Enhancement in a Formal V&V Approach for Real-Time Databases

Andleeb Shahnaz
National University of Sciences and Technology (NUST), H-12, Islamabad

Farooque Azam
National University of Sciences and Technology (NUST), H-12, Islamabad

ABSTRACT
With the evolution of concepts in databases' field, the complexity of systems is also increasing along with their efficiency. In conjunction with consistency and concurrency related problems, a related issue is data verification and data validation. To cater to the data collisions and synchronization problems, data verification and validation comes to play its part. This paper is motivated to quantify and enhance one of the available "formal verification and validation approach for real time databases" presented by "Ribeiro Neto, Perkusich, Oliveira De Almeida and Perkusich". This existing V&V technique uses the object oriented data models for real time databases. In this paper, first of all we highlight the issues associated with the use of object models and also their impacts on the accuracy of this verification and validation approach which is being analyzed. Secondly, a comparison of different alternatives for real time data modeling is carried out and on the basis of this comparison, an alternative for minimizing or solving these issues of object data models is proposed. Thirdly, an additional step, for gathering requirements for real time databases is added in this V&V approach.

General Terms
Real-time Databases, Verification and validation approaches, Data modeling.

Keywords
Real-time databases, Formal Verification and validation approach, Data modeling, Comparison of data models, Relational data models, Temporal relational data models, Temporal data models, Temporal object-oriented data models, Pros and cons of object data models for RTDBs.

1. INTRODUCTION
In this ever changing world of information technology, databases are considered the backbone of any software project. The reason for this is that data management, if done properly, enhances the strength of a project and if not then adds to complexity and results in increase of risks and cost of a software project. Some time ago, conventional databases were more in field, but their usage has reduced quite a while ago, due to rapidly growing need of time constrained data & transaction requirements of computer systems and applications. **[1]** Nowadays, real-time processing and data handling has become the need of almost every kind of computer system. For example, data intensive smart space applications, online trading, e-commerce, sensor data fusion, embedded control systems etc.

[2] Furthermore, with the fast growth of internet, the concept of information system is being globalized and need of an efficient RTDB is exploding due to the time constrained requirements in a globalised environment. But the database system can be the main service bottleneck in this upcoming global information infrastructure because of the **[3]** issues of limited performance, low predictability, consistency problem, data and resource conflicts, transaction aborts and the resulting rollbacks and restarts, that are inherent in current RTDBs.

Keeping these issues in mind, we can say that the fundamental requirements for achieving the success of real time database services are:

- Transaction timeliness
- Data freshness

But problem is that both of the above requirements can be conflicting to each other and a balance is needed to be achieved intelligently. This part is quite tricky and needs to be strictly verified and validated after being done.

So, this means that it is high time to think about verification and validation of RTDBs and to devise such methods and techniques for it that can **[4]**:

- Ensure the fulfillment of intended fundamental RTDB requirements
- Help to reduce their problems and issues to the greatest possible extent
- Most importantly help to convince the users, designers or certification authorities that it's safe to deploy the developed computer system.

Some research has been done in this field and some methods like different concurrency control techniques have been introduced which are quite good, but still a lot more research is required for RTDB. Specially, despite of the huge demand and need, there has been very scarce research regarding real time database modeling that allow a formal analysis, considering verification and validation characteristics. The existing tools for supporting modeling process especially do not present simulation capacity. One example of this is UML modeling, as UML approach presents a number of favorable characteristics for modeling complex real-time systems **[5]**, but the existing tools for UML modeling do not present simulation capacity (*Ribeiro-Neto, Perkusich, Oliveira De Almeida, Perkusich, 2009)**[6]***.

The main concept of verification and validation technique presented by "Hermann Kopetzis" **[4]** and also in **[7]** is described above in figure 1. Quite recently "A formal verification and validation approach for real-time databases" was introduced in a research paper by "*Ribeiro-Neto, Perkusich, Oliveira De Almeida, and Perkusich".*

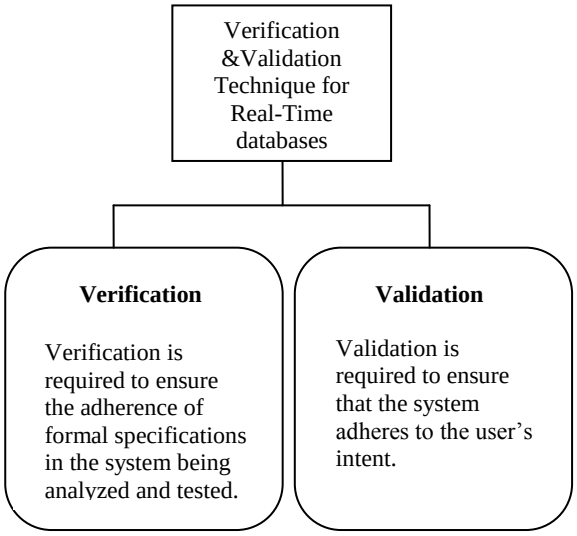

Figure 1: Main Concept of Verification & Validation Technique for RTDB

This technique has application of five major steps, which include:

1. Building an object model
2. Building a process model
3. Generating an occurrence graph
4. Generating a message sequence chart
5. Generating a timing diagram

Static analysis is done in first step, dynamic in the second and other three steps are dedicated to the validation of the model. This approach uses Hierarchial coloured Petri nets as a formal language for describing RTDB models (Jensen, 1998) **[8]** and it can be applied to various database systems that are developed for computer-integrated manufacturing, network management, stock exchange, command and control applications and multimedia systems (*Ribeiro-Neto, Perkusich, Oliveira De Almeida, Perkusich, 2009)**[6]***.

This writing is intended to analyze the above mentioned formal V&V approach, to specify its limitation and to suggest an alternative for compensating for that limitation. The remainder of this paper is presented in the following order. First of all a background is presented, in which the concepts of real time databases are described. Secondly, it compares the temporal, conventional and real-time databases. Thirdly, that formal V&V approach for RTDB is described which we intend to analyze in this paper. Fifth, its limitations are specified and then suggestions are made. Sixth, an additional enhancement in this V&V approach is proposed and finally, future trends and conclusion are presented.

2. BACKGROUND

2.1 Real-Time Databases (RTDB)

[1] "RTDB is by definition, a database system that has time constrained data and transactions.**"** The different types of possible restrictions regarding time can be: start times, end time limit, etc and the time semantics are associated with data to show that the data is valid for specified time limit. This is also worthwhile to notice that time constrained behavior is not the requirement of every database process. But it is possible that such real time applications require time constrained behavior in at least few of their processes or functions. (Gultekin Ozsoyoilu and Richard T. Snodgrass) **[9]**.

2.2 Data Types in RTDB

The data objects reflect the status of the environment and in real time databases; they are classified into following two categories as shown in figure 2.

a. *Static Data:* Static data is the form of data that exists in all typical databases and its main characteristic is that its value doesn't change with the passage of time i.e. it doesn't become outdated **[10]**.

b. *Dynamic Data:* Dynamic data is the data that always keeps on changing for reproducing the changes in the actual environment **[2]**. It is further classified into following two subparts:

- Base Data: It imports the outside environment's view
- Derived Data: It's a data item which is derived by using more than one base data items.

2.3 Transaction Types in RTDB

Transactions in a real-time database are classified into three main categories, which are further divided into the sub-categories, on the basis of variant criteria as shown in figure 2. One of the categorization based on real-time constraints **[4]** is:

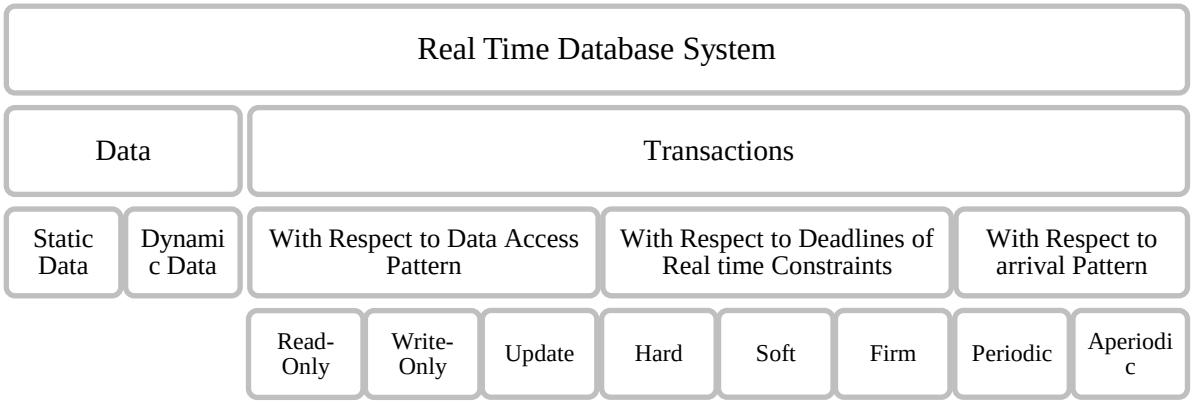

Figure 2: Parts of Real Time Database System

a. *Hard:* A hard transaction is the transaction that has most strict deadline and if its deadline is not met in time then it causes a serious damage and can result in a catastrophe.

b. *Firm*: Firm transactions are those which can be aborted and removed from the system if their deadline is not met, so that wastage of resources can be avoided that can occur if these tardy transactions are executed.

c. *Soft:* The transactions with soft deadlines can be finished by the system (i.e. commit/ terminated), even after their deadline. **[6]** Though, it's undesirable, but still if such transactions are completed after their deadlines, even then they can commit the system performance.

