RT-ETM: Toward Analysis and Formalizing Transaction and Data Models in Real-time Databases

Mourad Kaddes[1,2], Laurent Amanton[2], Bruno Sadeg[2], Mouez Ali[1], Majed Abdouli[1], Rafik bouaziz[1]
[1]Multimedia, Information Systems and Advanced
Computing Laboratory, Sfax university,
Tunis km 10, Sfax,Tunisia
{majed.abdouli, mouez.ali}@gmail.com

[2]UFR Sciences et Techniques
Université du Havre, 25 rue
Philippe Lebon BP 540,76058 Le Havre Cedex
FRANCE
{Mourad.kaddes@gmail.com, Laurent.Amanton@univ-lehavre.fr}

ABSTRACT: *Due to the diversity of extended transaction models, their relative complexity and their lack of formalization, the characterization and comparison of these models become delicate. Moreover, these models capture only one subset of interaction which can be found in the spectrum of the possible interaction. In front of this established fact, the framework ACTA was introduced. Our contribution in this field is twofold: (i) we extend ACTA by adding many dependencies for capturing interaction between transactions in real time environment on the one hand. We extend ACTA to take into account temporal characteristics of real-time data item, On the other hand, and (ii) we propose a meta-model, called RT-ETM, that capture concept of an extended real-time transaction model by using UML class diagram and its formal description using Z language.*

Keywords: Real-Time Transaction, ACTA, Dependency, Temporal Data, Data Freshness, Meta-model

Received: 13 February 2011, Revised 19 February 2011, Accepted 24 February 2011

1. Introduction

In a number of real-time applications, e.g., stock trading and traffic control, real-time databases (RTDBS) are required to process transactions in a timely fashion using a large number of temporal data, e.g., current stock prices or traffic sensor data, representing the real world status. A lot of real-time database research has been done to process transactions in a timely fashion using fresh data reflecting the current real world status. In other words, we must satisfy simultaneously freshness of data and make it available to the controlling system for its decision-making activities. Real time database systems are best suited for manipulating such applications since they also handle both large amounts of data and time constraints. Hence, many transactions models are proposed. However, there has been a few work to formalize the properties of transactions and data in real-time databases (RTDB). In this paper, we attempt to address this lacuna, based on the ACTA framework [1].

ACTA is a tool used to specify and reason out the effects of transactions on objects and the interactions between transactions [2]. Specifically, it can be used to specify the properties of atomic and extended transaction models such as nested transactions, split/join transactions and saga transactions, and to synthesize new transaction models [2]. However, it has not been used to specify time related requirements which are essential to the specification of realtime databases.

ACTA is a tool used to specify and reason out the effects of transactions on objects and the interactions between transactions [2]. Specifically, it can be used to specify the properties of atomic and extended transaction models such as nested transactions, split/join transactions and saga transactions, and to synthesize new transaction models [2]. However, it has not been used to specify time related requirements which are essential to the specification of realtime databases.

In this paper, we develop a transaction framework based on ACTA to facilitate the formal description of transaction properties in real-time databases. We present an overview of ACTA and our motivation in Section 2. Section 3 extends ACTA formalism to take into account temporal characteristics. Section 4 proposes a meta-model, called RT-ETM (Real-Time Extended Transaction Models) that captures concept of an extended real-time transaction models by using UML class diagram. We conclude our work in section 5.

2. Overview of ACTA

ACTA is not a new model of transactions, but rather a formalism allowing the formal description of properties of the complex transactions. Precisely, using ACTA, we can specify and reason about the effect of transaction on objects and the interactions between the transactions in a particular model.

ACTA describes the behavior of the transactions (active components) and their interactions which occur according to a correction criterion to generate acceptable results while acting on the objects (components passive). In other words, in ACTA, the interactions between transactions are represented by the effects which they cause. They are formulated by:

• The effects of the transactions on each other,

• The effects of the transactions on the objects which they handle. (cf. Fig 1).

This section provides an overview of the main concepts of ACTA which are extended in [3], [4], [5], [6]. ACTA is composed of five blocks: in history, dependencies between transactions, the view of transaction, the conflict set of transaction and delegation.

