Journal of Computer Science and Cybernetics, V.28, N.4 (2012), 323–332

EMBEDDED SYSTEM DESIGN AND CODE GENERATION BY USING THE DSL AND T4*

PHAM VAN HUONG, NGUYEN NGOC BINH

University of Engineering and Technology Vietnam National University, Hanoi, Vietnam;
Email: huongpv@gmail.com

Tóm tắt. Trong xu hướng phát triển mạnh mẽ của công nghệ nhúng, các phương pháp thiết kế hệ thống nhúng cũng được nghiên cứu và triển khai rộng rãi. Bài báo này trình bày cách tiếp cận mới để thiết kế và sinh mã cho hệ thống nhúng dựa trên ngôn ngữ miền xác định và công nghệ sinh mã T4. Chúng tôi định nghĩa ba ngôn ngữ miền xác định, xây dựng siêu mô hình tương ứng và phát triển công cụ để thiết kế mô hình kiến trúc, mô hình thành phần của hệ thống nhúng và mô hình hoạt động của các thành phần phần mềm trong hệ thống nhúng. Từ mô hình thiết kế, dựa trên công nghệ sinh mã T4 để sinh mã theo các ngôn ngữ khác nhau.

Abstract. In the development trend of embedded technology, the methods of embedded system design are studied and widely utilized. This paper presents a new approach to design embedded system and generate code from models based on Domain Specific Language and Text Template Transformation Toolkit. We define three Domain Specific Languages, build the corresponding meta-models and develop our framework. The framework is to design the models of embedded system such as the architectural model and the component model. It is also to design the flow chart of embedded software. Based on the designed models, we apply the Text Template Transformation Toolkit to generate code from models automatically.

Keywords. Design Embedded System, Embedded Software, DSL (Domain Specific Language), Code Generation, T4 (Text Template Transformation Toolkit), SoC (System on Chip)

1. INTRODUCTION

Nowadays, embedded systems technology have strongly developed and the embedded system design plays an important role in the embedded system development. A few authors have improved UML 2.0 to support on designing the embedded system [11, 13]. However, UML 2.0 has the following limitations:

- Each research group develops a concrete UML 2.0 profile. There is not still a general normalization of UML for embedded system.
- UML tools store the model file in the different formats so it is not portable among these UML tools.

*This research is partly supported by a VNU scientific project (group A) for 2012-2013

- There is not a general normalization for generating the code from model. Each UML tool has a concrete code generation method.
- UML is a multi-purpose language so it is not complete to specify detail information of embedded systems.

In order to solve this problem, we propose an approach to design models of the embedded systems and automatically generate code from the models based on DSL and T4. In recent years, because XML is applied widely for storing the model file and meta-model file, DSL and T4 have developed quickly. T4 code generation technology based on XML allows generating the code. The different languages can format generated code. DSL and T4 are prospect trends deployed in many fields [1, 8]. Figure 1 [9] shows comparison of the software development cost between the conventional methodology and the DSL – methodology. In this paper, we define three DSLs to design the architectural model, the component model and the flow chart. Then we build the framework that contains these defined DSLs. After that, we also integrate T4 templates of the framework to generate parameters and code from the models automatically. The paper is organized as follows: Section 2 – Related work; Section 3 – Defining the DSLs and developing the framework used to design the models of embedded system; Section 4 – Presenting experiments; Section 5 – Conclusion and future work.

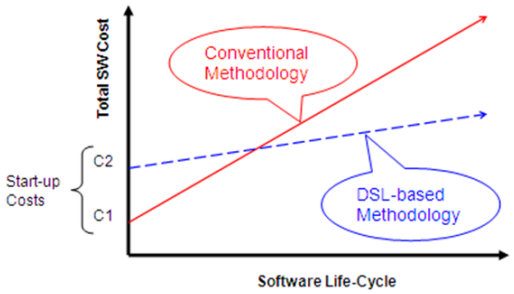

Figure 1. The comparison on the software development cost

2. RELATED WORK

In paper [2, 4], there are some UML 2.0 tools which are developed to model embedded system. For example, SYSML tool regards to embedded systems and SoCs particularities, there are quite similarities between the methods used in the area of System Engineering and complex SoC design, such as the need for precise requirements management, heterogeneous system specification and simulation [5, 8]. One of the major contributions of SYSML in the area of embedded system and SoCs is the support for requirements of modeling. We have studied the other tool that is UML-SOC [7, 12]. The profile of this tool intends to describe SoCs using the UML. UML-SOC focuses on the structural diagrams. It proposed the stereotypes that allow the structural modeling, communication modeling, operation and property modeling. Although there is some UML 2.0 tools but these tools have separated UML profiles and they also support to design the different kinds of model. Moreover, most of UML tools do not support optimal techniques at the model level.

