Possibility of SystemC Code Generation from SDL Specification

Pavel Morozkin
State University of Aerospace Instrumentation
Saint-Petersburg, Russia
pavel.morozkin@guap.ru

Abstract

Technology of code generation for models of systems is described by means of DSL *(Domain-Specific Language)* languages, based on MDA *(Model Driven Architecture)* design, provides a basis for further simulation and testing. In particular, the use of SDL language for design of formal specification of communication systems has more than thirty years story. Due to rapidly increasing complexity of systems, recently a group of researchers from SUAI and NOKIA Research Center has proposed and successfully implemented method for SDL and SystemC languages co-modeling use. This method has some significant drawbacks, seriously complicating process of SystemC components integration with SDL models. This paper describes a solution, which consists of method of SystemC code generation firstly by parsing textual representation of SDL specification, then by generating AST *(Abstract Syntax Tree)* tree and finally by generating of SystemC code from this tree.

Index Terms: SDL, SystemC, code generation.

I. Introduction

MDA [1, 2] approach is not widespread. Nevertheless, usually facilities of surrounding world are researched by developing models. Problem-oriented languages are used to design the models of different nature systems. In particular, the use of an SDL [3] double-representations language opens possibility to design systems at high abstraction level. An important point is that SDL language representations have same model semantics. By means of automatic code generation, models in form of executable programs are used especially for simulation and testing. An important advantage of SDL modeling tools is ability to support connection between an executable model and its graphical representation during the simulation. It can be possible by observe the model behavior and correct it. Monitoring is one of the simplest ways to test such models. But usually while making a research of the SDL models, developers are faced with the problem of choosing of most suitable technologies for testing. A several proprietary technologies are normally developed specially for large projects of industrial companies. Due to this fact, obtaining information about these technologies becomes impossible.

Previously developed technology of SDL and SystemC [4] co-modeling has opened up new possibilities for testing and research [5]. Implementation of this technology has been presented in [6]. Problems of current SDL-SystemC co-modeling technology and possible solution are described below.

With the financial support of the Ministry of Education and Science of the Russian Federation

II. From problem to solution

A. The problem overview

During testing of SDL-SystemC models developers are faced with problem of SDL models with SystemC components synthesis. Developers must use the models, obtained by the code generation technology, made by software tools manufacturer. Of course, this imposes a significant impact on processes of models integration. For example, integration of SDL generated C-code (with SDL modeling kernel) with SystemC components needs a lot of preparation steps. Examples of such steps — design of special SDL-SystemC signal converting wrappers and redefining the SDL scheduling. Due to the described difficulties, there is a need for a new solution to eliminate disadvantages of current co-modeling methods. Automatic SystemC code generation from SDL specification is one of the most interesting and efficient solution.

B. Related work

Previous applicability analysis [5, 7, 8] of SystemC and SDL languages showed potential possibility to represent of SDL models by means of SystemC language. The analysis is based on a detailed comparison of the both languages appointment and comparison of its elements. Then it has been presented a method of SDL and SystemC co-modeling. Main feature is an integration of SDL and SystemC models especially for modeling. Another feature is development of large models, which have a lot of similar type components. Also this method makes new features for model testing [8].

Examples of the SDL-SystemC co-modeling method, such as dynamic object creation for model of network switch, testing of the models, described earlier in [5].

C. Description of the method

The workflow graph of proposed method is shown at the Figure 1. SDL has two representations — graphical and textual [3]. Acronyms GR and PR means Graphic Representation and Phrase Representation. This terminology is also used by SDL developers (from SDL forum [9] or paper writers) as terms called *Graphic Notation* and *Textual Notation*. Firstly Graphic Representation of SDL model must be developed using SDL graphical editor. This representation is transformed to Phrase Representation by conversion tools. Phrase Representation means textual representation. Note that GR and PR are both equivalent representations of the same semantics. Then textual model must be checked for its correctness by analyzer. If model is incorrect, analyzer produces some errors and developer must eliminate these errors by model redesign. If model is correct, a code generator produces source code and custom makefile for SDL model. It should be mentioned that pure SDL model can't be exist as self-sufficient one. Model needs a modeling kernel, which carries all concepts of SDL language. Modeling kernel implements a control functions for SDL model and performs the SDL scheduling. After that the makefile must be processed by make utility. During a build process, make utility uses compiler. The compiler builds the generated SDL model source code with SDL modeling kernel and produces an object files. Then linker combines the object files with other libraries and produces executable model. As result of these steps, executable model can be simulated and tested.

Our method proposes analysis of textual representation, based on SDL grammars [3]. After that AST *(Abstract Syntax Tree)* [10] tree is generated and then source code is generated from this tree. This method considers following problems. The problem of textual representation analysis can be reduced to construction of AST tree, reflecting a structure of the SDL model.

