Research on Real-Time Software Development Approach

Wei Qiu
School of Computer Science, Jia Ying University, Meizhou, Guangdong, 514015, China
Email: qiuwei@jyu.edu.cn

Li-Chen Zhang
Faculty of Computer Science, Guangdong University of Technology, Guangzhou, Guangdong, 510090, China
Email: lchzhang@gdut.edu.cn

***Abstract*—Model Driven Architecture (MDA) is a development method of which can generate useable software products directly by the model. It includes a series of standardized modeling, transformation rules and other relevant standards architecture. Real-Time systems have been applied in many areas widely, but they have many non-functional requirements, which always crosscut the whole system modules. That may cause the code tangle and scatter, make the systems hard to design, reuse and maintain, and affect performance of systems badly. AOP is a new software development paradigm, which could attain a higher level of separation of concerns in both functional and non-functional matters by introducing aspect, for the implementation of crosscutting concerns. Different aspects can be designed separately, and woven into systems. This article introduces the technology of MDA, aspect-oriented, real-time systems and UML. This article takes the Aspect-oriented to the MDA modeling by the UML extension mechanisms, and presents a method, which is Aspect-Oriented MDA. In this article, UML profile is utilized to construct the meta-modal specifications respectively for common Aspect-Oriented and AspectJ. So the core business logic and the crosscutting aspects can be modeled as separate, modular Aspect-Oriented PIM's and PSM's. The authors analysis non-functional requirements of the real-time systems, and then apply aspect-oriented MDA modeling to develop an example of real-time systems, and propose how to model aspects of timer and real-time constraints. Finally, in order to more clearly understand how to complete the MDA in the aspect-oriented modeling, especially in real-time system. This paper through the example of a real-time system is discussed. The system simulates the operation of self-automatic washer process.**

***Index Terms*—Aspect-oriented, Real-time systems, Model-Driven Architecture, Platform Independent Model, Platform Specific Model**

I. INTRODUCTION

Model Driven Architecture (MDA) is a development method of which can generate useable software products directly by the model [1]. It includes standardized modeling, transformation rules and other relevant standards a series of architecture In MDA, the software development process is driven totally by models; model played a very important role. At first you should model the business logic of the system at a high level of abstraction, the initial model is not concern with any real platform and environment, that is platform-independent model (PIM) and then it will be transformed into PSM through the mapping mechanism according to the actual software operating environment, it is platform and technology-related business logic model. Finally, PSM become to practical code through a lower mapping. Developers do not need to consider platform and technology details, the specific code will be automatically generated from these models [2]. In MDA the model is no longer designed aids, but products in the process of system development. Its design has to fulfill demanding requirements with respect to limited resources, real-time requirement, reliability, cost, and re-usability.

A model-driven approach to component specification and generative techniques can help to simplify the process for component based development, and at the same time can enforce standard implementation of design patterns and ensure adherence to architectural guidelines. This paper summaries lessons from several projects where a model-driven approach and component based development was implemented to address productivity problems. The important issues are surprisingly non-technical and require paying careful attention to team structure and project organization. In the paper, the author analysis non-functional requirements of the real-time systems at first, and then apply aspect-oriented MDA modeling to develop an example of real-time systems, and propose how to model aspects of timer and real-time constraints. Finally, the author use extended sequence diagram to dynamically show how aspect impact real-time systems.

II. MODEL DRIVEN SOFTWARE DEVELOPMENT

Model-Driven Software Development provides a set of techniques that enable the principles of agile software development (www.agilealliance.org) to be applied to large-scale, industrialized software development. A fundamental concept in software product line engineering

This work is supported by the Major Program of National Natural Science Foundation of China under Grant No.90818008, and by the National Natural Science Foundation of China under Grant No.60774095..

doi:10.4304/jsw.7.7.1593-1600

is domain, which is defined as a bounded area of knowledge. Domains can relate to knowledge about vertical industries (business domains) and also to knowledge about specific software implementation technologies (technical domains) [4]. Productivity gains in building software product lines are achieved by raising the level of abstraction through the use of formalized domain-specific modeling languages that is not only understandable by domain experts, but also machine readable.

Domain-specific modeling languages are usually defined through a meta-model, i.e., an abstract description of the language elements and the rules for composing expressions using the elements of the language [5]. In MDSD we recognize that in some cases domain-specific notations add significant value if their development goes beyond the lowest common denominator that is harmless to share with competitors in an industry. The translation between domain-specific modeling languages and software implementation concepts is achieved through model transformations, which either transform a domain-specific model directly into programming language constructs of a target platform, or use a staged approach via intermediate, less-abstract models to finally reach the level of abstraction of the target platform[6]. Direct transformations between a domain-specific language and programming languages are often expressed in template languages, which have proved to be a critical element in model-driven approaches to developing software. In fact, template languages have a long track record in code generation technology, and are not an invention of the MDA era as claimed by some MDA tool vendors. In MDSD, model transformations and code templates are first-class artifacts, as important as the models that are transformed. Domain-specific frameworks are another important part of MDSD [7].

