## **Real-Time Database Systems Patterns**

The Brogrammers

Tugce Akin <tugce.akin@sjsu.edu>,

Navit Gaur <navit.gaur@sjsu.edu>,

Fabrizio Pisciotta <[fabrizio.pisciottasanabria@sjsu.edu](mailto:fabrizio.pisciottasanabria@sjsu.edu)>

Fall 2015

CMPE 202-02

**Table Of Contents**

Abstract

1. Introduction

2. Patterns

2.1 Transaction Stable Design Pattern(M)

2.1.1 Context

2.1.2 Problem

2.1.3 Solution

2.1.4 Application Mapping

2.1.5 Case Studies

2.2 Responsiveness Stable Analysis Pattern(S)

2.2.1 Context

2.2.2 Problem

2.2.3 Solution

2.2.4 Application Mapping

2.3 Duration Stable Design Pattern(S)

2.3.1 Context

2.3.2 Problem

2.3.3 Solution

2.3.4 Application Mapping

2.4 Criticality Stable Design Pattern(M)

2.4.1 Context

2.4.2 Problem

2.4.3 Solution

2.4.4 Application Mapping

2.5.5 Case Studies

3. Consequence

4. Appendix

**Abstract**

The motivation of the essay is to design explicitly define patterns for Real-Time Database Systems using Software Stability Model(SSM). These systems are getting more and popular everyday as the demand for real-time applications increase. So it’s crucial to come up with an enduring solution. We will utilize the concepts of Criticality, Duration, Responsiveness and Transaction to develop a more stable Real-Time Database System model. This will eliminate the need of rethinking a Real-Time Database problem every time from scratch.

**1. Introduction**

Traditionally, real-time systems manage their data (e.g. chamber temperature, aircraft locations) in application dependent structures. As real-time systems evolve, their applications become more complex and require access to more data. It thus becomes necessary to manage the data in a systematic and organized fashion. Database management systems provide tools for such organization, so in recent years there has been interest in merging database and real-time technology. The resulting integrated system, which provides database operations with real-time constraints is generally called a real-time database system (RTDBMS).

Like a conventional database system, a RTDBMS functions as a repository of data, provides efficient storage, and performs retrieval and manipulation of information. However, as a part of a real-time system, whose “tasks" are associated with time constraints, a RTDBMS, has the added burden of ensuring some degree of confidence in meeting the system's timing requirements.

Example applications that handle large amounts of data and have stringent timing requirements include telephone switching (e.g. translating an 800 number into an actual number), radar tracking and others. Arbitrage trading, for example, involves trading commodities in different markets at different prices. Since price discrepancies are usually short-lived, automated searching and processing of large amounts of trading information are very desirable. In order to capitalize on the opportunities, buy-sell decisions have to be made promptly, often with a time constraint so that the nancial overhead in performing the trade actions are well compensated by the benefit resulting from the trade.

This paper starts with a description of 4 different patterns in Section 2 which consists of Stable Analysis and Stable Design Patterns. In Section 3, we talk about the consequences of RTDBS. Finally, the paper concludes with a appendix in Section 4.

**Properties :**

Real-time database systems are the most promising alternative to manage the data with a structured and systematic approach. The **Timeliness** property is very important since it’s very important to complete the transactions a timely manner. Also the system should be efficient enough to handle all the data and the queries. So **Efficiency** is very much needed.

In every real time database system you expect every data to be accurate and critical for you. For example in a missile launch every data related to missile launch must be accurate and critical every second. Hence **Criticality** and **Accuracy** is an important factor.

There should always be a **Consistency** in your data otherwise that database is of no use most of the times. We describe most of these properties and many more in our document which follows.

**2. Patterns**

**2.1 Pattern Name: Transaction Stable Design Pattern(M)**

**2.1.1 Context :**

The **Goal** of this article is to understand the blissful way of designing a model using concepts of the knowledge map or the stability model. They are made of EBT’s (Enduring business themes), BO’s (Business objects) and IO’s (Industrial Objects).

Thus, the motivations behind this article are as below:

· Security

· Reliability

· Performance

In a Real-Time Database System all the transactions are important. So we try to come up with a Stable Design Pattern for all kind of Transactions i.e. **Any Transaction**. This would be applicable to all instances of Transaction and that is why it is called Stable Design Pattern. The BO described in this report is Any Transaction which is a Stable Design Pattern.

**Transaction** – Transaction is a Business Object which defines the sequence of operations. Transaction is an important element which determines the completeness of a real time database system. Transaction also helps us determine the performance of a system which is important.

