Modeling Structural, Temporal and Behavioral Features of a Real-Time Database

Nada Louati[1], Rafik Bouaziz[1], Claude Duvallet[2] and Bruno Sadeg[2]
[1]*MIRACL-ISIMS, Sfax University, BP 1088, 3018, Sfax, Tunisia*
[2]*LITIS, UFR des Sciences et Techniques, BP 540, 76 058, Le Havre Cedex, France*

Keywords: Real-time, Database, MARTE, UML-RTDB, Profile.

Abstract: Real-time databases are different from traditional databases in that they have timing constraints on data and on transactions upon the data. The design of this kind of databases must consider both temporal aspects of data and timing constraint of transactions in addition to the logical constraints of the database. This paper proposes a new UML profile that extends UML with concepts related to real-time databases design. These extensions aim to accomplish the conceptual modeling of a real-time database according to three aspects: structural, temporal and behavioral. Our propositions is based on MARTE (Modeling and Analysis of Real-Time and Embedded systems) profile which provides capabilities of modeling concepts to deal with real-time and embedded system features. The proposed profile is illustrated by a case study in the context of Ait Traffic Control System.

1 INTRODUCTION

Real-Time DataBases (RTDBs) are typically used to manage environmental data in computer control applications, such as air traffic control, automated manufacturing, and military command and control (Ramamritham, 1993). The design of such RTDBs differs from both that of real-time systems and that of conventional database management systems. The designers of RTDBs much consider both temporal aspects of data and timing constraints of transactions in addition to logical constraints of the database

The design of RTDBs is performance-and semantic-dependent. It must consider factors such as sensor data, derived data and quality of data management, temporal semantics in transactions scheduling algorithms and concurrency control protocols, to meet the timing constraints defined by the real-time applications (Idoudi et al., 2010). This paper proposes a new UML profile that incorporates these concepts. It is based on a subset of concepts inspired from the HLAM, NFP, and TIME packages of MARTE. The motivations behind these extensions are three-folds: (i) to have new notations distinguishing quantitative and qualitative features of RTDBs, (ii) to facilitate the modeling of timing aspects of data and transactions, (iii) to accomplish the conceptual modeling of RTDBs. This profile not only captures the structural aspects of a RTDB features, but also the temporal and behavioral aspects.

The remainder of this paper is structured as follows: in Section 2, we present the related works. In Section 3, 4, and 5 we respectively address the modeling of structural, temporal, and behavioral aspects of RTDBs. In Section 6, we present a case study to more illustrate our proposals. In Section 7 we conclude the paper and we give some perspectives of our work.

2 RELATED WORK

In recent years, several UML profiles have been proposed for modeling real-time systems to depict its real-time constraints. Only a few of these UML profiles address RTDBs modeling.

In (DiPippo and Ma, 2000), DiPippo and Ma describe an UML package, called RT-Object, for specifying RTSORAC object. This UML package contains real-time attributes, real-time methods, compatibility functions and constraints. The RT-Object package is based on a past standard of UML which is UML 1.3 Extension Mechanisms package. Furthermore, imprecise computation encapsulated within the RTSORAC object model is described in the context of Epsilon Serializability (Ramamritham and Pu, 1995),

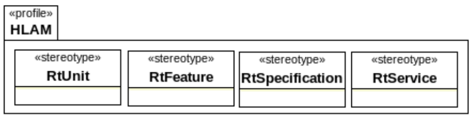

Figure 1: HLAM package specification in MARTE.

and does not support the notion of quality of data introduced in (Amirijoo et al., 2006).

Another work proposed the UML-RTDB profile (Idoudi et al., 2008a) (Idoudi et al., 2008b) to express RTDB features in a structural model. It supplies concepts for RTDBs modeling such as real-time attributes, real-time methods and real-time classes. However, the proposed extensions are based on the Evolutionary Stereotype concept (Debnath et al., 2003) which is not a standard way of extending UML.