2.1 Effects of the Transactions on each othe*r*

The dependencies provide a practical manner to simplify and to reason on the behaviour of the concurrent transactions. In fact, the dependencies describe the effects of the transactions on other transactions, and represent constraints on possible stories. By examining the possible effects of the transactions acting ones on the ‘others, it is possible to determine the dependencies which can be developed between them.

The first two and important dependencies presented in [2] are Commit dependency and Abort dependency defined in the following way, where ti and tj are transactions:

Commit Dependency (t_i CD t_j). If t_i and $t_j$are committed, then the commit of t_j must precede the commit of ti.

Abort Dependency (t_i AD t_j). If t_j aborts,t_i has to abort too.

Many others dependencies are added to extend ACTA to expressive more behaviour [3], [4], [5], [6], [7].

2.2 Effects of Transaction on Objects

A transaction effects on objects are characterized by the set of effects which are visible to it, the whole conflict operations that it carries out, and the whole effects that it delegates to other transactions. ACTA allows to capture these effects by introducing three sets (ViewSet, AccessSet, ConflictSet) and by using the concept of delegation (cf. Fig 1). In fact, ACTA allows finer control over the visibility of objects by associating three entities, namely ViewSet, ConflictSet and AccessSet with every transaction. We mean by Visibility the ability of one transaction to see the effects of another transaction on objects while they are executed.

The ViewSet contains all the objects potentially accessible to the transaction that can be operated by transaction. In addition, it specifies the state of these objects.

• The conflictSet contains objects that t wants access and which are already held by incompatible operations.

• AccessSet contains all the objects which are already accessed by a transaction.

Delegation: traditionally, the committing or aborting of an operation is part of the responsibility of the invoker. However, the invoker and the one committing operation may be different when a transaction delegates its responsibility to another transaction.

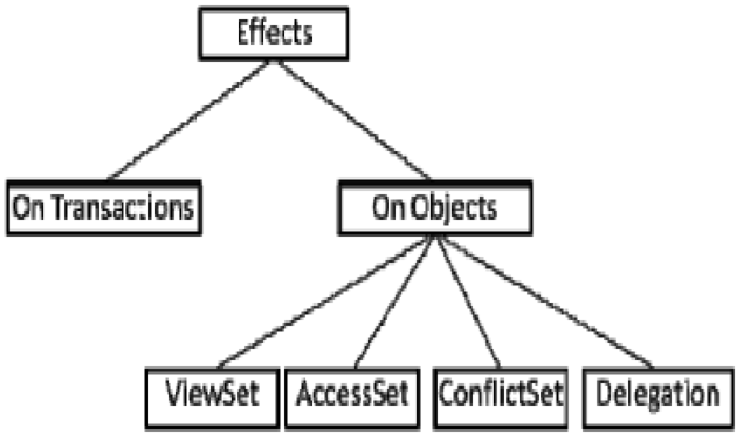

Figure 1. ACTA: Effects of transaction

3. Temporal Characterstics In ACTA

A great deal of ACTA work concerns the formal description of properties of extended transaction models, such as nested transactions, split/join transactions, transactions in active databases and the synthesis of extended transaction models [2]. Basically, transaction models in non real-time databases are the major concern of the past work. Through these last two decades, a lot of work has been done in real-time databases which investigate transaction scheduling in real-time database systems with transaction and data timing constraints [1]. So, Kaddes *et al.* [7] have proposed a Real-Time ACTA framework (CRT-ACTA) as an extension of ACTA to specify and formalize the transaction and data timing constraints. Unlike RT-ACTA [1], which is based on E-C-A (eventcondition- action) model and deals only with active databases, CRT-ACTA deals with all RTDBMS. In this section, we briefly recall the concept of effects of transactions and transaction dependencies invoked in [7], and we introduce the new concepts of temporal validity, priority and dependencies on occurrence. Then, we deal with effects of transactions on data, data time constraints and data freshness.

3.1 Temporal Characteristics of Transactions

The distinction between real-time and non real-time database systems lies in the timing constraints associated with transactions and data values in a real-time database. In realtime database systems, transactions have to complete before deadlines, and they have to read data values which are temporally valid, because data values may become stale as time goes by.

In other words, a transaction is characterized not only by its effects on objects and other transactions as we have seen in previous section, but also by temporal validity and priority (cf.Figure2).