**Scenario 1:** Consider a situation of a Bank i.e. **Party** (BO). Every **Transaction** (BO) is important for Bank. So keeping the transaction secure is their main goal. Since every transaction has many **Type** (BO) like money transfer or balance checking that leads to successful transaction i.e. **Outcome** (BO).All bank transactions are reflected in our bank account which can be checked online i.e. **Media** (BO) Also a bank keeps a record of all the transaction i.e. **Log** (BO). This results in a perfect **Trading** (EBT).

**Scenario 2:** Consider a student and professor i.e. **Actor** (BO). Here whatever information or knowledge they share with each other is a **Transaction**(BO). Every information exchanged has a **Type**(BO). They can exchange data using a flash or notes or a compact disk etc i.e. **Media**(BO). Also a student can maintain a activity **LOG**(BO) to keep track of all the data exchanged. This gives us a **Trading**(EBT) of data between students and professor.

The BO’s systematize properties for achieving the goal. Thus these properties makes design to ultimately achieve the EBT’s mentioned. The BO’s corresponding to the mentioned EBT are:

|  |  |
| --- | --- |
| 1. Any Actor | 2. Any Party |
| 3. Any Log | 4. Any Outcome |
| 5. Any Transaction | 6. Any Type |
| 7. Any Media | 8. Any Entity/Event |

Our pattern is independent of any domain and is applicable to any instance or application of Transaction.

**Known As:**

1 Occurrence Stable Design Pattern

2 Operation Stable Design Pattern

3 Performance Stable Design Pattern

**2.1.2 Problem:**

Problem includes designing a stable pattern which can be used by all the entities which require a real time database system to carry any transaction for achieving their desired outcome. The suggested pattern should be dynamic and versatile. What's more, it can be utilized on any structural engineering, idea or media. The proposed example ought to have the capacity to apply in any application regardless of any space which is the thing that makes the example steady and viable. The aim of developing such a pattern is to resolve the core issue of defining transaction for any kind of trading.

**There are 2 types of requirements:**

**Functional Requirements:**

This set of requirements defines what a system is supposed to do. The EBTs come at the very core of the system and represent the core knowledge of the system. EBTs being the ultimate business goals, we want to keep them constant even if we offer business in different ways over time. It also describes the BOs involved in the Stability Model. BOs form the concrete classes of the system and generally referred as “Workhorses” of the system.

**EBT Stable Design Patterns for our architecture:**

**Trading:** Trading is the process of carrying out transactions to successfully complete any operation or task. It is the process of carrying out sequence of transactions to produce an outcome that is secure and reliable. The process includes tracking the transaction, identifying new ones and evaluating transaction process effectiveness.

**The Related BOs are:**

**Any Actor:** Actor is outside the scope of the system. Actor is representing the users in system. It has four types: Software, Hardware, People, and Creature.

**Any Party:** Party is a collectible group who works in collaboration. It is the group which does the transaction.

**Any Log:** Log is a record of all the procedures taking place in the system. It is basically a log of something or the log of results obtained from it.

**Any Transaction:** Any Transaction is the occurrence of certain events or procedures that results into some outcome. It depends on an event, entity and actor.

**Any Media:** The outcomes and entities reside on a medium i.e. the means of communication.

**Any Type:** The various types of transactions i.e. of any category or things with common characteristics. So type defines the different characteristics of the related issue.

