Aspect-Oriented QoS Modeling for Cyber-Physical Systems

Lichen Zhang
Guangdong University of Technology, Guangzhou, China
Email: zhanglichen1962@163.com

***Abstract*—Cyber-physical systems (CPSs) are physical and engineered systems whose operations are monitored. Cyber-physical systems having quality-of-service (QoS) requirements driven by the dynamics of the physical environment in which they operate, the description, control, management, consultation and guarantee of QoS are very complex and challenging work, Quality of Service(QoS) is directly related to system's performance. This paper proposes an aspect-oriented QoS modeling method based on UML and formal methods. We use an aspect-oriented profile by the UML meta-model extension, and model the crosscutting concerns by this profile. In this paper, we build the aspect-oriented model for specifying Quality of Service (QoS) based on the combination of UML and RTL. We believe that the two types of notation, graphical (semi-formal) and, respectively, formal, can efficiently complement each other and provide the basis for an aspect-oriented specification approach that can be both rigorous and practical for QoS modeling. Two examples depict how aspect-oriented methods can be used during QoS analysis and design process.**

***Index Terms*—QoS, Cyber Physical systems, Aspect-Oriented**

I. INTRODUCTION

A cyber-physical system (CPS)[1] is a system featuring a tight combination of, and coordination between, the system's computational and physical elements. Today, a pre-cursor generation of cyber-physical systems can be found in areas as diverse as aerospace, automotive, chemical processes, civil infrastructure, energy, healthcare, manufacturing, transportation, entertainment, and consumer appliances. The dependability of the software [1]has become an international issue of universal concern, the impact of the recent software fault and failure is growing, such as the paralysis of the Beijing Olympics ticketing system and the recent plane crash of the President of Poland. Therefore, the importance and urgency of the digital computing system's dependability began arousing more and more attention. A digital computing system's dependability refers to the integrative competence of the system that can provide the comprehensive capacity services, mainly related to the reliability, availability, testability, maintainability and safety. With the increasing of the importance and urgency of the software in any domain, the dependability of the distributed real-time system should arouse more attention.[2]

Fundamental limitations for Cyber-Physical Systems (CPS) include[3][4]:

• Lack of good formal representations and tools capable of expressing and integrating multiple viewpoints and multiple aspects. This includes lack of robust formal models of multiple abstraction layers from physical processes through various layers of the information processing hierarchy; and their cross-layer analyses.

• Lack of strategies to cleanly separate safety-critical and non-safety-critical functionality, as well as for safe composition of their functionality during humanin-the-loop operation.

• Ability to reason about, and tradeoff between physical constraints and QoS of the CPS.

Aspect-oriented programming (AOP) [5] is a new software development technique, which is based on the separation of concerns. Systems could be separated into different crosscutting concerns and designed independently by using AOP techniques.

When designing the system, aspects are analyzed and designed separately from the system's core functionality, such as real-time, security, error and exception handling, log, synchronization, scheduling, optimization, communication, resource sharing and distribution. After the aspects are implemented, they can be woven into the system automatically. Using AOP techniques will make the system more modular, and developers only design the aspect without considering other aspects or the core component.

Real-time feature is the most important aspect of the cyber physical system, which determines the performance of the system. So we describe the real-time feature as an independent aspect according to the AOP techniques, and design a time model to realize and manage the time aspect in order to make the system easier to design and develop and guarantee the time constraints.

The QoS of dependable cyber physical system is very complex, currently the QoS research still hasn't a completely and technical system, and there isn't any solution meeting all the QoS requirements. We design the QoS of dependable real-time system as a separate Aspect using AOP, and proposed the classification of complex QoS, divided into the timing, reliability and safety and other sub-aspects. These sub-aspects inherit t members and operations from the abstract QoSaspect. We design each sub-aspects through aspect-Oriented modeling, to ensure the Quality of dependable real-time system meeting the requirements of the dependability.

doi:10.4304/jsw.7.5.1083-1093

This paper proposes an aspect-oriented QoS specification method based on UML, mainly form an aspect-oriented profile by the UML meta-model extension, and model the crosscutting concerns by this profile.

II. QoS of Cyber physical systems

Cyber physical systems are characterized by their stringent requirements for quality of service (QoS), such as predictable end-to-end latencies, timeliness and scalability. Delivering the QoS needs of cyber physical systems entails the need to configure correctly, fine-tune and provision the infrastructure used to host the cyber physical systems, which crosscuts different layers of middleware, operating systems and networks. These tangled deployment and configuration concerns of cyber physical systems requires integrating the principles of Aspect-Oriented Software Design (AOSD) with MDD. Cyber physical systems share the following characteristics giving rise to tangled concerns in its development and maintenance lifecycle[6]:

Heterogeneity. Large-scale cyber physical systems often run on a variety of computing platforms that are interconnected by different types of networking technologies with varying QoS properties. The efficiency and predictability of cyber physical systems built using different infrastructure components varies according to the type of computing platform and interconnection technology.

Deeply embedded properties. cyber physical systems are frequently composed of multiple embedded subsystems. For example, an anti-lock braking software control system forms a resource-constrained subsystem that is part of a larger cyber physical application controlling the overall operation of an automobile.

Simultaneous support for multiple QoS properties. cyber physical software controllers are increasingly replacing mechanical and human control of critical systems. These controllers must simultaneously support many challenging QoS constraints, including (1) real-time requirements, such as low latency and bounded jitter, (2) availability requirements, such as fault propagation/recovery across boundaries, (3) security requirements, such as appropriate authentication and authorization, and (4) physical requirements, such as limited weight, power consumption, and memory footprint. For example, a distributed patient monitoring system requires predictable, reliable, and secure monitoring of patient health data that can be distributed in a timely manner to healthcare providers.

Large-scale, network-centric operation. The scale and complexity of cyber physical systems makes it infeasible to deploy them in disconnected, standalone configurations. The functionality of cyber physical systems is therefore partitioned and distributed over a range of networks. For example, an urban bio-terrorist evacuation capability requires highly distributed functionality involving networks connecting command and control centers with bio-sensors that collect data from police, hospitals, and urban traffic management systems.

Dynamic operating conditions. Operating conditions for large-scale cyber physical systems can change dynamically, resulting in the need for appropriate adaptation and resource management strategies for continued successful system operation. In civilian contexts, for instance, power outages underscore the need to detect failures in a timely manner and adapt in real-time to maintain mission-critical power grid operations. In military contexts, likewise, a mission mode change or loss of functionality due to an attack in combat operations requires adaptation and resource reallocation to continue with mission-critical capabilities.