Second categorization that is based on arrival pattern of transactions is as follows:

a. *Periodic:* A periodic transaction is the one that occurs in a regular pattern i.e.at every periodic interval and is predictable by database systems e.g. sensor readings, robot position after every 10 sec, etc.

b. *Aperiodic:* Aperiodic transactions are those that are not predictable by the database system and occur only if the data value is changed. Furthermore, aperiodic updates can be modeled as sporadic. For example, stock price updates can be modeled as sporadic based on the highest update frequency in the peak time, therefore, a validity interval can be associated with a stock item. An existing stock price can be marked stale if an update actually arrives in the validity interval. Otherwise, the validity interval can be extended for another period by renewing the timestamp as the current time (Kyoung-Don Kang, December 6, 2001) **[2]**.

Another categorization based on data access type is as under **[11]**:

a. *Write:* Such transactions are also called sensor transactions. These write-only transactions are specifically designed for the purpose of periodically updating your base data, with the intention that it replicates the current status of the actual environment.

b. *Update:* They are also called re-computation transactions. For maintaining the temporal consistency, these transactions are used for the purpose of re-computing the derived data - whenever the related base data changes - and then write/store that in the database.

c. *Read:* These transactions are also called user transactions. These are user-level transactions (aperiodic in nature) that have a time limit. They have the access right for reading and writing non-sensor data, however sensor data cannot be accessed by using them.

An important issue that is related to transactions is how to avoid conflicts that occur due to concurrent execution of transactions. A lot of work has been done in this dimension and many concurrency control techniques have been proposed. But it is a fact that has been proposed by most of the researchers that abnormal termination of transactions and resulting restarts cannot be entirely eradicated due to the presence of conflicting transactions, no matter how good concurrency control technique is used. The only condition in which it is possible is to get rid of them entirely, is to execute them serially. But this is not possible in real time applications **[12, 13]**.

2.4 Data Consistency:

Normally correctness of a database depends upon data consistency and in a real time database; by data consistency, we mean both logical and temporal consistency.

Logical data consistency is attained by using concurrency control techniques whereas temporal data consistency is defined by using following categories of validity interval.

[1] An RTDB makes validity intervals explicit as part of its database schema.

a. *Absolute validity Interval:* **[2]** It is the time interval for which a dynamic data object will be considered valid or temporally consistent. An absolute validity interval of a data item x is valid only, if it is longer than the time interval between current and last update time of the data being tested i.e. (avi(x) >= (current_time – timestamp(x))).

b. *Relative validity Interval:* A relative validity interval specifies that how old your base data can be. Suppose data item y is derived from data set $R=\{x1, x2, ..., xk\}$, then y is temporally consistent if the base data items in R are temporally valid and the $|timestamp\ (xi\ R) - timestamp\ (xj\ 0\ R)| \leq rvi\ (y)$.

Data is said to be temporally consistent only if it is both relatively as well as absolutely consistent.

3. A COMPARISON OF CONVENTIONAL, TEMPORAL & REAL-TIME DATABASES

"Conventional databases characterize the condition of an environment at a particular time instant because when new information is added, the contents of a database get changed and with this, the previous outdated data is deleted from the database.**"**

Furthermore, in conventional database management systems, queries and transactions are executed depending on their arrival order and their completion times are not assured as well **[9]**.

Table 1: Data Types Supported by Conventional & Temporal databases [10]

Snapshot Data Type	Static Data Type	Temporal Data Type
A snapshot data type value is the one that is only valid for the current instant.	A static data type is a constant that is defined over whole universe of time or we can say that it is a value that is valid for forever.	A temporal data type value is the one whose validity can be assured for only a particular interval of time.

According to the definition provided by **[9]**, **[10]** Chris Klassen and Richard Snodgrass, temporal databases have a fixed timescale (e.g. seconds or even milliseconds) for efficiently storing a time series of data and then after that fixed timescale only the changes in the measured data are recorded/ stored.