**Any Entity:** Entity is a participant of the complete system that the pattern represents.

**Any Event:** Any Event that poses exposure to threat.

**Any Outcome:** A measure or result of a transaction.

**Non-Functional Requirements**

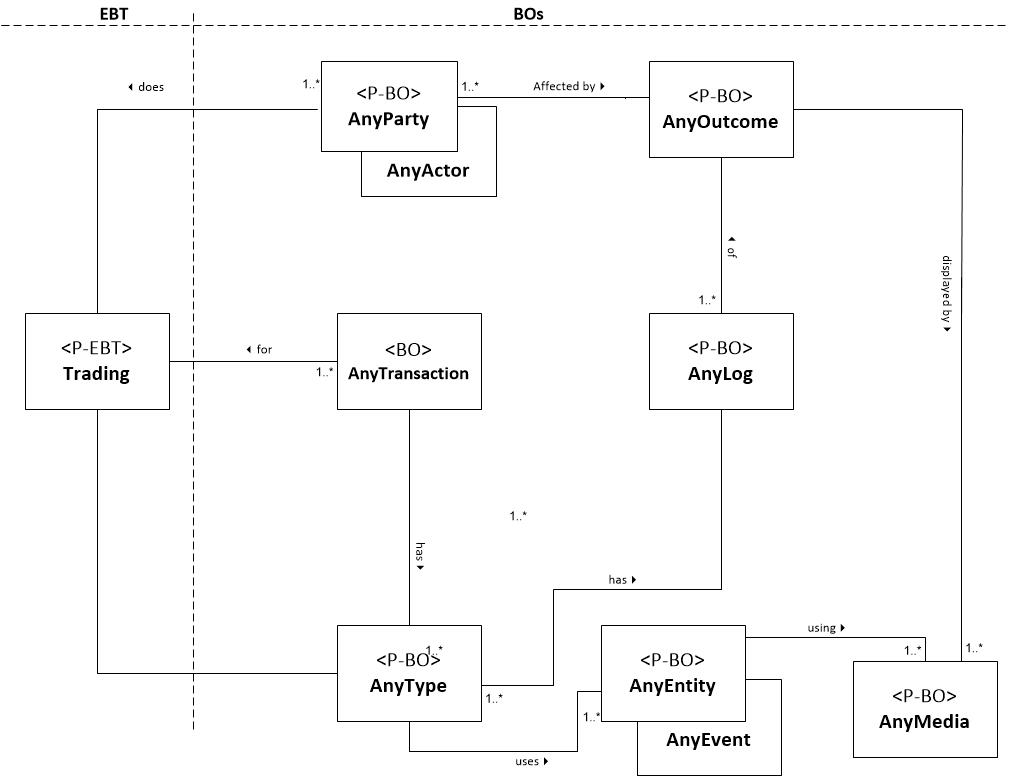
**Security:** A transaction has to be secure as per the extent. We need to measure how secure the transaction is to define its capabilities. For example, if we know how secure a transaction is, then we can use that in a Bank or in any business firm.

**Performance:** A transaction must define its performance. This defines how efficient it is to produce the desired outcome. The main focus is on the efficiency and the outcome of the transaction. For example in a spacecraft management system whenever we do a transaction or need any result of our query, then the transaction should be quick and produce the correct result, i.e. it should perform.

**Reliability:** Transactions should always be reliable. For example, suppose in a Bank there are lot of customers and a lot of transactions takes place daily for every customer. So every transaction should produce the same result for every customer. For example any Transaction of $500 should transfer an amount of $500 for all customers who initiate that transaction.

**2.1.3 Solution: Pattern Structure and participants**

Figure 1: Stable Design Pattern of Any Transaction



**Transaction Stable Design Pattern:**

|  |  |
| --- | --- |
| <P-EBT> | Patterns Enduring Business Theme |
| <P-BO> | Patterns Business Objects |
| <EBT> | Enduring Business Themes |
| <BO> | Business Objects |

**Description:**

1. Transaction produces an outcome and is logged properly.

2. Trading uses a mechanism that produces a transaction.

3. Transaction can be of any type that uses any entity or event.

4. Any Actor or any party is affected by the outcome of the transaction.

**Participants:**

Trading: Trading is the process of carrying out transactions to successfully complete any operation or task.

Any Actor/Any Party: Are the authorized participants who will initiate the transaction.

Any Type: The Transaction can be of any type or any category.

Any Entity/Event: Are the resource around which the transaction of Any Actor/Any Party takes place.

Any Transaction: Any Transaction is the sequence of operations that take place. It depends on an event, entity and actor.

Any Outcome: A measure or outcome of a transaction.

Any Log: Log is the measure of records of the transaction. It is basically a record or log of all the transactions.

Any Media: Media is the source on which actions are described or recorded.

**2.1.4 Application Mapping using the Pattern**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EBT | BO | Bank  Transaction-IO | ATM Transaction-  IO | E-Trading  Transaction - IO | Employee Time Clock System - IO | E-commerce-  IO |
| Trading | AnyTransaction | Account Opening | Cash Withdrawal | Stock Exchange | Time Monitoring | Shopping |
| Trading | AnyParty | Bank Of America,Citi Bank | Wells Fargo | Bombay Stock Exchange | Cisco,  T-Mobile | Ebay |
| Trading | AnyActor | Customer,  Manager | Customer,  Service Provider | Person, Broker | Employee,  Receptionist | Customer,  Sales Person |
| Trading | AnyOutcome | Successful, Failure | Success,  Time Out | Money, Shares | Hours or Duration | Purchase |
| Trading | AnyType | Money Transfer | Balance Check | Stocks, Bonds | Time Capture | Money, Cash |
| Trading | AnyEntity | Money | Cheque | Stocks | Swipe Card | Items |
| Trading | AnyEvent | Transfer | Withdrawal | Shares Crashing | Capturing | Shopping |
| Trading | AnyMedia | Computer | ATM Machine | Internet | Display Screen | Internet |
| Trading | AnyLog | Record Files | ATM Receipt | Stock Record | Employee Records | Shopping Cart |