III. Aspect-Oriented QoS modeling

The integration of physical systems and processes with networked computing has led to the emergence of a new generation of engineered systems: Cyber-Physical System s(CPS). Such systems use computations and communication deeply embedded in and interacting with physical processes to add new capabilities to physical systems. These cyber-physical systems range from miniscule (pace makers) to large-scale (the national power-grid). Because computer-augmented devices are everywhere, they are a huge source of economic leverage. A CPS is a system in which computation/information processing and physical processes are so tightly integrated that it is not possible to identify whether behavioral attributes are the result of computations (computer programs), physical laws, or both working together; Real/Continuous time models take physical durations into account. These are important for doing various time-related analyses (e.g., deadline matches) and, in particular, for real-time scheduling as in RMA approaches [6]. They are also used for modeling the temporal characteristics of the physical environment or system with which the embedded system is interacting (usually before discretization).A real-time and embedded modeling language for cyber physical systems needs concepts for dealing with different models of time. In order to meet the challenge of cyber-physical system design, we need to realign abstraction layers in design flows and develop semantic foundations for composing heterogeneous models and modeling languages describing different physics and logics. One of the fundamental challenges in research related to CPSs is accurate modeling and representation of these systems. The main difficulty lies in developing an integrated model that represents both cyber and physical aspects with high fidelity. Among existing techniques, aspect-oriented modeling is a suitable choice, as it can encapsulate diverse attributes of cyber physical systems. The control-centric nature of the programmatic QoS adaptation extends beyond software concepts, e.g., issues such as stability and convergence become paramount. In cyber physical system, QoS is specified in software parameters, which have a significant impact on the dynamics of the overall physical system. Due to complex and non-linear dynamics, it is hard to tune the QoS parameters in an *ad hoc* manner without compromising

the stability of the underlying physical system. The QoS adaptation software is, in effect, equivalent to a controller for a discrete, non-linear system. Sophisticated tools are therefore needed to design, simulate, and analyze the QoS adaptation software from a control system perspective.

Programmatic QoS adaptation approaches offer a lower level of abstraction, i.e., textual code based. For example, implementing a small change to CDL-based adaptation policies requires manual changes that are scattered across large portions of the cyber physical system, which complicates ensuring that all changes are applied consistently. Moreover, even small changes can have far-reaching effects on the dynamic behavior due to the nature of emergent crosscutting properties, such as modifying the policy for adjusting communication bandwidth across a distributed surveillance system.

QoS provisioning also depends on the performance and characteristics of specific algorithms that are fixed and cannot be modified, such as a particular scheduling algorithm or a specific communication protocol. These implementations offer fixed QoS and offer little flexibility in terms of tuning the QoS. Consequently, any QoS adaptation along this dimension involves structural adaptation in terms of switching implementations at run-time, which is highly complex, and in some cases infeasible without shutting down and restarting applications and nodes. Moreover, issues of state management and propagation, and transient mitigation, gain prominence amid such structural adaptations. Programmatic QoS adaptation approaches often offer little or no support for specifying such complex adaptations.

Object-Oriented Programming (OOP) has been the dominant programming methodology that is being used in all kinds of software development today. The main focus of OOP is to find a modular solution for a problem by breaking down the system into a collection of classes that encapsulates state and behavior. However, In Object-Oriented Programming, crosscutting concerns are elements of software that can not be cleanly captured in a method or class. Accordingly, crosscutting concerns has to be scattered across many classes and methods. OOP fails to provide a robust and extensible solution to handle these crosscutting concerns. AOP is a new modularity technique that aims to cleanly separate the implementation of crosscutting concerns. It builds on Object-Orientation, and addresses some of the points that are not addressed by OO. AOP provides mechanisms for decomposing a problem into functional components and aspectual components called aspects[7]. An aspect is a modular unit of crosscutting the functional components, which is designed to encapsulate state and behavior that affect multiple classes into reusable modules. Distribution, logging, fault tolerance, real-time and synchronization are examples of aspects. The AOP approach proposes a solution to the crosscutting concerns problem by encapsulating these into an aspect, and uses the weaving mechanism to combine them with the main components of the software system and produces the final system. We think that the phenomenon of handling multiple orthogonal design requirements is in the category of crosscutting concerns, which are well addressed by aspect oriented techniques. Hence, we believe that system architecture is one of the ideal places where we can apply aspect oriented programming (AOP) methods to obtain a modularity level that is unattainable via traditional programming techniques. To follow that theoretical conjecture, it is necessary to identify and to analyze these crosscutting phenomena in existing system implementations. Furthermore, by using aspect oriented languages, we should be able to resolve the concern crosscutting and to yield a system architecture that is more logically coherent. It is then possible to quantify and to closely approximate the benefit of applying AOP to the system architecture.

UML is acquainted to be the industry-standard modeling language for the software engineering community, and it is a general purpose modeling language to be usable in a wide range of application domains. So it is very significant to research aspect-oriented real-time system modeling method based on UML[8]. However they didn't make out how to model real-time systems, and express real-time feature as an aspect. In this section, we extend the UML, and present an aspect-oriented method that model the real-time system based on UML and Real-Time Logic (RTL)[9]. Real Time Logic is a first order predicate logic invented primarily for reasoning about timing properties of real-time systems. It provides a uniform way for the specification of both relative and absolute timing of events. Real-time logic (RTL), which is based on first-order logic with restricted features, captures the timing requirements of real-time systems. Real-Time Logic provides a uniform way for the specification of relative and uniform timing of events. It is an extension of integer arithmetic without multiplication (Presburger arithmetic) that adds a single uninterpreted binary *occurrence function*, denoted by @, to represent the relationship between events of a system, and their times of occurrence. The equation @(*e; i*) = *t* states that the time of the i_ith occurrence of event *e* is *t:* Let us denote with Z, N and N+ the set of integers, positive integers, and strict positive integers, respectively. The time occurrence function is a mapping @: *E £* N+ *!* N, where *E* is a domain of events, and such that @ is strictly monotonically increasing in its second argument, i.e., @(*E; i*) <@(*E; i*+ 1), for any *i 2* N+*:* It is supposed that all events may occur infinitely often. There are no event variables, or uninterpreted predicate symbols. So, RTL formulas are boolean combinations of equality and inequality predicates of standard integer arithmetic, where the arguments of the relations are integer valued expressions involving variables, constants, and applications of the function symbol@. The correctness of a real-time system can be achieved by Computing the

satiability of an associated propositional formula[14] Extended Real Time Logic (ERTL)[10] is a formalism for the modeling and analysis of relative and absolute timing properties of cyber physical systems (systems that combine continuous variables and discrete event dynamics). The extensions provided by ERTL enable the modeling of system behaviour ranging from activities of the physical entities that form part of the environment of a computing system, to the temporal ordering of the computational tasks of the computing system itself, thus providing a formal notation that can be used in all stages of software development.. The specification of the Object Constraint Language (OCL) [11] is a part of the UML specification, and it is not intended to replace existing formal languages, but to supplement the need to describe the additional constraints about the objects that cannot be easily represented in graphical diagrams, like the interactions between the components and the constraints between the components' communication . Since OCL is an expression language, it can be checked without an executable system. All these features turn out to be useful in representing QoS properties, which can be represented by the combination of precondition, post-condition and invariant in OCL. The QoS attributes are represented by the member variables of the class, and the QoS actions are represented by the methods. They are checked at run time, before and after the calls so that the change of the QoS parameters of the system is monitored in a timely basis.