Real-time databases are those databases which have time constrained transactions. That's why they are also termed as time-constrained databases. **[11]** The importance of real-time databases is that they incorporate the features of both, a conventional database, as well as an RTDB, e.g. transaction processing with timing constraints and temporal data integrity (Stankovic and Son, 1999; Kang et al., 2002). The schema illustrating the properties of a real time database in comparison with traditional database systems is shown in figure 3 below. **[1]** The difference between temporal and RTDB is that both of them support different aspects of time. Temporal databases support information related aspects of time, e.g. stock quotes, whereas an RTDB tries to provide support for operation related aspects of time.

The different data types that are supported by conventional and temporal databases, including snapshot, static and temporal data types are described in table1 above, and a comparison of conventional and temporal databases on the basis of supported data types is shown in table2 below. And the data types that are supported by real-time databases, including static and dynamic data types, have already been defined and described in the previous sections of this writing, in detail.

It's a common thinking that the only necessary requirement for real-time databases is that they should respond quickly and with high speed and due to this, conventional databases can suffice for real time applications, but in reality, it's not true because in an RTDB, fast service doesn't necessarily mean predictable service.

Conventional databases are inapt for data intensive real time applications because they are unaware of data freshness and timing constraints. Though it's possible to support the time constraints in conventional databases and some designers believe that they can force these real-time principles into conventional databases, but still the fact is that **[1]** there is a high probability that the resulting real time database system will not suffice in terms of efficiency. An example of this scenario can be that, suppose you've added a validity interval field in each of your conventional database table and the transactions themselves monitor them for verifying the absolute and relative validity. This force fitting way is quite inefficient because now every transaction must provide this monitoring ability itself.

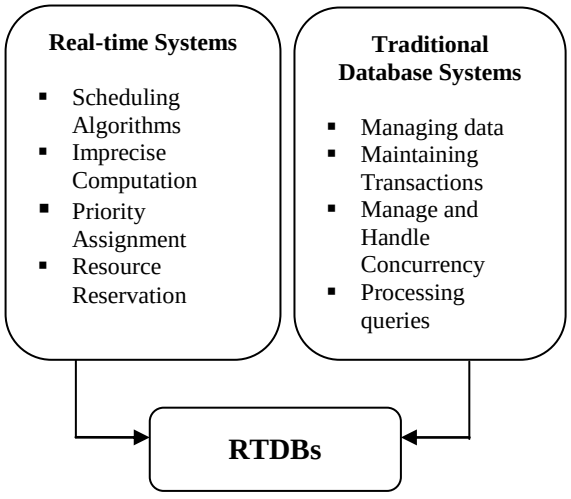

Figure 3: Schema for RTDB Systems

Table 2: Comparison of Conventional & Temporal databases

	Snapshot Data Type	Static Data Type	Temporal Data Type
Conventional Database	✓		
Temporal Database	✓	✓	✓

In most of the existing research work for real-time databases, the **[9]** two major concepts that are always discussed are temporal data and temporal consistency constraints, which I've discussed above as well. But the point that hasn't been given a thought yet is that if time constrained data and consistency limitations are necessary to describe the real-time databases then why not to use temporal models and temporal query languages for RTDBs.

The reason along with benefit for this cross-infusion, presented by **[9]** is that: **"**some benefits may directly be achieved regarding real-time database research, when temporal data models are used for specification and management of queries and transactions in real-time databases. Due to this, there arises the possibility that querying capabilities in RTDBs might get enhanced.**"**

4. REAL-TIME DATABASE VERIFICATION AND VALIDATION METHOD

A fairly good verification and validation technique is presented by "*Ribeiro-Neto, Perkusich, Oliveira De Almeida, Perkusich, 2009)**[6]**"* for real-time databases. Design/CPN tool package (Jensen et al, 1999) **[8]** is used in this approach for HCPN modeling. This V&V technique is comprised of following five steps:

- Build an object model
- Build a process model
- Generate an occurrence graph
- Generate a message sequence chart
- Generate a timing diagram

These steps, when followed together provide you with a simple way of verifying and validating your real-time database system.

4.1 Step 1- Build an Object Model

As described earlier in this writing, this V&V approach for an RTDB uses the concept of a real-time object model and then all the further analysis is based on this model. At this initial step, that main object model is created. The main dissertation for the process model is defined by this object model and **[6]** following sub-steps are used for creating it, which are shown in figure 4.

