Approaches of using UML for Embedded System Design

Sudeep D. Thepade
Lecturer,
Dept. of Information Technology,
Thadomal Shahani Engg. College,
Bandra, Mumbai
sudeepthepade@gmail.com

Swapnil D. Thepade
M.E. (Appeared)
CAD/CAM and Robotics,
Fr. Conceicao Rodrigues College of Engg,
Bandra, Mumbai
friendlyswapnil@gmail.com

Abstract

New approaches are required for analysis, design and specification of Embedded Systems, beyond conventional boundary of hardware and software methods. As per demand of embedded systems these approaches should allow one to combine hardware and software design. Now a days software plays major role in embedded systems. Hence latest trends in software analysis and design must be considered in embedded system analysis and design. The development of Unified Modeling Language (UML) and number of extension proposals in real-time domain are encouraging new design methods above static and traditional biasing of hardware and software design methods UML has some lacunas to support embedded system design and some strengths as well. Several approaches are suggested for using UML to design embedded systems. This paper is an overview of proposals 'How UML can be used for Embedded System Design (ESD)?'

Keywords : UML, ESD.

1. Introduction

ESD is about implementation of a set of functionalities satisfying number of constraints like performance, cost, power consumption, size and weight etc. The platform of implementation may be software or hardware.

1.1. Embedded System

Modern embedded systems are assemblage of multiple functional units or subsystems that carry out computation and communication using various models of computation.

These functional systems are nothing but flexible architectures, implemented using composition of variety of components both hardware and software. Choice of system components depends upon optimality of

- Implementation of function
- Analysis of alternatives

in both hardware and software domains. Specification of flexible hardware-software architectures should be possible. These systems demand new ways of merging hardware and software design methods to cut time required and the risk involved in development.

1.2. ESD requirements

In recent years, the complexity of embedded system has exponentially grown because of the growing number of functionalities to be implemented in it. As a result the development time is increasingly difficult to predict and to keep in check. The increased complexity and constantly evolving specifications

have motivated designers to look at implementations that can be changed rapidly. As hardware- based implementation is time consuming and expensive, the interest in software-based implementation has risen to previously unseen levels.

1.2.1. Assemblage

Today's embedded systems are usually assemblage of subsystems, each of which may be based on a different Model of Computation (MOC). Both functional and architectural descriptions of system may be based on compositions of subsystems. For this kind of design methodology the importance should be equally given to the validity of assemblage and the correctness of assembled subsystems.

1.2.2. Heterogeneous Implementation

In recent years embedded systems have moved from single function products to incorporating multiple functions. The functionalities from multiple domains of computing, signal processing of audio and video, wired and wireless communication have led to complex embedded systems. These systems are integration of subsystems reflecting a heterogeneous set of Models of Computations. Designing such systems requires set of heterogeneous notations to reflect: continuous and discrete events, data flows, finite state machines, etc. There are also multiple physical implementation domains for embedded systems in addition to these logical domains. Embedded system functions can be realized with the help of assemblage of dataflow and control software/hardware components such as microprocessors, DSPs, digital hardware blocks, optical components. These heterogeneous implementation choices need to be modeled during design.

1.2.3. Reuse and Synthesis

Modern embedded system products are too complex to be designed from scratch. The complexity of the embedded software for these systems is growing exponentially. Complexity from heterogeneous functional requirements demands more design knowledge than single product group used to have. All these factors lead the development process towards reuse and synthesis methods driven from system-level models [1].

1.2.4. Attention to QoS Factors

So far, in embedded software design little attention has been paid to hard constraints of software like reaction speed, memory consumption and power consumption, etc.

Along with these, the conventional quality measuring factors should also be considered during designing phase.

1.2.5. Context

Along with converting functional requirements of the embedded system in to models and executable specifications, the environmental and user driven context for system should also be reflected in design.

2. Unified Modeling Language (UML)

UML is an object-oriented modeling language standardized by Object Management Group (OMG) mainly for software systems

development. UML is used to specify, visualize, construct and document the artifacts of a software system. It captures decisions and understanding about systems to be constructed. It is used to understand, design, browse, configure, maintain and control information about such systems. UML captures static as well as dynamic behavior of a system.

