Proceedings of the 29th Chinese Control Conference
July 29-31, 2010, Beijing, China

A Semantic-Based Model-Driven Design Method for Real-Time Control Software*

WANG Bin, ZHANG Yun-Sheng, XIONG Xin, XUE Jie, ZHANG Jing

Faculty of Information Engineering and Automation, KunMing University of Science and Technology, KunMing 650051, P.R.China
E-mail: wangbin1@vip.sina.com

Abstract: A semantic-based model-driven software design method for embedded real-time control system is proposed in this paper. The graphic model and semantic model of main objects are abstracted from popular real-time operating systems. An object-related behavior modeling and sorting method is presented for effective interactive between object models. A real-time data acquisition and display application designed in this method is given to validate the feasibility and rationality of the theory.

Key Words: Model-driven, Semantic-based Model, Embedded Real-time Control Software, Object-related Behavior Model

1 INTRODUCTION

Embedded real-time control system has three special characters: the first is strict time restraint: a correct real-time application depends not only on the logical correctness of the computation but also on the time at which the result is produced [1]. The second is concurrency: a number of action sequences must be implemented in a processor or in multiple processors at the same time. The last one is cooperative performance: interactive works between multiple tasks must be effective and distinct. Because of these characters, software design for embedded real-time control applications is complex and time-consuming.

Generally embedded real-time control software is developed in four steps: ①system analysis; ②conceptual design; ③programming according to the selected hardware and operating system; ④test and debugging. Code completed in this method can be used only for the specific hardware and operating system determined in step③, which means the codes can't be reused for another application. If our control system requires modification, we need change the configuration of hardware platform and operating systems; as a result we have to do the similar development which just repeats above steps. In addition system verification only can be carried out after all code having been completed, but the modification and debugging will occur again and again in the verification process, which also will induce repeating the four steps. Both of the situations will prolong the development cycle and increase the development cost. Although embedded hardware is developed in very fast pace, many defects of traditional software design approach have restricted the development of embedded real-time control systems. Therefore, a model-driven software design methods is introduced to embedded real-time control system design.

OMG proposed Model Driven Architecture (MDA [2]) in 2001. MDA is a new specification and development method. It wants to get the uniformity of analysis model, design model, programming model and test model in the process of software design. The main idea of MDA is to abstract Platform-Independent Model (PIM) and then make mapping rules which can convert PIM to Platform Specific Model (PSM); finally PSM will be converted into code framework. This method separates functional analysis from concrete realization and reduces the dependence between platform and design as much as possible, which also can improve the reusability and portability of software.

There are some mature model-driven embedded software research projects: Ptolemy [3] developed by UC Berkeley EECS Dept is a heterogeneous model framework, which studies modeling, simulation and design of concurrent real-time embedded systems. The focus of Ptolemy is on assembly of concurrent components. GME [4] project carried out by Vanderbilt is a configurable environment for domain-specific modeling and synthesis programming. It provides a set of universal notions for multiple application fields so that user can customize model for specific domain. Rhapsody [5] is model-driven integration environment of embedded software development by Telelogic. MoBIES[6] funded by the Defense Advanced Research Projects Agency (DARPA) Information Technology Office (ITO) is studied for the design of process-oriented components and their runtime environments for real-time embedded systems. DRM [7] of Peking University is a feature-oriented domain modeling method, etc.

Most researches mentioned above have a close relationship with MDA and want to provide common

*This work is supported by National Nature Science Foundation under Grant (No. 60863004), National 863 Project（No.2007AA0104 09）,Natural Science Foundation of Yunnan Province, China (No.2007F040M)

modeling environment as broadly as possible at the cost of ignoring detailed design, which means we can only get a framing by using them and most of the concrete design and program must be finished by hand. Furthermore, although these methods can be used to design embedded real-time system software, the correctness of embedded real-time application software not only depends on functional correctness, but also depends on the satisfaction of non-functional constraint. There are also many limitations when we use them.

In view of current situation, on the basis of some research production about real-time software design[8], this paper makes a research on semantic-based model-driven design method specifically for embedded real-time control software design. By analyzing and generalization for several universal operating systems such as VxWorks, ucOS / II, eCos and RT-Linux, a semantic-based modeling method for real-time system is proposed, the graphic model and semantic model are abstracted for every necessary unit (In section 2). As an emphasis an object-related behavior model description and classification method is presented to provide more efficient interaction between tasks, and the behavior collections with semantic model and graphic model are summarized (In section 3). Finally, an example is given to validate the feasibility and rationality of this method in an embedded real-time control software design platform which is under construction (section 4).