As the QoS concern needs to be considered inmost parts of the system, it is a cross-cutting concern.[12] Cross-cutting concerns[30] are concerns that span multiple objects or components. Cross-cutting concerns need to be separated and modularized to enable the components to work in different configurations without having to rewrite the code. If the code for handling such a concern is included in a component, it can make the component tied to a specific configuration. This code will typically be scattered all over the component implementation and tangled with other code in the component. Modularizing it will make it more robust for change, and separating it totally from the component implementation will save the component programmers from having to implement it. Aspect oriented programming is a new method for modularizing cross-cutting concerns. By using AOP, concerns can be modularized in an aspect and later weaved into the code. Fig.1 shows aspect model of QoS framework.

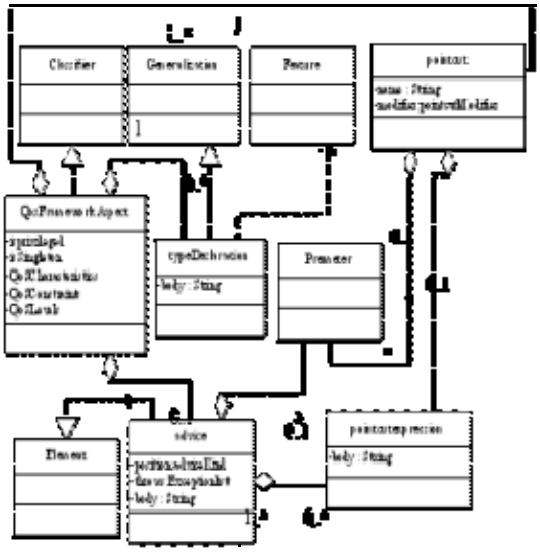

Figure 1. Aspect model of QoS framework

IV. Related Works

Developing cyber physical systems is hard since it requires a coordinated, physics-aware allocation of CPU and network resources to satisfy their end-to-end quality-of-service (QoS) requirements. Jaiganesh Balasubramanian et al. make two contributions to address these challenges[13]. First, Jaiganesh Balasubramanian et al. present model-driven middleware called NetQoPE that shields application developers from the complexities of programming the lowerlevel CPU and network QoS mechanisms by simplifying (1) the specification of per-application CPU and per-flow network QoS requirements subject to the physical constraints and dynamics, (2) resource allocation and validation decisions (such as admission control), and (3) the enforcement of per-flow network QoS at runtime. Second, they empirically evaluate how NetQoPE provides QoS assurance for CPS applications. The results demonstrate that NetQoPE provides flexible and non-invasive QoS configuration and provisioning capabilities by leveraging CPU and network QoS mechanisms without modifying application source code.

The development of Distributed Real-Time and Embedded (DRE) systems is often a challenging task due to conflicting Quality of Service (QoS) constraints that must be explored as trade-offs among a series of alternative design decisions. The ability to model a set of possible design alternatives, and to analyze and simulate the execution of the representative model, offers great assistance toward arriving at the correct set of QoS parameters needed to satisfy the requirements for a specific DRE system. Jeff Gray et al. present a model driven approach for generating QoS adaptation in DRE systems[14]. The approach involves the creation of high-level graphical models representing the QoS adaptation policies. The models are constructed using a domain-specific modeling language - the Adaptive Quality

Modeling Language – which presents a view-oriented separation of common concerns. The proposed method motivates the need for aspect weavers at the modeling level, and provides a case study that is based on bandwidth adaptation in video streaming of an Unmanned Aerial Vehicle (UAV).

Carsten Köllman, Lea Kutvonen, Peter Linington, Arnor Solberg presents an approach for managing several dependability dimensions[15]. They use aspect oriented and model driven development techniques to separate and construct QoS independent models, and graph-based transformation techniques to derive the corresponding QoS specific models.

QoS-UniFrame[16] classifies quantifiable QoS requirements into static and dynamic. *Static QoS* is design-related, and *dynamic QoS* is substantially influenced by the deployment environment. Many of the static QoS requirements can be evaluated at component assembly time, yet dynamic QoS requirements need either simulators or virtual machines to monitor, predict, and adapt the QoS concerns. However, several dynamic QoS requirements can be assessed by referring to a component's previous state and observations, as stored in a knowledge base at assembly time. Static and dynamic QoS parameters may be further subclassified into strict and non-strict, and orthogonal and non-orthogonal QoS. *Strict QoS* requirements (e.g., hard deadlines) force DRE systems to meet the requirements. Otherwise, the system will be incorrect because it cannot meet its QoS. *Non-strict* QoS requirements (e.g., soft deadlines) allow margins of error when meeting QoS requirements. The performance of the system will be degraded according to the magnitude that non-strict QoS requirements are not assured. *Orthogonal QoS* implies that its adaptation will not influence other QoS, yet *non-orthogonal QoS* substantially affects other QoS directly or indirectly. According to the hierarchy of classification, QoS-UniFrame separates static and dynamic QoS into a two-level assurance process.

Bikash Sabata et al. specify QoS as a combination of metrics and policies[17]. QoS metrics are used to specify performance parameters, security requirements and the relative importance of the work in the system. They define three types of QoS performance parameters: Timeliness, Precision, and Accuracy. QoS policies capture application-specific policies that govern how an application is treated by the resource manager. Examples of such policies are management policies and the levels of service.

Mohammad Mousavi et al. present an extension to the GAMMA formalism[18], which they name AspectGAMMA[18], and we show how non-computational aspects can be expressed separately from the computation in this framework. They discuss the main characteristics of an aspect-oriented formal specification framework, which is based on a multiset transformation language called GAMMA, a formalism based on multiset rewriting they illustrate how having a tailor-made formalism for each aspect that is abstracted from other aspects is a key benefit of such a formal design framework. To clarify the discussions, they sketch an architecture specification and design method for reactive distributed real-time embedded systems. In the approach they describe in their paper, they propose separating the concerns of computation, coordination, timing, and distribution, through different simple and abstract notations for these aspects. They also describe a weaving process that maps all these different aspects to a single semantic domain. The method is based on a formal semantics that should ultimately enable automated reasoning about designs. The idea exploited in this method can be extended to other aspects, and extended with more complex weaving criteria.