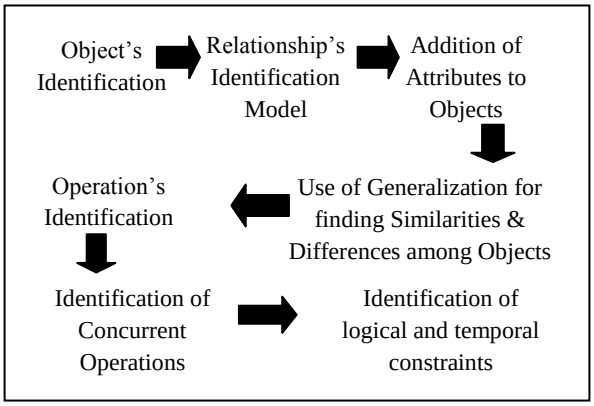

Figure 4: Steps for Building an Object Model

4.2 Step 2 - Build a Process Model

For objects identified in the above step, both functional and dynamic properties are captured by the process model. At this step, HCPNs are used, through which objects with their respective operations are first described and then with each of these HCPN modules the system behavior is analyzed. Design/CPN tool package was used for HCPN modeling in this step. The sub-steps involved in this phase are shown in figure 5.

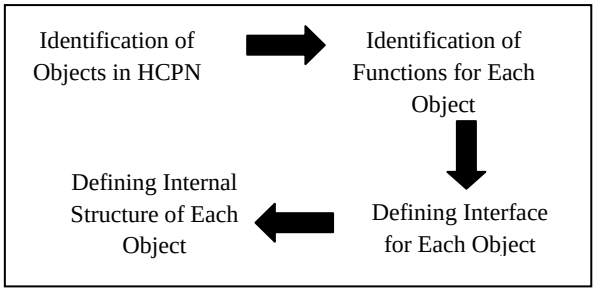

Figure 5: Steps for Building a Process Model

4.3 Step 3 - Generate an Occurrence Graph

For the HCPN model created in the above step, now an occurrence graph is generated so that its state space can be represented and as a result the occurrence possibility of different transitions/transactions can also be identified. The benefit of using an occurrence graph is that through it, the inherent properties of an HCPN model can be verified, which include liveness properties (which shows the dead markings. dead/ live transition instances), boundedness properties (which shows marking limits for each place instances) etc.

Design/CPN tool package that is mentioned above was used for generating an occurrence graph. This tool provided them with some simple and useful features like: **[6]** built-in standard queries (e.g. *Reachable*, *AllReachable* etc), a number of powerful search facilities allowing formulating non-standard queries, reports generation (these reports contain information about the graph and its metaproperties).

4.4 Step 4 - Generate a Message Sequence Chart (MSC)

An MSC is a language through which communications or connections between parts of the system are specified and following **[6]** benefits can be achieved.

- Specifying the requirements
- Simulating and Validating the system
- Specifying the test-cases
- Documenting Real-time Systems

A message sequence chart is generated at this stage for each possible situation, considering a particular order of execution. It represents both data properties and transaction properties along with their respective temporal constraints and thus makes it possible to verify the properties of an RTDB along with the validation of:

- behavior and relationships of objects
- scenarios where concurrent access to the RTDB takes place by using object operations

Design/CPN tool is again used for generating a messaging sequence chart as well.

(*Ribeiro-Neto, Perkusich, Oliveira De Almeida, Perkusich, 2009, **[6]**)*

4.5 Step 5 - Generate a Timing Diagram

At this last stage, a timing diagram is generated through design/CPN tool for simulation based performance analysis of HCPN and for showing the timing constraints in time sample.

5. ANALYSIS

5.1 Limitations

A lot of research has been conducted regarding the use of object data models in real-time databases, which shows that there are certain merits and demerits linked to their use for real-time database modeling, as summarized in table 3 below. The advantage is that the large variety of data semantics of object oriented data models can be used for dealing with real-time transactions **[14]** and its benefits or merits may be needed for complex real-time applications **[15]**, e.g. methods, encapsulation, complex objects **[9]**. But following problems can arise in relation to their use.

a. **"**Existing research is not sufficient for incorporating these features in time-constrained real-time databases. E.g. For real-time databases, the high time costs related with the storage and implementation of complex objects must be minimized. Along with that, the extremely parameterized time cost formulas for such operators make it difficult to guarantee probable complex-object algebra time costs with small variances.**" [9]**

Table 3: Merits and Demerits of Using Object Data Models for RTDB

Advantages	Issues
Complex objects can be represented easily through its rich semantics support	Significant research, effort and modification of existing object-oriented data models is required for providing the features of object models in time-constrained RTDBs.
Object oriented features improve maintainability and add extensibility	Encapsulation restricts the ways of accessing object, which can possibly result in delayed transaction execution.

b. **"**Along with providing a number of benefits, encapsulation also limitizes the ways of accessing the objects, which can result in deferred transactions execution.**"[9] [16]**

c. Coupling, Cohesion, inheritance and complexity issues of object oriented environments can also be a factor that can result in fault proneness in the system being verified and validated **[17**]. This is because of the same reason as described above, that they restrict the ways of accessing the real time objects that are identified in the real time environment of the system under test.