There are also some studied using the DSL to design model of the embedded system now [3]. For example, the paper [8] defined DSL and developed the framework to specify and design

real time embedded system. In the paper [1], the authors also studied and developed the DSL to co-design hardware and software for FPGA. Based on these results, we propose a new approach to embedded system design using DSL presented in the next sections.

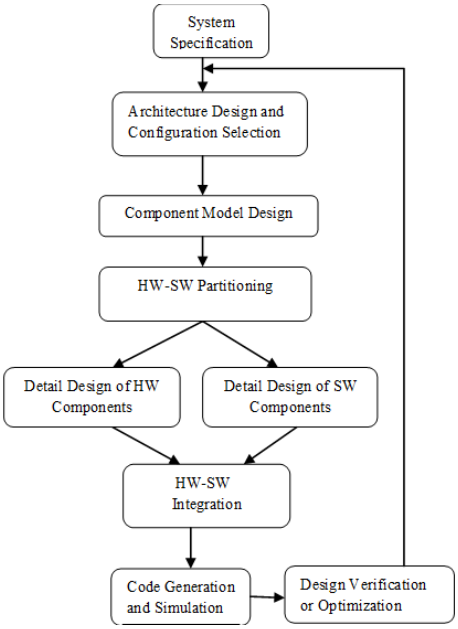

Figure 2. The design process of embedded system

3. DEFINING THE DSLS AND DEVELOPING THE FRAMEWORK

In this paper, we apply the design process of embedded system as shown in Figure 2. First, we define a DSL to design the architectural model of the embedded system. From the architectural model, we use T4 to generate parameters, which are to have multi-objective optimization for embedded systems on other our research. Second, in order to design detail of the embedded system, we divide the system into components. These components can belong to software components or hardware components. We define the second DSL and build the meta-model for designing the component model of the embedded system. Based on the component model, we optimize the embedded systems by hardware-software partitioning. Finally, we define the third DSL and build the meta-model to design the flow chart of each software component. Any languages can be automatically generated the code of software components of the embedded system. We will deploy the definition of DSL for designing the hardware of embedded systems in further research.

3.1. Defining the DSL and building the meta-model for designing the architectural models

There are some types of the embedded system architecture but the basic architecture of the embedded system is as shown in Figure 3. An embedded system normally consists of CPU, RAM and ROM, instruction cache, data cache, input ports and output ports [1, 6]. The components of the system communicate together through the bus system, which includes the

system bus and the local bus. Based on the basic architecture of embedded systems, we define and build a DSL by the following steps:

- Define the logical classes used to express the meaning of the elements and the relationships between two elements as shown in Table 1.
- Define the visual classes used to express graphical elements in our framework. Each visual class is corresponding to a logical class. The visual classes are shown in Table 1.
- Create the XML file that stores the definitions and links between the logical classes and the visual classes. We use the Visual Studio.NET 2010 to build the meta-model as shown in Figure 4.

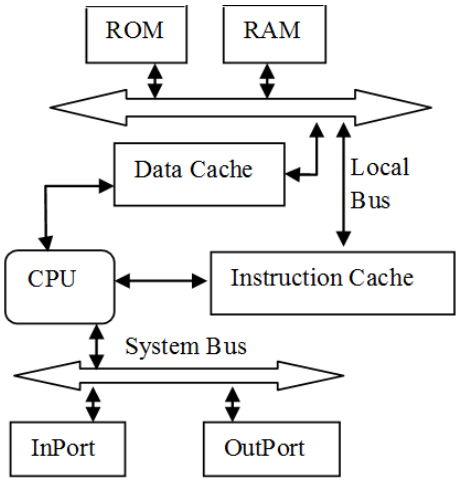

Figure 3. An architecture of embedded systems

Table 1. The main classes of DSL for the architectural model

Logical Classes	**Shape Classes**
ESArchitectureModel	ESArchitectureModelDiagram
CPU	ComponentShape
RAM	RAMShape
ROM	ROMShape
Cache	CacheShape
BusLocalCPU_Cache	BusLocalCPU_CacheShape
BusLocalMem_Cache	BusLocalMem_CacheShape
InPort	InPortShape
OutPort	OutPortShape
BusSystem	BusSystemShape
Comment	CommentShape

