

Embedded System Design and Code Generation by Using the DSL and T4

¹Pham Van Huong and Nguyen Ngoc Binh

University of Engineering and Technology

Vietnam National University, Hanoi, Vietnam

Abstract. In the strong development trend of embedded technology, design methods of embedded system also are researched and deployed widely. The 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 defined three Domain Specific Languages, built the correspondence meta-models and developed the framework which is used to design the architectural model, the component model and the flow chart of embedded systems. From designed models, we applied the Text Template Transformation Toolkit to automatically generate code from models.

Keywords: Embedded System Design, Embedded Software, DSL – Domain Specific Language, Code Generation, T4 – Text Template Transformation Toolkit.

1. Introduction

Nowadays, embedded system technology has strongly developed so design plays an important role in the embedded system development. A few of researches have improved UML 2.0 to support on designing embedded system [5, 6, 8, 9]. However, UML 2.0 has the following limitations:

- Each research group develops a concrete UML 2.0 profile. There is not still a common standard of UML for embedded system
- UML tools store model file in the different formats so it is not portable among these UML tools
- There is not a common standard for generating code from model. Each UML tool has a concrete code generation method
- UML is a multi-purpose language so it does not completely specify detail information of embedded systems.

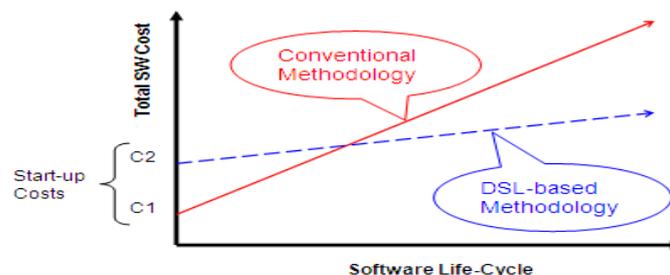


Fig. 1: The comparison on the software development cost

To solve this problem, we propose the approach in which DSL and T4 are used to design models of the embedded system and automatically generate code from models. In the recently years, because of the application of XML for storing model file and meta-model file, DSL and T4 have developed widely. T4 code generation technology based on XML allows strongly generating code. Generated codes can be formatted by the different languages. DSL and T4 are prospect trend and they are deployed in many fields [10, 11]. Fig. 1 shows the comparison on the software development cost between the conventional methodology and the DSL-based methodology. In this research, we will define three DSLs to design the architectural model, the component model and the flow chart. Then we will build the framework which contains these defined DSLs. And we will also integrate T4 templates to the framework to automatically generate parameters and code from the models. The rest sections of this paper are arranged as follows: *Section 2* – Defining DSLs and developing framework which is used to design models of embedded system; *Section 3* – We present an experiment; *Section 4* – Conclusion and future work.

2. Defining DSLs and developing the framework

In this paper, we firstly define a DSL to design the architectural model of embedded system. From the architectural model, we use T4 to automatically generate parameters which are used to do multi-objective optimization for embedded systems on the other research of ours. After that, to show more detail of the system architecture, we divide the system into components. These components might belong to software or hardware. We define the second DSL and build the meta-model for designing the component model of the system. Based on the component model, we will do optimization of embedded systems based on partitioning hardware – software. Finally, we define the third DSL and build meta-model to design flow chart of each component. The code of software components of the embedded system can be automatically generated by any languages. The definition of DSL for designing hardware of embedded systems will be deployed in the further researches.

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

There are some types of embedded system architecture but the basic architecture of the embedded system is described in Fig. 2. An embedded system normally includes CPU, RAM, ROM, instruction cache, data cache, input ports and output ports [1-4]. The components of the system communicate through the Bus system which includes the system Bus and the local Bus.

Based on the basic architecture of embedded system, we define a DSL and build the correspondence meta-model as following steps:

- Defining logical classes: to express meaning of the logical elements and their relationships as in Table 1
- Defining visual classes: the visual classes express the graphic interface elements which are used to design. These classes are text classes or shape classes. 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 domain classes and the shape classes. This XML file is the meta-model file that is built by us and is packaged to design the architectural model of embedded system. Using Visual Studio.NET 2010, we built meta-model as Fig. 3.