2 SEMANTIC-BASED MODEL-DRIVEN DESIGN METHOD

In embedded real-time application the problem is divided into multiple concurrent tasks in the design process, which means the basic functional unit is task. Comparing with the realization of a single task, design of coordination and cooperation between the multiple tasks is more important. Firstly, communication between two tasks is needed in order to achieve information interaction. Secondly, a task may also communicate with the environment, such as reading data from a sensor, or sending command to an actuator. In order to finish these works, we should use interrupt, mailbox, semaphore and other units.

2.1 Object Model Abstract Method

In embedded real-time control systems, task management mechanisms, interrupt management mechanisms, as well as synchronization, communication management mechanisms between tasks and interrupts in the design implementation process must be taken into account as important factors. We studies several typical embedded real-time operating system such as VxWorks, RT-Linux, ucOS / II and eCos in kernel preemption, priority changes, the number of tasks, time determinacy, synchronization, communication mechanisms and scheduling algorithms to make a compare and summary [9-13]which is shown in table 1.

According to Tab. 1, we find that in embedded real-time control system software design, the most commonly used objects are task, resource, event, interrupt, mailbox, message, message queue, alarm and semaphore. We call those elements as object in this paper. In our semantic-based model-driven software design method an object is a unique conceptual entity with identities, attributes and behaviors. From a more general sense, an object can be seen as an autonomous entity. As we known, human's body is a composition of different organs, and different organs are build-up by different types of cells, so cells are the most basic self-government units with different properties, behaviors, status, roles and functions. Different cells may collaborate organically through different actions to complete a higher level of collaboration such as digestive organ. The concept of an object is just like a cell. So each object model must be integrity and functional independent. At the same time abstraction of object model must be accurate, clear and reasonable, which determines whether the whole model-driven design method will be validity or not.

Tab.1 Popular Operation Systems Feature Comparison

operating system(OS)	VxWorks	RT-Linux	ucOS/II	eCos
Kernel Preemption	true	true	true	true
Number of assignments	No Limit	No Limit	64 (56 for user)	No Limit
Priority assignment Mechanism	dynamic	static (default); dynamic	dynamic	dynamic
Time predictability	true	true	true	true
Scheduling	time slice rotation	Priority; shortest time first	Fixed Preemptive Priority	Preemptive Priority time slice rotation based on Priority
Synchronization and communication mechanisms	FIFO queue; shared memory; Mutex; Condition variable; semaphore	FIFO queue; shared memory; Mutex; queue; semaphore	semaphore; Mutex; mailbox; queue; Event flags	Mutex; Condition variable; semaphore; mailbox; Event flags

2.2 Graphic Model and Semantic Model of Objects for Embedded Real-Time System

We have established graphic model and semantic model for every object individually. When presenting semantic model we adopt ontology semantic describing method in our research. Main idea of ontology is abstracting businesses of objective world to a series of conceptions and relationships between those conceptions[12],Now we present task, semaphore and interrupt as examples to illustrate.

- **Task**

A task, also called a thread, is a most basic self-government unit, which is a simple program segment formally. The graphic model of task is shown in Fig. 1.The semantic of task is described as:

TASK (*task_id, TP*)

……

……

END_TASK

-*task_id*: task identifier;

-*TP*: task priority.

It should be noted that the task object can change status and properties of itself or another task, which are not defined in the task model but in the behavior model, details as section 3.2.

- **Semaphore**

Semaphore is invented by Edgser Dijkstra in 1960'.It is a conventional mechanism in multi-task communication. Semaphore has three functions:

（1）control the authority for using a shared resource in multiple tasks;

（2）mark the occurrence of an event;

（3）ensure the synchrony of two tasks.

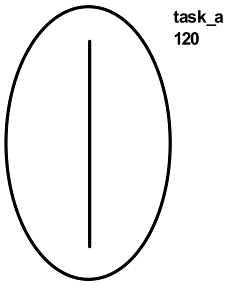

Fig.1 Task Model

There two kinds of semaphore: binary semaphore and counting semaphore. Semaphore may have two options to deal with several queued tasks waiting for the semaphore: FIFO (First In First Out) or highest priority (Priority). The graphic model of semaphore with FIFO is shown in fig.2 (a) and the semaphore with Priority is shown in figure2 (b). The semantic model of semaphore is described as:

SEMAPHORE (*sem_id,[MV],[IV],[QHM]*)

- *sem_id*: semaphore identifier;

- *MV*: maximum value of semaphore;

-*IV*: initial value of semaphore;

-*QHM*: queue handling mode.