From these above described limitations it is evident that further research is needed to make it certain that in database operations timeliness is ensured when new data modeling features are provided. Because of this analysis regarding the use of object oriented data models for an RTDB, a point arises that if we use them in the verification and validation process for an RTDB then these same limitations will be inherited by that V&V technique as well. This as a result will affect the efficiency of that V&V technique.

Therefore, as "formal verification and validation approach for real time databases" by "Ribeiro Neto, Perkusich, Oliveira De Almeida, Perkusich, 2009", also uses object models in its first step then this means that this also makes even this technique vulnerable to the above discussed limitations and problems. So, an alternative needs to be considered for getting rid of these issues.

5.2 Existing Work on Providing Counter Measures for These Limitations

A lot of object oriented metrics have been proposed to provide counter measures for these issues of object oriented modeling environment and also to control and handle their effects.

The vital argument regarding object-oriented metrics development is that by using them, faulty classes can be identified prior to testing or field failures. **[17].** In past, at least fourteen empirical validations have been carried out for judging the validity of these metrics **[17].** However, according to some recent researches it has been deduced that these metric's impacts may be overstated if effect of size of class is not controlled during a validation study. **"***Khaled El Emam***" [17]** illustrated and simplified the confusing class size effect on object oriented metrics validity**,** through path diagram in figure 6.

So now, after this research indication only two metrics are considered valid enough for the prediction models creation. But they are only for coupling **[17].**

If there would have been validated metrics for other issues and concepts of object oriented modeling besides coupling then there might have been a possibility of incorporating some of the best metrics for covering these major object oriented modeling issues in our V&V approach. But now as they cover only for coupling then this leaves us in jeopardy again. Also, it might be possible in theory to apply these metrics in this V&V approach under test and get rid of these modeling issues and limitations pointed out above, to the greatest possible extent. But in reality the fact is that applying these metrics effectively has a pre condition of controlling the class size.

Achieving this pre condition of controlling the class size in case of real time systems is not that easy and most of the time it may require a compromise on timeliness behavior and requirements of transactions.

Due to these problems associated with the existing counter measures regarding object data models limitations and due to severity of the lacking in object data models in case of real time databases, it's worthwhile to consider other data modeling alternatives as well, for being used in V&V approach being analyzed.

5.3 Data Modeling Alternatives

Besides real-time object models, we've following other options for real-time data modeling that are:

- Relational data models
- Temporal relational data models
- Temporal data models
- Temporal object-oriented data models

All four of them are evaluated as follows by keeping in view their merits, demerits and applicability and along with each of them; the reasons for using or not using them in V&V approach are specified.

5.3.1 Relational Data Model:

It can be used for real-time database modeling but the main demerit is that time semantics are not supported in them (as described by "Gultekin Ozsoyoilu and Richard T. Snodgrass" **[9]**) which means that temporal dimensions are not present in relational data models. So, this means that relational models alone can't be sufficient for real-time database modeling.

5.3.2 Temporal Relational Data Model:

An alternative for covering the limitation of relational data model – that is described above - is to use relational models that have been extended by using temporal models concepts.

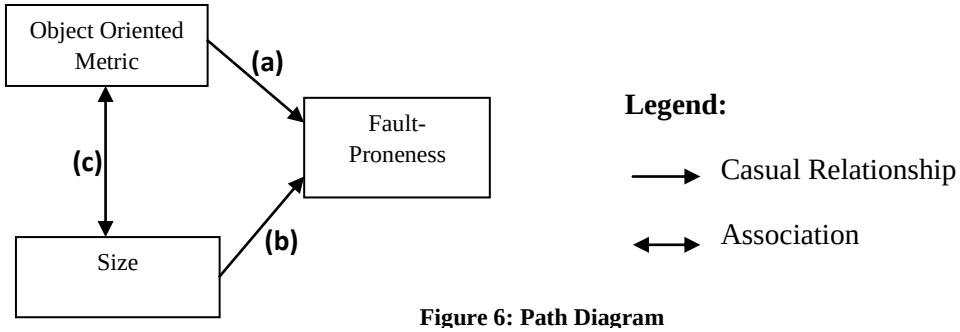

Figure 6: Path Diagram

The basic concept of such temporal relational data models is described by figure 7. One such "logical temporal relational data model" **[18]** has been presented by "Nadeem mahmood, Aqil burney and Kamran ahsan" which provides a fairly simple and easy way of modeling for real-time databases, by adding a temporal layer in relational model. A lot of similar work has been done in this dimension, and can be used.