Lynne Blair proposes multi-paradigm approach to formal specification and shows how this approach can be successfully used in the specification of distributed multimedia systems[26]. He takes an example, a published description of an algorithm to establish the initial synchronization of distributed stored media streams that avoids the need for large buffers (e.g.if the locations of the media sources are widely distributed). He shows how this algorithm can be specified using a combination of real-time temporal logic and timed automata. He then describes how the different specifications (languages) can be combined in order to analyze the overall behaviour[19].

Jochen Hoenicke uses a combination of three techniques for the specification of processes, data and time: CSP, Object-Z and Duration Calculus[20]. The basic building block in our combined formalism CSP-OZ-DC is a class. First, the communication channels of the class are declared. Every channel has a type which restricts the values that it can communicate. There are also local channels that are visible only inside the class and that are used by the CSP, Z, and DC parts for interaction. Second, the CSP part follows; it is given by a system of (recursive) process equations. Third, the Z part is given which itself consists of the state space, the Init schema and communication schemas. For each communication event a corresponding communication schema specifies in which way the state should be changed when the event occurs. Finally, below a horizontal line the DC part is stated. The combination is used to specify parts of a novel case study on radio controlled railway crossings. Johannes Faber formally specifies a part of the European Train Control System (ETCS) with the specification language CSPOZ-DC treating the handling of emergency messages.

Hybrid systems are models for complex physical systems and are defined as dynamical systems with interacting discrete transitions and continuous evolutions along differential equations. With the goal of developing a theoretical and practical foundation for deductive verification of hybrid systems, Andre Platzer introduces a dynamic logic for hybrid programs, which is a program notation for hybrid systems. As a verification technique that is suitable for automation, he introduces a free variable proof calculus with a novel combination of real-valued free variables and Skolemisation for lifting quantifier elimination for real arithmetic to dynamic

logic. The calculus is compositional, i.e., it reduces properties of hybrid programs to properties of their parts. The main result proves that this calculus axiomatises the transition behaviour of hybrid systems completely relative to differential equations. In a case study with cooperating traffic_ agents of the European Train Control System, the case study show that our calculus is well-suited for verifying realistic hybrid systems with parametric system dynamics[21][22][23].

B. Mahony and J.S. Dong propose a timed, multithreaded object modeling notation for specifying real-time, concurrent, and reactive systems. The notation Timed Communicating Object Z (TCOZ) builds on Object-Z's strengths in modeling complex data and algorithms, and on Timed CSP's strengths in modeling process control and real-time interactions. TCOZ is novel in that it includes timing primitives, properly separates process control and data/algorithm issues and supports the modeling of true multi-threaded concurrency. TCOZ is particularly well suited for specifying complex systems whose components have their own thread of control. The expressiveness of the notation is demonstrated by a case study in specifying a multi-lift system that operates in real-time[24][25].

Sandeep Neema et al. present a model-driven approach for generating quality-of-service (QoS) adaptation in Distributed Real-Time Embedded (DRE) Systems. The approach involves the creation of high-level graphical models representing the QoS adaptation policies. The models are constructed using a domain-specific modeling language – the Adaptive Quality Modeling Language (AQML). Multiple generators have been developed using the Model-Integrated Computing (MIC) framework to create low-level artifacts for simulation and implementation of the adaptation policies that are captured in the models. A simulation generator tool synthesizes artifacts for Matlab® Simulink ®/Stateflow® (a popular commercial tool), providing the ability to simulate andanalyze the QoS adaptation policy. An implementation generator creates artifacts for Quality Objects (QuO), a QoS adaptation software infrastructure developed at BBN, for execution of QoS adaptation in DRE systems. A case study in applying this approach to an Unmanned Aerial Vehicle – Video Streaming application is presented. This approach has goals that are similar to those specified in the OMG's Model- Driven Architecture initiative[26].

Although UML is a general purpose modeling language, it contains extensibility mechanisms that can be used to tailor it to specific domains (QoS information specification, for instance). These extensibility mechanisms can be understood as indirect modification, at the model level, of the UML meta-model [8.The standard extensibility mechanisms of UML are *stereotypes, tagged values and constraints*. These extensibility mechanisms are called "lightweight extensibility mechanisms" in contrast to the direct manipulation of the UML "meta-model" that can be interpreted as "heavyweight extensibility mechanisms" (addition of new meta-classes, meta-associations, etc.). In order to give support to the gradual adoption of "standard" UML extensions, OMG has introduced the concept of "UML profile" which, in spite of the lack of a normative definition, has already been used in several OMG technical groups. A "profile" might be defined as a "specification that specializes one or several standard meta-models, called "reference meta-models". OMG defines two UML profiles in order to use this modeling language for the specification of QoS information related to distributed object-based applications and for the modeling of mechanisms for monitoring the specified QoS information. A QoS characteristic represents some aspect of the QoS of a system, service or resource that can be identified and quantified. A QoS statement expresses some QoS by constraining values of QoS characteristics. A QoS relation specifies the mutual obligation of an object and its environment with respect to QoS. These concepts are related to the UML meta-model in order to define a UML profile for QoS[27][28].

V. Case Study One: Aspect-Oriented QoS Modeling of Fire alarm systems

An automatic fire alarm system is designed to detect the unwanted presence of fire by monitoring environmental changes associated with combustion. In general, a fire alarm system is either classified as automatically actuated, manually actuated, or both. Automatic fire alarm systems can be used to notify people to evacuate in the event of a fire or other emergency, to summon emergency services, and to prepare the structure and associated systems to control the spread of fire and smoke.

QoSConstraint Q1,Q2, Q5 of the fire alarm system is expressed as follows with formal technique RTL[9]**:**

[Q1]: ・i・j@(data.collect,j)-
@(・stop,i)・COLLECT_MIN_TIME・@(・data.open, j) -@ (・stop,i)・COLLECT_MAX_TIME
[Q2]: ・i・j@(data.process,j)-
@(・stop,i)・DATA_PROCESS_MIN_TIME・ @
(・data. process, j)-
@(・stop,i)・COLLECT_MAX_TIME
[Q5]: ・i・j@(alarm.process.,j)-
@(・command.send,i)・ALARM_PROCESS_ MINTI ME ・@(alarm.process,j)-
@(・command.send,i)・ALARM_PROCESS_MAXTI ME

QoSConstraint Q3 and Q4 of the fire alarm system is expressed as follows with XML[12]:

```
[Q3]: <QoS type="Level">
        <Firelevel  val = "FIRE_MAX_LEVEL"/>
      </QoS>
[Q4]: <QoS type="Contraint">
        <frame_rate val =
"FRAME_RATE_CONSTRAINT"/>
        <audio_sample_rate val =
"FRAME_RATE_CONSTRAINT"/>
      </QoS>
```

We separate QoS from real-time fire system as an aspect, the aspect-oriented model of QoS of real-time fire system is shown as Fig.2.

