Modeling Real Time Embedded Systems with UML2.0

Sina Tebiani  
MDH University  
IDT Department  
Email:sti10003@student.mdh.se

ABSTRACT
The error-proneness of traditional V model development of embedded systems with manually written specification, system designing based on these specifications and manual written code, leads to the seek of alternative and more efficient methodologies. This paper will present an approach for model driven development of real time embedded systems through the Unified Modeling Language (UML) 2.0. This methodology is object oriented and involves all aspects of real time behaviors like interrupts and timeouts. Recent approaches for applying UML2.0 into the C based embedded systems will be discussed in this paper. UML profile extension for real time embedded systems (MARTE) is an approach for MDD adoption to the real time system domain. MARTE consists of foundations for model based description of real time embedded systems.

Categories and Subject Descriptors
D.4.7 Organization and Design

General Terms
Design

Keywords
Embedded system design, Object oriented designing

1. INTRODUCTION
The characteristics of modern embedded systems require new development process methods to cover specification, design and implementation phases. Such new methods are needed to deal with complexity of the systems as well as holding the productivity in high level. The Unified Modeling Language (UML) is general purpose solution for embedded systems design. The using of UML in development process of embedded systems differs from standard code-based approaches in two ways: the UML is object oriented while embedded systems are by nature functionality oriented and a direct consequence of this is that, while OO languages (e.g., C++) are usually employed with UML, embedded systems are usually implemented using C. These are only two of the multiple reasons that make UML challenging to use when developing embedded systems. The challenges arise from features diversity of UML and C language. The UML integration to the modeling of embedded systems is the integration of two different development orientations: object oriented (OO) characteristics of UML and function oriented characteristics of C language makes the integration difficult. On the other hand, the C developers face with many unfamiliar OO contexts to deal with while the UML developers are more comfortable to overcome complex object oriented designs by applying the UML advantages and code generation technologies. The other reason developers are reluctant in switching to model-based development is the difficulty in re-engineering legacy systems.

In this paper we present a set of possible guidelines for using UML in the development of real time embedded systems. The C language is the main goal of the implementation platform while the UML 2 notations used.

The order of the paper follows: Section 2 gives an overview on the embedded system characteristics. Section 3 highlights UML 2 usage in embedded systems. Section 4 discusses the UML profile for modeling and analysis of real time embedded system. Section 5 is a discussion about the modeling of real time embedded systems. In section 6 transformations of models from UML models to C will be analyzed. Section 7 mentions an example to demonstrate the usage of UML in the real time systems. Section 8 gives a summary on this paper. Section 9 introduces the literature and references.

2. REAL TIME EMBEDDED SYSTEM CHARACTERISTICS
The real time embedded systems are the specific microcontroller based systems that have been designed for dedicated functions. These systems have time constraints to manage the tasks concurrency execution in the time frame. It means that the correct behavior of the system depends not only to the value of computation also the time that the results are gained. The late response to the events can have catastrophic consequences.

The real time systems have a major role in our life. They are applied in many fields like railway systems, automotive applications, flight control systems and etc. Despite of the extensive application of these systems, developers and researches still put a lot of efforts to increase the applicability of real time features in designing and implementation of these time constraint systems. Some features can be declared as follow [2], [3]:

1. The real time embedded systems run complex algorithms. The design of these algorithms needs domain specific knowledge.

2. The real time systems can be considered embedded system that embedded in large hardware or software systems. The picture 1 shows the state of real time
3. Interaction between real time systems and sensors and actuators is through the interfaces. There is interface to transmit the data from sensors as input to the actuators and outputs.

4. The operations have to run with deadlines. The hard real times cause the failures in the system and consequences are catastrophic but soft real times cause some errors inside the system.

5. The real time systems are event triggered systems. They are stimulated by the external stimuli. Many concurrent tasks are handled with timing algorithms.

With respect to these features, the design and modeling of real time embedded systems will be discussed during this paper. The concentration of the design will be on the interaction and timing algorithms of systems.