However, this approach has been criticized for **"**violating Codd's Informational Rule of relational databases and for not being subject to the full normalization process**"** by Paul Schleifer, Yuan Sun, Dilip Patel **[19]**.

So, this means that we're left with only one good alternative that was proposed by "Snodgrass" **[9]** and "Paul Schleifer, Yuan Sun, Dilip Patel"**[19]**. This alternative modeling method is described in detail as under.

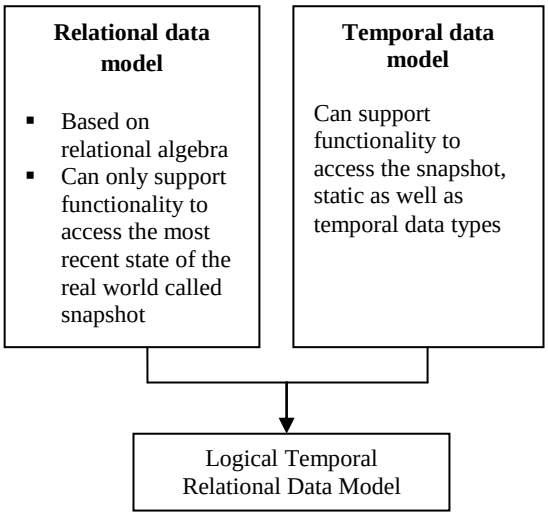

Figure 7: Temporal Relational Data Model

5.3.3 Temporal data model:

By using the temporal models and somewhat enhancing them to cater to the additional requirements of an RTDB – that are not present in a temporal database – it would be possible to provide one of the best possible alternatives. But the problem is that applying them along with petrinets and in the flow used in V&V approach being analyzed; might not be easy. It might require us to completely modify our V&V flow.

5.3.4 Temporal Object-Oriented data Model:

Among the various temporal models, the one that can work better for RTDB is the "Temporal element based data models", which is also considered best for temporal databases by "Seo-Young Noh, Shashi K. Gadia" **[20]**.

"Paul Schleifer, Yuan Sun, Dilip Patel" **[19]** have proposed a temporal object-oriented data model that can be used as an alternative for object data models. This approach uses the "Temporal element based data models" along with object data models concepts. By using such a model the enhancement and modification in the existing formal V&V approach for an RTDB will be easy because of two main reasons. One, the same flow will have to be followed and only first phase will have to be changed. Second, by adding the temporal models concept, the limitation of object models will be covered to some extent.

6. ADDITIONAL ENHANCEMENT IN FORMAL V&V APPROACH

Due to the inherent complexity of these real time systems, it'll be worthwhile to incorporate a "requirement gathering" step before "Building an object model" step in the V&V approach being analyzed. This will help out in identifying the true, complete and more accurate requirements for a real time system and thus will improve the accuracy of our verification and validation approach. Various researches have proved this fact that formal methods and formal specifications help in improving the clarity and precision of requirements specifications. Requirement gathering through formal methods has also been used as an integral part of various V&V approaches; an exemplary scenario is **[21].** So, it's advisable to use formal methods and specifications for requirement gathering before starting off with building an object model in the verification and validation process.

7. ENHANCED FORMAL V&V APPROACH FOR AN RTDB

After incorporating the enhancements proposed above, following is the anticipated improved verification and validation approach for real time databases:

Step 1: ***Gather Requirements through Formal Specifications***

Step 2: ***Build a Temporal Object Oriented Data Model***

Step 3: Build a Process Model

Step 4: Generate an Occurrence Graph

Step 5: Generate a Message Sequence Chart

Step 6: Generate a Timing Diagram

8. FUTURE TRENDS

Some surveys have already been conducted regarding temporal object oriented data models, temporal query languages and temporal access methods. The available temporal object oriented data models should be analyzed and one that appears to be the best among them should be implemented in this verification and validation technique. Through this, it'll become possible to quantify and verify that whether temporal object oriented data models worked in the expected ideal manner for an RTDB. Along with that it should also be discovered that whether the existing temporal query languages and temporal access methods can give some input in enhancing this verification and validation method.

Moreover, as also proposed by "Ribeiro Neto, Perkusich, Oliveira De Almeida and Perkusich", the automatic code generation from this modified verification and validation method should be considered. It will be highly satisfying to get to know that your automatic code generator is based on a verified and validated model. In recent years, graphical modeling environments consisting of block diagrams and state machines have been used to analyze, simulate, prototype, specify, and deploy software algorithms within a variety of embedded systems and applications. The main benefit of using model based design is to verify that model or design was fully implemented in the code and also that generated

code should behave like the simulation model, as is evident from **[22]**. So, it'll be worthwhile to consider using model based design prior to doing automatic code generation.