3.2. Defining the DSL and building meta-model to design the component models

In this section, we define and develop a DSL for designing the component model of embedded systems. The component model is also to express the architectural aspect of the embedded

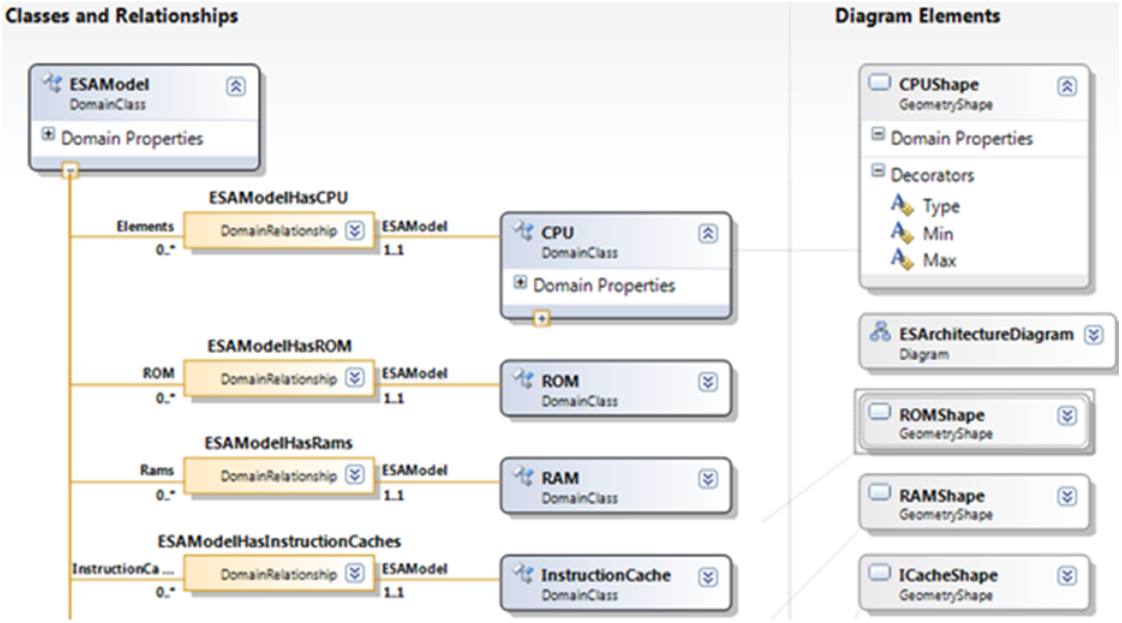

Figure 4. A part of the meta-model of DSL to design the architectural model of embedded systems

systems [5]. We do the same steps as in the previous section to define the elements of this DSL as shown in Table 2. We build the meta-model as shown in Figure 5.

Table 2. The main classes of the component model - DSL

Logical Classes	Shape Classes
Model	ESComponentDiagram
Component	ComponentShape
InPort	InPortShape
OutPort	OutPortShape
ComponentLinkComment	LinkShape
Comment	CommentShape
OutPortLinkToInPort	OutPortLinkToInPortShape

3.3. Defining the DSL and building the meta-model to design flow charts

By the same way as in the previous sections, we define the third DSL and build the corresponding meta-model to design the flow charts of the software components of the embedded system. This DSL aims to design dynamic aspect of the software components. Based on the flow chart, the source code of the embedded software components can be generated by T4 in any languages automatically. We define this DSL as shown in Table 3 and build the meta-model as shown in Figure 6.