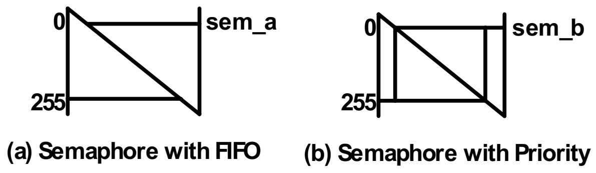

Fig.2 Semaphore Model

- **Message, mailbox and message queue**

In an embedded real-time system, Messages are delivered between two tasks or between a task and an interrupt service for communication. In our semantic-based model-driven design approach, we use message, mailbox and message queue for message transmitting. As shown in Figure 3, (a) is the model of a mailbox with 50 messages; (b) is the model of a single message and (c) is the model of a message queue with 20 messages.

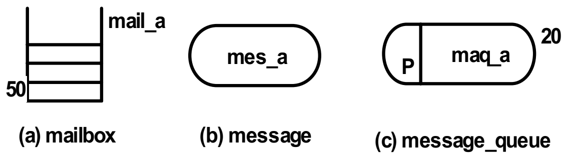

Fig.3 Model of Mailbox ,Message and message_queue

The semantic model of them is described as:

MAILBOX (*mbx_id, [MV], [QHM]*)

- *mbx_id*: mailbox identifier;

-*[MV]*: maximum number of messages;

-*[QHM]*:queue handling mode;

MESSAGE (*mes_id*)

- *mes_id*: message identifier;

MESSAGE_QUEUE (*mes_id*, *N*)

- *mes_id*: message_queue identifier;

-*N*: number of message;

3 OBJECT-RELATED BEHAVIOR MODELING METHOD FOR EMBEDDED REAL-TIME SYSTEM

In the model-driven software design method, interaction between objects is completed by the behavior models. All the actions of a model will be achieved through a set of operations eventually, which means we should make a clear definition of every executable behavior and establish behavior model combination for all objects, the behaviors of the a model should be restrained by the behaviors supplied by model collection .So an effective behavior modeling method will be very important and necessary.

3.1 Object-Related Behavior Model Approach

In the current research work about model-driven method, the behavior model can be divided into three categories according to how simple the behavior is and what degree the behavior depends on the history. They are simple behavior model, state behavior model and continuous behavioral model.

As mentioned above, in the traditional modeling method most detail actions should be programmed by hand. However, the method proposed in this paper is a specific design method for embedded real-time control system which wants to realize detail programming as much as possible and relieve manual work done by user as few as possible. Therefore current behavior modeling method is unreasonable and inadequate for our method. For this reason we propose object-related behavior modeling and sorting method in this article which establishes model from the view of object and keeps focus on the basic objects of embedded real-time control system such as task, interrupt, and semaphore. Because these object models correspond to the basic elements mentioned above, all the behaviors are carried out between the object models, so construct behavior model in such a manner is reasonable. "Object-related" means behavior model is defined and created according to which object a behavior is launched or which object a behavior is directed. There are many non-functional constraints to be met in embedded real-time control system. In the design process, the designers pay more attention to what objects an action is associated with than other factors because the behavior is connected with two objects in order to achieve a specific operation. Therefore this method is more close to the designer's manner of thinking. Furthermore, when extracting model in object-related way the collection of behavioral models will form a classification naturally, and this kind of category makes the call of behavior model closer to the designer's thinking, so that the design process will go more smoothly.

3.2 Behavior Model Categorization and Collection

In our current study, the behavior model is divided into 10 categories: creation behavior model, delete behavior model, algorithms behavior model, task behavior model, interrupt behavior model, resource behavior model, mailbox behavior model, semaphore behavior model, alarm behavior model, event behavior model. It should noted that the creation behavior of all objects have been created as a separate class, because the behavior in the model-driven design method will indicate its' two coherent objects directly, there is no need to establish the model for every object respectively. Delete behavior model and algorithm behavior model are also under similar circumstances.