MARTE (OMG, 2009) is an industry standard of the OMG for model-driven development of embedded systems (OMG, 2009). It aims at providing support for specification, design, validation, and verification stages in real-time and embedded system development. The richness of MARTE profile in terms of concepts offers an interesting common modeling basis to adequately specify many design features of RTDBs. However, the notions of real-time data and real-time transactions, which are the basis features of RTDBs are missing in MARTE.

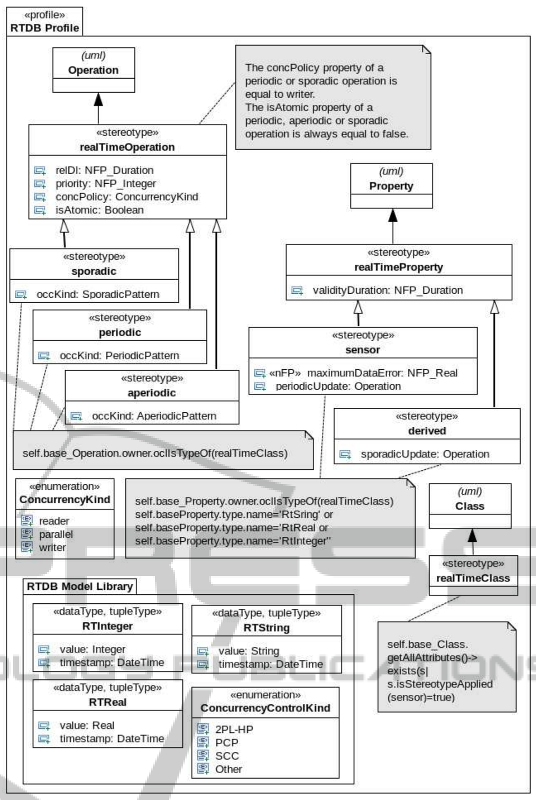

Figure 2: RTDB Profile Metamodel.

3 MODELING OF STRUCTURAL ASPECTS

3.1 High Level Application Modeling in MARTE

The HLAM package of MARTE proposes concepts that enable to describe both quantitative and qualitative features of real-time applications at a high abstraction level. Figure 1 depicts the basic stereotypes associated with the HLAM package. An *RtUnit* may be seen as an autonomous execution resource that owns one or more schedulable resources to handle incoming messages. *RtUnits* may provide real-time services. The *RtService* stereotype has been introduced for that purpose. The *RtFeature* stereotype is used to annotate model elements with real-time features according to set of *RtSpecification* associated with this stereotype. It can be attached to multiple kinds of modeling elements (i.e., behavioral features, actions, messages, signals, and ports). For a full description of all these stereotypes, the reader may refer to the specification document of MARTE (OMG, 2009).

3.2 Modeling of RTDB Structural Features

We propose new stereotypes expressing RTDB features in a structural model on the one hand, and according to the MARTE profile on the other hand. Figure 2 shows the extensions proposed to some metaclasses belonging to the class diagram. In order to take advantages of some MARTE concepts, the proposed profile imports stereotypes from HLAM, NFP, and VSL sub-profiles.

3.2.1 Stereotypes for Real-Time Data

Real-time attributes are divided into two types: *sensor attributes* and *derived attributes* (Ramamritham et al., 2004). Sensor attributes are used to store sensor data which are issued from sensors. Derived attributes are used to store derived data which are calculated from sensor data. Thus, we define two stereotypes, *sensor* and *derived*, in order to declare respectively sensor and derived attributes in the class diagram. As illustrated in Figure 2, we define an abstract stereotype, called *realTimeProperty*, that factorizes *validity duration* property which indicates the amount of time during which the attribute value is considered valid. We use the *MARTE NFP_Duration* datatype as a type for the *validity duration* feature.

We characterize each *sensor* stereotype by two properties : *maximum data error* and *periodicUpdate*. *Maximum data error* indicates the maximum deviation tolerated between the current attribute value and the updated value. It is of type *NFP_Real*. *Maximum data error* represents a non-functional property specifying the upper bound of the error. We propose to associate the *Nfp* stereotype of MARTE profile to the *maximum data error* attribute. *PeriodicUpdate* refers to a periodic operation of the class that owns the sensor attribute. The role of this periodic operation is to update the current *value* and the *timeStamp* fields of the *sensor* attribute. It is of type *Operation*.