3.4. Applying the T4 to generate code automatically from model

In Visual Studio.NET, a T4 text template is a mixture of text blocks and the control logic that can generate a text file [7]. We can write the control logic as the fragment of program code

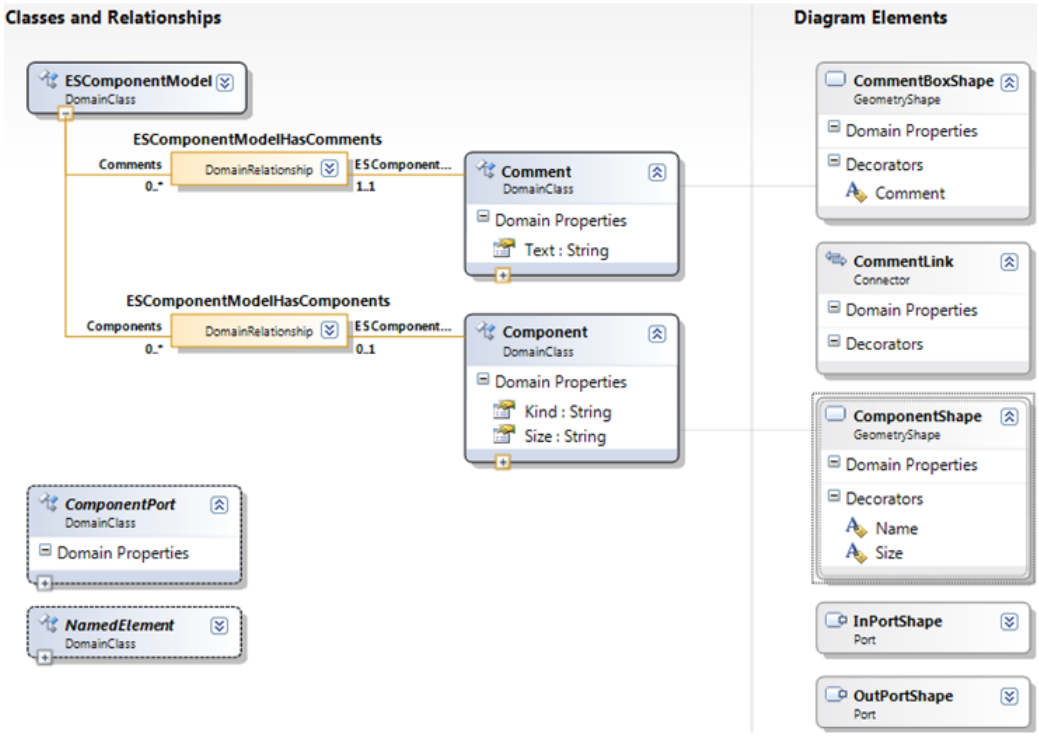

Figure 5. A part of the meta-model of DSL used to design component models

Table 3. The main classes of the flow chart - DSL

Logical Classes	Shape Classes
FlowChart	FlowChartDiagram
Component	ComponentShape
Element	ElementShape
Operation	OperationShape
OperationIO	OperationIOShape
Start	StartShape
End	EndShape
Branch	BrachShape
Synchronization	SynchronizationShape
Comment	CommentShape

in C# language or Visual Basic.NET language. T4 is the part of our framework to automate the creation of text files with a variety of parameters. These text files can ultimately be considered as any text format such as C# code, Java code, XML, HTML or XAML [13]. In this paper, we apply T4 to generate source code and parameters automatically from three kinds of DSL models defined and developed by us. We will present some experiments of the design and code generation in the next section.