Figure 1. Real time embedded systems.[1]

3. UML2.0 FOR REAL TIME EMBEDDED SYSTEMS

For designing and modeling of real time embedded systems the UML version 2.0 and its profiles are commonly used today. The designs and structures will be modeled based on specifications and requirements. UML 2.0 has three model categories and thirteen types of diagrams. The categories and their types include [4]:


UML as a general-purpose modeling language represents a possible solution for modeling software systems and there exist several tools supporting it. Nevertheless, it lacks the real time embedded systems domain support. The modeling of time constraints, components and signals are not supported in some aspects [5]. To overcome these shortages some extension proposals suggested. The Object Management Group (OMG) suggests the different profiles including the required extensions. The CASE tool vendor proposes some extensions to the UML [5]. “Rational and Telelogic adapts UML for modeling embedded real-time systems by combining it with the modeling languages from the real-time (ROOM) and telecommunication domains (SDL), while I-Logix stays with Standard-UML, but provides a very powerful implementation of state charts”[5].

The UML 2.0 and its profiles introduce a rich set of elements and concepts for modeling real time embedded systems. Its extension ability allows the designers to model different specific domains. These extensions can be used effectively in real time concepts.

4. THE UML PROFILE FOR MODELING AND ANALYSIS OF REAL TIME EMBEDDED SYSTEMS (MARTE)

To Enable the UML to model real time embedded systems, MARTE specification is added to the UML. This profile supports all specification, design and testing phases. This extension considers the model based development in two aspects: modeling and analyzing. The modeling parts provide support from specification to the detail design and the analysis annotates models with required information to establish required analysis.

[16] The advantages of this profile can be mentioned as [16]:

1. Provides common communication way in software and hardware modeling aspects from the early stages of development until testing between developers.

2. The interoperability between different tools used for specification, designing and validation/ verification and code generation is provided.

3. Promote the model constructions to make quantitative predictions from real time and embedded systems features view.

As the wide range of knowledge areas are involved in this profile extension and many different tools from different domains must be adapted to these specification, to ease this adoption this specification profile defines modular approach for conformance.

[16] The extension unities are used in order to identify concrete MARTE elements. These are UML profiles or model libraries that MARTE proposes as language extension packages. In the figure below we can see these units and their names:

Table1: Extension Units [16]

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Name, description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFP</td>
<td>Non-Functional Properties</td>
</tr>
<tr>
<td>Time</td>
<td>Enhanced Time Modeling</td>
</tr>
<tr>
<td>GRM</td>
<td>Generic Resource Modeling</td>
</tr>
<tr>
<td>Alloc</td>
<td>Allocation Modeling</td>
</tr>
<tr>
<td>GCM</td>
<td>Generic Component Model</td>
</tr>
<tr>
<td>HLAM</td>
<td>High Level Application Modeling</td>
</tr>
<tr>
<td>SRM</td>
<td>Software Resource Modeling</td>
</tr>
<tr>
<td>HRM</td>
<td>Hardware Resource Modeling</td>
</tr>
<tr>
<td>RTM</td>
<td>Real-Time objects Modeling (RTO, MoC)</td>
</tr>
<tr>
<td>GQAM</td>
<td>Generic quantitative Analysis Modeling</td>
</tr>
<tr>
<td>SAM</td>
<td>Scalability Analysis Modeling</td>
</tr>
<tr>
<td>PAM</td>
<td>Performance Analysis Modeling</td>
</tr>
<tr>
<td>VSL</td>
<td>Value Specification Language</td>
</tr>
<tr>
<td>CFL</td>
<td>Clock Handling Facilities</td>
</tr>
<tr>
<td>RSM</td>
<td>Repetitive Structure Modeling</td>
</tr>
</tbody>
</table>
To better target the user need, the MARTE implementers need level of conformance definition. The conformance use cases can be considered as following [16]:

1. Software modeling
2. Hardware modeling
3. System architecture
4. Performance analysis
5. Schedulability analysis
6. Infrastructure provider
7. Methodologist

In the table below the extension unites that must be supported in the compliance cases are illustrated:

<table>
<thead>
<tr>
<th>CASE</th>
<th>Level</th>
<th>ORM</th>
<th>MP</th>
<th>VOS</th>
<th>TrE</th>
<th>Eff</th>
<th>HRI</th>
<th>CDM</th>
<th>SM</th>
<th>AIV</th>
<th>LGM</th>
<th>OCM</th>
<th>RAM</th>
<th>SMT</th>
<th>RM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>Full</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Hardware</td>
<td>Full</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>System</td>
<td>Full</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Performance</td>
<td>Full</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Schedulability</td>
<td>Full</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Infrastructure</td>
<td>Full</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Methodologist</td>
<td>Full</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

The compliance cases are introduced in two levels in order to speed up adoption: base level and full level. The mandatory extensions in the full level can be considered optional extension for the base level.

