TRANSFORMAÇÃO AUTOMÁTICA DE DIAGRAMAS UML PARA MODELOS *SIMULINK*

ANDREI COSTA[1]; VINÍCIUS PAZZINI[2]; SIMONE CAVALHEIRO[3]; LISANE BRISOLARA[4]; FLÁVIO RECH[5]; LUCIANA FOSS[6];

[1,2]Universidade Federal de Pelotas, Bacharelado em Ciência da Computação;
[3,4,6]Universidade Federal de Pelotas, Centro de Desenvolvimento Tecnológico;
{acosta, vspazzini, simone.costa, lisane, lfoss}@inf.ufpel.edu.br
[5]Universidade Federal do Rio Grande do Sul, Instituto de Informática – flavio@inf.ufrgs.br

1. INTRODUÇÃO

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.

Atualmente, sistemas embarcados estão sendo utilizados nas mais diversas áreas e esta demanda está aumentando a complexidade destes sistemas. Métodos de engenharia de software tradicionais podem não atender à qualidade destes sistemas de tempo real, como segurança, manutenibilidade, reuso de código, etc.

A UML é uma linguagem amplamente aceita na área de modelagem para sistemas embarcados. De acordo com [Kaur 2012], o padrão UML é muito útil na captura de requesitos, na decomposição dos sistemas em objetos e na definição de suas relações, além de prover um alto nível de abstração e de manutenibilidade.

Simulink é uma ferramenta baseada em modelos que vem sendo muito utilizada para modelagem de sistemas embarcados. Esta ferramenta tem foco no processamento de sinal e controle do sistema, além de prover suporte à geração de código e possuir componentes reusáveis em suas bibliotecas.

Pesquisas mostram que tanto UML quanto *Simulink* são considerados atrativos para modelagem de sistemas embarcados [Brisolara et al. 2008, Sjöstedt et al. 2008, Farkas et al. 2009], o que motiva pesquisadores a descobrir um modo de explorar simultaneamente os benefícios de ambas linguagens.

O mapeamento de diagramas UML para modelos *Simulink* foi proposto por Brisolara [Brisolara et al. 2008], através do uso dos diagramas de sequência e de implantação da UML. Contudo, este mapeamento foi definido de modo informal, através de linguagem natural, o que torna o processo de tradução suscetível a erros e imprecisões. Este trabalho não só propôs a formalização como também automatizou tal mapeamento. A tradução foi implementada utilizando Java e formalizada através da linguagem formal de gramática de grafos [Bisi et al. 2011, Foss et al. 2013].

O processo de tradução começa construindo o grafo inicial da gramática de grafos a partir dos diagramas UML. Então utilizando a ferramenta GROOVE [Rensink et al. 2010] e aplicando as regras definidas na gramática, o grafo inicial é transformado em um grafo que representa o modelo *Simulink.* Finalmente este grafo é transformado em um modelo Simulink. O processo de tradução está completamente automatizado.

2. METODOLOGIA

A tradução proposta está definida para diagramas UML com algumas restrições. O diagrama de implantação permite a especificação e organização dos elementos de processamento ou de software do sistema. Nós utilizamos tais diagramas para representar a estrutura do sistema embarcado. Um *Node* representa um processador e um *Artifact* modela uma *thread* dentro de um *Node*.

O diagrama de sequência representa as interações entre os objetos, que se comunicam por troca de mensagens. Objetos são instâncias de classes e uma mensagem é uma invocação de um método. Um objeto é representado por um nome, pelo nome da classe, e por um esteriótipo. Utilizamos um conjunto de esteriótipos do pacote MARTE GRM [OMG 2008], cujo objetivo é modelar uma plataforma geral para execução de aplicações embarcadas. Os objetos devem ser de um destes cinco esteriótipos: <<ComputingResource>> (representa um processador), <<Scheduler>> (modela o escalonador, que inicializa a execução das threads), <<SchedulableResource>> (identifica threads, ou objetos escalonáveis), <<DeviceResource>> (modela sistemas externos (IO)) e <<lib>> (deve ser definido pelo próprio modelador, define uma biblioteca de funções que podem ser reusadas em vários projetos).

O mapeamento começa com os diagramas UML, que são convertidos em um grafo inicial, representando o modelo UML. Esta parte está automatizada por uma ferramenta desenvolvida em Java. Este grafo inicial é importado em uma gramática que possui regras que definem formalmente o mapeamento proposto. As regras são aplicadas sucessivamente, transformando o grafo inicial em um grafo final, que representa um modelo *Simulink*. Este passo utiliza a ferramenta GROOVE.

O último passo do mapemento é a geração do modelo Simulink a partir do grafo de saída da gramática. Esta etapa também está automatizada por uma ferramenta desenvolvida em Java.

