TRADUÇÃO FORMAL DE DIAGRAMAS UML PARA MODELOS *SIMULINK*

PAZZINI, Vinícius[1]; BISI, Nícolas[2]; CAVALHEIRO, Simone[3]; FOSS, Luciana[4]; BRISOLARA, Lisane[5]

[1,2]Universidade Federal de Pelotas, Bacharelado em Ciência da Computação; [3,4,5]Universidade Federal de Pelotas, Centro de Desenvolvimento Tecnológico.
vspazzini@inf.ufpel.edu.br; nnbisi@inf.ufpel.edu.br; simone.costa@inf.ufpel.edu.br; lfoss@inf.ufpel.edu.br; lisane@inf.ufpel.edu.br.

1 INTRODUÇÃO

O presente trabalho tem por objetivo apresentar os resultados iniciais de uma pesquisa na área de métodos formais aplicada ao desenvolvimento de sistemas embarcados. Essa pesquisa é orientada pelas professoras Simone Cavalheiro e Luciana Foss, e está sendo realizada por membros do grupo de pesquisa LaPeGE (Laboratório de Pesquisa para Grupos Emergentes) da UFPEL. Além disso, este trabalho está inserido no âmbito de um projeto denominado Núcleo de Excelência em Engenharia de Software para Sistemas Embarcados, coordenado pelo professor Flávio Wagner Rech (UFRGS) e desenvolvido em cooperação com pesquisadores da UFRGS, UFPEL e UNIPAMPA. Este projeto tem como objetivo definir uma metodologia para o desenvolvimento de sistemas embarcados confiáveis em um tempo de projeto reduzido, garantindo a qualidade e a eficiência do conjunto hardware/software, bem como dando suporte a sua evolução.

Uma grande redução do esforço no desenvolvimento de sistemas embarcados pode ser obtida com o uso de modelos. Porém, ferramentas disponíveis para modelagem e geração de código normalmente são dependentes de domínio e o software embarcado normalmente possui comportamento heterogêneo, requerendo suporte a múltiplos modelos de computação. Entre as possibilidades para a realização de um projeto e o desenvolvimento de sistemas embarcados destacam-se UML (OMG, 2010) e *Simulink* (Mathworks, 2011).

O UML fornece abstrações de alto nível adequadas para a modelagem de software orientado a modelos, já o Simulink permite uma melhor descrição de fluxo de dados. As especificidades de cada modelo motivam propostas de unificação e mapeamento entre UML e *Simulink*, com o objetivo de permitir que desenvolvedores utilizem UML como linguagem de modelagem e ao mesmo tempo se beneficiem com as facilidades de *Simulink* para simulação e geração de código.

Um mapeamento de diagramas UML para modelos *Simulink* foi definido informalmente em BRISOLARA (2007, 2008), através de linguagem natural. Porém, devido à natureza da linguagem utilizada, esta definição pode ser ambígua e inconsistente. O uso de uma linguagem formal para definir a tradução permite eliminar a ambiguidade e identificar inconsistências. Portanto, neste trabalho, propõem-se uma definição formal para este mapeamento utilizando Gramática de Grafos (ROZENBERG, 1997; EHRIG, 1999) como linguagem formal de especificação e a ferramenta Groove (RENSINK, 2004) para automatizar o processo de transformação. Uma gramática de grafos é composta de um grafo inicial que modela o estado inicial da transformação e de um conjunto de regras que descrevem as possíveis mudanças de estados grafos.

2 METODOLOGIA (MATERIAL E MÉTODOS)

A tradução proposta em Brisolara (2007) está definida para diagramas UML que possuem algumas restrições. Estas restrições são impostas para que exista um modelo *Simulink* equivalente.

A transformação dos diagramas UML em modelos *Simulink* começa com a construção de um grafo que representa os diagramas. Este grafo é usado como grafo inicial em uma gramática contendo regras que definem formalmente o mapeamento proposto. Uma regra transforma um grafo em outro criando e/ou apagando elementos, bem como preservando outros. A Fig. 1 exibe as principais regras que descrevem a tradução (regras que apagam arcos auxiliares foram omitidas por questão de espaço).