1.Temporal Validity

Temporal validity permits to specify the validity interval of transaction, periodicity of transaction and the type of transaction: hard, firm, soft. A hard deadline transaction has hard timing constraints that must absolutely be met. Firm deadline transactions should also be met but may be missed occasionally. A firm transaction that misses its deadline adds no value to systems. A soft transaction is not interrupted once its deadline missed.

2. Priority

Priority permits to specify how priority is assigned to transaction. Priority is used in most conflict resolution protocols, so the priority assignment policy plays a crucial role.

3 Effects of transactions

The effects of transaction are divided on three subcategories: on transaction, on occurrence and on objects.

- On transaction

Recent researches in real-time transactions in RTDBS have focused on the idea of using partial results so that transactions meet their deadlines [6], [8]. Transactions are composed of two types of sub-transactions: optional and required. When required sub-transactions of transaction ti are executed, ti transaction pre-commits. Optional sub-transactions may be aborted during execution if they do not have enough time to complete. Consequently, pre-committed transactions will not abort.

Due to induction of pre-commit event, transaction relationships and behaviour between transactions are modified. So, we propose many new dependencies for more expressiveness of these relationships. For example, if two transactions t_i and t_j must commit, we can define only one dependency “t_i CD t_j”. So with such dependency, we can’t express the relationships between precommit transactions. We define following new dependencies for more precision in execution of transaction:

- **Pre-Commit dependency** (t_i PCD t_j). Transaction t_i can’t pre-commit until transaction t_j pre-commits
- **Strict-Pre-Commit dependency** (t_i SPCD t_j). Transaction t_i can’t pre-commit until transactiont_j commits
- **Weak-Commit dependency** (t_iWCD t_j). Transaction t_i can commit if t_j is already pre-committed.
- In the same way, we can introduce the pre-commit in the exclusion dependency.
- **Pre-Exclusive dependency** (t_i PED t_j). If the transaction t_j pre-commits then t_i must abort.
- **Pre-Exclusive** Pre-Exc(t_i, t_j). Two transactions t_i and t_j are excluded mutually each transaction develops a relation of pre-exclusive dependency on the other.

- On occurrence

The effects of transaction on occurrences are defined by the dependencies between different occurrences of transactions and permit to take into account the periodicity of transaction. Particularly, when a transaction is repetitive, an occurrence of transaction can depend on the other occurrences, e.g., in m-k firm model only m occurrence must commit among k occurrences. If m occurrences are already committed, the remainder occurrences can be aborted or not initialed [9], [10]. So, for taking into account the behaviour between different occurrences of a transaction, we define these dependencies:

Not ADmit Occurrence on Commit Dependency

NADOCD (t_{ji}, K, m): the occurrence i of the transaction t_j is not admitted if the k precedent occurrences are committed.

Aborted Occurrence on Commit Dependency AOCD (t_{ji}, k, m): the occurrence i of transaction t_j is aborted if the k precedent occurrences are committed.

In dynamic systems, such web servers and sensor networks with non uniform access patterns, the workload of RTDB cannot be precisely predicted and, hence, RTDB can become overloaded. As a result, uncontrolled deadline misses may occur during the transient overloads. So, many researches proposed [11] to not admit or to abort an occurrence in some conditions. This will imply modifications on the behavior of the transactions and their interrelationships. We can drive new dependencies by applying the concept of conditional dependency [7] on the previous dependencies and relaxing them.

A conditional dependency is noted as follows: ti (dependency [C]) tj, where the condition C is an optional part. When condition is mentioned and satisfied, the dependency must be respected. If the condition is omitted, the dependency must be respected all the time.

Conditional Not Admit occurrence on Commit Dependency CNADOCD (t_{ji}, K, m, [C]): the occurrence i of transaction tj is not admitted if the k previous occurrences are committed and if condition C is true.

Conditional Aborted Occurrence on Commit Dependency CAOCD (t_{ji}, k, m, [C]): the occurrence i of transaction tj is aborted if the k previous occurrences are committed and if condition C is true.

- On objects

The effects of transactions on objects are described by the states and status of objects (cf. section 3.2).