3. RESULTADOS E DISCUSSÃO

Um estudo de caso foi realizado, construíu-se um modelo UML na ferramenta Papyrus [Papyrus 2008]. A Figura 1 mostra um grafo que representa os diagramas de sequência e implantação UML do modelo desenvolvido. A Figura 2 mostra o mesmo sistema da Figura 1, porém após a aplicação das regras da gramática.

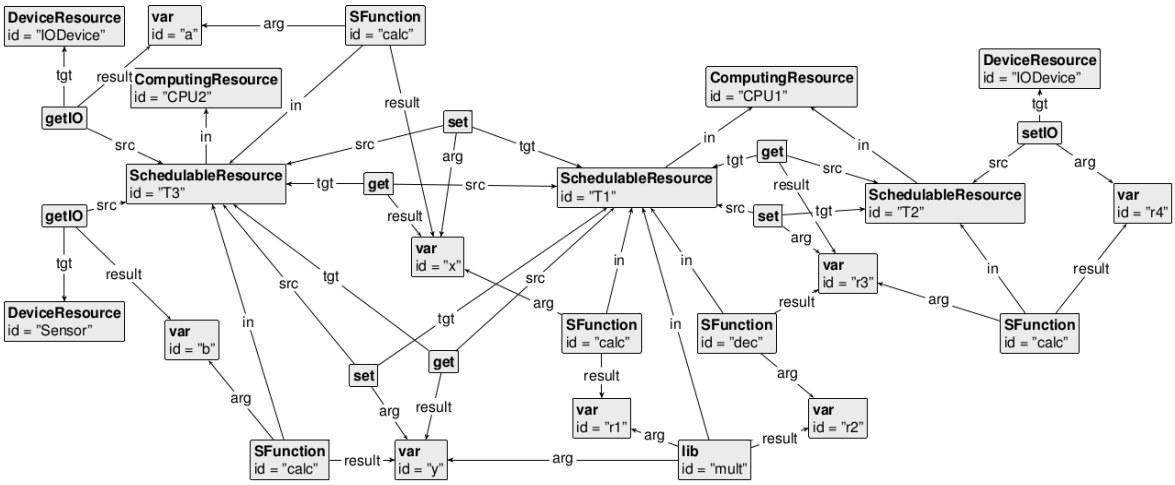

Figura 1 – Grafo inicial representando diagramas de sequência e implantação UML

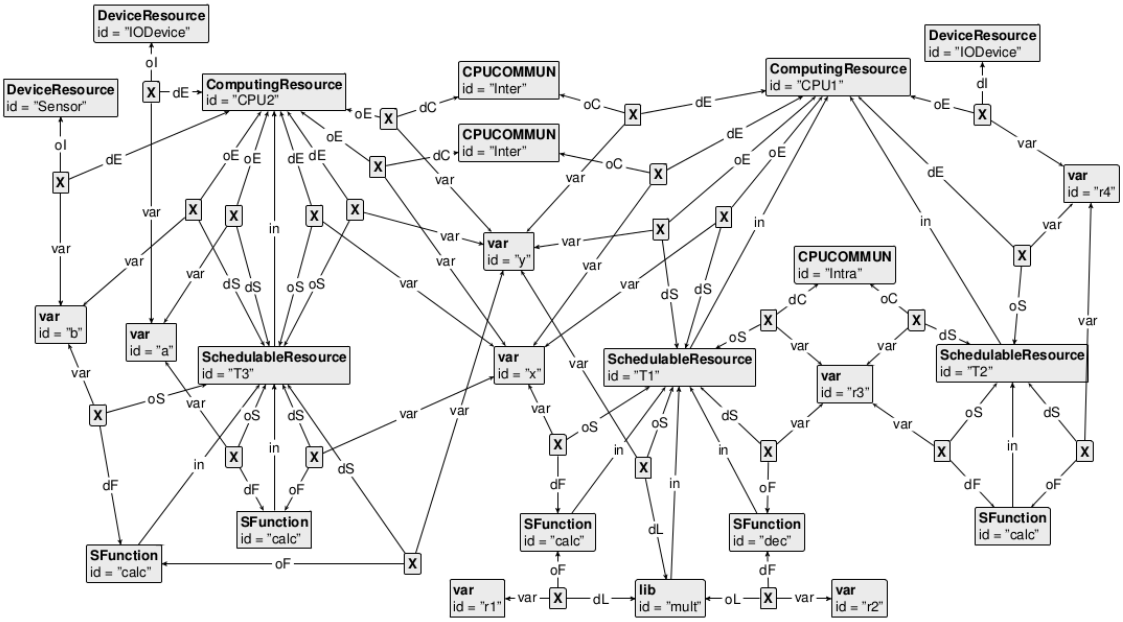

Figura 2 – Grafo final representando um modelo *Simulink*.

