IEEE

10th ANNIVERSARY

AICT2016
International Conference
12-14 October 2016, Baku

www.aict.info/2016

APPLICATION OF INFORMATION AND COMMUNICATION TECHNOLOGIES - AICT2016

CONFERENCE PROCEEDINGS

12-14 October 2016, Baku, Azerbaijan

www.aict.info/2016

SESSION 3. CYBER SECURITY ISSUES

SESSION 4. LATEST TRENDS IN ICT APPLICATION

Using Model-Based Methods for Embedded Software Development

S.Jalilian, T.Boroumandnejad
Satellite Research Institute, Iranian Space Research Center
Tehran, Iran
shjalilian@gmail.com

Kazimov Tofig H.
Azerbaijan National Academy of Sciences Institute of Information Technology
Baku, Azerbaijan
tofig@mail.ru

***Abstract*—today, using of embedded systems and complexity of these systems are increased. As a result of this complexity, development of this systems based on model-based process is needed. But the Standard-UML based tools can't be used because there is significant difference between general–purpose software and embedded software. Therefore, the integrated tools must be used for covering non functional, scheduling characteristics of these systems. In this paper we will review three methods that are used for developing embedded software.**

***Index Terms*—Embedded systems, Model-based Development, UML, code generation.**

I. Introduction

Today, Embedded Computer systems are used in a wide range of systems from airbag control systems to automotive and airplanes. The design of these systems is a complex task because these systems include hardware and software components that must be highly optimized for the application needs special in real time embedded systems. Also, the increasing complexity of these systems has impact on its development life cycle. Because of these reasons, system development methodologies must be used in order to manage the team size, the requirement and project's constraints. Furthermore, embedded software is brain and intelligent part of these systems that controls the behavior of embedded systems, so it plays an important role in this type of systems due to its flexibility [4], [5].

Nevertheless to say, Model-based development methods that uses only in classical software engineering such as UML cannot be applied for creating embedded software and must be integrated with another tools. Because the development process of general-purpose software and embedded software is differ significantly. The embedded software is deal with non-functional issues such as performance, reliability, safety, timeline and concurrency in a physical context. Therefore the integrated methodology must be used for covering these issues [1], [7].

Standard UML-based modeling has several shortcomings. For example while deployment diagram attempt to provide a modeling capability for supporting component and system configuration but it is weak because it can't map component to processors, component to threads and so on. Also, UML cannot express system behavior under fault conditions or can't define concurrent behavior of component that is important in the embedded systems [2],[6].

Because of reasons that mentioned above, hybrid methodology or tools must be used for making embedded software. In this paper these kinds of methods will be reviewed.

II. Embedded Software Design Methodology

Some useful methods and techniques for embedded software development is explained in this section. These methods are as follow:

- *Embedded System Modeling Language(ESML)*
- *An Integrative approach with UML and Simulink*
- *Template-based code generation*

A. Embedded System Modeling Language(ESML)

ESML is a system modeling language for embedded systems that is used for integrated modeling of software component, component interaction, hardware configurations, scheduling policies and other aspects relevant for system developer. Procedure invocation via component and event propagation through public/subscribe are two mechanisms that is used in this language. These two mechanisms which is used in ESML are combined in a "push-directed-pull" interaction pattern. As illustrated in fig.1, in this mode of operation, subscribers are notified by a publisher that data is available for retrieving. After that, the subscribers will be triggered and a call "back" will be send to their facets by their receptacles. This technique provides more adaptive behavior to support non functional characteristic of embedded systems [2].

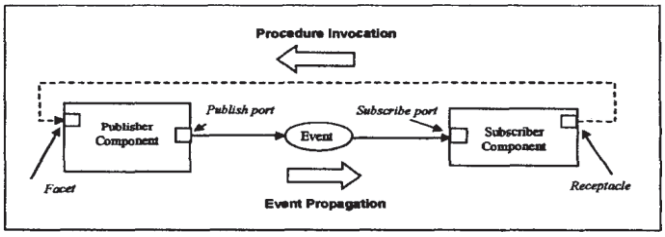

Fig. 1. Model of computation in ESML

Nevertheless, ESML isn't a replacement for UML or implementation language. It is in conjunction with these tools for designing embedded software. Figure 2 illustrates how this is done. In development process of this kind of software, components are designed by Rational Rose and header files are generated through its code generator. UML model must be annotated with "tagged values". Next, annotated model will be import in ESML modeling environment and ESML component will be construct by interprets the annotations. Now, the modeler can construct other models for example interaction diagrams and scheduler diagram [2].