3.2 Temporal characteristics of Data
In RTDBS, we distinguish: update transactions and user transactions. Update transactions are used to update the values of real-time data in order to reflect the state of real world. We can divide this class in two sub-categories. Sensor transactions composed of a simple operation, executed periodically and having only to write a real-time data item. Recomputation or sporadic transactions have to derive a new data item from basic data items when these later are updated [11]. User transactions, representing user requests, arrive aperiodically and may read real-time data items, and read or write non realtime data items.

Each type of transactions has its own structure and has a different behavior, e.g., different interactions with the others transactions and objects. These items are considered as passive components.

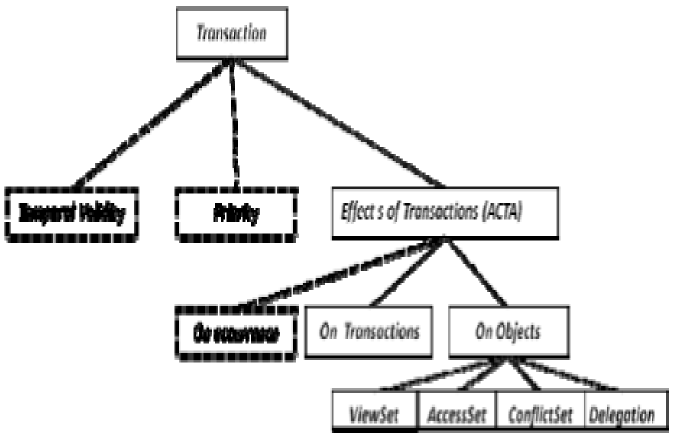

Figure 2. Transaction characteristics and effects

The non temporal data items are found in traditional databases and their validity is not affected as times goes by. The temporal data items, which can be sensor data items or derived data items, represent the states of real-time objects which change continuously. These states may become invalid as time goes by. To reflect these changes, data objects that model the entities need to be updated. Only non temporal data items are taken into account by ACTA. So, we propose to extend Acta to formalize temporal characteristics.

We categorize data freshness into "database freshness" and "perceived freshness".

• Database freshness (DF) describes the state of data. In addition, data freshness specifies temporal status and it describes the method of freshness [12].

• Perceived freshness (PF) is defined for the data accessed by user transactions [12].

1) Database Freshness
Data-base freshness is divided into four components: validity interval, value, methods of freshness and impact on other data.

- **Validity interval:** leads to the notion of temporal consistency. Temporal consistency has two classes:

• Absolute-consistency is the state between the environment and its reflection in the database. As mentioned earlier, this arises from the need to keep the controlling system's view consistent with the actual state of the environment.

• Relative-consistency is among data used to derive other data. A RTDB contains basic data items which record and model a physical real world environment. These basic data items are summarized and correlated to derive views, so when the environment changes, basic data items are updated, and subsequently view recomputations are triggered [13].
Figure 3 shows the structural relationship of these components and update transactions.

- ***Value****:* it defines the contents of the data. A data can be imprecise and multi-versions.

• ***Data error***: As we have already mentioned, a RTDBS can become overloaded and this overload is unpredictable. Hence, many approaches propose through the transient overload to discard many update transactions to reduce the workload avoiding consequently the degradation of system and eventually its crash. The discard of update transactions implies a certain degree of deviation compared to realworld value and then the decrease of data quality. In order to measure data quality many researches introduce the notion of data error, denoted DE. A data error which gives an indication of how much the current value of a data object di stored in the RTDB deviates from the corresponding real-world value, given by the latest arrived transaction updating di. The upper bound of the error is given by the maximum data error (MDE) [14].

• ***Versioning:*** the conflict between transactions is one of major factor that makes some transactions blocked, or aborted and restarted and thereafter this may lead to the miss deadlines, especially in transient overload. To address this problem, many approaches proposed a multi-versions data [11]. An object is defined by a set of values and can be accessed by many read and write operations which belong to different transactions. So, we limit data access conflicts between transactions, enhance the concurrency and limit the deadline miss ratio.