We take "task" object as example to show how object-related behavior modeling and sorting method work. Task-related behavior model collection includes seven behavior models: DELAY, SUSPEND, RESUME, CHANGE_PRIORITY, LOCK, UNLOCK and INQUIRE. The graphic model and semantic model of every behavior will be shown as follows.

- DELAY (DS) :Delay behavior, which extend the disable interrupt time, DS is the delay time of interrupt，which is numbered as formula (1).

$$DS = T_{dl}+T_{c1} \qquad (1)$$

T_{dl}: the longest time of disable interrupt;

T_{c1}: implementation time of the first Interrupt Service Routine instruction.A delay behavior model with 10ms interrupt latency time is shown in fig. 4.

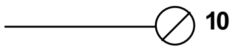

Fig.4 DELAY Model

- SUSPEND (*task_id*): Change the state of a task to "suspend", which means the task is waiting for the occurrence of an event, such as waiting for an interrupt from some external device. It is shown in fig. 5.

  -*task_id*: target task identifier, it may be the active task (which promote this behavior) itself or another task.

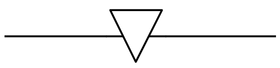

Fig.5 SUSPEND Model

- RESUME (*task_id*): Change the state of a task from "suspend" to "ready",

-*task_id*: target task identifier, it may be the active task itself or another task. It is shown in fig. 6.

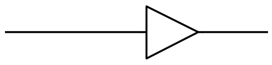

Fig.6 RESUME Model

- CHANGE_PRIORITY (*task_id, TP*): Priority reset, whose function is to alter the priority of a task.
  -*task_id*: target task identifier, it may be the active task itself or another task.

-*TP*: is the designated new priority of the target task.

A change priority behavior model with new priority of 115 is shown in fig. 7.

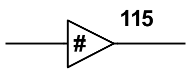

Fig.7 CHANGE_PRIORITY Model

- LOCK: Prohibit of task scheduling, until the task is completed and the scheduler is unlock. It is shown in fig. 8.

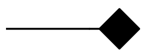

Fig.8 LOCK Model

- UNLOCK: Unlock scheduler. It is shown in fig. 9.

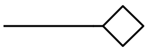

Fig.9 UNLOCK Model

- INQUIRE (*object_id*): Query to obtain the information of an object; it is shown in fig. 10.

Object _id : is the target object identifier, it may be the active task itself or another object

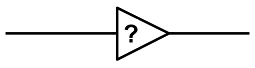

Fig.10 INQUIRE Model

In addition, create behavior model, delete behavior model and algorithm behavior model are also applicable for task.

4 AN EXAMPLE

On the basis of our research, we are building a platform to confirm this method. In this platform, people can develop their embedded real-time applications by joint graphic models together, at the same time the corresponding semantic models will be generated automatically. The semantic model can be converted into target codes according to specific system by custom-interpreter, which is another emphasis of our research.

Fig. 11 is a real-time data acquisition and display application design with our model-driven design method mentioned above.