4.1 OMG REAL TIME EMBEDDED SYSTEM RELATED STANDARDS

The MARTE profile consists of group of related OMG specification. The superstructure UML 2 and OCL 2 specification are used in this profile. The figure below shows these combinations:

5. MODELING REAL TIME EMBEDDED SYSTEMS

The modeling of real time embedded systems consists of gathering data, analysis and design a model. With UML 2.0 the system and its behavior are described by thirteen different diagrams. Each sort of diagrams has specific concepts and elements. A subset of diagrams will be chosen to design a specific domain of embedded systems.

5.1 MODELING REQUIREMENTS WITH SYSML

The OMG system modeling language is a general purpose graphical modeling language for specifying, analyzing, designing and verifying complex systems. These systems can include software and hardware, information and procedures. The SysML is subset of UML 2 with extensions to meet the requirements of the UML for system engineering.

The block is the basic structure in SysML and can be used to show the hardware, software, personnel, facilities or other systems. The diagram types of SysML are shown in this picture:

The system structure is illustrated by block definition diagrams and internal block diagrams. A block definition diagram shows the system hierarchy system classification. The internal block diagram represents the internal structure of a system like parts, ports and connectors. To organize the model the package diagram is used. [17]

The behavior diagram includes the use case diagram, activity diagram, sequence diagram and state machine diagram. [17] In the real time embedded system domain models most include the interactions between the objects. To show these behaviors and interactions between them UML 2.0 uses four different
behavioral modeling packages: Use Cases, Activities, Interactions and state machine.

To represent the descriptive requirements and relate them to other model elements SysML includes a graphical construction. The requirement diagram provides a relation between the typical requirement management tools and the system models. [17]

The parametric diagram is used to show the constraints on system property values such as performance, reliability and integrate the specification and design models with engineering analysis models. [17]

The SysML also includes the allocation relationships to show the allocation of functions to components, logical to physical components and software to hardware. [17]

6. TRANSFORMATION

The conversion set of elements to the new set of target elements can be fulfilled by transformation. This transformation would be the model to text or code, text or code into model or transformation of model to the new models in different level of abstraction. In this paper we give an overview on mapping UML models to C concepts. The following UML objects can be converted into C code based on UML model to the real time C code transformation [6]:

1. Passive classes
2. Models containing passive classes
3. Packages containing passive classes

The active classes like protocols will not be transformed to the C code. In this paper our goal is just transforming the UML models to the C code manually. The UML does not support specific programming language. We specify transformation mapping strategy for the system. The mapping strategy consists of set of rules for UML elements exclusively for diagram types. The use cases in the Use Case diagrams map to the .C and .H files.

The classes map to the struct and attributes and operations map to variables and functions. To identify the sequence of function calls we use the communication and sequence diagram. Timing diagram and state machine diagram are used when the system object has different states. Event triggers in the diagrams are implemented as triggering the timer and interrupts target emulation environment.

Only UML elements that are mapped to the C language can be transformed while other elements will be ignored. The transformation sources must have names because the names for UML models are transformed to the names in the C code. [6]

In the table 3 we can set some mapping rules from UML model to C code transformation. [3]

Table 3. UML 2.0 to C mapping [3]

<table>
<thead>
<tr>
<th>UML feature</th>
<th>C</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case</td>
<td>C files</td>
<td>Create one .C and one .H file for each Use Case.</td>
</tr>
<tr>
<td>Class</td>
<td>struct</td>
<td>Each class in the UML model corresponds to a struct defined in the .H file.</td>
</tr>
<tr>
<td>Attributes</td>
<td>Variables</td>
<td>Public attributes can be declared in the .C file while private attributes are included inside the struct as variable.</td>
</tr>
<tr>
<td>Operations</td>
<td>Functions</td>
<td>Operations map to functions declared in the header files and implemented in the .C file.</td>
</tr>
<tr>
<td>States</td>
<td>Switch-case</td>
<td>Each state name is used as enumeration type of a variable identifying the current state. State transition can be supported both from one state to another or because of incoming timed or asynchronous event triggers.</td>
</tr>
</tbody>
</table>

7. EXAMPLE

In this part an example will be demonstrated for applicability of the MDD in real time embedded system by modeling the Stop Watch system [2]. In this example the different UML diagrams will be applied to model the Stop Watch application.

The Stop Watch is the equipment to measure the elapsed time. Two buttons designed to control the performance of the watch.

