciências exatas e da Terra, 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 dado sequência ao projeto anterior, implementado agora a interoperabilidade para o FeatureIDE, fazendo com que o mesmo receba arquivos do SMartyModeling. 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 o FeatureIDE aceite diagramas feitos no SMartyModeling. A tradução foi realizada utilizando algoritmos para a leitura do arquivo de texto contendo o diagrama exportado do SMartyModeling e montagem de um novo arquivo no formato aceito pelo FeatureIDE.

Introdução

Inicialmente foram 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 SMartyModeling para o FeatureIDE.

Materiais e métodos

Foram considerados artigos sobre interoperabilidade (ABUKWAIK; ROMBACH, 2017), (REZAEI, 2014) e (DA SILVA SERAPIÃO LEAL; GUÉDRIA; PANETTO, 2019), 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, dessa vez o padrão do FeatureIDE.

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, possibilitando a criação de um algoritmo para ler este arquivo.

```
<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 foram estudadas formas de montar um algoritmo que consiga ler o arquivo .smty gerado pelo SMartyModeling e foi notado que o SMartyModeling, salva suas informações separadas por tags, logo uma leitura simples, buscando pelas tags features, variability e connection, que são as tags que guardam todas as informações necessárias para a montagem do diagrama do SMartyModeling, a leitura de todas as informações importantes vai ser feita. Um detalhe a se observar, é que antes da leitura, precisamos primeiramente converter o arquivo de texto para a extensão .xml, visto que a ferramenta de leitura só aceita arquivos xml. Essa conversão é feita de forma simples, apenas copiando os bytes do arquivo de entrada em formato .smty e copiando seus bytes para um novo arquivo e salvando-o com a extensão xml.

```
private void readFeatures() throws XPathExpressionException {

    expression = "/project/diagram/feature";
    nodeList   = (NodeList) xPath.compile(expression).evaluate(document, XPathConstants.NODESET);

    for(int i = 0; i<nodeList.getLength(); i++) {
        Node node = nodeList.item(i);
        Element element = (Element) node ;
        Feature feature = new features.featureide.Feature(element);
        diagrama.setFeatures(feature.getId(), feature);
    }
}
```

Imagem 3. Exemplo do código de leitura das tags features.

Com o algoritmo de leitura pronto, ficamos apenas com o problema da montagem de um novo arquivo utilizando o padrão do FeatureIDE, que acaba sendo um pouco complexa, visto que, diferente do SMartyModeling, o FeatureIDE usa um aninhamento de tags para representar hierarquias entre elas, se assemelhando a uma árvore, logo sua montagem deve levar em conta as features acima do nível atual do diagrama, ou seja, trocamos o problema das posições que ocorreu na fase 1 do projeto pelo problema de aninhamento das tags do FeatureIDE.

Foi criado um algoritmo que utiliza as informações já lidas do arquivo xml do SMartyModeling para montar o diagrama em memória, com objetos que guardam suas informações como nome, pai e filhos de cada feature. Com isso, foi realizada a tradução para o padrão utilizado no FeatureIDE.

```
public void arrumarDiagrama() {
    findRoot();
    ArrayList<Feature> nivel0 = new ArrayList<Feature>();
    nivel0.add(getRoot());
    niveis.add(nivel0);
    pais = nivel0;
    while (pais.size() > 0) {
        // System.out.println("NOVO NIVEL");
        ArrayList<Feature> novo_nivel = new ArrayList<Feature>();
        pais.forEach((pai) -> {
            filhos = acharFilhos(pai);
            filhos.forEach((f) -> {

                novo_nivel.add(f);
            });

        });
        niveis.add(novo_nivel);
        pais = novo_nivel;

    }
    arrumarNiveis();

}
```

Imagem 4. Exemplo de uma das funções utilizadas para a montagem em memória do diagrama do SMartyModeling**.**

Ao final, tem-se 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 SMartyModeling e o tradução para o formato aceito pelo FeatureIDE.

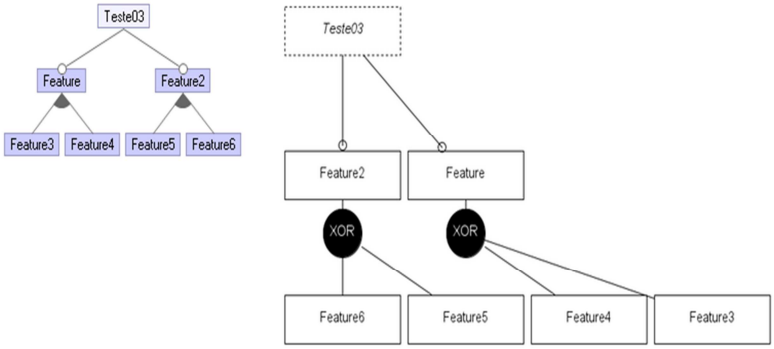

Imagem 5. Exemplo de diagrama montado no SMartyModeling à direita e sua tradução para o FeatureIDE à esquerda**.**

Conclusões

O projeto de iniciação científica permitiu o estudo da interoperabilidade entre sistemas de uma forma prática, onde foi concluída a interoperabilidade entre as duas ferramentas estudadas, dando sequência ao projeto anterior, fazendo com que agora as duas ferramentas possam importar e exportar arquivos de diagramas de features entre si.

Agradecimentos

Agradeço ao meu orientador Prof. Dr. Edson A. Oliveira Junior e ao 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.