2.2. Defining a DSL and building meta-model to design the component model and the flow chart

In this section, the first we aim to define and develop a DSL for designing the component model of embedded system. The component model is also used to express the architectural aspect of the embedded system [3]. We do the same steps in the section 2.1 to define elements of DSL and build the correspondence meta-model.

To express dynamic aspect of the component, we developed the DSL to design the flow chart of a component. Based on the flow chart, the code of the embedded system can be automatically generated by T4 on any languages. We also completely defined the DSL and built the meta-model of the flow chart.

2.3. Applying T4 to generate code from model

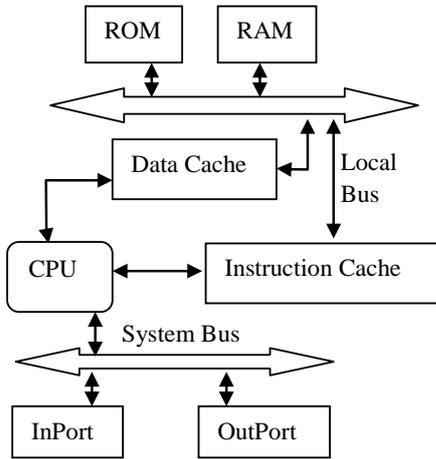


Fig. 2: Architecture of embedded system

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

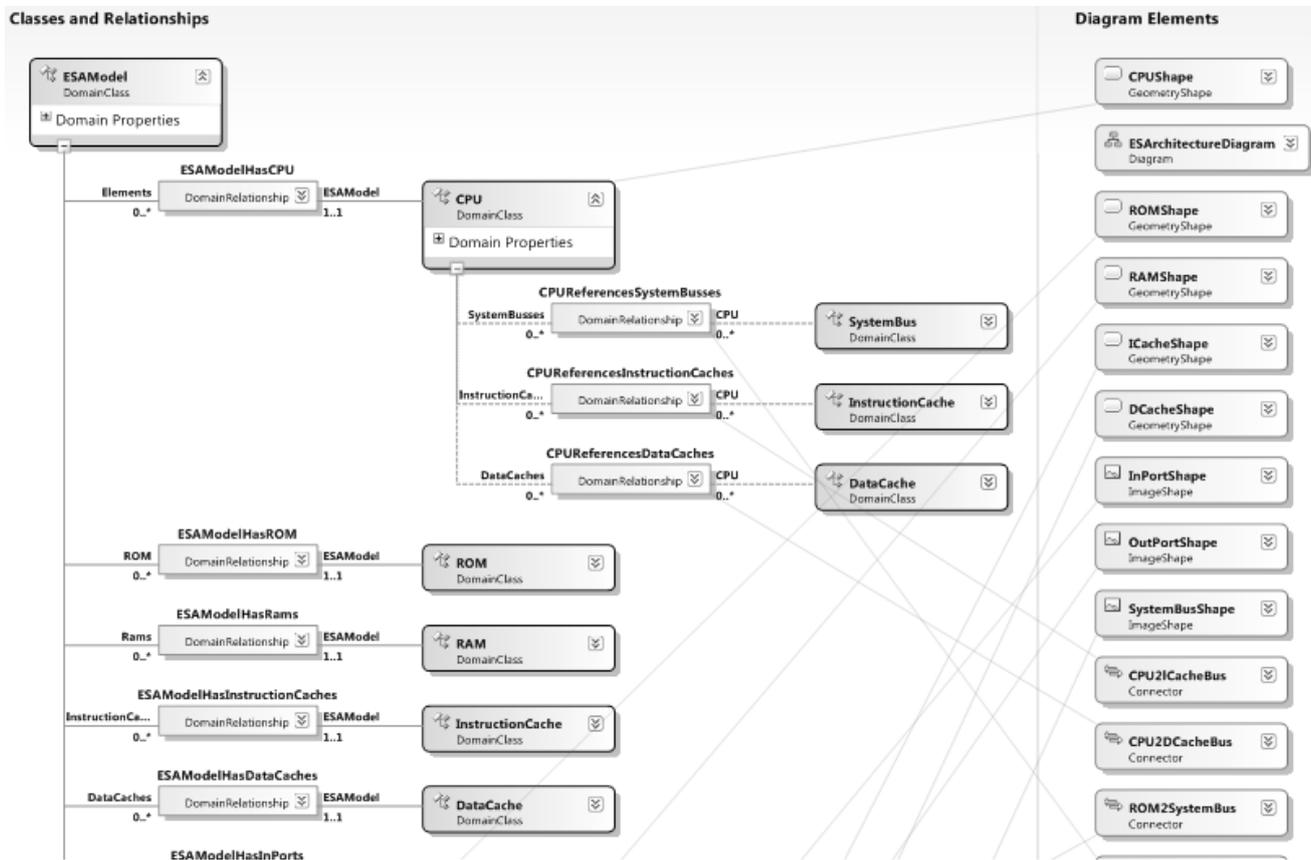


Fig. 3: A part of meta-model of DSL for embedded system architecture

In Visual Studio, a *T4 text template* is a mixture of text blocks and control logic that can generate a text file. The control logic is written as fragments of program code in Visual C# or Visual Basic.NET. T4 is used by developers as part of a framework to automate the creation of text files with a variety of parameters. These text files can ultimately be any text format, such as C# code, Java code, XML, HTML or XAML [7]. In this paper, we apply T4 to automatically generate code and parameters from three kinds of DSL model which are defined and developed by us. The experiment of design and generation will be presented in the next section.

3. Experiment

As mentioned in the previous sections, we have defined three DSL and completely built DSL framework to design embedded system. In this experiment, we will design and generate code for an embedded system. Here the embedded system is Telegraph which is used to connect a printer to a network. This embedded system 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.

3.1 Designing architecture mode and automatically extracting information