The first button is the PAUSE/RUN button. Pressing the RUN starts to count time until the PAUSE button pressed. The elapsed time will be displayed when the button pressed. The twice press button can continue running form the point that it was stopped. The RESET button will reset the watch to zero.

7.1 System Specification

The Stop Watch system only shows the seconds, minutes and hours. It has two buttons RESET and RUN/PUASE as user interfaces and the actuator will be screen display. One function will execute to construct the second, minute and hours and one function will process the clock ticks.

7.2 UML Use Case Diagram

UML 2.0 provides the possibility to extend and tailor the language to a specific domain by means of profiles, stereotypes and constraints. This feature of UML let us to model real time embedded systems applications and their schedulability and time constraints features.

Modeling process with UML 2.0 includes modeling requirements, behavior and structure of the system. This process is independent from programming language and can be transformed to any required language automatically or manually.

The real time features can be expressed in the requirement development process. In the Use Case diagram we can use the graphical notations to show the associations between actors and systems. [3]

The requirements modeling are the first step of modeling procedure. The use case diagram of UML is utilized to model the
system and actors. The interaction between the system and users are modeled with diagram elements. Although the UML 2.0 is applied like the previous versions but the related concepts can be owned by classifiers when it is extending and also the conditions are applied when the use cases are extended for another one. The following picture shows the UML use case diagram of Stop Watch:

![UML use case diagram of Stop Watch](image)

In this model we can understand the relation between the system users and the interfaces (buttons) and actuators.

### 7.3 Activity Diagram

In this diagram the functionality of system will be modeled. For instance, the function `tick()` will be modeled in our example. The `tick()` function processes the second minute and hour counting. The event `evSecond` is the timing event to produce the one second. The activity diagram has been shown in picture 6:

![Activity diagram of tick() function](image)

### 7.4 The State Machine Diagram

The state machine diagram is the set of elements to describe the behavior of the system under different states. The UML 2.0 presents a protocol state machine diagram to make the classifier trigger transition legally.

The state machine diagram in this example will show the transition between two states: Timer On and Timer Off. This refers to the functionality of system which is controlled by two events: system reset and system run/stop: `evReset` for system resetting and `evToggle` for RUN/PAUSE. In the figure below we can see the model of states.

![State Machine diagram of Stop Watch](image)

### 7.5 Sequence Diagram

The sequence diagram supports the time constraints in the form of time duration and specific time instant. The time domain concept categorized into four structures [3]: 1. Modeling time and timing values 2. Modeling events in time and time related stimuli 3. Modeling timing mechanism (Clock, Timer), 4. Modeling timing services [3]

In the detailed system modeling, the interaction between elements plays a significant role. The functionality of a system that has been designed with use case diagram can be defined in sequence diagram. This diagram could define all the model interactions. This diagram model interactions while considers their message exchange sequence and their corresponding sequence occurrence in lifeline.

Communication diagram is the simplest model of a sequence diagram. This diagram focuses on the internal structures of the system. It uses the sequence numbering scheme without considering the time line.

![Sequence diagram of Stop Watch](image)
When the time is the main purpose in our model structure the timing diagram is used. This diagram is a combination of the sequence diagram and the interaction diagram.

In this diagram the interaction of the functions through to the messages between them will be demonstrated. The sequences of the events and function calls have been shown in the next picture.

After modeling the system functionalities with UML diagrams the source code in C language will be generated based on the transformation rules automatically or it can be implemented manually.

8. SUMMARY

This paper presented how the UML 2.0 can be used in real time embedded systems as the target implementation language can be C. This paper described some of the challenges, focusing on the ground misalignments between object orientation of UML and functionality orientation of real time embedded systems. A possible way to mitigate such differences is to extend the UML through appropriate profiles to be able to model the domain-specific concepts in the real time embedded domain. The paper presented the fundamental real time embedded systems features. It also discusses in which concepts the UML 2.0 can be adopted to the real time embedded system that consequently comes up with extension profile MARTE. The related standards contributed in the MARTE profile have been mentioned. This paper described the modeling diagrams and the UML requirement modeling extension for system engineering. Moreover, the application of UML and its provided diagrams for the modeling of simple software systems is described. The description of existing model transformations for generating implementation code from UML models in the scope of real time embedded systems is only sketched in the paper and left as possible future enhancement.

9. REFERENCES

[1] Li, Q., Yao, C. “Real time concepts for embedded systems”. Published by CMP Books.