9. CONCLUSION

In this paper, we identified the limitations and issues that were inherent in the verification and validation approach that was presented by "Ribeiro Neto, Perkusich, Oliveira De Almeida and Perkusich" in a paper named "A formal verification and validation approach for real-time databases". Along with that various modeling options for real-time databases were described and analyzed. Then from all those modeling methods, temporal object-oriented data models were considered as best suited ones for real-time databases and were proposed as an alternative for object data modeling. This selection was made by keeping in view the merits, demerits, applicability as well as the availability and level of research work available for that model. Along with that it's proposed to add a step for gathering requirements for real time database systems through formal methods, prior to building an object model in V&V process. These two enhancements when incorporated are likely to enhance the accuracy of the existing formal verification and validation approach for real time databases.

10. REFERENCES

[1] Misconceptions about real time databases, John A. Stankovic and Sang Hyuk Son (University of Virginia, Charlottesville), Jorgen Hansson (University of Skovde, Sweden), 1999

[2] PhD Proposal on qRTDB: QoS-Sensitive Real-Time Database, Kyoung-Don Kang, Department of Computer Science, University of Virginia, December 6, 2001

[3] K. Ramamritham. Real-Time Databases. International Journal of Distributed and Parallel Databases, 1(2), 1993.

[4] Real-time Systems: Design Principles for Distributed Embedded Applications, Hermann Kopetz, **2011**

[5] Designing concurrent, distributed, and real-time applications with UML, Hassan Gomaa, George Mason University, Fairfax, VA, **2006**

[6] "A Formal Verification And Validation Approach For Real Time Databases", Pedro Fernandes Ribeiro Neto, Maria Ligia Barbosa Perkusich, Hyggo Oliveira De Almeida, Angelo Perkusich, **2009**

[7] **"**Guidance on Environmental Data Verification and Data Validation**"** *U.S. Environmental Protection Agency Quality System Series, 2002*

[8] Jensen, K. (1999). Design/CPN 4.0. Meta software Corporation and Department of Computer Science, University of Aarhus, Denmark. Retrieved May 19, 2006, from (http://www.daimi.aau.dk/designCPN/)

[9] Temporal and Real-Time Databases: A Survey, Gultekin Ozsoyoilu and Richard T. Snodgrass, **2008**

[10] What is a Temporal Database, Chris Klassen, **2008**

[11] "Real-Time Database Modeling Considering Quality of Service", Pedro Fernandes Ribeiro Neto, Maria Ligia Barbosa Perkusich, Angelo Perkusich, 2003

[12] An Approach for Real-Time Database Modeling and Performance Management - Jisu Oh and Kyoung-Don Kang, Department of Computer Science, State University of New York at Binghamton, *appeared in:* Real Time and Embedded Technology and Applications Symposium, **2007**. RTAS '07. 13th IEEE

[13] Fundamentals of Database Systems, Addison Wesley, R. Elmasri and S. B. Navathe. (4th edition, 2003.) & (5^{th} edition, **2008**)

[14] L.B.C. Di Pippo and V.F. Wolfe, "Object-based semantic real-time concurrency control," Proc. 14th IEEE Real-Time Systems Symp., Dec. 1993.

[15] J. Lee and S.H. Son, ''Issues in developing object-oriented database systems for real-time applications," Proc. IEEE Workshop Real-Time Applications, Washington, D.C., 1994.

[16] F. Bancilhon, "Object-oriented database systems," Proc. ACM SIGACTNGMOD PODS Conf., Austin, Texas, pp. 152-162, 1988.

[17] A Validation of Object Oriented Metrics, Khaled El Emam, Saida Benlarbi Nishith Goel, Shesh Rai, **2010**

[18] A Logical Temporal Relational Data Model, Nadeem Mahmood, Aqil Burney and Kamran Ahsan, January **2010**

[19] The Role of Object-Oriented Temporal Databases in Information Systems, Paul Schleifer, Yuan Sun, Dilip Patel, **2007**

[20] Benchmarking temporal database models with interval-based and temporal element-based time-stamping, Seo-Young Noh, Shashi K. Gadia, **2008**

[21] Formal Validation and Verification of Space Flight Software Using Statechart-Assertions and Runtime Execution Monitoring, Miriam C. Bergue Alves, Doron Drusinsky, James Bret Michael and Man-Tak Shing, **2011**

[22] Verification, validation and test with Model Based Design, Tom Erkkinen, Mirko Conrad, **2008**