The Unified Modeling Language (UML 2.0) offers an unprecedented opportunity to describe component-based embedded systems [8]. It capabilities for architectural modeling have been drawn from three architectural description languages (ADLs), viz., UML-RT (ROOM) by Selic and Rumbaugh, ACME by Garlan et al., and SDL by the ITU-T[9]. The Object Management Group (OMG) has proposed the Model Driven Architecture (MDA) approach [10], which aims to allow developers to create systems entirely with models. MDA envisages systems being comprised of many small, manageable models rather than one gigantic monolithic model, and allows systems to be designed independently of the technologies they will eventually be deployed on. The approach is based on two essential concepts, viz., the Platform Independent Model (PIM) and the Platform Specific Model (PSM), which separates the specification of system functionality from the specification of the implementation of that functionality on a specific technology platform and provides a set of guidelines for structuring specifications expressed as models [11]. Both PIM and PSM can be expressed in extended UML [8]. The advantages of the MDA approach are: (1) PIM provides specifications of the structure and functions of the system that abstract away technical details, (2) implementations on several platforms can be derived from one PIM, (3) system integration and interoperability can be anticipated and planned in the PIM but postponed to the PSM's, and (4) it is easier to validate the correctness of models. Therefore, it is useful and significant to combine the MDA approach and UML models with the component technique to develop software for embedded systems. In the following sections, we shall present how to design a platform-independent component model employing the MDA approach and UML notations based on component-based software engineering.

This article introduces the technology of MDA, aspect-oriented, real-time systems and UML [3]. This article takes the AO to the MDA modeling by the UML extension mechanisms, and presents a method, which is Aspect-Oriented MDA. In this article, UML profile is utilized to construct the meta-modal specifications respectively for common Aspect-Oriented and AspectJ. So the core business logic and the crosscutting aspects can be modeled as separate, modular Aspect-Oriented PIM's and PSM's.

III. SPECIAL REQUIREMENTS AND REAL-TIME COMPONENT MODEL

A. Special Requirements of Real-time Systems

Real-time systems are computing systems whose correct behavior not only depends on the computation results, but also on the time when these results are provided. Embedded real-time systems contain computers as parts of larger systems and interact directly with external devices [8]. They usually have to meet stringent specifications for safety, reliability, limited hardware etc. Thus, there is a strong relationship between hardware and software. Real-time systems consist of different, independent processes or threads that communicate with each other. This communication has to be described, e.g., which system parts may talk to each other and which may not, which messages they can send and receive, or protocols that must be followed. In addition, real-time systems often have to deal with task management as this kind of software often consists of several processes and threads whose scheduling has to be regulated. Therefore, certain aspects of task management like priorities have to be modeled, too. Real-time applications are composed of one or more tasks that are required to perform their functions under strict timing constraints. Most typically, tasks are built of concurrent programs. This concurrency can be implemented as a pseudo-concurrency where multiple tasks compete for the use of a single processor, or true concurrency on multiprocessor systems. When a single processor has to execute a set of concurrent tasks, the CPU is assigned to various tasks according to a pre-defined scheduling policy. Real-time tasks can be categorized as periodic and aperiodic according to whether their arrivals repeat regularly [9].

A typical timing constraint of a task is its deadline, i.e., the instant before which the task should complete its execution. For process scheduling, the attributes of real-time tasks are: Worst Case Execution Time (WCET), which is the maximum time necessary for the processor to execute a task without pre-emption, release time, which is the time at which a task becomes ready for execution, and precedence constraints defining an order of task execution. The smallest structural element in embedded real-time systems should be the task. Each task has a set of input ports and output ports, which are used for communication with other tasks and also constitute the interface to the environment. As shown in Figure 1, here real-time tasks are considered as leaf components. Since a component can also be a composition of other components, a component consists of at least one task, possibly more. Any task is executed starting at its entry point and running to its exit point. A task can assume the states ready, running, waiting, or suspended. The task state transitions are shown in Figure 1.