**2.1.5 Case Studies**

**Case Study 1: Bank Transaction Failure**

After selling his house, John transferred the proceeds from the current account he held in his sole name to a savings account that he asked the bank to open for him and his sister, Sara.

John had previously been bullied and intimidated by a guy named Sam, who had - over time - persuaded John to pay him substantial amounts of money. It was to try and prevent a recurrence of this situation that John and his sister asked for the savings account to be set up so that both of them had to sign for all withdrawals.

Unfortunately, however, the bank ignored these instructions. John and his sister were each able to withdraw money from the account using just their own signature. And in less than a year, John had withdrawn and paid over to Sam some $11,000. In Sara’s view, this had only been possible because of the bank's error. The bank disagreed with this.



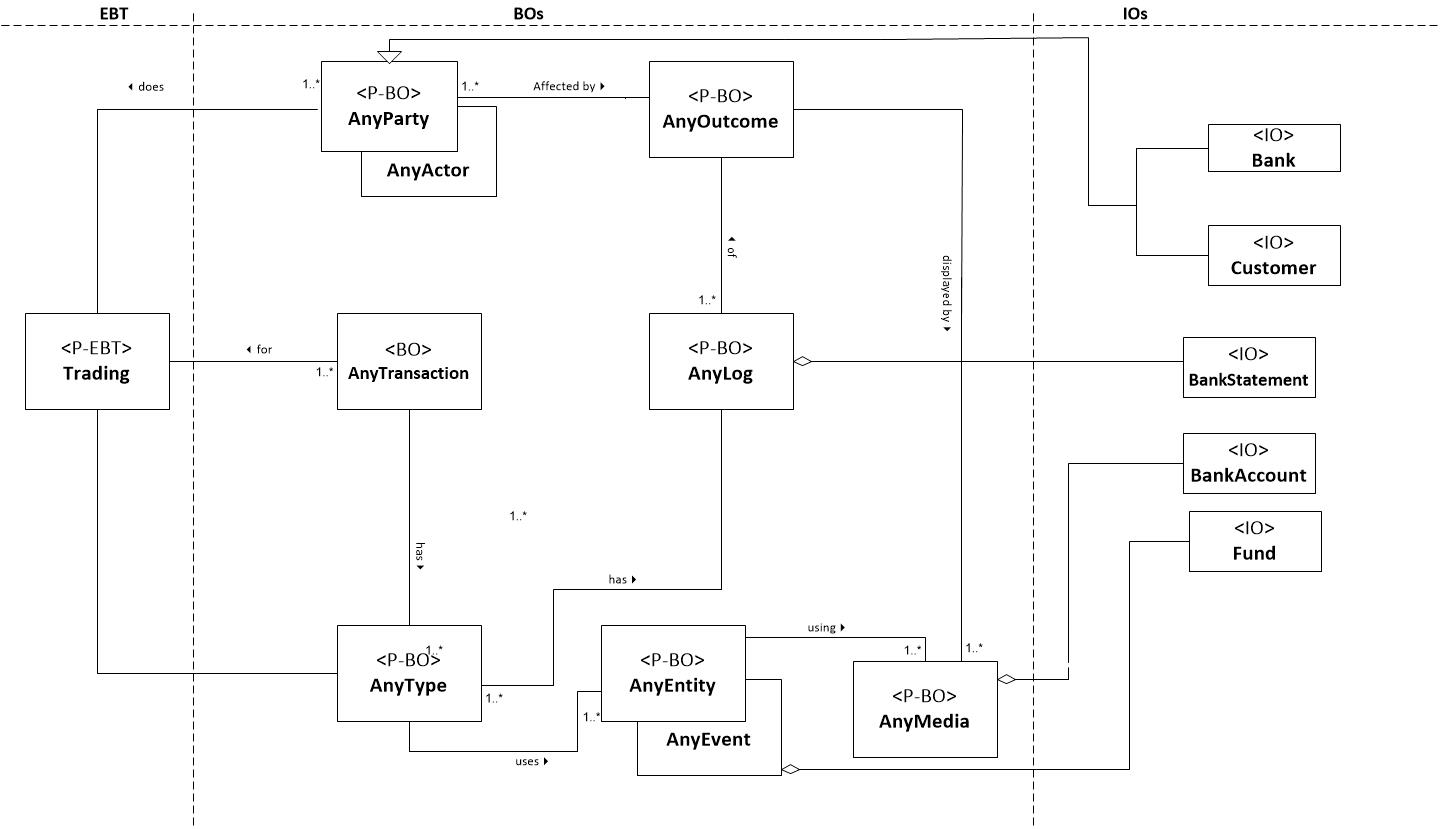
It was not in dispute that the bank should have set up the account so that both Sara and John had to sign for any withdrawals. However, the bank said that the money in the account belonged to John and he was entitled to pay it over if he wanted to - which is exactly what he had done.