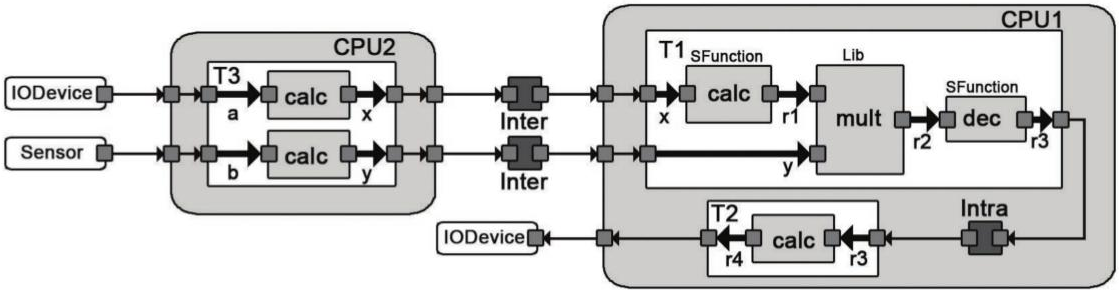

Figura 3 – Modelo *Simulink* gerado.

A Figura 3 ilustra o modelo do estudo de caso no *Simulink* após ter sido gerado pela nossa ferramenta a partir do grafo final, a vizualização é gerada pela própria ferramenta *Simulink*.

4. CONCLUSÕES

Neste trabalho apresentamos uma tradução automatizada de diagramas de sequência e implantação UML em modelos *Simulink*, permitindo a modeladores aproveitar, ao mesmo tempo, UML como linguagem de mapeamento e as facilidades de simulação e geração de código disponíveis no *Simulink*.

O uso de uma especificação formal na tradução, com gramática de grafos, nos permite, no futuro, fazer verificações formais de propriedades desta. No momento, uma propriedade interessante que pode ser analisada é a consistência nas conexões do grafo do modelo *Simulink* gerado, garantindo que todos argumentos dos métodos que são previamente chamados, serão produzidos, e seus resultados utilizados.

Nós também pretendemos provar formalmente que a semântica dos componentes mapeados é preservada, fazendo com que o fluxo de dados representados pelo diagrama de sequência inicial seja mantido no modelo *Simulink* final.

5. AGRADECIMENTOS

Os autores agradecem à FAPERGS, ao CNPq (ARD-11/0764-9) e ao projeto NESS (PRONEX-10/0043-0), pelo suporte financeiro na realização do presente trabalho.

6. REFERÊNCIAS BIBLIOGRÁFICAS

KAUR, A. Application of uml in real-time embedded systems. **International Journal of Software Engineering and Applications (IJSEA)**, 3(2):59-70, 2012.

SJÖSTEDT, C. J.; SHI, J.; TÖRNGREN, M.; SERVAT, D.; CHEN, D.; AHLSTEN, V.; LONN, H. Mapping Simulink to UML in the Design of Embedded Systems: Investigating Scenarios and Structural and Behavioral Mapping. **OMER4 Post-proceedings**, 2008.

FARKAS, T.; NEUMANN, C.; HINNERICHS, A. An integrative approach for embedded software design with UML and Simulink. **Proceedings of the 2009 33rd Annual IEEE International Computer Software and Applications Conference - Volume 02**. pages 516-521. Washington, DC, USA. COMPSAC'09, IEEE Computer Society. 2009.

BRISOLARA, L. B.; OLIVEIRA, M. F. S.; REDIN, R.; LAMB, L. C.; CARRO, L.; WAGNER, F. Using UML as front-end for heterogeneous software code generation strategies. **Proc. of the conference on Design, automation and test in Europe**. pages 504–509. NY, USA. ACM. 2008.

BISI, N. N.; PAZZINI, V.; FOSS, L.; DA COSTA CAVALHEIRO, S. A.; DE BRISOLARA, L. B.; WAGNER, F. R. Using graph grammars to develop embedded systems based on uml models. **2011 Workshop-School on Theoretical Computer Science, WEIT 2011**. pages 81-87, Pelotas, Brazil. IEEE Computer Society. 2011.

DE BRISOLARA, L. B. **Strategies for embedded software development based on high-level models**. Phd thesis, Federal University of Rio Grande do Sul. 2007.

RENSINK, A.; BONEVA, I.; KASTENBERG, H.; STAIJEN, T. **User Manual for the GROOVE Tool Set**. Department of Computer Science, University of Twente. 2010.

Papyrus. **Open source tool for graphical uml2**. Acessado em 29 de Março de 2013. Disponível em http://www.papyrusuml.org/. 2008.

OMG Object Management Group. **A UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded systems**. Object Management Group. 2008.