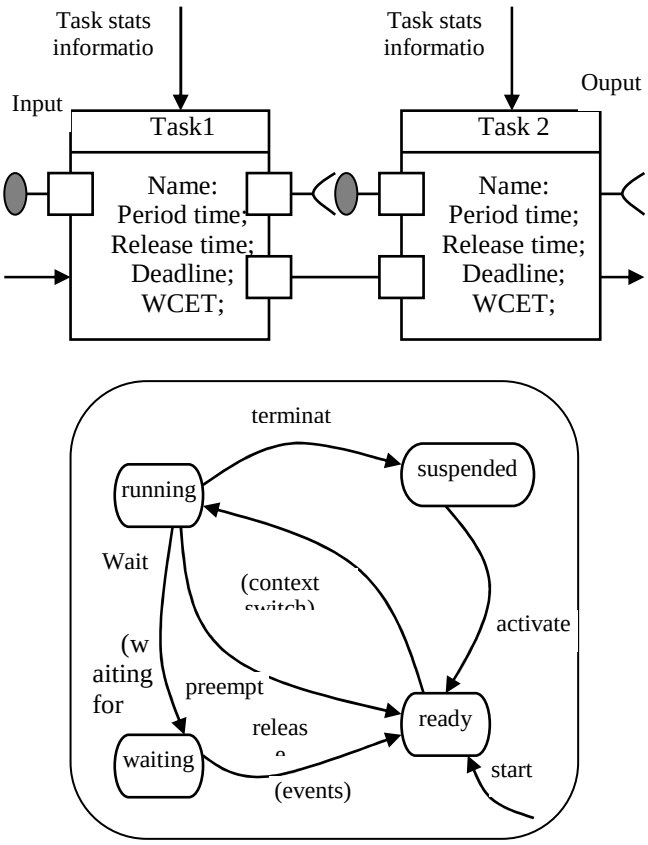


Figure.1. Real-time task as component

A task is running if a processor is assigned to it. At any point in time, only one task can be in the running state. A task is ready, if all the requirements for transition into the running state are met, but no processor is assigned to it, yet. The scheduler decides when a ready task becomes a running one. A ready task can pre-empt a running one if required by priority and enabled by pre-emptivity, otherwise, it has to wait until all tasks with higher priorities have finished.

A task in the state suspended is passive. Before it can enter the state ready, it must be activated. Tasks enter the state waiting when they need to wait for at least one event before they can resume execution.

B. Separation of Cross-Cutting Concerns

AOP is a separation of concerns techniques, namely from the sources to the target under the mapping of the mapping relations, an element of sources scattered in the some of the elements belong to target, and at least there is an element struggle with the certain elements of source in these elements, so we can say that it happened “cross” on the element. Source is the crosscutting concern of the target. Basic idea is to describe the different concerns of the system and their relationship, and achieve a single concern through a loosely coupled way, and then rely on the support mechanism for AOP environment and compile these concerns points to the final run program [5].

As the differences required, it caused the different criteria to achieve the act of crosscutting concerns. We must also consider crosscutting data or property and judge whether they are same. In avionics systems, fast alignment, gyrocompass alignment and calibration alignment complete the basic function of alignment, but they have differences principle, and lead to the differences of established class, so much of their crosscutting concerns should be separately classified as different aspects. During this procedure, this will require strict semantic restrictions to control the balance between the local property and consistency problem. After the crosscutting concerns summarized the corresponding area, the separate process about the aspects from the OO model of avionics software system is an insinuation extraction and iterative process.

Set up a insinuate set. Class set

$$\{C1, C2, Cm\} \quad (1)$$

and crosscutting concerns set

$$\{A1, A2, \ldots, An\} \quad (2)$$

Sort of concerns set. Accordance with the number of quantity and complexity associated to sort the set

$$\{A1, A2, \ldots, An\} \quad (3)$$

and has the

$$A1', A2', \ldots, An' \quad (4)$$

This will single out the large particle size or scope and will help simplify the separate process.

Insinuate. Pick the first aspect A1', and pick the class which may be include A1 and form a new set

$$\{C1, C2, Ck\} \quad (5)$$

The new set may be the true subset of

$$\{C1, C2, Cm\} \quad (6)$$

and also a subset may not be true.

Extraction. Extracting the data and method form the new set one by one in the (3), and making the corresponding connection point, the corresponding Ci recorded as Ci', until the A1 extract from the last Cp. Then bring the new tagged set

$$\{C1', C2', \ldots, Cp'\} \quad (7)$$

back to the original set

$$\{C1, C2, \ldots, Cn\} \quad (8)$$

and separated aspect will cleared from the aspect set.

Repeatedly (3) and (4), until the aspects are null, and then finish the extract job.

It can be seen that, the determine of crosscutting concerns is to reduce the coupling degree of the system process, which improve the system scalability, maintainability; aspects of the separation system enhance the analytical, makes sensors logic behavioral more easily implant into system through aspect, and pave the way for further aspects of the restructuring. After implementation the separation of crosscutting concern, it will combine the concerns points with software real-time techniques, using real-time implanted theology, restructure the crosscutting concerns to aspect oriented. In this way, the real-time system will collect procedures status information, and save to the information database, it extend the assertion system and debugging system, and make it can declare assertion in anywhere, provide the more information; aspect oriented technology separate the time and error-control behavior, and form an aspect to achieve real-time control, such the time of expression system and the error will not be distributed at each module, it easy to implementation and manage system when all the time and wrong behavior centralized managed. Thus the system real-time, low-aspect-oriented coupling, the flexibility of the system can be good play.

Aspect-oriented restructuring is to have the completed functional design of the real-time system implanted into the original components that is the inverse process of crosscutting concerns separation. Analyzing the property of concerns points in concerns set {A1, A2, …, An}, concerns raised here are the smaller particle size of concern, such as variety requirements exist in real-time requirements. The failure time of avionics system, there are 50ms, 2.5min and so many cases, the details of the sensor design in different circumstance should be taken into account.