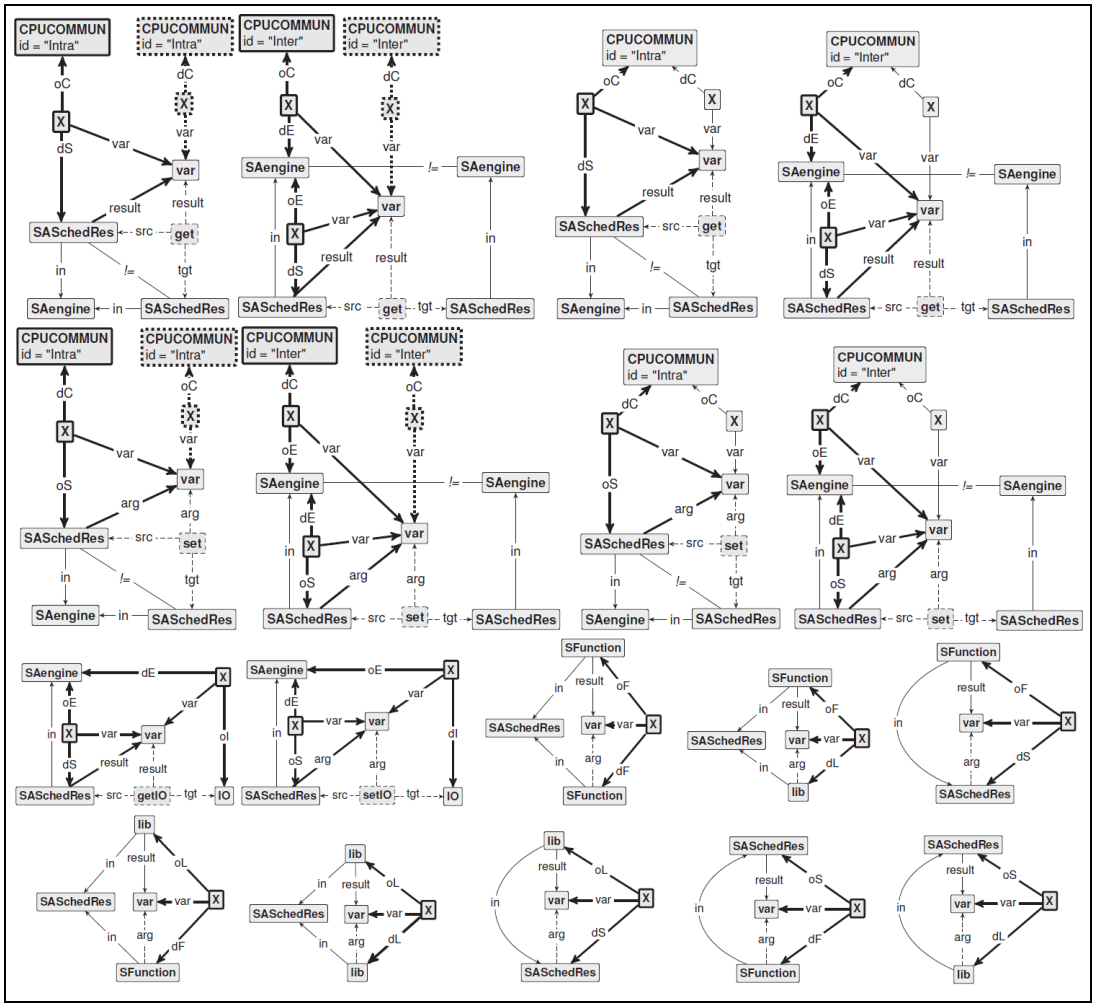

Figura 1 - Principais regras que especificam a transformação de UML para *Simulink*.