In the previous section, we have defined a DSL, built meta-model and created a tool which is used to design architecture model of embedded system. Using our tool, designers can design the architecture of embedded system and automatically generate configuration information of embedded system from the architectural model. This information including type of components, minimal value and maximal value is used to do Pareto optimization and choose the best configuration. Fig. 4 shows graphical interface of the tool and the architectural model which is designed based on this tool.

To automatically calculate parameters from architectural model, we built T4 template and integrated it to this DSL framework. Fig. 5 shows the information that is automatically generated from model by T4. Parameters are calculated based on this information.

3.2 Designing the component model and generating code from the model

After designing the architecture of embedded system in the previous section, designers can use our framework to design the component mode of embedded systems. Fig. 6 shows the toolbox of this framework and the designed component model of Telegraph embedded system. And the generated code is shown in Fig. 7.

3.3 Designing detail of a software component and generating code

After designing the component model, designers can design more detail of each component by the flow chart. In Fig. 8, we use the developed framework to design the flow chart of the *SortDataPackets* component in the Telegraph embedded system. Fig. 9 shows the generated C code.

4. Conclusion and future work

In summary, in this paper we proposed and developed a new approach to design embedded system and generate code from the models of embedded system. Theoretically, we define three Domain Specific Languages which are used to design the architecture, the component model and the flow chart of embedded systems. This is the prospect trend in the embedded system development. Conclusions to the experiment, we completely built the framework which includes three DSLs and the integrated T4 templates used to generate code.

Based on this research, we will do further researches for design and optimization of embedded system such as developing DSL to design hardware components, do multi-objective optimization of embedded system to select the best configuration of the system architecture and optimize embedded system at model level based on partitioning hardware – software.