As shown in Figure 2, we characterize *derived* stereotype by *sporadicUpdate* property, that refers to a sporadic operation of the class that owns the derived attribute. This sporadic operation is used to update the *current value* and the *timeStamp* fields of the *derived* attribute.

Each real-time attribute value is characterized by a *timestamp*, which indicates the time at which it was last updated. So, for each real-time attribute value corresponds a *timestamp*, which distinguishes it from other attribute's values. Whereas, the values of the *validity duration* and *maximum data error* fields are the same for the same real-time attribute.

3.2.2 Stereotypes for Real-Time Transactions

A real-time transactions may be aperiodic, periodic, or sporadic (Ramamritham et al., 2004). It has timing constraints such as deadline and period. We define three stereotypes, aperiodic, periodic, and sporadic (cf. Figure 2) in order to declare respectively aperiodic, periodic, and sporadic operations in the class diagram. Each of these stereotypes is characterized by a *deadline*, which indicates the last time by which the method execution must be completed. Thereby, we define an abstract stereotype, called *realTimeOperation*, which is used to annotate model elements (i.e. Operations) with real-time features according to set of RtSpecification associated with this stereotype. RtSpecification possesses nine tagged values among which: relDl, occKind, and priority. The relDl attribute specifies the deadline of a method execution. The occKind attribute indicates the arrival transaction specification (i.e. periodic, aperiodic, or sporadic). For a periodic (respectively aperiodic and sporadic) transaction, the PeriodicPattern (respectively AperiodicPattern and SporadicPattern) enumeration literal is selected for ArrivalPattern attribute of the occKind property. The priority attribute specifies the priority

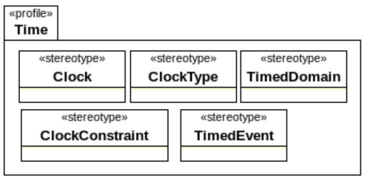

Figure 3: Time Specification in MARTE.

order of a transaction. The *realTimeOperation* stereotype factorizes also two properties: concPolicy and isAtomic. When the value of *isAtomic* is *true*, the method execution is done as one individual unit. This fact coincides with the *atomicity* property of a transaction. However, in RTDBs, this feature is relaxed in order to allow the validation of a transaction even if only a part of its actions have been executed (Agrawal et al., 1994). So, in our work, we consider that the value of the *isAtomic* property is always *false*. The *concPolicy* property specifies the concurrency policy of a transaction. The values of this property may be: *reader*, *writer* or *parallel*. A *writer* transaction implies that multiple calls from concurrent transactions may occur simultaneously and will be treated as soon as concurrency on data allows its execution. A *reader* transaction implies that multiple calls from concurrent transactions may occur simultaneously and will be executed simultaneously if there is no writer transaction using one or more data that the *reader* transaction needs. A *parallel* transaction is a transaction whose actions do not use any data of the database in reading mode nor in writing mode. In our work, we consider that for an update transaction, which can periodic or sporadic, the *concPolicy* property is *writer*.