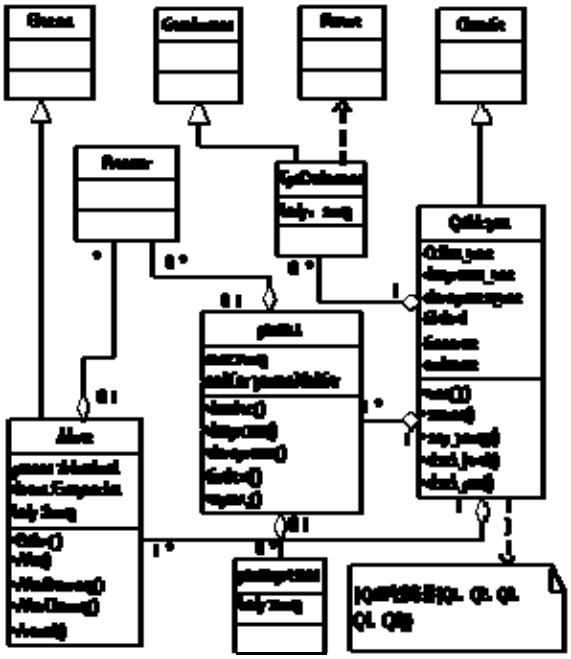

Figure 2. QoS aspect-oriented model of Fire real-time system

We use QoSAspect to express QoS of real-time fire alarm system. The class diagram of Fire Real-time System with the aspect-oriented extension is shown as Fig.3.

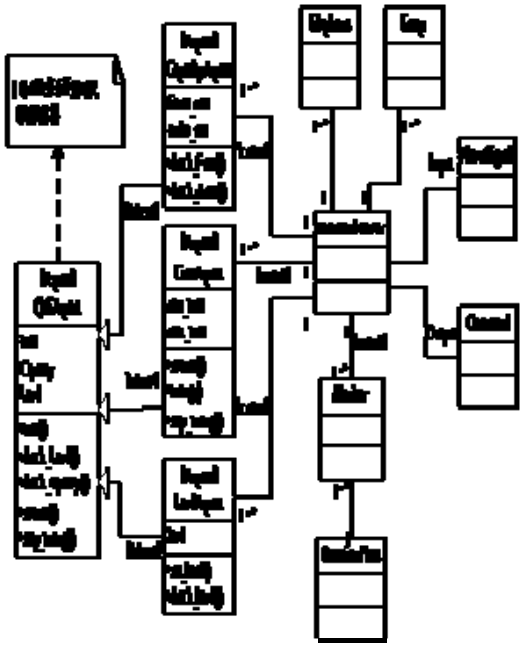

Figure 3. Class Diagram of Fire Real-time System

The QoS Aspect Weaving Diagram of real-time fire alarm system is shown as Fig.4.

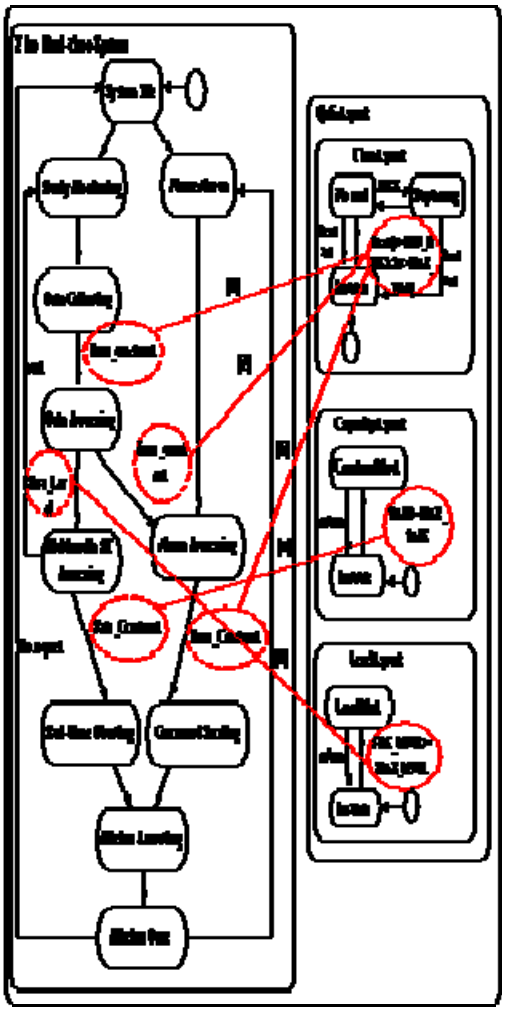

Figure 4. QoS Aspect Weaving Diagram of Fire Real-time System

VI. Case Study Two: Aspect-Oriented QoS Modeling of Elevator System

To satisfy the requirements of AOSD modeling cyber physical systems, UML should be extended by introducing a new stereotype called <<aspect>>. The <<aspect>>is a stereotype for the base class <<class>> that is part of the <<classifier>> element , in order to make sure that <<aspect>> has the same behavior as class. Real-time feature is described as an instance of <<aspect>> and called TimeAspect as shown in Fig.5.

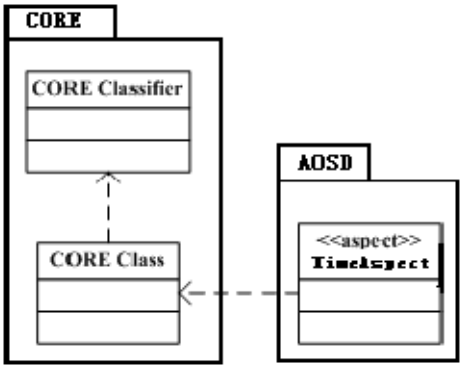

Figure 5. Relationship of class, classifier and <<aspect>>

The time aspect of real-time systems can be designed independently and expressed as a time aspect model. The time aspect model can be defined and designed based on the general time model in Fig.6.