Uma regra define os elementos que devem ser preservados (desenhados com linhas simples e contínuas), apagados (desenhados com linhas simples e tracejada) e criados por sua aplicação (desenhados por linhas densas e contínuas), além de descrever elementos proibidos. Para que uma regra possa ser aplicada a um grafo, os elementos a serem preservados ou apagados devem estar presentes neste grafo, porém os que são proibidos não podem estar presentes.

Para se obter o grafo que constitui uma representação correspondente a um modelo *Simulink*, aplicam-se as regras sucessivas vezes começando a partir do grafo inicial que modela os diagramas UML.

3 RESULTADOS E DISCUSSÃO

Um estudo de caso foi realizado, onde diagramas UML foram representados pelo grafo da Fig. 2. Aplicando-se as regras definidas é obtido o grafo que representa um diagrama em *Simulink*, que é apresentado na Fig. 3.

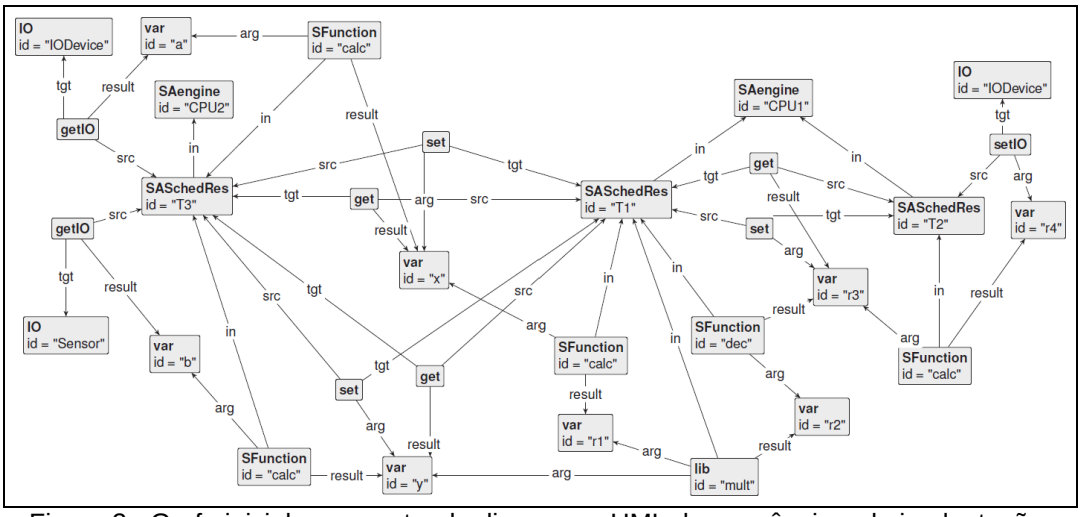

Figura 2 - Grafo inicial representando diagramas UML de sequência e de implantação.