3.2.3 Real-Time Class Stereotype

The design of a RTDB, which is by definition a database system, has to take into account the management of many components such as queries, schemas, transactions, commit protocols, concurrency control protocol, and storage management (Stankovic et al., 1999). In order to deal with time-constrained data, time-constrained operations, parallelism, and concurrency property inherent to RTDBs, we introduced the *RealTimeClass* stereotype. This stereotype specifies that instances of a class will encapsulate real-time data and real-time operations and a local concurrency mechanism. Because of the dynamic nature of the real world, more than one transaction may send requests to the same object. Concurrent execution of these transactions allows several methods to run concurrently within the same object. To handle this essential property of RTDB systems, we associate to each object a local concurrency control mechanism, named *local controller*, that manages the concurrent execution of its methods. Thus, the object receives messages in its mailbox awaking its local controller that checks the timing constraint attached to messages and selects one message following a special scheduling algorithm. The local controller verifies the concurrency constraints with the already running methods of the object. Then, it allocates a new thread to handle the message when possible. When a method terminates its execution, the corresponding thread is released and concurrency constraints are relaxed. In our work, the scheduling algorithm adopted by the mailbox is EDF (EarliestDeadlineFirst). For that purpose, we import from MARTE library the *SchedPolicyKind* enumeration which defines the most common kind of scheduling policy. Add to that, the concurrency control protocol adopted by the local controller is 2PL-HP (Two Phase Locking-High Priority). In 2PL-HP, all data conflicts are immediately resolved by aborting lower priority transactions. Thereby, we enhance our profile model library by a new enumeration type, called *ConcurrencyControlKind*, in order to define concurrency control policies (i.e. 2PL-HP, PCP (Priority Ceiling Protocol), SCC (Speculative Concurrency Control), etc.). Our *RealTimeClass* stereotype overlaps with the UML extensions presented by MARTE profile especially those relative to the mailbox and the local controller. In fact, a *RealTimeClass* may be considered as an autonomous execution resource, able to handle different messages at the same time. It can manage concurrency and real-time constraints attached to incoming messages. This has the same meaning as the *RTUnit* stereotype defined in MARTE (cf. Figure 1). Thus, we consider that the *RealTimeClass* concept as a specialization of MARTE *RtUnit* concept. Thereby, a *RealTimeClass* is a *real-time unit* but with a RTDB modeling semantics. It represents the RTDB entity which encapsulates time-constrained data and time-constrained operations. It also deals with parallelism and concurrency features.

4 MODELING OF TEMPORAL ASPECTS

4.1 Time Specification in MARTE

The sub-profile TIME of MARTE has been introduced to model timing aspects. Figure 3 depicts the MARTE extension about time modeling in UML. The *Clock* stereotype is a model element that represents an instance of *ClockType*. The *ClockType* stereotype is related indirectly to the *Clock* stereotype. Its properties specify the kind of clock (chronometric or log-

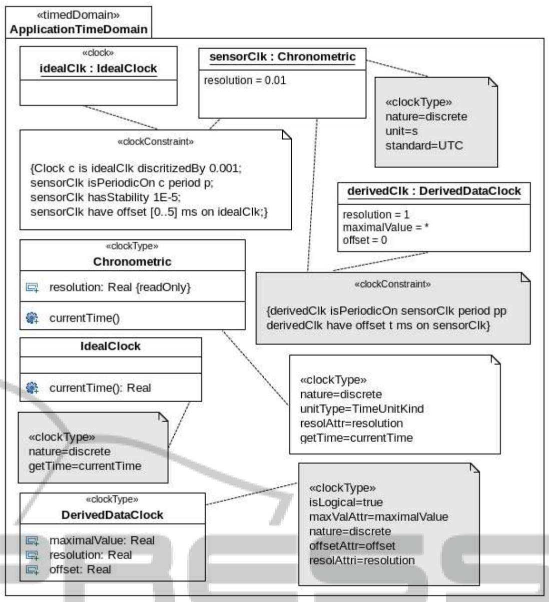

Figure 4: Clocks Specification in our RTDB Profile.

ical), the nature (dense or discrete) of the represented time, a set of clock properties (e.g., resolution, maximal value, etc.) and a set of accepted time units. The *ClockConstraint* is a stereotype of the UML *Constraint* concept. The clock constraints are used to specify the time structure relations of a time domain. A *ClockConstraint* is a constraint that imposes dependency between clocks or between clock types.

4.2 Modeling of RTDB Temporal Properties

Time is present in different RTDB features and more specially in real-time data. Real-time data are divided into two classes: sensor data and derived data. As previously mentioned in Section 3.2.1, we have defined for each sensor data a periodic method (i.e. *periodicUpdate*) which is periodically executed to update the values of the corresponding sensor data. In the same way, we have defined for each derived data a sporadic method (i.e. *sporadicUpdate*) which is sporadically executed to update the values of the corresponding derived data. In this section, we define for the sensor and derived data clocks as well as their associated properties in a high level specification using the TIME sub-profile of MARTE. The temporal unit that we are looking for can be used in two ways: (i) to reference the physical time and adopt a chronometric clock for sensor data, (ii) to reference a logical time that is incremented each time sensor data have been updated.