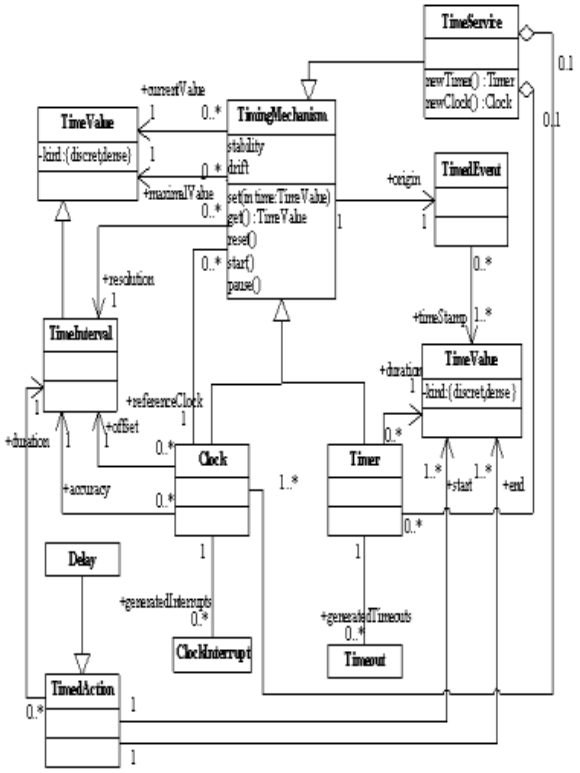

Figure 6. Time Aspect Model

An elevator control system[24][29] illustrates the development process including static structure, dynamic behaviors and the weaving of time aspect by integration of Informal Specification and formal Specification with aspect-oriented approach. We consider that every floor has a pair of direction lamps indicating that the elevator is moving up or down. There is only one floor button and one direction lamp in the top floor and the bottom floor. Every floor has a sensor to monitor whether the elevator is arriving the floor.

We consider that the elevator is required to satisfy the following timing constraints:

[T1] After the elevator has stopped at a particular floor, the elevator's door will open no sooner than OPEN_MIN_TIME and no later than OPEN_MAX_TIME.

[T2] After the elevator has stopped at a given floor the elevator's door will normally stay open for a STAY_OPEN_NORMAL_TIME. However, if the CloseDoorButton on board of the elevator is pressed before this timeout expires, the door will close but no sooner than STAY_OPEN_MIN_TIME.

[T3] After the door is closed, the movement of the elevator can resume, but no sooner than CLOSE_MIN_TIME, and no later than CLOSE_MAX_TIME.

Separation of Concerns From Elevator System

Several concerns can be separated from the elevator control system, such as time aspect, control aspect, and concurrency aspect. The development process of the elevator control system is shown as Fig.7. However, in this paper we only simply consider the time aspect, and will complete other aspects in our future work.

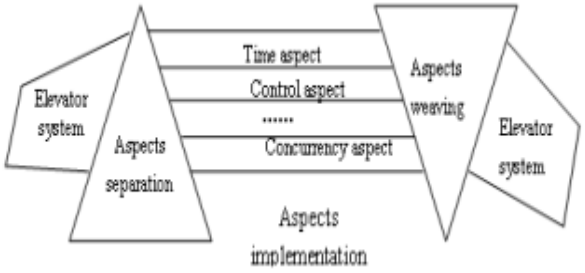

Figure 7. The Development Process of The Elevator Control System

Structural Description Using Class Diagrams

The real-time feature of real-time systems can be modeled using UML by extending stereotypes, tagged values, and constraints before. For example, timing constraints can be added on the class to express the time feature in class diagram. But the implementation of the time feature were still scattered throughout, resulting in tangled code that was hard to develop and maintain. So we describe the real-time feature as an independent aspect according to the AOP techniques, and design a time model to realize and manage the time aspect in order to make the system easier to design and develop and guarantee the time constraints.

We separate the real-time feature as a TimeAspect, which is an instance of <<aspect>> in the elevator control system. The TimeAspect crosscuts the core functional class by stereotype <<crosscut>> in class diagram. Also timing constraints can be attached to the TimeAspect explicitly. The elevator control system class diagram is shown in Fig.8.

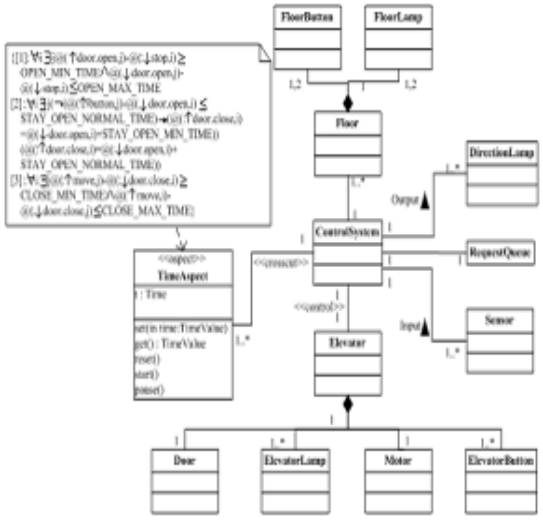

Figure 8. The Elevator Control System Class Diagram

Behavioral Description

UML has five behavioral diagrams to describe the dynamic aspects of a system as representing its changing

parts. Use case diagram organizes the behaviors of the system. Sequence diagram focuses on the time ordering of messages. Collaboration diagram emphasizes on the structural organization of objects that send and receive messages. Statechart diagram focuses on the changing state of a system driven by events. Activity diagram focuses on the flow of control from activity to activity messages. Use case diagram, collaboration diagram, and sequence diagram belong to Inter-Object behavior diagrams. While statechart diagram belong to Intra-Object behavior diagrams. The time behavior is depicted by extending timing marks in the statechart traditionally. can model the intra-object aspectual behaviors well in the object-oriented programming paradigm. However, current specification of statecharts doesn't support aspect-oriented modeling. To support aspect-orientation within the context of statecharts, we need to provide a mechanism by which the modeler can express these aspects. Statecharts modeling aspects should consider the association between aspects and transitions instead of states. Orthogonal regions, which are shown as dashed lines in statecharts, combine multiple simultaneous descriptions of the same object. The aspects can be expressed as objects, which have their own sub-states. Interactions between regions occur typically through shared variables, awareness of state changes in other regions and message passing mechanisms such as broadcasting, and propagating events .

Collaboration diagram emphasizes on the structural organization of objects that send and receive messages. A collaboration diagram shows a set of objects, links among those objects, and messages sent and received by those objects. It shows classifier roles and the association roles. A classifier role is a set of features required by the collaboration. Classifier roles for core classes implement the core features required by the system. Classifier roles for aspects are services required by the core classes which are otherwise tangled with the roles of the core functional features . The time aspect is time service required by the core classes in real-time systems.

The elevator control system expresses the time features as an object of TimeAspect. The behavior of the time object interacting on other objects of the system is shown in Fig. 9.

The statecharts of the elevator control system is shown in Fig.10. Timing behaviors are described by the advanced features of statecharts, and the time concern is achieved implicit weaving with the core functionality of the system. Statecharts refine the model and aspects codes can be generated automatically by existing CASE tools.

Weaving of Time-Aspect

The time aspect can be woven into the real time system by using the UML's statecharts, as statecharts refine the model. Libraries of core and time aspect statecharts can be developed concurrently and independently, and combined only when needed for a particular application.