- **Method of freshness*:*** It is important to guarantee the freshness of data independently. Hence, many approaches, particularly for derived data items, are proposed to guarantee the freshness of data with inducing an acceptable number of triggered recomputation transactions. We can mention delayed forced update, periodic update, on demand update and deferred update [15]. Similarly, for sensor data, different approaches are proposed to maintain the freshness; e.g., periodic update, on demand, deferred update…

- **Impact on others data:** define the set of objects whose values are derived from current object. The change of its value implies the obsolescence of related objects and triggers their recomputation.

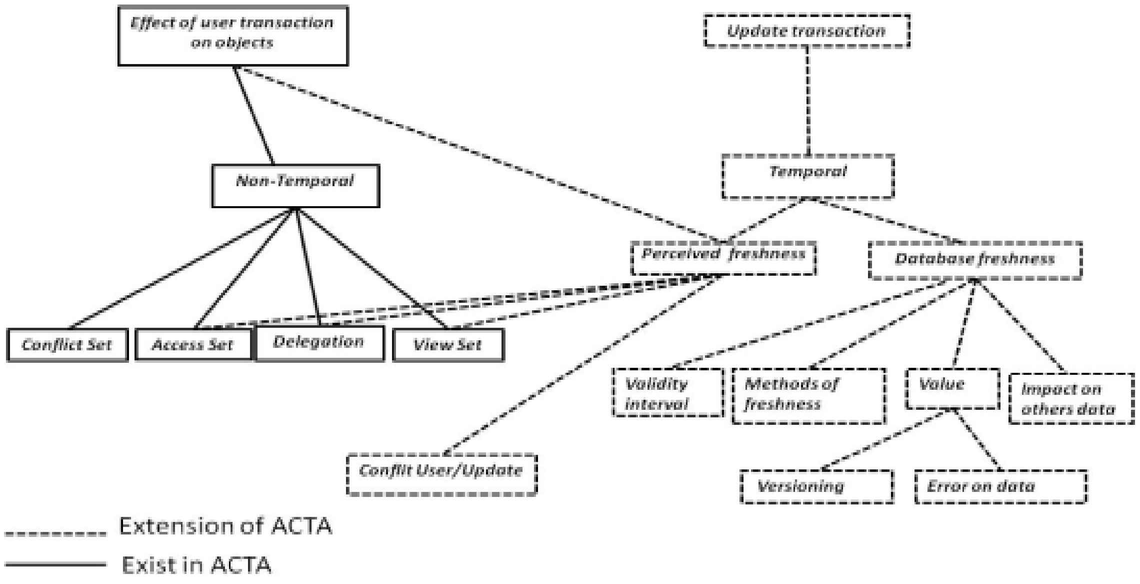

Figure 3. Effects of user and update transaction on temporal and non temporal data

2) Perceived Freshness

Perceived freshness describes the effect of a user transaction on real-time data. It is represented by AccessSet, ViewSet, delegation and ConflictUser/Update Set. This later describes the conflicts between user and update transactions. We extend ViewSet and AccessSet by adding temporal constraints to allow finer control over the visibility of objects.

- The **ViewSet** contains all the valid objects potentially accessible to the transaction. It specifies what objects can be operated by the transaction, i.e., the state of these objects that is visible to the operations invoked by the transaction and specifies temporal data constraints, i.e., only fresh data are allowed to be viewed. Otherwise, this constraint is relaxed.

- The **AccessSet** contains all objects already accessed by a transaction. AccessSet is extended to specify temporal access policy, e.g., a transaction must be temporaly correct or relaxed: absolute consistency and/or relative consistency and/or timely commit. more detail are given in [1].

- The **Conflict User/Update Set** of a user transaction contains objects that user transaction t wants read and which are already held by sensor or recomputation transaction.

Figure 3 shows the structural relationship of elements used by a perceived freshness of a user transaction with update transactions.