**a. Case Studies:**

The Bank was criticised for:

1. Transacting money without Sara’s signature.
2. Making such a big error of ignoring Sara’s role
3. The money John withdraw was of no good use to him.

**b. Class diagram**



**c. Use Cases:**

1. Authenticate all the Transactions

Actor : John, Sara

2. Re-Verify every Transaction

Actor : John , Sara

**d. Use Case Description :**

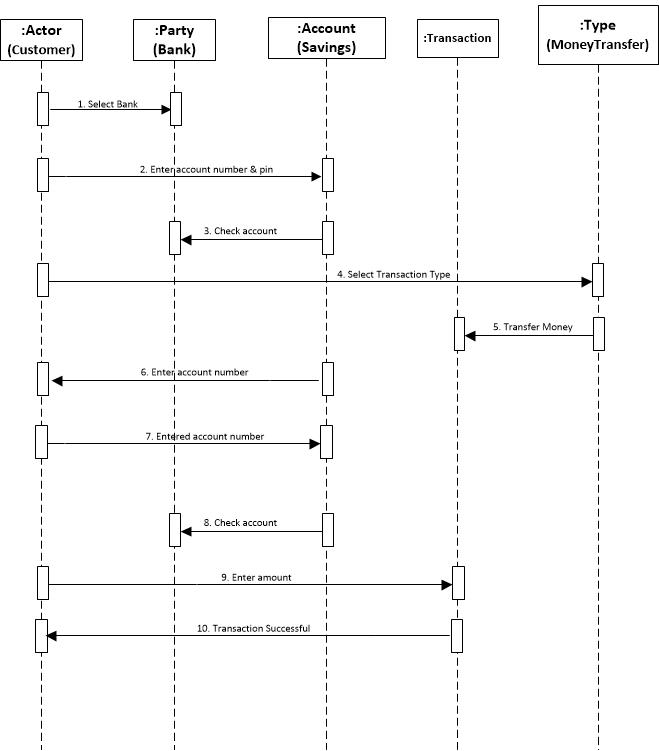
**Use Case 1: Authenticate all Transactions**

1. Bank should authenticate every transaction
2. All the details of every actor i.e. customer should be checked
3. Although the bank had been correct in noting that John had made the withdrawals himself, he had received no benefit from the money.
4. The Bank should have understood the significance of the request.
5. Bank should re-work the savings account as though the disputed withdrawals had not been made.
6. Every authenticated transaction needs both the customers approval.

**Use Case 2 : Re-Verify all Transactions**

1. All customer should reverify all the transaction they make.
2. There should be a common mechanism for verification.
3. Every transaction should have a buffer time of 1 or 2 days to be completed.
4. Bank representatives log the details of transaction
5. Bank should re-work the savings account as though the disputed withdrawals had not been made
6. Bank should reverify the transaction to avoid errors and inconvenience caused to customers.