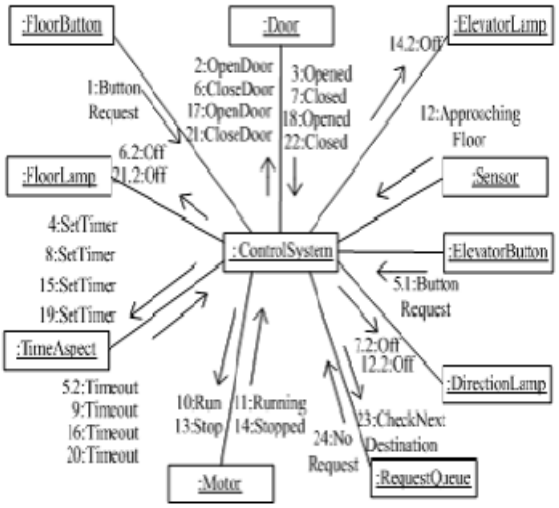

Figure 9. Collaboration Diagram of The Elevator Control System

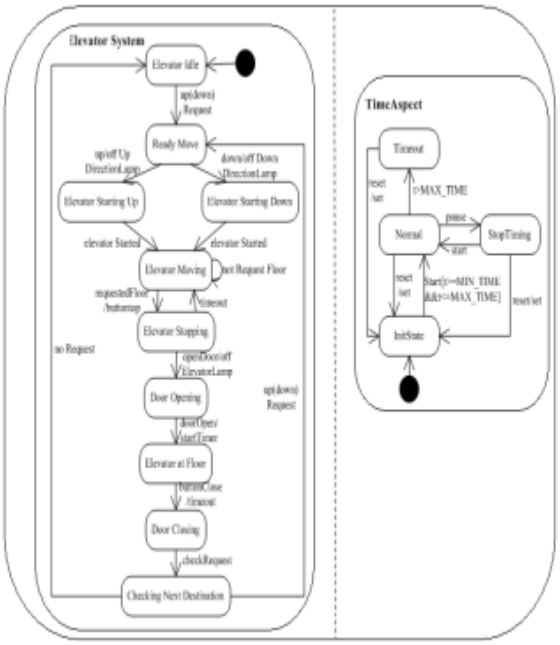

Figure 10. Statechart of the Elevator Control System

In this paper, we weave the time aspect with high-level declarations about how an event in the time statechart can be treated like a completely different event in the core statechart. The weaving framework permits statecharts design to be translated into skeleton code for a class. The time aspect and core statecharts objects may be joined to create orthogonal regions. In addition, the time aspect statechart can be woven by specifying which events shall be reinterpreted to have meaning in other statechart. The declarations of events reinterpretation of the elevator example can be described as follows according to the timing constraints:

[1] If the core statechart is in the 'ElevatorStopping' state and a 'openDoor' event is introduced, and if the time aspect statechart is in the state 'InitState' satisfying 't>=Min_Time&&t<=Max_Time'. Then the time aspect statechart transfers to the 'Normal' state, the

core statechart treats the ‘openDoor’ event exactly and transfers to the ‘DoorOpening’ state.

[2] If the core statechart is in the ‘DoorOpening’ state and a ‘closeDoor’ event is introduced, and if the time aspect statechart is in the state ‘InitState’ satisfying ‘t>=Min_Time’. Then the time aspect statechart transfers to the ‘Normal’ state, the core statechart treats the ‘closeDoor’ event exactly and transfers to the ‘DoorClosing’ state.

[3] If the core statechart is in the ‘DoorOpening’ state and no any event is introduced, and if the time aspect statechart is in the state ‘Normal’ satisfying ‘t>Max_Time’. Then the time aspect statechart transfers to the ‘TimeOut’ state, the core statechart transfers to the ‘DoorClosing’ state.

[4] If the core statechart is in the ‘ReadyMove’ state and a ‘down(up)’ event is introduced, and if the time aspect statechart is in the state ‘InitState’ satisfying ‘t>=Min_Time&&t<=Max_Time’. Then the time aspect statechart transfers to the ‘Normal’ state, the core statechart treats the ‘down(up)’ event exactly and transfers to the ‘Elevator Starting Down(Elevator Starting Up)’ state.

The time aspect and core statecharts will only make the transitions based on the declarations defined above. So it makes sure that the system will implement strictly relying on the timing constraints and guarantee the real time feature. Weaving the time aspect of the elevator system is shown in Fig.10. We extend the reinterpretation function so that an aspect can be woven into other aspects or core classes. In the example above, weaving can be specified using a reference to the core ‘statechart’ object and a reference to the aspect ‘statechart’ object:

AspectID= core.crosscutBy(TimeAspect);

This specifies how an aspect is woven into other aspects or core classes. Every weaving of aspect has unique AspectID. When aspects are weaving, methods will be called to map events in the core and aspect statecharts. The declarations will hold which events need to be reinterpreted. These details will be filled in while a specific aspect is woven. The declarations above are equivalent to the expressions as follows:

[1]. reinterpretEvent (core,”ElevatorStopping”,”openDoor”,”InitState”,”start/t>=OPEN_MIN_TIME&&t<=OPEN_MAX_TIME”,AspectID,Statechart.PREHANDLE);

[2]. reinterpretEvent (core,”DoorOpening”,”button”,”InitState”,”start/t>=STAY_OPEN_MIN_TIME”,AspectID,Statechart.PREHANDLE);

[3]. reinterpretEvent (core,”DoorOpening”,” ”,”Normal”,”start/t>STAY_OPEN_NORMAL_TIME”,AspectID,Statechart.PREHANDLE);

[4]. reinterpretEvent (core,”ReadyMove”,”down(up)”,”InitState”,”start/t>=CLOSE_MIN_TIME&&t<= CLOSE_MAX_TIME”,AspectID,Statechart.PREHANDLE);

The time aspect can be woven into the real time system by using the UML’s statecharts as shown in Fig.11.

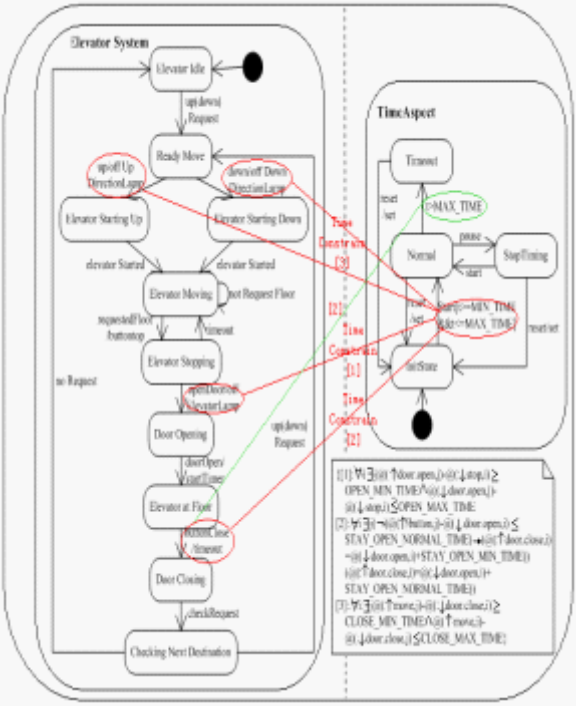

Figure.11. Weaving the Time Aspect

VII. Conclusion

In this paper, we presented an aspect-oriented model for specifying Quality of Service (QoS) based on the combination of UML and RTL. Two types of notation, graphical (semi-formal) and, respectively, formal, can efficiently complement each other and provide the basis for an aspect-oriented specification approach that can be both rigorous and practical for QoS modeling. Two examples depicted how aspect-oriented methods can be used during QoS analysis and design process.