C. Modeling Aspect-oriented PIM for Real-time System

The processes of designing platform independent real-time component models in the course of developing embedded real-time systems were shown [10]. A PIM component model captures the essential features of a system, and specifies what it does, while a PSM component model describes how the PIM component model is implemented on a specific platform. Platform-independent component models are designed by defining a set of Stereotypes, Tagged Values and Constraints according to the requirements for software in embedded systems, and assigned to a Component Meta-model [11]. Platform-Independent Model and Platform-Specific Model are two important models proposed in MDA.PIM is the business model of which reflects functional requirements, and PSM is the mapping from PIM to implementation technology [5]. Models for non-functional requirements such as security, functionality, memory management and communication are not given in MDA. An aspect diagram notation is represented based on UML Profile according to Aspect-Oriented Programming concept [6]. The aspect diagram is applied to build Aspect-Oriented Model reflecting non-functional requirements in MDA. Model-weaving process is added to MDA, and therefore a more powerful MDA development approach based on aspect is obtained.

An aspect is then combined with the classes it crosscuts according to specifications given within the aspect. Moreover, an aspect can introduce methods, attributes, and interface implementation declarations into types by using the inter-type declaration construct. The essential mechanism provided for composing an aspect with other classes is called a join point. A join point is a well-defined point in the execution of a program, such as a call to a method, an access to an attribute, an object initialization, exception handler, etc. Sets of join points may be represented by pointcuts. AspectJ provides various pointcut designators that may be combined through logical operators to build up complete descriptions of point cuts of interest. An aspect can specify advices that are used to define some code that should be executed when a point cut is reached. An advice is a method-like mechanism that consists of a piece of code to be executed before, after, or around a point cut. An AspectJ program can be divided into two parts: a base code part which includes classes, interfaces, and other language constructs for implementing the basic functionality of the program, and an aspect code part which includes aspects for modeling crosscutting concerns in the program. For further information about AspectJ, one can refer to [5].

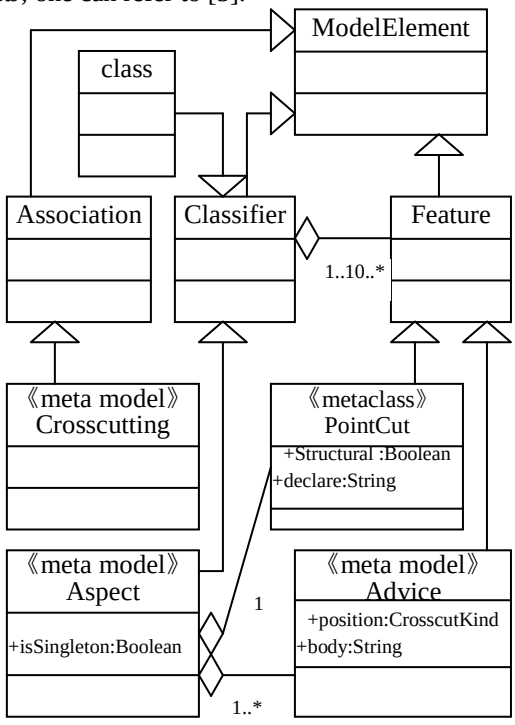


Figure 2. Meta-model of Aspect-oriented Real-time PIM.

Aspect-oriented platform-independent PIM needs to be expressed in the form's structure and behavior characteristics, such as crosscutting aspects of the operation, the selected connection point of the structure of crosscutting relations. If the direct use of object-oriented model or a simple extension that will lead to the case in the absence of the model specification appears

ambiguous. In the MDA model transformation in the end lead to failure. Therefore need to use PIM meta-model to define.

In addition, cross-platform support of model exchange and reuse, aspects of the PIM meta-model level should also be defined, in order to have good features and unified platform-independent specifications. Conforming to the MDA approach, a platform-independent real-time component model for embedded real-time systems is defined as shown in Fig. 2, in which it addresses functionality constraints, and specifies efficient functional interfaces and architectural constraints. It clearly separates the requirements and their implementation in terms of a particular execution platform.

According to the basic concepts of AOP, the aspect PIM meta-model describes the modeling aspect-oriented crosscutting model elements required for the establishment of aspect-oriented specification provides the definition of PIM, the aim is to develop the framework of MDA can First, the platform independent model, namely the establishment of aspect-oriented PIM.Meta-model of the various aspect-oriented platform has some common characteristics of abstract, semantically covers a total of aspect-oriented nature and behavior, ignoring the details of the specific platform that can guide the establishment of aspect-oriented PIM.

D. Modeling Aspect-oriented PSM for Real-time System

Which specifies how the system is implemented? It determines how the PIM executes in the target deployment environment. A PIM is used as foundation for mapping into one or more platform specific models.

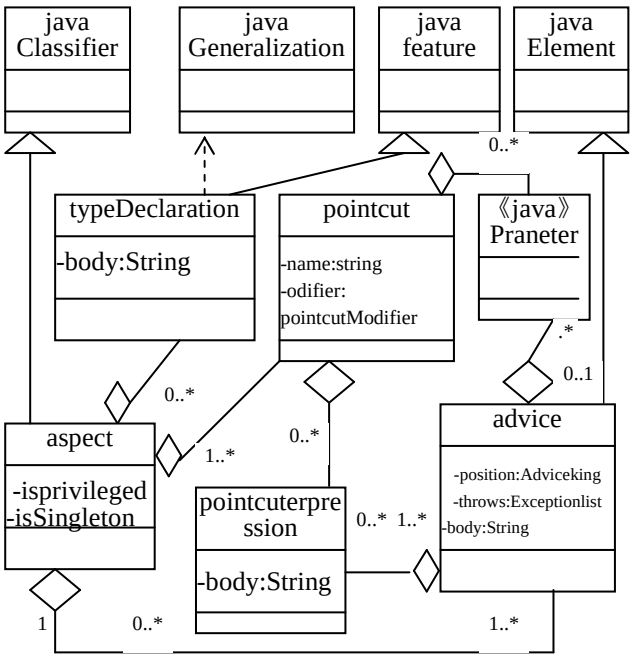


Figure 3. Meta-model of Aspect-oriented Real-time PSM.