The MARTE *TimeLibrary* provides a model for the *ideal time* used in physical laws: *idealClk*, which is an instance of the class *IdealClock*, stereotyped by *ClockType*. The *IdealClock* represents the time evolution. This time is expressed in seconds in the international system units. Starting with *idealClk*, we define new discrete chronometric clock (cf. Figure 4). First, we specifies *Chronometric* (a class stereotyped by *ClockType*) which is discrete, not logical (therefore chronometric), and with a read only attribute (resolution). Clocks belong to timed domains. In Figure 4, a single time domain is considered. It owns two clocks: *idealClk* and *sensorClk*, an instance of *Chronometric* that both uses the second (s) as a time unit; and whose resolution is 0.01 s. The two clocks are a priori independent. A clock constraint specifies relationships among them. The first statement of the constraint defines a clock c local to the constraint. C is a discrete time clock derived from i*dealClk* by a discretization relation. The resolution of this clock is 1 ms. The next statement specify that *sensorClk* is subclock of c with a rate *p* times slower than c. The fourth statement indicate that *sensorClk* is not a perfect clock. Flaws are characterized by non functional properties like stability and offset. Its rate may have small variations (a stability of 10^{-5} implicitly measured on *idealClk*). The last statement claims that the clocks is out of phase, with an offset value between 0 and 5 ms measured on *idealClk*.

Figure 4 depicts the definition of a logical clock dedicated for derived data. We have created a class *DerivedDataClock* with *ClockType* as stereotype.This class has three attributes: *maximalValue*, *offset* and *resolution*. *MaximalValue* specifies the maximal value of the associated clock, value at which the clock rolls over. The *offSet* property determines the initial instant of the associated clock. The *resolution* attribute defines the resolution of the associated clock. Now that we have defined the clock, we need to instantiate it. Figure 4 depicts the instantiation of the clock *DerivedDataClock* called *derivedClk*. The unit of *DerivedClk* is the sensor data update.

Sensor data are periodically updated causing the update of each derived data that uses those sensor data. This is equivalent to say that the derived data update operation is activated every tick of the clock associated with sensor data. We can deduce an affine relation between *sensorClk* and *derivedClk* as specified in the clock constraint specified in Figure 4: *derivedClk isPeriodicOn sensorClk*, period = pp, offset=t. Such a constraint states that each pp^{th} occurrence of *sensorClk* there will be an occurrence of *derivedClk* and the first occurrence of *derivedClk* appears at the t^{th} instant of *sensorClk*.

5 MODELING OF BEHAVIORAL ASPECTS

5.1 Behavior Specification in MARTE

State machines are often used in the real-time and embedded domain. They allow the description of a system behavior in terms of states and transitions between these states. MARTE introduces mainly two time-related concepts that can be used to improve the usage of the UML behaviors, such as state machines, for developing real-time applications: timed processing and timed events (cf. Figure 3). The *TimedProcessing* stereotype enables modelers to specify duration for a behavior. The *TimedProcessing* stereotype represents activities that have known start and finish times or a known duration, and whose instants and durations are explicitly bound to clocks. It can also reference events triggered when a behavior or processing starts and ends. The duration can be specified using the VSL language, which supports time expressions.

The *TimedEvent* stereotype represents events whose occurrences are explicitly bound to a Clock. It extends the *TimeEvent* concept of UML. It allows to characterize the logical or physical clock on which a time event relates.