Future works will focus an automatic weaver for aspect oriented model of QoS.

Acknowledgment

This work is supported by the Major Program of National Natural Science Foundation of China under Grant No.90818008, National Natural Science Foundation of China under Grant No. 61173046 and Natural Science Foundation of Guangdong province under Grant No.S2011010004905.

References

[1] Edward A. Lee, Sanjit A. Seshia, Introduction to Embedded Systems, A Cyber-Physical Systems Approach, Published by authors, First Edition, 2011, 978-0-557-70857-4G.

[2] Lui Sha, Sathish Gopalakrishnan, Xue Liu and Qixin Wang: Cyber-Physical Systems: A New Frontier. ISBN 978-0-387-88734-0, Springer，2009

[3] Wolf.W. Cyber-physical Systems. Computer,Volume: 42 Issue: 3,88-89,2009

[4] Edward A.Lee. Cyber Physical Systems Design Challenges. Object Oriented Real-Time Distributed

Computing (ISORC), 2008.11th IEEE International Symposium. January 23,2008.
[5] Aspect-Oriented Software Development. http: //aosd.net/.
[6] Aniruddha Gokhale and Jeff Gray, An Integrated Aspect-oriented Model-driven Development Toolsuite for Distributed Real-time and Embedded Systems, *Workshop on Aspect-Oriented Modeling Workshop*, held at *AOSD 2005*, Chicago, IL, March 2005
[7] Carsten Köllman ; Lea Kutvonen ; Peter Linington ; Arnor Solberg, An aspect-oriented approach to manage Qos dependability dimensions in model driven development, International Workshop on Model-Driven Enterprise Information Systems, p85-94, 2007
[8] Aldawud, T. Elrad, and A. Bader. A UML Profile for Aspect Oriented Modeling, Workshop on AOP, 2001.
[9] Farnam Jahanian, Aloysius K. Mok.Safety Analysis of Timing Properties in Real-Time Systems. IEEE Trans. Software Eng. 12(9): 890-904 (1986)
[10] R. de Lemos and J. Hall. ERTL: An extension to RTL for requirements analysis for hybrid systems. Technical report, Department of Computing Science, University of Newcastle upon Tyne, UK., 1995.
[11] UML2.0 OCL Specification. http: //www.comp.nus.edu.sg/~yangfei/HYP/UML2.0ocl.pdf
[12] Jeff Gray, Ted Bapty, and Sandeep Neema, Aspectifying Constraints in Model-Integrated Computing, *Workshop on Advanced Separation of Concerns in Object-Oriented Systems*, held at *OOPSLA 2000*, Minneapolis, MN, October 2000
[13] Jaiganesh Balasubramanian et al.,A Model-driven QoS Provisioning Engine for Cyber Physical Systems, http: //citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.151.1140&rep=rep1&type=pdf)
[14] Jeff Gray, Sandeep Neema, Jing Zhang, Yuehua Lin, Ted Bapty, Aniruddha Gokhale, and Douglas C. Schmidt, "Concern Separation for Adaptive QoS Modeling in Distributed Real-Time Embedded Systems," Behavioral Modeling for Embedded Systems and Technologies: Applications for Design and Implementation, edited by Joa M. Fernandes and Luis Gomes, 2009.
[15] Carsten Köllman ; Lea Kutvonen ; Peter Linington ; Arnor Solberg, An aspect-oriented approach to manage Qos dependability dimensions in model driven development, International Workshop on Model-Driven Enterprise Information Systems, p85-94, 2007
[16] QoSUniFrame.http: //www.cis.uab.edu/liush/QosUniFrame.htm.
[17] Bikash Sabata et al., Taxonomy for QoS Specifications, the proceedings of WORDS '97, February 5-7, 1997 in Newport Beach, California
[18] M.R. Mousavi, G. Russello, M. Chaudron, M. Reniers, T. Basten, A. Corsaro, S. Shukla, R. Gupta, D. Schmidt, Using Aspect-GAMMA in the Design of Embedded Systems, *Proceedings of the Seventh IEEE International Workshop on High Level Design, Verification and Test (HLDVT'02)*, Cannes, France, pp. 69--75, IEEE CS, October 2002
[19] Blair L., *The Role of Temporal Logic and Time Automata in Distributed Multimedia Systems*, Proceedings of Modal & Temporal Logic Based Planning for Open Networked Multimedia Systems (PONMS '99), Cape Cod, MA, November 5-7, pp 1-7, 1999
[20] Jochen Hoenicke.Specification of Radio Based Railway Crossings with the Combination of CSP, OZ, and DC. http: //citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.21.4394.
[21] André Platzer. Differential dynamic logic for hybrid systems. Journal of Automated Reasoning, 41(2), pp 143-189, 2008.
[22] André Platzer. Differential dynamic logic for verifying parametric hybrid systems. LNCS 4548, pp 216-232. Springer, 2007.
[23] André Platzer,*Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, 2010. 426 p. ISBN 978-3-642-14508-7.
[24] B. Mahony and J.S. Dong. Timed Communicating Object Z**.** IEEE Transactions on Software Engineering, 26(2): 150-177, Feb 2000
[25] B. P. Mahony and J.S. Dong. Blending Object-Z and Timed CSP: An introduction to TCOZ. ICSE'98, April 1998.
[26] Sandeep Neema,Proceedings of the 1st ACM SIGPLAN/SIGSOFT conference on Generative Programming and Component Engineering, 2002)
[27] OMG, "UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms," Request forProposal, ad/02-01-07, January, 2002.
[28] OMG, "UML Profile for Schedulability, Performance, and Time Specification," Final Adopted Specification, ptc/02-03-02, March, 2002.
[29] Carsten Suhl, An Integration of Z and Timed CSP for Specifying Real-Time Embedded Systems, Ph.D thesis, Technischen Universitat Berlin, 2002
[30] Wehrmeister, M.A., Freitas, E.P., and Pereira, C.E., et al., "An Aspect-Oriented Approach for Dealing with Non-Functional Requirements in a Model-Driven Development of Distributed Embedded Real-Time Systems ", 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, Santorini Island, Greece, May7-9, 2007, IEEE Computer Society, pp.428-432.