B. An Integrative approach with UML and Simulink

This approach tries to integration of exiting artifact like a C legacy code and the migration to a model-based development. Specially, it shows when modeled and non-modeled artifacts exist how UML & Simulink is used for functional specification and automatic code generation for control device. Because of using UML for modeling the architecture, simulation and integration aspects of embedded software artifacts, this method defines an extensible UML profile. This profile has different stereotypes such as Matlab, LegacyClass to reference several artifacts such as Matlab function, legacy C code. The process of development steps is given in Fig. 3. A system developer designs new capability based on exiting artifacts and models. Next, this artifact must be integrated into the architecture view of UML model. The legacy code, network models and real time operating system (RTOS) models are directly included in the UML via references. Finally, code will be generated automatically. This approach uses a stepwise migration model-based development due to the fact that industries can't switch-over immediately from C-code to model-based development.

This method is useful for existing embedded systems that has huge amount of legacy C-code and want switch to model-based development as soon as possible [8].

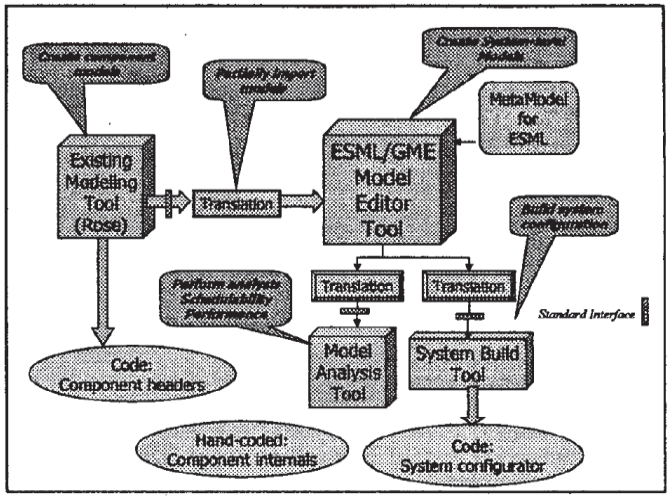

Fig. 2. The process of using ESML

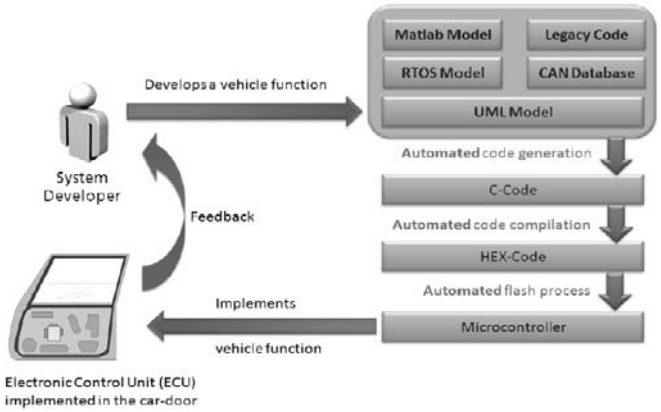

Fig. 3. Process of automated development step

C. Template-based code generation

There are many repetitive problems in the context of embedded systems like process management, scheduling and so on. Solutions for these problems exist but the heterogeneity of embedded software doesn't allow reusing components. This approach, as shown in Fig. 4, uses a technique like preprocessor macros for solving this problem which is called template-based development. This method use application-independent templates instead of generating machine code from an application model or attempt to reuse precompiled libraries. One of the advantages of this approach is that it can be adapted automatically to the application requirements on the base of the model. Also flexibility is big advantage of this method [3].

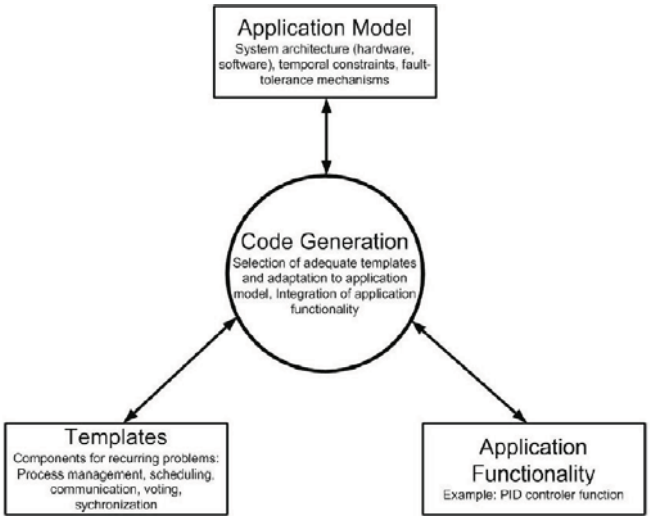

Fig. 4. Code Generation Architecture

The code generation process steps for developing embedded software as shown in Fig. 5 is as bellow:

- First, developer specifies the model of system. The model must be included the information about the software architecture, the tasks of application and timing constraints.
- The information about non-functional aspects like OS, programming language, performance is also needed for selecting the right template.
- In the next step, the correctness of model will be check by model parser and the information will be store into database.
- The second step for developer is to implement the application code like control function. This task can be performed by other model-based tools such as Matlab/Simulink.
- At the last step, the parser combines the generated code with implemented code by the developer. The result is source code that can be compiled and executed on any platform [3].

III. Conclusion

As we mentioned in this paper, to cope with complexity, uncertainly and encompassing heterogeneity of embedded systems, model-based approaches are more convenient than traditional approaches. General-purpose model-based development is an ongoing approach for developing embedded software. It can't cover all aspects of embedded software such as non-functional characteristics; therefore new approaches such as integrative or template-based approach have been provided to cope with UML weaknesses. In this paper 3 approaches were discussed. The advantages of these approaches are high-level modeling of the software, possibility facilitating to achieve early simulation and formal verification based on model and flexibility in source code generating. Main problem in this area is tools support. Although UML is widely supported by software development tools, tool facilities for developing embedded software are not as rich as UML. There are some commercial tools such as IBM Rhapsody or Eclipse model-based development tools which support UML for embedded software development. They help developer to cope with software complexity but to cover non-functional aspects of embedded software development, advanced model-based methods must be included.

In the future, we are going to inspect aspect-oriented approaches for developing embedded software, its advantages and disadvantages.

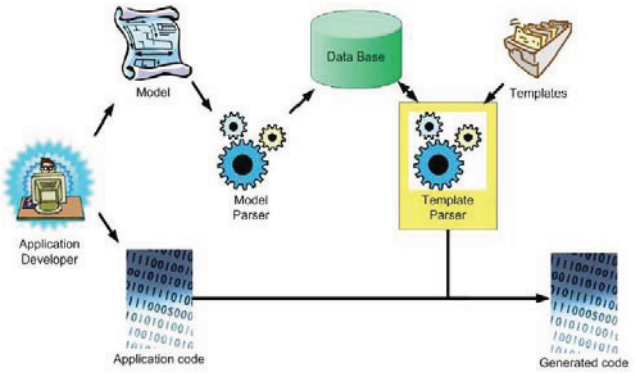

Fig. 5. Code Generation Step

References

[1] B. Dion, Th. Le Sergent, B. Martin, "Model-based development for Time-Triggered Architectures," the 23rd digital Avionics Conference, vol. 2, pp.6. D.3-6.1.7, IEEE, Oct 2004.

[2] G. Karsai, S. Neema, B. Abbott, D.Sharp, "A Modeling Language and its supporting Tools for Avionics Systems", the 21rd digital Avionics Conference, vol.1, pp.6A3.1-6A3.13, IEEE , 2002.

[3] Ch. Buckl, A. Knoll, and G. Schrott, "Model-Based development of Fault-tolerant Embedded Software", second international symposium on leveraging applications of formal methods, IEEE, pp. 103–110, Nov 2006.

[4] L. Cordeiro, C. Mar, E. Valentin, F.Cruz,"A Platform-Based Software Design Methodology for Embedded Control System: An Agile Toolkit,"15th annual IEEE International Conference and workshop on the Engineering of Computer Based Systems, pp. 408–417, Apr 2008.

[5] M. A. Wehrmeister, C. E. Pereira, L. B. Becker, "Object-Oriented to the Development of Embedded Real-Time Systems," 8th annual IEEE International symposium on object-oriented Real-Time Distributed Computing, pp.125-128, May 2005.

[6] M. A. Wehrmeister, C. E. Pereira, F. Rammig, "Aspect-Oriented Model-Driven Engineering for Embedded Systems applied to Automation Systems," IEEE Transactions on Industrial Informatics, vol.pp, jan 2013 .

[7] Ph. Conmy, R. F. Paige, "Challenges when using Model Driven Architecture in the development of Safety Critical Software,"4th international workshop on Model-Based Methodologies for pervasive and embedded software, IEEE, vol. 2, pp. 127–136, March 2007.

[8] T. Farkas, C. Neumann, A. Hinnerichs, "An Integrative Approach for Embedded Software Design with UML and Simulink,"33rd Annual IEEE International Conference on Computer and Software, vol. 2, pp.516-521, July 2009.