5.2 Modeling of RTDB Behavior

A RTDB is a collection of real-time objects which are used to manage time-critical dynamic systems in the real world. RTDB behavior model building consist of describing the behavior with state-transition diagram for each real-time object of the RTDB. Stereotypes *TimedProcessing* and *TimedEvent* are used to specify behavior, duration of a behavior, and events bounded to a clock. The tagged value *queueSchedPolicy*, defined in MARTE, are associated to the state-transition diagram in order to indicate the queue scheduling policy of a behavior. We apply the *TimedProcessing* stereotype on the state machine itself and we specify the corresponding clock using the meta-attribute *on*. We employ the *TimedEvent* stereotype in order to characterize the logical and physical clocks on which a time event relates.

6 CASE STUDY

Throughout, this section, we will use a running example to illustrate the use of our profile. We illustrate our proposal on an air traffic control system. The aim of the air traffic control is to separate aircrafts,

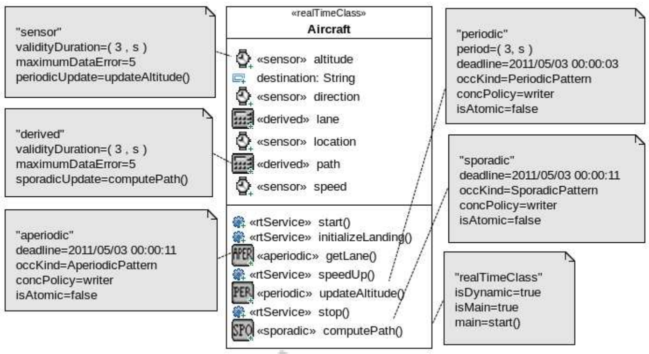

Figure 5: Aircaft real-time class.

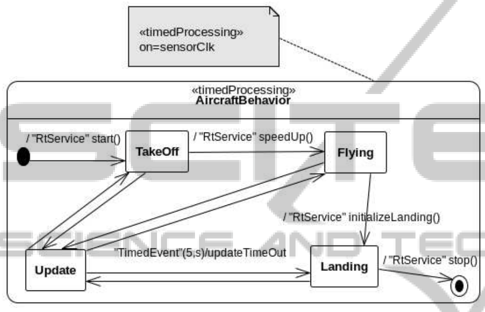

Figure 6: Aircraft behavior modeling.

to avoid collisions and to organize the flow of traffic. It consists of a large collection of data describing the aircrafts, their flight plans, and data that reflect the current state of the controlled environment (Locke, 2001). This includes flight information, such as aircraft identification, speed, altitude, origin, destination, route and clearances. In fact, each aircraft has three sensor data which are speed, altitude and location and two derived data which are path and lane. Sensor data are periodically updated to reflect the state of an aircraft. The derived data is calculated based on altitude, location, and direction values, in order to verify if the aircraft deviates from a predetermined path or lane. All these data values are published periodically by sensors supervising the aircrafts controlled elements.

For sake of clarity, only the *Aircraft* class and a subset of its methods are specified. Each *Aircraft* in the airspace is stereotyped by *realTimeClass*. We characterize the *Aircraft* class by three sensor attributes: direction, location, and altitude. Each attribute is periodically updated in order to closely reflect the real world state of the application environment. Thereby, we associate to each sensor attribute a periodic method: updateAltitude(), which periodically updates the value and the timestamp of the altitude. We characterize also the Aircraft class by two derived attributes: lane which is calculated from location and altitude values, and path, which is calculated from location and direction values. Each derived attribute has its own sporadic method: computePath(), which sporadically updates the value and the timestamp of the path. Figure 5 depicts a simplified view of the Aircraft class. We indicates that the Aircraft class is the main unit of the application (property *isMain* is set to true) and it starts a main *rtService*, called *start()*. Its *isDynamic* property is set to true. In this case, the schedulable resources are created dynamically when required. The attribute *altitude* has a validity duration equal to (*3*, *s*) and its maximumDataError is 5. Additionally, it is updated periodically using the *updateAltitude()* operation, which has a period equal to (*3*, *s*). The *concPolicy* attribute of that operation is *writer* and its *isAtomic* attribute is set to *false*. In the same way, we characterize the derived attribute path by a *validityDuration* and a *maximumDataError* and it is sporadically updated by means of the *computePath()* operation.