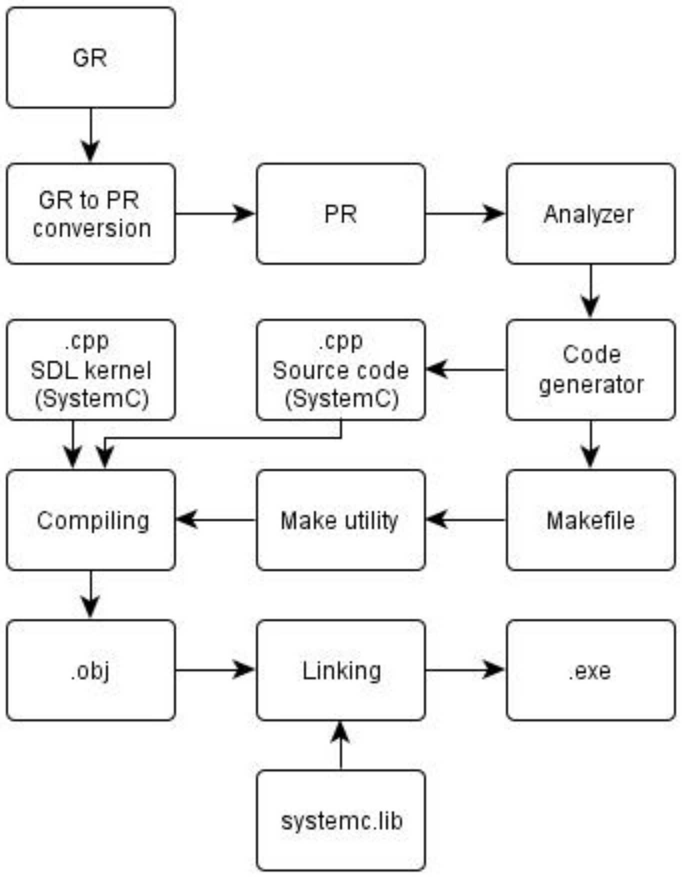

Fig. 1. Workflow graph

In addition, at this stage textual representation of SDL model is checked for correctness of syntax and semantics. After the model correctness analysis with help of available tools, it can be taken as an axiom that the model is correct. Hence, the problem reduces by development of tool, which can generate AST tree and then generate SystemC code from this tree. To solve the textual representation analysis problem by Terence Parr and others the ANTLR [11] tool is implemented. ANTLR takes as input a grammar that specifies some language and generates as output source code for a recognizer for that language. Also AST tree is automatically generated by ANTLR. The task is further reduced, because earlier Michael Schmitt developed and implemented a parser for the SDL-2000 language, based on the ANTLR tool [12]. A peculiarity of this parser — ability to output the generated AST tree that displays the structure of SDL model. Therefore, this parser can be used as a basis for development of a tool that generates SystemC code from the AST tree.

Now some words about code generation. At present there are a limited number of code generators for SDL models. The most widely used of them are briefly described in [13-16]. An important detail, that these different products were originally created to solve a specific problems — simulation and testing. These problems can be solved by execution of simulators on computers or real physical devices. Note that in the latter case the generated code is intended to compile for specific hardware platform. SDL specification model is based on the concept of communication EFSM *(Extended Finite State Machines)* [3]. The generated code, which is reflection of the model, is also based on EFSM. This means that SystemC code should meet all concepts [3] of SDL language.

In all well known SDL code generation tools the generated code is not self-sufficient in sense that it only reflects the functionality of original SDL model. Functioning of this code becomes possible after dipping it into a modeling kernel that carries SDL language concepts, which are declared and implemented as functions. The functioning of SystemC code requires development of the custom modeling kernel. It is important that the SystemC language, which is developed as part of the C++ language and implemented as an independent library, already provides a powerful modeling kernel that carries the language concepts, many of them are in consistence with the SDL language [7]. So, our task is even more simplified and development of SDL modeling kernel means the design of the control functions, which implement the concepts of SDL language. It must be mentioned that the SystemC modeling kernel supports two types of modeling —an event-triggered modeling and time-triggered modeling. Since the SDL language uses [17] concept of event-triggered modeling, it is one more argument to possibility of solving problem of automatic SystemC code generation.

D. Application of the method

The method of SystemC code generation for SDL models is primarily intended for testing of SDL specification of protocol. Testing is performed by technology based on model insertion into test environment, which is responsible for sending/receiving test data and monitoring results of the simulation [8].

Simple application of this method is showed at the Fig. 2. This is point-to-point data transmission model. Dataflow is generated by special tester — a component written on SystemC. This tester contains a traffic generator and data flow receiver. SDL under test model describes a network protocol specification and contains two entities, each of which implement a similar protocol stack. First instance of protocol stack presented as transmission part and second instance presented as receiver part. These parts connected by channel model.