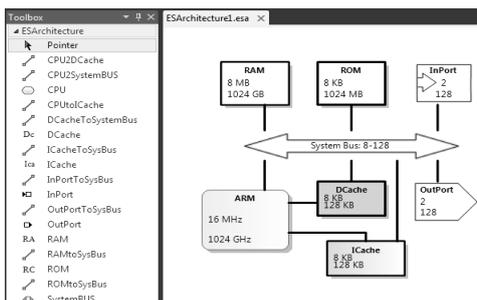


Fig. 4: Toolbox and the architectural model

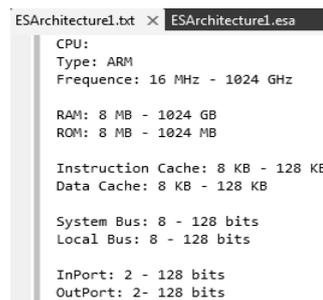


Fig. 5: Information generated

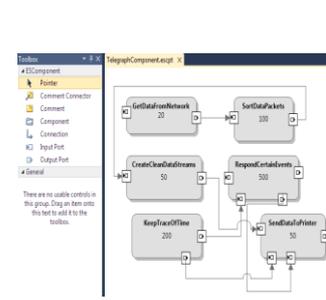


Fig. 6: Toolbox and the component model

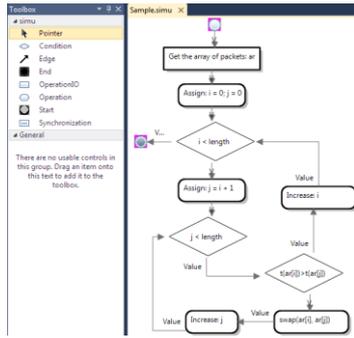


Fig. 8: The flow chart of the SortDataPackets component from model

```
ESComponentReportVB.txt x TelegraphComponent.escript
component GetDataFromNetwork
{
  size: 20 loc
  InPort: serial, 8 pin
  OutPort: 1 byte
}

component SendDataToPrinter
{
  size: 50 loc
  InPort: 1 byte
  InPort: 1 byte
  OutPort: 1 byte
}

component SortDataPackets
{
  size: 100 loc
  InPort: byte chain
  OutPort: sorted byte chain
}
```

Fig. 7: Code generated from the component model

```
simuReport.txt x Sample.simu
Packet[] ar = arPacket;
int i = 0;
for(i = 0; i < length; i++)
{
  int j = 0;
  for(j=i+1; j<length; j++)
  {
    if(time(ar[i] > time(ar[j])))
    {
      Packet temp = ar[j];
      ar[i] = ar[j];
      ar[j] = temp;
    }
  }
  j++;
  i++;
}
```

Fig. 9: C code generated from the flow chart

5. References

- [1]. C. Dorotska, D. Frohlich, and B. St einbach. *Synthesis of UML-Models for Reconfigurable Hardware*. In proceeding, 2nd UML for SoC Design Workshop at 42nd Design Automation Conference (DAC), Anaheim, California, 2005.
- [2]. D. Frohlich. *Object-Oriented Developm ent for Reconfigurable Architectures*. Dissertation. Von der Fakultat fur Mathematik und Informatic. Der Technischien Universitat Bergakademie Freiberg. 20. Juni 2007.
- [3]. D.D Gajski, F. vahid, S. Narayan, an d J. Gong. *Specification and Design of Embedded Systems*. Published by Prentice Hall. Englewood, Newjersey 07632. 1994.
- [4]. A.Gerbi and K. Ferhat. *UML Profiles for Real-Time Systems and their Applications*. Journal paper. JOT, vol. 5, no. 4, pp. 149-169, May-June 2006. http://www.jot.fm/issues/issue_2006_05/article5/
- [5]. OMG. *UML Profile for System on a Chip*. OMG Available Specification, version 1.0.1 formal /06-08-01, August 2006.
- [6]. OMG. *Systems Modeling Language (SysML) Specification*. OMG document: ad/2006-03-08-01, version 1. Draft, April 2006.
- [7]. Eugen Wachtel, Marco Kuhrmann, Georg Kalus, "A D omain S pecific L anguage for Project E xecution Models", 14 Boltzmannstr , pp.1-15, Germany 2007.
- [8]. Y. Vanderperren, and W. Dehaene, *SysML and Systems Engineering Applied to UML-Based SOC Design*, In Proc. 2nd UML-SOC Workshop at 42nd DAC, Anaheim (CA), USA, pp.1-13, 2005.
- [9]. OMG. *UML Profile for System on a Chip*. OMG Available Specification, version 1.0.1 formal /06-08-01, pp.5-15, August 2006.
- [10]. Kevin Hammond and Greg Michaelson, *A Domain-Specific Language for Real-Time Embedded Systems*, pp.37-56, GPCE 2003.
- [11]. Jason Agron, "Domain-Specific Language for HW/SW Co-design for FPGAs", pp.262-284, DSL 2009.