Such a PSM describes in detail how the PIM is implemented on a specific platform, or in a certain technology. PSM's are also expressed in UML adding constraints and implementation details. PSM is generally from the top of the PIM transform, but now there is uniform conversion standard. Therefore, in order to establish respect for the PSM and the corresponding transformation in, you can first increase the PSM of the specification describing. Modeling after the PSM research based on the relationship between the specific transform. Here's a specific platform technology selected AspectJ, need to build AspectJ meta-model, to ensure the establishment of a unified description of the form of PSM. When the platform-specific model has a unified standard, be possible to achieve the mapping PIM to PSM and PSM to code to achieve the conversion from the basis.

The PSM model is the concept of aspect-oriented expression of the PIM is almost the same with the previous chapter, so you can see from Figure 3 added element model changes are not large. The biggest difference lies with the AspectJ and specific combination of the specific syntax of the language. Element model is with the name to distinguish PIM, PSM meta-model name all lower case letter.

IV. APPLYING MDA TO PIM AND PSM

The principle of this development process is to establish, first, a PIM, then, to refine it according to a specific situation and, next, to derive one or more PSM's from this PIM. These PSM's are the results of mappings to several platform models. Both PIM and PSM's are developed according to the rules defined within a meta-model of which can be described in UML. If necessary, the PIM and PSM's can be optimized to reduce their complexity. The PSM produced last is used to generate executable code, preferably in an automatic way. The transformation from one PIM to several PSM's is the core of MDA, while the one from a PSM to executable code is straightforward and nowadays supported by a number of commercial tools.

A. Analysis of Real-time Aspects for the Washer

In order to more clearly understand how to complete the MDA in the aspect-oriented modeling, especially in real-time system. Following through the example of a real-time system is discussed. The system simulates the operation of self-automatic washer process.

Overview of washer system functions:

l) Users before washing clothes texture and weight according to the control panel, select the amount of water, washing time and washing methods. Then there is the remaining amount will include the laundry card into the card slot.

2) When you press the Start button, less the cost of laundry card, while washer started to work in accordance with user requirements and tips ringing sound. After running, the motor starting time decreases the cycle time, when you press the Cancel button to stop working.

3) When the washer work will be suspended when the outer door laundry work, stop the timer, close the outer door, then continue to work while continuing to timing.

4) The time taken to reach zero when the laundry is completed, run the washer motor power and lights will automatically turn off prompt, suggesting that the bell will ring three times to prompt the laundry work is done.

5) The motor record time and with each washing had been running all the time before adding, in the control panel shows the total hours of work.

6) Require the user card, the controller displays in the three seconds less than the balance of the card.

7) Require the user presses the start button a second charge.

From the above system (show figure 4) can be summarized in the basic classes, as shown, which includes the user, the washer controller (selection panel), motor, door sensors, indicators, and they may have various properties, such as the controller of the water, laundry format and more.

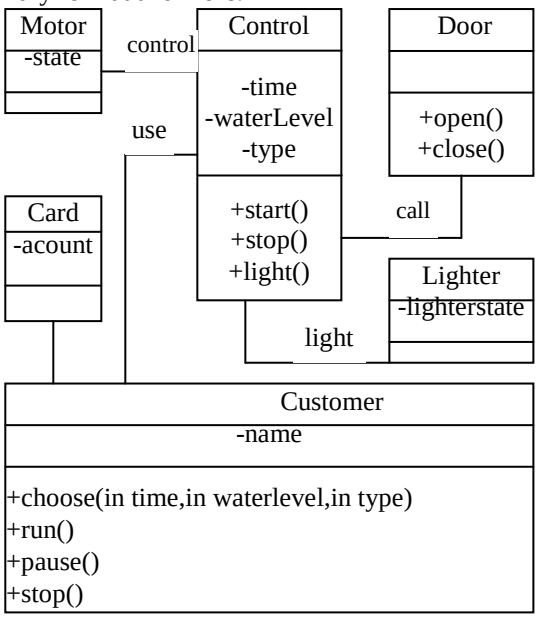


Figure 4. Core Class of Washer

The washer system has the typical characteristics of real-time systems. Mentioned earlier, the biggest feature of real-time system is required for the system within a certain time limit to respond. In this system, after the display card charge back fees, and laundry are two functions need to be completed within a certain time interval. In addition, after the start of washing laundry in the countdown and begin recording the motor to run this time. These four mechanisms function with time can be divided into two categories; the first two and the time constraints related to the last two and timing related. Since this is a very simplified system, the system in practical applications there will be many areas with similar time constraints or timer functions. Should therefore be both time-related aspects in the modeling.