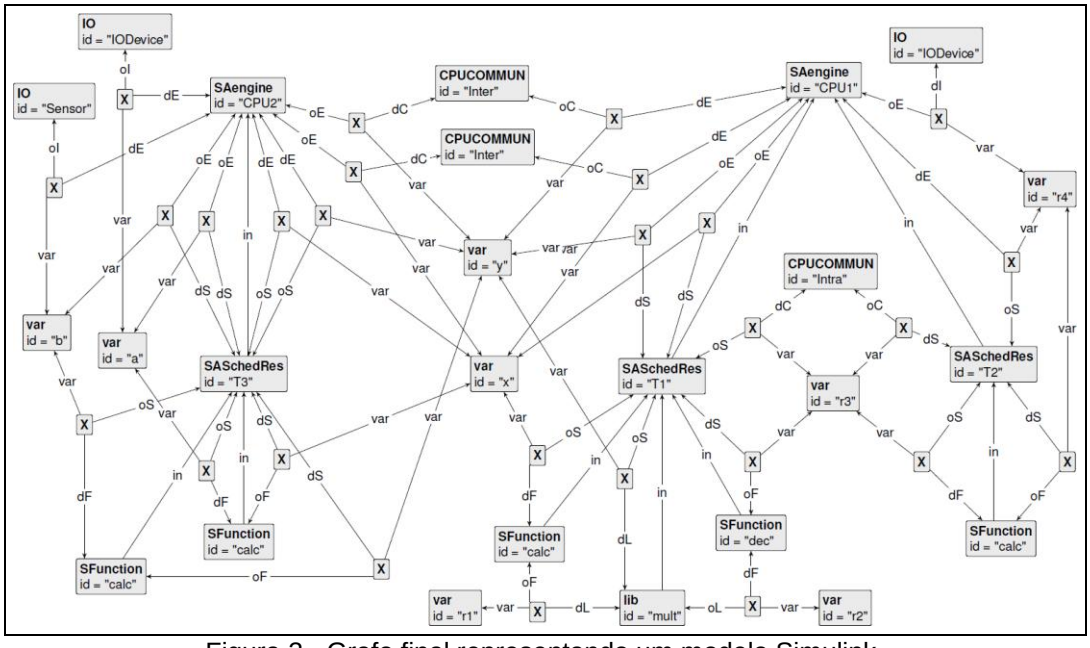

Figura 3 - Grafo final representando um modelo Simulink.

Até o presente momento, foi definida formalmente a transformação entre os grafos que representam os diagramas UML e os modelos *Simulink.* Além disso, estas transformações foram automatizadas com o uso da ferramenta Groove. Para podermos ter todo o processo formalizado e automatizado, ainda deve-se, definir formalmente as construções do grafo inicial a partir dos diagramas UML e a

construção dos modelos *Simulink* a partir do grafo gerado pela aplicação das regras, bem como automatizá-los.

4 CONCLUSÃO

Neste trabalho apresentou-se a definição formal da tradução de UML para *Simulink* proposta em BRISOLARA (2008), utilizando Gramática de Grafos. O uso de uma linguagem de especificação formal não só permitiu eliminar possíveis ambiguidades na definição do mapeamento previamente descrita em linguagem natural, como também permitiu o uso da ferramenta Groove para automatizar a tradução. Desta forma, um fluxo de projeto que começa em um modelo UML e provê um mapeamento (semi)automático para um diagrama de blocos *Simulink* é proposto.

Como trabalhos futuros são propastas as automatizações das traduções do diagrama UML para a entrada da ferramenta Groove e a automatização da tradução do grafo resultante, representando um diagrama Simulink, para um diagrama Simulink. Também se pretende explorar a verificação de propriedades importantes em transformação de modelos, como consistência e terminação, para o mapeamento definido.

5 AGRADECIMENTOS

Os autores agradecem à FAPERGS e ao CNPq pelo suporte financeiro na realização do presente trabalho.

6 REFERÊNCIAS

BRISOLARA, L. B. et. al. Using UML as front-end for heterogeneous software code generation strategies. In: **CONFERENCE ON DESIGN, AUTOMATION AND TEST IN EUROPE**. New York, 2008. Proceedings... New York: ACM, 2008. p. 504–509.

BRISOLARA, L. B. **Strategies for Embedded Software Development Based on High-level Models**. 2007. Tese de Doutorado em Computação, Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2007.

EHRIG, H.; ENGELS, G.; KREOWSKI, H.-J.; ROZENBERG, G (eds.). **Handbook of graph grammars and computing by graph transformation: applications, languages, and tools**. River Edge: World Scientific Publishing Co. editors, 1999.

MATHWORKS. **Simulink**. 2011. http://www.mathworks.com/. Acesso em julho de 2011.

OMG. **Unified Modeling Language (UML) infrastructure version 2.3**. Technical Report Formal/2010-05-03, 2010.

RENSINK, A.. The GROOVE simulator: A tool for state space generation. In **APPLICATIONS OF GRAPH TRANSFORMATIONS WITH INDUSTRIAL RELEVANCE**. Charlottesville, 2003. Proceedings… Berlin: Springer, 2004. p. 479– 485.

ROZENBERG, G. (ed.). **Handbook of Graph Grammars and Computing by Graph Transformation: Foundations**. River Edge: World Scientific Publishing, 1997.