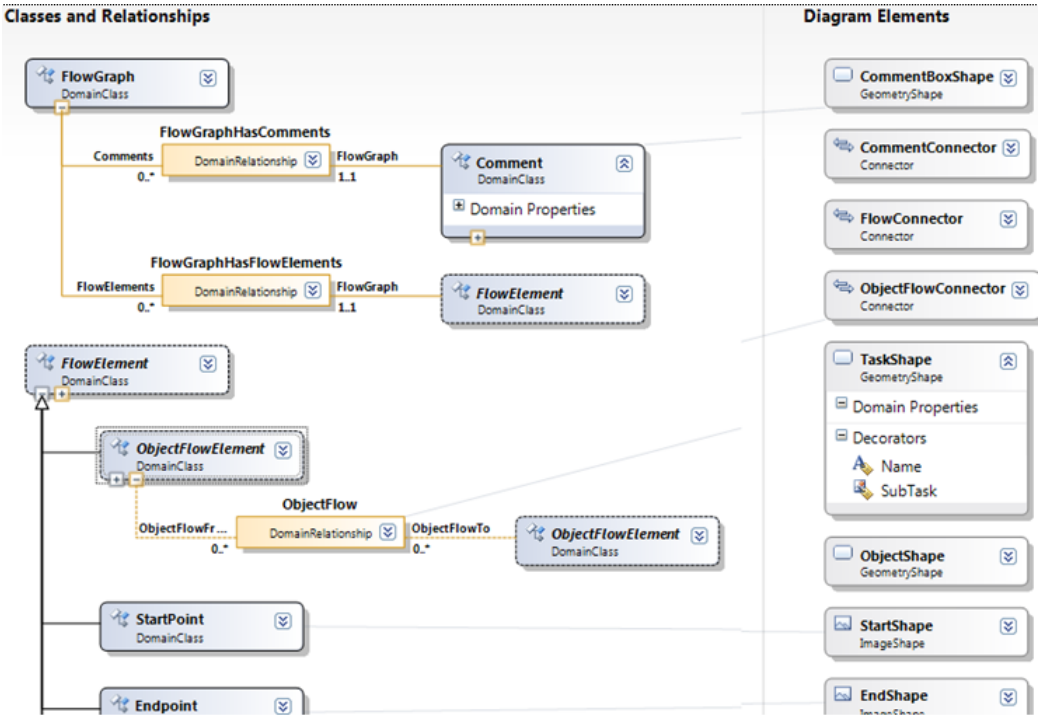

Figure 6. A part of meta-model of DSL used to design the flow chart

4. EXPERIMENT

As mentioned in the previous sections, we have defined three DSLs and built the DSL framework to design the embedded system and the software component. In this experiment, we will design and generate code for an embedded system. The embedded system is Telegraph used to connect a printer to a network. This embedded system is as shown in Figure 7 [10]. It includes functions as follows: receive data from network; copy data to serial port of printer; sort unordered data packets and provide a clean data stream to printer; feed printer one print job at a time and hold off all other computers; respond rapidly to certain events; keep trace of time. In the next section, we will present steps to design and generate code for this system.

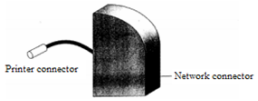

Figure 7. Telegraph embedded system

4.1. Designing the architecture mode and automatically extracting the information

We have defined a DSL, built the meta-model and created a tool used to design the architectural model of embedded system. Using our tool, designers can build the architecture of embedded systems and automatically generate the configuration information of the embedded system from the architectural model. The configuration information includes type of compo-

nents in an embedded system, the minimal value and the maximal value. This information is to solve the Pareto optimization problem deployed in our further research. Figure 8 shows the graphical interface of our tool and the architectural model designed based on this tool. In order to generate parameters automatically from architectural models, we build T4 template and integrated it to this DSL framework. Figure 9 shows the information generated from the model by T4 automatically.

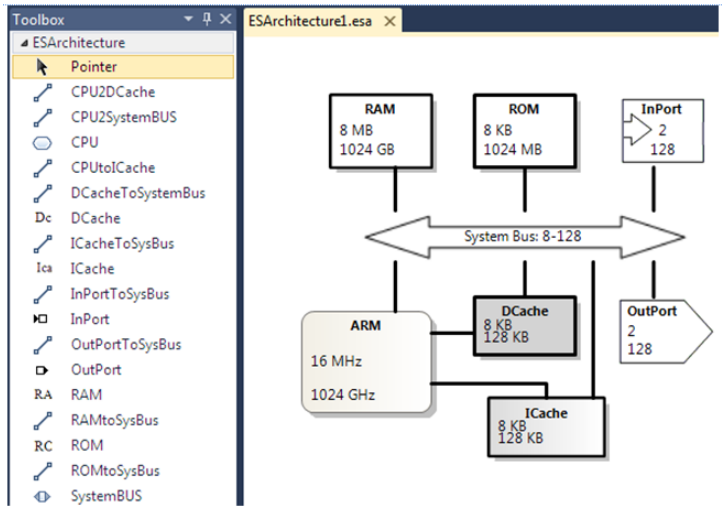

Figure 8. Toolbox and the architectural model

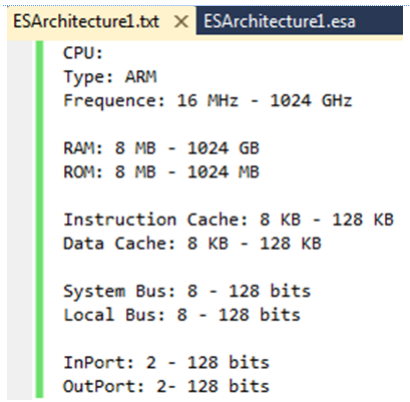

Figure 9. Information generated from model

4.2. Designing the component model and generating the code from the model

After designing the architecture of embedded system in the previous section, designers can use our framework to build the component model of the embedded system. Figure 10 shows the toolbox of our framework and the designed component model of the embedded system Telegraph. In addition, the generated code is as shown in Figure 11.