B. Aspect Time for PIM and PSM

Timer to time constraints and modeling, the literature OMG's modeling time model can be used as a case. PIM to establish aspect-oriented, by analyzing the demand for shows, you should create a time dimension, which it has to count down the operation of the motor cycle time and total time. Processes for each washing into a timer set for each user laundry countdown, but also to motor run time, the motor controller should calculate the total time to retain a variable to be updated after each washing. Show Figure 5.

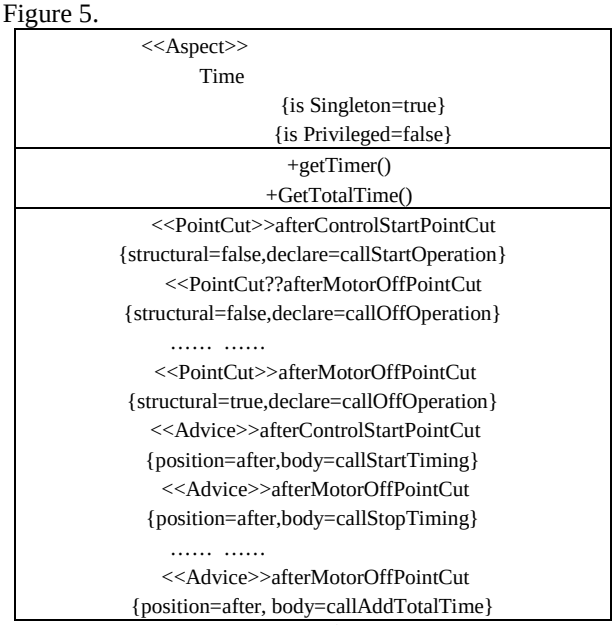
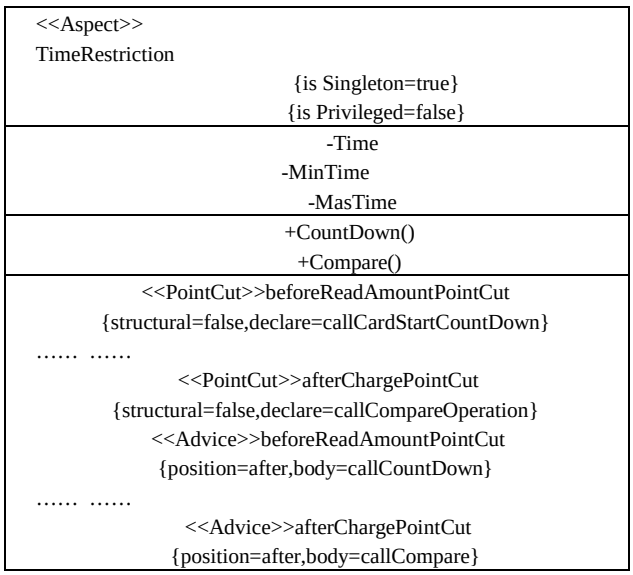


Figure 5 PIM of Aspect Time

In order to run the motor all the time, time, Time to introduce a total Time variable, in the laundry after each time the user running the second time to the total running time of the motor. Can also be seen from the figure are structural cross-section of the process.

Figure 6 is a time-bound aspect of PIM; it will be time aspects of a property as their own. Cut into the time constraints the timer before running the method, while the value of the specified time from the constraint starts to count down.

Figure 6 PIM of Aspect Time Restriction



Countdown to zero, if the methods have not received the news of the end, it means that did not meet the time constraints, go to the appropriate error handling. In the

end of the method (after) compare method is called, first stop the countdown, then the time spent compared with minTime and maxTime obtained meets all of the time constraints.

Figure 7 is in accordance with the terms of the PSM element model for the establishment of the time constraint specification aspects of PSM.

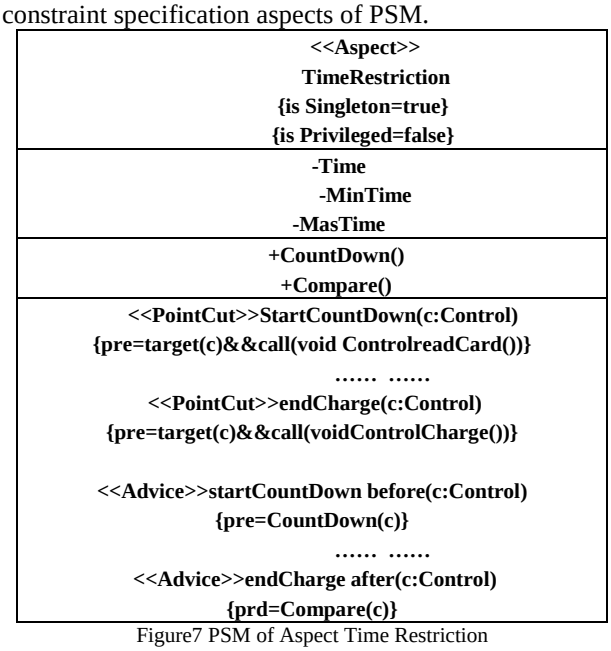


Figure7 PSM of Aspect Time Restriction

C. Added the Core Aspect to PIM and PSM

Washing machine is in fact the basic class diagram has been described in its core function as the core PIM can be transformed. After transformation is in Figure 5, the core of PSM. A clearer response to cross-cutting impact in this figure only those associated with the timing of the core classes that are modeled with 00 ideas, and because the specific implementation technologies through the combination of model transformation, which expressed as ordinary Java classes. The area has been by the time was converted to Figure 6, Figure 8 how a time-based Aspect PSM. Aspects and core classes for the connection to the following figure to express their relationship with the crosscut.