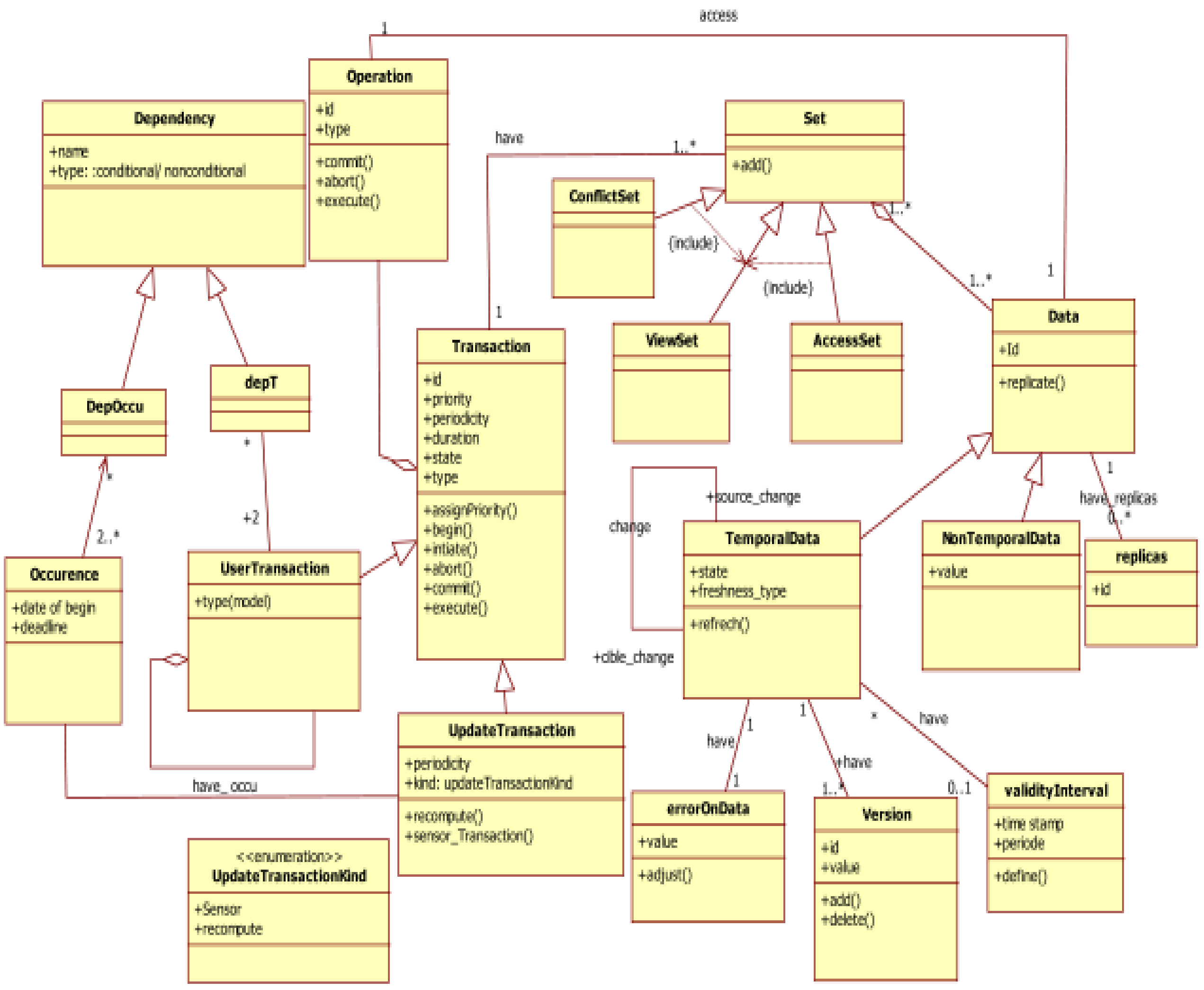

Figure 4. Meta-model of Real time extending transaction Model

4. Formal Definition Of RT-ETM

In this section we present a metal-model of real-time extended transaction (RT-ETM) using the semi-formal language UML (Unified Modeling Language). Figure 4 shows a class diagram where each concept as is represents by a class e.g. Transaction is presented by class transaction. It has priority, periodicity, duration … as attribute and begin (), abort ()… as methods. The interrelations among concepts are expressed by different types of association e.g. to express that temporal data is particular data we have make a Generalization- Specialization between the class “Data” and class “TemporalData”. The reflexive dependency “change” express that the refresh of temporal data (base data) induces the refresh of all related temporal data (derived data).

The instantiation of this meta-model permits the definition of new extended transaction model. The correctness of metamodel can be verified by simple translation to formal language Z for more information you can see [16].