4.3. Designing the detail of software components and generating the code

After designing the component model, designers can build more details of each software component by the flow chart. In Figure 12, we use our developed framework to design the flow chart of the SortDataPackets component in Telegraph embedded system. Figure 13 shows the generated code in C language.

5. CONCLUSION AND FUTURE WORK

In this paper we have proposed a new approach to support designing the embedded system and generating the code from models of embedded systems. Based on this approach, we have done the experiment of building three Domain Specific Languages used to design the architectural model, the component model and the flow chart of embedded systems and software components. In order to do the experiment, we first define the domain classes, the visual

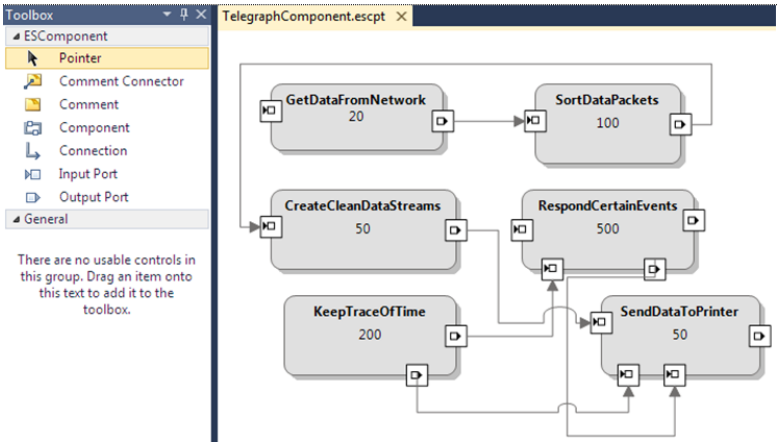

Figure 10. Toolbox and the component model

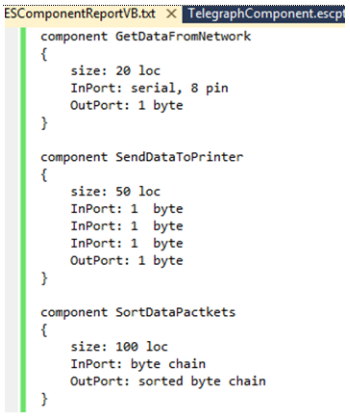

Figure 11. Code generated from the component model

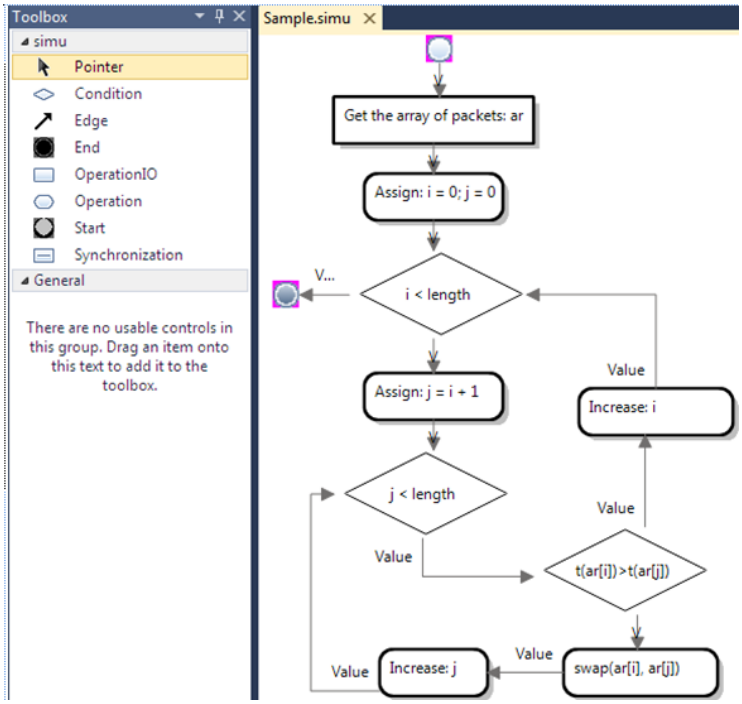

Figure 12. The flow chart of the SortDataPackets component

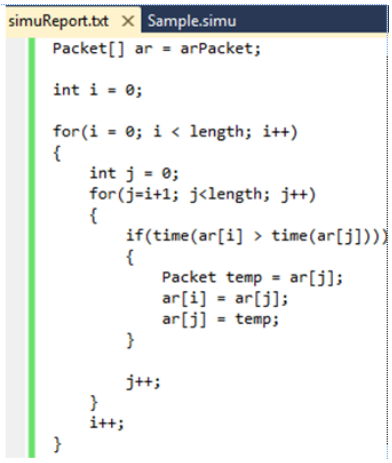

Figure 13. C code generated from the flow chart

classes and the relationships among the classes. Then we use the Visual Studion .NET 2010 to design and build the meta-model. Finally, we build and integrate T4 templates used to generate the code automatically.

Moreover, embedded systems normally have constraints such as performance, memory size, the battery lifetime and the real time. In addition, these constraints may be different from the embedded system types. Therefore, the embedded systems are high specific and they need the specific languages to design embedded system. Because the meta-model is the model used

to create other models, the meta-model and DSL are specific and flexible to design embedded systems and embedded software. This is the prospective trend in the embedded system development. In further study, we will develop and improve these DSLs to suit constrains of embedded systems and embedded software. After that, we will study optimization problem of embedded systems and embedded software based on DSL and T4.

REFERENCES