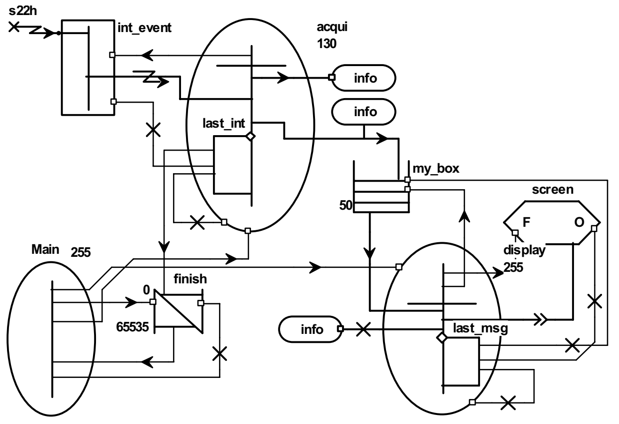

Fig.11 A Model-Driven Data Acquisition and Display Application

There are eight object models using to design this example: three concurrent tasks: “main” task, “acqui” task and “display” task; a interrupt named “int_event” to get a data from extern device; a mailbox named “my_box” and a message named “info” to deliver this message to “display” task for display; a semaphore named finish to judge the last interrupt; a source named “screen” to show the data. There are twenty-four behavior models using to complete interactive between the eight objects: seven create behavior models, seven delete behavior models, four algorithms behavior models, two mailbox-related behavior models, two semaphore-related behavior models, a interrupt-related behavior model and a resource-related behavior model. At the mean time we can get the semantic model of this application as shown in fig. 12.

```
INTERRUPT(int_event,s22h)
    SIGNAL_IT(int_event);
END_INTERRUPT

TASK(acqui,130)
    CREATE_INTERRUPT(int_event,s22h);
    FOREVER
    CREATE_MESSAGE(info);
    WAIT_IT(int_event,);
    SEND_TO_MBX(my_box,info,,);
    IF last_int THEN
        SEND_TO_SEM(finish,1,);
        DELETE_INTERRUPT(int_event);
        DELETE_TASK(acqui);
    ELSE
    END_IF
END_TASK

TASK(display,255)
    CREATE_RESOURCE(screen,F,O);
    CREATE_MAILBOX(my_box,50,F);
    FOREVER
    WAIT_ON_MBX(my_box,);
    ACCESS(screen,W,);
    DELETE_MESSAGE(info);
    IF last_msg THEN
        DELETE_MAILBOX(my_box);
        DELETE_RESOURCE(screen);
        DELETE_TASK(display);
    ELSE
    END_IF
END_TASK

TASK(Main,255)
    CREATE_TASK(display,255);
    CREATE_SEMAPHORE(finish,65535,0,F);
    CREATE_TASK(acqui,130);
    WAIT_ON_SEM(finish,1,);
    DELETE_SEMAPHORE(finish);
END_TASK
```

Fig.12: Semantic Model of Fig.11

5 CONCLUSIONS

On the basis of model-driven software design theory we study a base level model-driven software design method especial for embedded real-time applications. It set emphasis on the detail design of application but not on the frame design which most researches focus on. Three major issues were addressed in this paper: (1) A semantic-based modeling method is effective for embedded real-time software design, and we can abstract the models of objects and behavios from standard real-time operation systems. (2) An object-related behavior modeling and sorting method is suitable for interaction between multiple tasks, which make the semantic-based model-driven method more efficiency. (3) Object models and behavior models are presented in graphic models and corresponding semantic models. An example of real-time data acquisition and display application has verified above issues.

For further study, we intend to complete and improve semantic model architecture; another emphasis is to study the transition method from PIM to PSM by using semantic model.

REFERENCES

[1]J.A. Stankovic, “Misconceptions about Real-Time Computing: a Serious Problem for the Next-generation Systems”, IEEE Computer, 1988,21(10):10-19.
[2]Object Management Group. Model driven architecture(MDA). OMG document, ormsc/2001-07-01,July2001.
[3] Edward A. Lee, "Overview of the Ptolemy Project," Technical Memorandum No. UCB/ERL M03/25, University of California, Berkeley, CA, 94720, USA, July 2, 2003.
[4] A.Ledeezi, M.Maroti, et.al.The Generie Modeling Envirotunent[C]Proceeding of IEEE Workshop on Itelligent Signal Proeessing, BudaPest, Hungary, 2001.
[5] http://www.ibm.com/us/en/,2009.
[6]MoBIES Automotive Open Experimental Platform [EB/OL].httP://vehiele.me.berkeley.edu.mobies/.2009.
[7] HUANG Gang;WANG Qian-xiang;CAO Dong-gang;MEI Hong(Software Institute;Peking University;Beijing 100871;China) PKUAS:A Domain-Oriented Component Operating Platform, Chinese of journal electronics2002,z1(30).
[8] J.-J.Schwarz,J.Skubich,P.Szwed,M.Maranzana.Real-Time Multitasking Design with a Graphical Tool, [C],First IEEE Workshop on Real Time Application,New York,USA, May, 1993
[9] Victor Yodaiken , Michael Barabanov. A Real-Time Linux. [C]In Proceedings of the Linux Applications Development and Deployment Conference (USELINUX), Anaheim, CA, January 1997. The USENIX Association.
[10] MicroC OS II: The Real Time Kernel, [M], Jean J.Labrosse, 2002-01, McGraw-Hill Europe.
[11]Studer, Benjamins, Fensel, Knowledge Engineering: Principles and Methods. RudiStuder, V. Richard Benjamins, and Dieter Fensel. Data and Knowledge Engineering,25(102):161-197, 1998.
[12]Anthony J Massa, Embedded Software Development with eCos, 2002.
[13] http://www.windriver.com/,2009.