5. Conclusion

In this paper, we have presented the components of ACTA and we have proposed many extensions to facilitate: (i) the formal description of properties of transactions in real-time databases, reasoning about the transaction interactions and effects on objects. To this purpose, we have added many new dependencies to capture more interactions between transactions in real-time environment and we have introduced many concepts such as perceived freshness, conflictSetUser/Update to formalize and to control the effect of user transactions on data (ii) the formal description of properties of temporal data. For this, we have defined many new concepts to allow a finer control of real-time data items. We have presented a meta-model of real-time extended transaction model.

In our future work, we plan to give a formal definition of new extended transaction model based on this work and to prove its correctness. We also project to extend our study to take into account transaction recovery.

References

[1] Xiong, M., Ramamirthan, K. (1997). Towards the Specification and Analysis of Transactions in Real-Time Active Databases, RTBD'97, Burlington Vermont, USA, p 327-348, .

[2] Chrysanthis, P.K., Ramamritham, K. (1994). Synthesis of Extended transaction Models Using ACTA, *ACM Trans, Database Syst.*, 19 (3) 450- 490.

[3] Schwarz, K., Turker, C., Saake, G. (1998). Analyzing and Formalizing Dependencies in Generalized Transaction Structures". *In:* Proc. of Int. workshop on Issues and Applications of database technology, July 6-9, 1998,Berlin, Germany,.

[4] Schwarz, K., Turker, C., Saake, G.. (1998). Transitive Dependencies in Transaction Closures. Database Engineering and Applications Symposium, July 8-10, Cardiff, Wales, UK,.

[5] Schwarz,K., Turker, C., Saake, G.. (1998). Extending Transaction Closures by N-ary Termination Dependencies, Symposium on Advances in Databases and Information System (Adibis'98), Semptember 8-11, Poznan, Poland.

[6] Abdouli, M. (2006). Study of Extended Transaction Model Adaptation to Real-time DBMS, PhD Thesis, Le Havre university, France (in French).

[7] Kaddes, M., Abdouli, M., Bouaziz, R. (2008). Adding New Dependencies to Acta Framework, *In:* Proceedings of 22nd European Simulation and Modelling (ESM'2008). Le Havre, France, 27-29, October .

[8] Haubert, J., Sadeg, B., Amanton, L . (2004). (m-k) firm real-time distributed transactions, *In*: Proc. of the 16th WIP Euromicro Conference on Real-Time Systems (ECRTS).

[9] Hamdaoui, M., Ramanathan, P., (1995). A Dynamic Priority Assignment Technique for Streams with (m,k)-Firm Deadlines, *IEEE Transactions on Computers*, 44 (4) 1325–1337.

[10] Koren, G., Shasha, D. (1995). Skip-over: Algorithms and complexity for overloaded system that allows skips, *In:* Real-Time System Symposium, p. 110–117.

[11] Bouazizi, E., Duvallet, Sadeg, B. (2005). Multi-Versions Data for Improvement of QoS in RTDBS, *In*: Proceedings of 11th IEEE International Conference on Real-Time and Embedded Computing Systems and Applications (IEEE RTCSA'2005), Hong Kong, China, pp 293-296, August 17-19.

[12] Kang, K., H Son, S., Stankovic, J.A., Abdelzaher, T.F., (2002). A QoSSensitive Approach for Timeliness and Freshness Guarantees In Real-Time Databases, *In*: the 14th Euromicro Conference on Real-Time Systems.

[13] Kao, B., Lam, K., Adelberg, B., Cheng, R., Lee,T. (2003). Maintaining Temporal Consistency of Discrete Objects in Soft Real-Time Database Systems, *IEEE Trans. Comput*., 52, 373- 389.

[14] Amirijoo, M., Hansson, H., Son, S. H (2006). Specification and Management of QoS in Real-Time Databases Supporting Imprecise Computations, *Transactions on computers*, 55 (3).

[15] Katet, M., Duvallet, C., Bouazizi, E., Sadeg, B (2006). Prise en compte des données dérivées temps réel dans une architecture de contrôle par rétroaction, *In*: MCSEAI'2006, Agadir, Maroc, p. 114-119.

[16] Ali, M., (2010). Verification Et Validation Formelles de Modeles UML: Approches et outils, Editions universitaires européennes, 17 December.