UML consists of a set of basic building blocks, rules that dictate the use and composition of these building blocks, and common mechanisms that enhances the quality of UML models [2]. Its rich notation has made UML a popular modeling language in multiple application domains for system analysis, design and specification. The use of UML with code generation tools is still limited, but is expected to increase as executable dialects are introduced [3][4][5]. To make system specification phase easier and more flexible, UML provides multiple diagrams to model a system from several perspectives and at multiple level of abstraction. It is necessary to follow rigorous design discipline to maintain consistency across several UML diagrams for a system.

2.1. UML Strengths for ESD

Some of the important strengths of UML(1.4) [2][6][7] and UML(2.0) [8][9][10] to support ESD can be listed as..

The promotion and evolution of UML is handled by Object Modeling Group (OMG) [6], a large standards organization. Additionally it supports Object-Oriented system analysis and design, most popular among software developers.

UML is set of heterogeneous notations which can allow design for varied applications.

Extensions of notations are allowed via stereotypes, tags for particular application.

System environments, user access, and test cases can be expressed using Use-Case Diagrams.

Class Diagram, Collaboration and Sequence Diagrams can be directly used for ESD. State-Chart diagrams support the behavior of Objects.

Deployment and Component diagrams give the mapping of logical concepts to actual implementation environment (platform) which is more important for ESD.

Because of growing interest of embedded system and real time system community in UML, OMG wishes to enrich the UML 2.0 by additional functionalities (notations & semantics) to make it more suitable for real time aspects like dataflow, schedulability, performance, time, etc.

Even one of the proposals for UML2.0 suggests a profile for SDL (used in real time and embedded software development) [11].

Architectural co-design can be supported be the concept Model-driven Architecture [6], which separates specification from platform.

2.2. UML Lacunas for ESD

One of the important deficiencies is the complication of communicating UML models among users due to incompatible stereotypes, tool-dependent semantics and limited capability of code generation from design diagrams.

The other key lacks remaining in UML to support embedded system design [12] are: the platform model, action semantics, mapping and refinement methodology to switch platform, the constraint definition and

budgeting methodology, communication synthesis, performance analysis.

3. Approaches to use UML for ESD

For using UML in specification and design of real-time embedded systems variety of supporting and competing approaches have been proposed as response to OMG RFP's. Good comparison and summary of some of these is found in [13]. These approaches can be studied under the categorization as discussed next.

3.1. UML without Alteration

Bruce Powel Douglass has promoted the concept of using UML as is for real time ESD [14]. The main focus was on taking advantage of extensibility in UML stereotypes.
Some of the examples can be…
Synchronization was represented using UML messages, UML active objects can be used as threads of OS and guarded operation stereotype as semaphores. Hardware design elements like processors, devices etc. can be shown using Node of UML and software elements can be shown using Component.

3.2. Translation from UML

SDL based toolsuits supports real time code generation, functional simulation and test generation. So one approach introduced by Telelogic in their Telelogic Tau toolsuite is translation of UML to SDL.

Long back Ek introduced the concept of SDL-oriented Object Modeling Technique (SOMT) [15]. More recently some specification proposals like [16] includes concept of SDL combined with UML that defines the mapping between the notations.

3.3. Object Structuring of System

Object structuring is nothing but functional decomposition of system. These are UML-RT and SDL-UML profiles based on existing languages like ROOM [17] and SDL. The UML-RT profile is now available as Rose-Real-Time tool from Rational, which is extension of UML for real time [18] [19].

3.4. UML with Action Semantics

This approach is applicable to all UML development for embedded systems as well as other software applications. The UML has a formal syntax, but not formal semantics. Lack of defined action semantics has kept UML in documentation domain only. There is a need of well defined, inter-operable semantics of action and operations [20]. The improvement in semantics will enable the specification of computation.

3.5. UML with QoS

In addition to resource (physical and non-physical) modeling various Quality of Service characteristics like time, clocks, concurrency, schedulibility and performance can also be represented in UML using some class-based stereotypes. The concept is the OMG RFP for a UML profile for scheduling, performance and time [21]. QoS characteristics will serve as the base for embedded system engineering [10].

3.6.Architectural to Physical Mapping

UML allows the mapping of architectural design to physical implementation using component and deployment diagrams. But the current notions of these diagrams are not sufficient to represent mapping of hardware-software co-design. By extension of these diagrams the better mapping can be achieved for embedded system design. The above mentioned OMG RFP for scheduling, performance and time [21] is a step towards this extension.