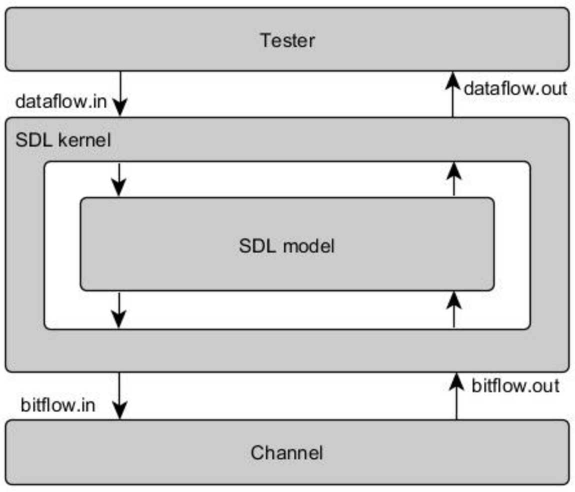

Fig. 2. Testing concept

First time the SystemC code for the SDL model must be generated. Since SDL core is also written on SystemC, the core simply handles input flow and executes the SDL model. Main property of proposed method is that a need of development of several SystemC-SDL wrappers and converters is absent now. SDL model receives, processes this flow and send performed bit flow to another SystemC component — channel. Main purpose of the channel is emulation of a real physical channel, which can introduce some errors during data transmission. In this case SystemC channel implemented in a way that some chosen bits purposely corrupted and then bit flow with correct bits and corrupted bits is sent again to SDL model. Owing to SDL kernel, SDL model receives this flow, processes and sends back to tester which performs a checking of the received data with transmitted data. If these data is not equal – tester writes error message in log file.

III. CONCLUSION

The proposed technology allows reducing of model testing time by automatically SystemC code generation from SDL models. Moreover, this technology allows performing SDL and SystemC co-modeling by simple integration of SDL models with test environment. Summarising the results of this work, it can be concluded that the presented technology allows using efficiently both of SystemC and SDL languages for protocol specification design.

REFERENCES

[1] Object Management Group, *Model Driven Architecture Guide*, Version 1.0.1, 2003.

[2] D. S. Frankel, *Model Driven Architecture: Applying MDA to Enterprise Computing*. John Wiley and Sons., 2003.

[3] International Telecommunication Union, *Recommendation Z.100. Specification and Description Language (SDL)*, Geneva, 2002.

[4] Black, D., & Donovan, J., *SystemC: From the Ground Up,* New-York: Springer Science+Buisness Media, Inc, 2004.

[5] Sergey Balandin, Michel Gillet, Irina Lavrovskaya, Valentin Olenev, Alexey Rabin and Alexander Stepanov, *Co-Modeling of Embedded Networks Using SystemC and SDL*, IJERTCS journal.

[6] Alexander Stepanov, Irina Lavrovskaya, Valentin Olenev, Alexey Rabin, Sergey Balandin, Michel Gillet, *SystemC and SDL Co-Modelling Implementation.* Proceedings of 7th Conference of Finnish-Russian University Cooperation in telecommunications (FRUCT) Program, pp. 130-137, Saint-Petersburg: Saint-Petersburg University of Aerospace Instrumentation (SUAI), 2010.

[7] Alexander Stepanov, *Comparison of SDL and SystemC Languages applicability for the protocol stack modeling*. Proceedings of the Saint-Petersburg University of Aerospace Instrumentation scientific student's conference (pp. 76-80). Saint-Petersburg: Saint-Petersburg University of Aerospace Instrumentation (SUAI), 2009.

[8] Alexander Stepanov, Irina Lavrovskaya, Valentin Olenev, *SDL and SystemC co-modelling: the protocol SDL models Tester*. Proceedings of 8th Conference of Finnish-Russian University Cooperation in telecommunications (FRUCT), 2010.

[9] SDL forum, http://www.sdl-forum.org.

[10] Joel Jones, *Abstract Syntax Tree Implementation Idioms*.

[11] ANTLR tool by Terence Parr, http://www.antlr.org.

[12] Michael Schmitt, *The Development of a Parser for SDL-2000*, Institute for Telematics, Germany.

[13] Leiming Chen, *Code Generation from Cinderella-SDL to Embedded Platforms*, Thesis in partial fulfilment of the degree of Master in Technology in Information and Communication Technology, 2007, Norway.

[14] Thomas Kolloch, *Scheduling with Message Deadlines for Hard Real–Time SDL Systems*, 2002, Germany.

[15] PragmaDev, *Real Time Developer Studio V4.0 Tutorial*.

[16] Joachim Fischer, Toby Neumann, and Anders Olsen, *SDL Code Generation for Open Systems*, Berlin, Germany.

[17] Annette Muth, *SDL-based Design of Application Specific Hardware for Hard Real-Time Systems*, 2002.