Figure 6 depicts a state-machine diagram that provides a simplified view of the *Aircraft* behavior. The Aircraft has four states: TakeOff, Flying, Update, and Landing. Periodically , it enters in the Update state for updating the Aircraft sensors values. The sensors values update has to be done with a period of 5 s and lasts 2 s. Then it returns to the state which activated the update transition. We apply the TimedProcessing stereotype on the state machine itself. We use the former to indicate the scheduling policy associated with the Aircraft behavior (i.e EDF). The latter is used to assign the sensorClk clock to the model. All the elements of the state machine have the same time reference. We apply the TimedEvent stereotype on the UML timed event that triggers the UpdateTimeOut transition.

7 CONCLUSIONS

The richness of MARTE profile in terms of concepts offers an interesting common modeling basis to adequately specify many design features of RT-DBs. In This paper, we showed how the concepts of the MARTE standard profile can serve to model RTDB features. We used the HLAM package to describe both quantitative and qualitative properties, and the TIME sub-profile to specify temporal properties. Moreover, we proposed UML/MARTE-based extensions for a complete and powerful RTDB modeling. Additionally, the proposed extensions not only capture the structural aspects of RTDB features, but also the temporal and behavioral aspects. We have invested some effort on ensuring that all concepts have a well defined basis semantics. This has been illustrated on a case study in the context of Air Traffic Control System.

We are currently working on the the integration of the RTDB development process in the context of a Model Driven Architecture. This way, the complex task of designing the whole RTDB is tackled in a systematic, well-structured and standard manner.

REFERENCES

Agrawal, D., Bruno, J. L., Abbadi, A. E., Krishnaswamy, V., El, A., and Krishnaswamy, A. V. (1994). Relative serializability: An approach for relaxing the atomicity of transactions. In *In Proceedings of the 13th ACM Symposium on Principles of Database Systems*, pages 139–149. ACM.

Amirijoo, M., Hansson, J., and Son, S. H. (2006). Specification and management of qos in real-time databases supporting imprecise computations. *IEEE Trans. Computers*, 55(3):304–319.

Debnath, N., Riesco, D., Montejano, G., Grumelli, A., Maccio, A., and Martellotto, P. (2003). Definition of a new kind of uml stereotype based on omg metamodel. In *Computer Systems and Applications, 2003. Book of Abstracts. ACS/IEEE International Conference on*, pages 49–54.

DiPippo, L. C. and Ma, L. (2000). A uml package for specifying real-time objects. *Comput. Stand. Interfaces*, 22:307–321.

Idoudi, N., Duvallet, C., Sadeg, B., Bouaziz, R., and Gargouri, F. (2008a). Structural model of real-time databases. In *ICEIS (3-2)*, pages 319–324.

Idoudi, N., Duvallet, C., Sadeg, B., Bouaziz, R., and Gargouri, F. (2008b). Structural model of real-time databases: An illustration. In *ISORC*, pages 58–65.

Idoudi, N., Louati, N., Duvallet, C., Sadeg, B., Bouaziz, R., and Gargouri, F. (2010). A framework to model real-time databases. *International Journal of Computing and Information Sciences (IJCIS)*, 7(1):1–11.

Locke, C. D. (2001). Applications and system characteristics. In *Real-Time Database Systems*, pages 17–26.

OMG (2009). A UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded systems, version 1.0, formal/2009-11-02.

Ramamritham, K. (1993). Real-time databases. *Distributed and Parallel Databases*, 1(2):199–226.

Ramamritham, K. and Pu, C. (1995). A formal characterization of epsilon serializability. *IEEE Trans. on Knowl. and Data Eng.*, 7:997–1007.

Ramamritham, K., Son, S. H., and DiPippo, L. C. (2004). Real-time databases and data services. *Real-Time Systems*, 28(2-3):179–215.

Stankovic, J. A., Son, S. H., and Hansson, J. (1999). Misconceptions about real-time databases. *IEEE Computer*, 32(6):29–36.