3.7. UML and Platforms

Embedded system platforms can be divided in to three abstraction levels-architecture (ARC), application programming interface (API) and application specific programmable (ASP). These can be respectively represented by extending deployment, component and class diagram.

UML platform introduces new building blocks to represent specific platform concepts, chooses proper diagrams to model platforms at different abstraction levels, quantifies QoS performance and budgeting constraints.

3.8. Embedded UML

Embedded UML was research project for defining and proposing, a UML profile suitable for ESD. Embedded UML retains best of UML and real time UML like class diagrams, use cases and sequence diagrams. As complement to these it supports platform model in hardware and software for implementation architecture. The collection of platform services can be considered as a system platform API.

For functional encapsulation the approach of embedded UML suggest use of blocks as extension of capsule notation. Collaboration diagrams are extended to netlists for functional composition and interfaces, channels with stereo types are used for communication specification. Finally mapping diagrams (extension of deployment diagrams) are used for performance analysis.

4. Need of Unified Approach

After analyzing the deficiencies of UML to support the specification, development and implementation of software-based embedded systems running on hardware-software platforms, the UML community must have to overcome these lacunas through hardware-software co-design research and extension to UML.

From the above discussed approaches, to have commonly acceptable extension, the researchers of hardware and software design fields must have to join hands and work together for ESD-UML.

5. Conclusion

Based on the nature of modern embedded systems, to support their design UML has its own strengths and weaknesses. To use UML for ESD many extensions have been proposed, but still the embedded system developer community is waiting for commonly acceptable complete solution.

6. References

[1]. Alberto Sangiovanni-Vincentelli and Grant Martin, “Platform-Based Design and Software Design Methodology for Embedded Systems”, IEEE Design and Test of Computers, Volume 18, Number 6, November-December 2001, pp. 23-33.

[2]. J. Rumbaugh, I. Jacobson, and G. Booch, The Unified Modeling Language User Guide, Addison-Wesley, 1998

[3].UML-RFP, http://www.omg.org/techprocess/meetings/schedule/Action_Semantics_for_UML_RFP.html

[4]. B. Selic, Complete High-Performance Code Generation from UML Models, Proceedings of Embedded System Conference, San Francisco, CA, USA, March 2002.

[5]. C. Raistrick, Executable UML for Embedded System Development, Proceedings of Embedded System Conference, San Francisco, CA, USA, March 2002.

[6]. The Object Modeling Group, URL : www.omg.org/.

[7]. Thomas Weigert, “What Really is UML?”, presentation, October 18, 1999.

[8]. Gjalt de Jong, “A UML-Based Design Methodology for Real-Time and Embedded Systems”, DATE 2002.

[9]. Bran Selic, “The Real-Time UML Standard: Definition and Application”, DATE 2002, March 2002.

[10]. Bran Selic, “A Generic Framework for Modeling Resources with UML”, IEEE Computer, June 2000, p.64-69.

[11]. Morgan Björkander, “Graphical Programming Using UML and SDL”, IEEE Computer, December 2000.

[12]. Grant Martin, Luciano Lavagno, and Jean Louis-Guerin, “Embedded UML: a merger of real-time UML and codesign”, CODES 2001, Copenhagen, April 2001, pp.23-28.

[13]. Ganssle, J. “Navigating through new development environments”, Embedded Systems Programming, May, 1999, pp. 22-30.

[14]. Douglass, B. P. Doing Hard Time – Developing Real-Time Systems with UML, Objects, Frameworks and Patterns, Addison-Wesley, 1999.

[15]. Ek, A. “The SOMT Method”, Telelogic white paper, Sept. 19, 1995.

[16]. Bikander, M. “Graphical Programming Using UML and SDL”, IEEE Computer, Dec. 2000, pp. 30-35.

[17]. Selic, B., Gullekson, G. and Ward, P. Real-Time Object-Oriented Modeling, John Wiley and Sons, New York, 1994.

[18]. Selic, B. and Rumbaugh, J. “Using UML for Modeling Complex Real-Time Systems”, white paper, Rational (ObjecTime), March 11, 1998.

[19]. Selic, B. “Turning clockwise: using UML in the real-time domain,” Comms. of the ACM, Oct., 1999, pp. 46-54.

[20]. OMG Request for Proposal: Action Semantics for the UML RFP, OMG document: ad/98-11-01.

[21]. OMG Request for Proposal: UML Profile for Scheduling, Performance and Time, OMG document: ad/99-03-13.