The principle of this development process is to establish, first, a PIM, then, to refine it according to a specific situation and, next, to derive one or more PSMs from this PIM. These PSMs are the results of mappings to several platform models. Both PIM and PSMs are developed according to the rules defined within a meta-model which can be described in UML. If necessary, the PIM and PSMs can be optimized to reduce their complexity. The PSM produced last is used to generate executable code, preferably in an automatic way. The transformation from one PIM to several PSMs is the core of MDA, while the one from a PSM to executable code is straightforward and nowadays supported by a number of commercial tools. functionality can be used as the core PIM for transformation, the core of Figure 7 PSM. In order to more clearly reflect the aspects of cross. The impact of cutting, Figure 5 and the timing of selected related core classes, after the model with specific implementation technology transformation, they become ordinary Java classes.

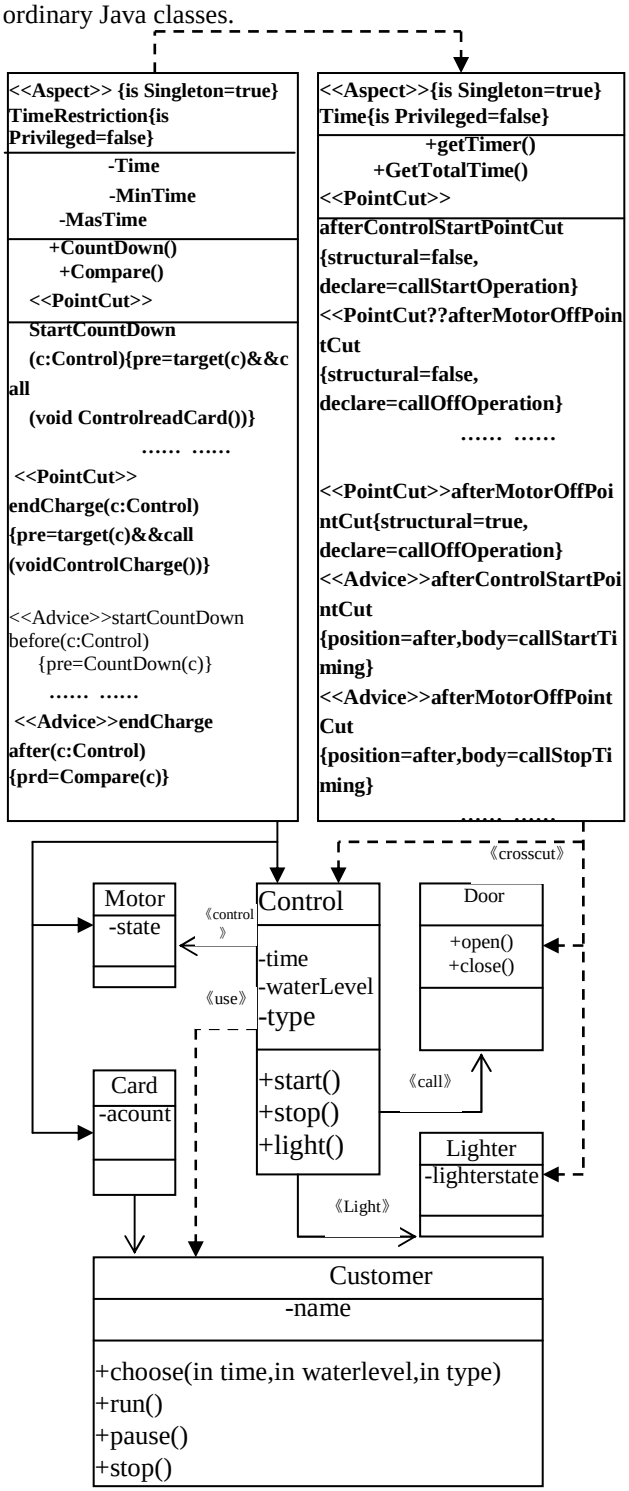


Figure 8 The Core Aspects of PSM by Adding Timing Aspects of PSM and Time Constraints PSM