[1] J. Agron, Domain-Specific Language for HW/SW Co-design for FPGAs, *Proceedings of the IFIP TC 2 Working Conference on Domain Specific Languages*, Springer-Verlag, Berlin, Heidelberg, 2009 (262–284).

[2] D. B. S. Christina, D. Fröhlich, Synthesis of UML-Models for Reconfigurable Hardware, *Proceedings of the 8th International Conference CADSM*, 2005 (260–267).

[3] W. M. K. Eugen, G. Kalus, A Domain Specific Language for Project Execution Models, *Proceedings of the 39th Annual Conference of the German Computer Society*, 2009 (35–44).

[4] D. Fruhlich, "Object-oriented development for reconfigurable architectures", PhD thesis, Freiberg University of Mining and Technology, Germany, 2001 (1–273).

[5] D. Gajski, F. Vahid, Specification and Design of Embedded Software/Hardware Systems, *IEEE Design & Test of Computers* **12** (1995) 53–67.

[6] A. Gherbi, F. Khendek, UML Profiles for Real-Time Systems and their Applications, *Journal of Object Technology* **5** 149–169.

[7] E. C. L. W. S. A. S.Gregor, B. Westfechtel, *Graph transformations and model-driven engineering*, in Gregor Engels; Claus Lewerentz; Wilhelm Schufer; Andy Schürr & Bernhard Westfechtel, ed., Springer-Verlag, Berlin, Heidelberg, 2010 (580–601).

[8] K. Hammond, G. Michaelson, Hume: A Domain Specific Language for Real-Time Embedded Systems, *Proceedings of Generative Programming and Component Engineering (GPCE '03), Lecture Notes in Computer Science*, Springer-Verlag, 2003 (37–56).

[9] P. Hudak, "Little Languages for Big Applications, Department of Computer Science", Yale University, 2003 (1–44).

[10] D. E. Simon, *An Embedded Software Primer*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[11] J. A. K. Vaibhav, P. R. Panda, A SysML profile for development and early validation of TLM 2.0 models, *Proceedings of the 7th European conference on Modelling foundations and applications*, Springer-Verlag, Berlin, Heidelberg, 2011 (296–311).

[12] B.Willard, UML for systems engineering, *Comput. Stand. Interfaces* **29** (1) (2007) 69–81.

[13] M. A. R. S. B. E. R. P. S. Y. V. Wolfgang, W. Dehaene, UML for ESL design: basic principles, tools, and applications, *Proceedings of the International Conference on Computer-Aided Design*, ACM, New York, NY, USA, 2006 (73–80).

Received on September 06, 2012
Revised on December 11, 2012