The timing has been shown in Figure 4, Figure 5 was converted to a timing based on AspectJ PSM. In Figure 8, the aspects and the connection between the core classes with a crosscut expression. Therefore, aspect-oriented technology solves real-time systems due to crosscutting concerns in code tangling and code-dispersion problem, and MDA eliminated because of technological change to achieve the impact on the software system will aspect-

oriented MDA real-time system development for real-time systems can improve the reusability and maintainability, reduce development costs.System class diagram describes the basic core, Real-Time systems have been applied in many areas widely, but they have many non-functional requirements, which always crosscut the whole system modules. That may cause the code tangle and scatter, make the systems hard to design, reuse and maintain, and affect performance of systems badly. AOP is a new software development paradigm, which could attain a higher level of separation of concerns in both functional and non-functional matters by introducing aspect, for the implementation of crosscutting concerns. Different aspects can be designed separately, and woven into systems. This article introduces the technology of MDA, aspect-oriented, real-time systems and UML. This article takes the Aspect-oriented to the MDA modeling by the UML extension mechanisms, and presents a method, which is Aspect-Oriented MDA. In this article, UML profile is utilized to construct the meta-modal specifications respectively for common Aspect-Oriented and AspectJ. So the core business logic and the crosscutting aspects can be modeled as separate, modular Aspect-Oriented PIM's and PSM's. The authors analysis non-functional requirements of the real-time systems, and then apply aspect-oriented MDA modeling to develop an example of real-time systems, and propose how to model aspects of timer and real-time constraints. Finally, the author use extended UML diagram to dynamically show how aspect impact real-time systems.

V. CONCLUSION

Real-time systems have many non-functional requirements, which crosscut entire system and are difficult to be handled. This paper presents a real-time system development method based on aspect-oriented Model-Driven Architecture. It separates the handing of non-functional requirements from the functional ones in design phase, which reduces the complexity of real-time systems development and improves reusability and maintainability of real-time systems, and modularization of crosscutting concerns. Application case proves that the method is effective.

In developing software for embedded systems one has to consider non-functional and resource constraints besides software quality aspects such as re-usability, because the correct operation of such a system is not only dependent on the correct functional working of its components, but also dependent on its non-functional properties. Embedded systems often have limited resources such as processing power, storage capacity and network bandwidth. A developer has to cope with these constraints and make sure that the software will be able to run on the constrained system. In addition, embedded systems also have timing constraints on their computations. Missing a time constraint can be catastrophic or annoying. In order to cope with these special requirements, it is indispensable to improve and enhance the component technologies to design software for embedded systems.

ACKNOWLEDGMENT

This work is supported by the Major Program of National Natural Science Foundation of China under Grant No.90818008, and by the National Natural Science Foundation of China under Grant No.60774095.

REFERENCES

[1] Baruah, S.K.: An improved EDF schedulability test for uniform multiprocessors. In: Proc. 16th IEEE Real-time and Embedded Technology and Applications Symposium (April 2010)

[2] P. Boulet, J. L. Dekeyser, C. Dumoulin and P. Marquet: MDA for SoC Embedded Systems Design, Intensive Signal Processing Experiment. Proc. SIVOES-MDA, Workshop at UML2003, San Francisco, 2003, pp.166–174.

[3] Yongfeng Yin, Bin Liu, Zhen Li, Chun Zhang, Ning Wu: The Integrated Application Based on Real-time Extended UML and Improved Formal Method in Real-time Embedded Software Testing. Journal of Networks, Vol. 5, No 12 (2010), pp1410-1416, Dec 2010.

[4] Youtian Qu, Chaonan Wang, Lili Zhong, Huilai Zou, Hua Liu: Research for an Intelligent Component-Oriented Software Development Approaches Journal of Software, Vol 4, No 10 (2009), pp1136-1144, Dec 2009.

[5] Youtian Qu, Chaonan Wang, Lili Zhong, Huilai Zou, Hua Liu: Research for an Intelligent Component-Oriented Software Development Approaches. Journal of Software, Vol. 4, No 10 (2009), pp1136-1144, Dec 2009.

[6] Elrad T, Filman R E. Aspect-Oriented Programming [J].Communication of ACM, 2010, 44(10): pp. 29-32..

[7] K. Sandstr¨om, J. Fredriksson and M. Akerholm: Introducing a Component Technology for Safety Critical Embedded Real-Time systems. Proc. Intl. Symposium on Component-based Software Engineering, Edinburgh. Vol. 2, pp. 740–741, August 2004.

[8] Papapetrou O, et al. Aspect Oriented Programming for a component-based real life application: A case Study .2004 ACM Symposiums on Applied Computing. 2004, pp.1555-1558.

[9] Mraidha C, et al. A Two-Aspect Approach for a Clearer Behavior Model .Proc.6th IEEE international Symposium on Object-Oriented Real-Time Distributed Computing (ISORC03).

[10] Zhang Lichen,Liu Ruicheng. Aspect-Oriented Real-time System Modeling Method Based on UML[C]//Proc.of RTETA'05.[S.l.]:IEEE Press,2005.pp.546-551.

[11] Kniesel H Y. Towards Visual AspectJ by a Meta Model and Modeling Notation[C]//Proc. of the 6th International Workshop on Aspect-Oriented Modeling. Chicago, Illinois, USA, 2005.

Wei Qiu, born in 1974, received M.S. degree in Software Engineering from Guangdong University of Technology of China in 2004. He is an Associate Professor at School of Computer Science and Jia Ying University. His main research interests Data engineering, data integration, intelligent software, Real-time Systems and intelligent mobile technology.

Li-Chen Zhang, born in 1962. He is a full professor and Ph.D in Guangdong University of Technology. His main research interests SOA, Real-time Systems.