**e. Sequence Diagram**



**Case Study 2: ATM Money Transaction**

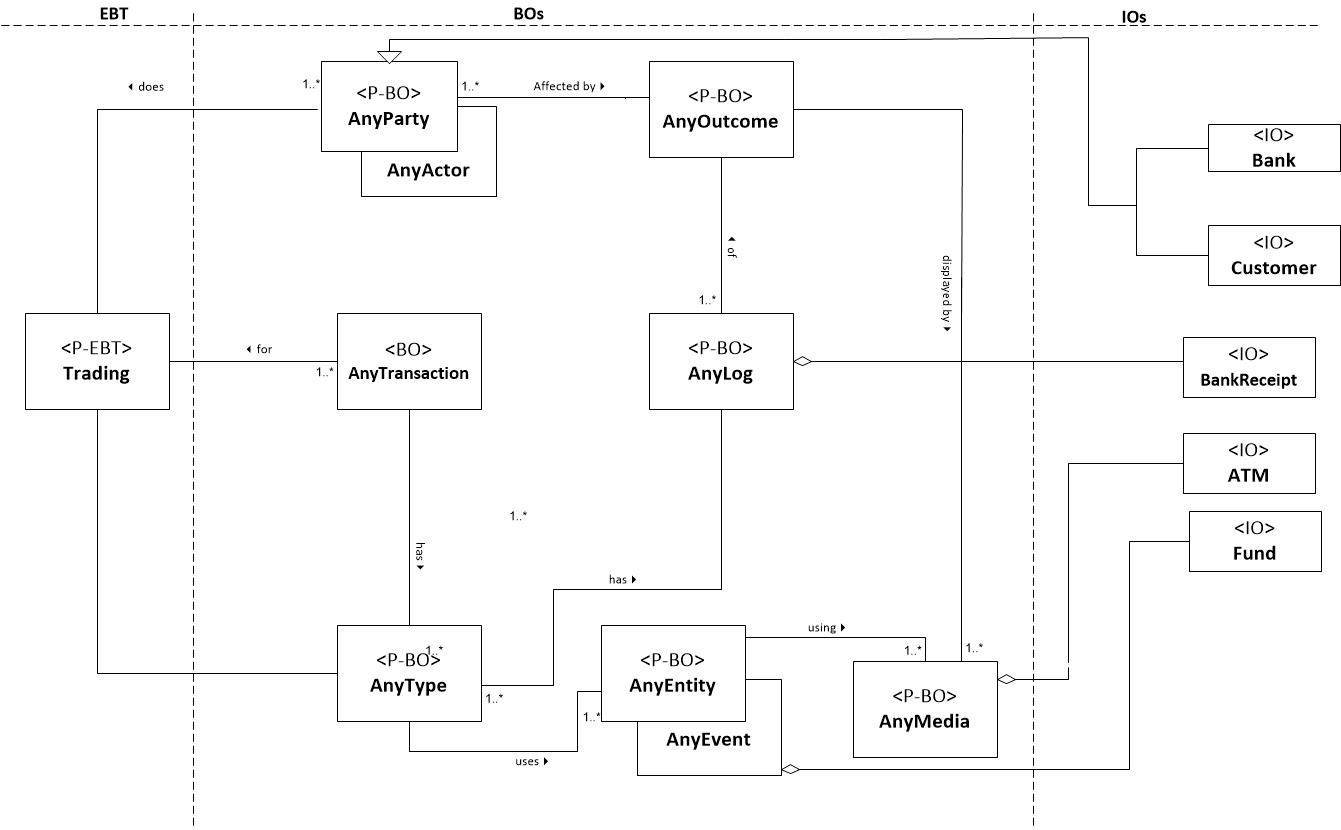
The ATM service was first introduced in Pakistan by Habib Bank Limited in 1987. The ATM infrastructure in Pakistan is developed, managed and supported by Commercial Banks, Microfinance Institutions, and by 3rd party commonly known as Services Providers. ATMs are owned, operated, deployed and managed by Commercial Banks established under the Banking Companies Ordinance 1962 and licensed by the State Bank of Pakistan (the Central Bank). In order to provide an easy and quick access to card holders, ATMs have

been placed at convenient locations called on-site locations (near or inside the bank branch premises) and off-site locations (shopping malls, universities, colleges, gas stations etc).



In ATM transactions, the role of Issuer and Acquirer bank occupies an important position. For example a banking company in Pakistan can become an acquirer or issuer or both for offering e-banking products and services to consumers. Banks issuing payment card such as Debit, Credit, Prepaid Debit or Remittance Cards are called issuing banks and they are not required to deploy ATMs or POS terminals. Payment cards issued by issuer bank can easily be used on any ATM or POS terminals owned and managed by another bank in Pakistan called acquirer bank. Most of the banks in Pakistan are both issuer as well as acquirer as the case with ACB Bank Limited.

**b. Class diagram**



**c. Use Cases:**

1. Money withdrawal from ATM

Actor: Bank, Bank Customer

2. Transaction Use Case

Actor : Bank , Customer

**d. Use Case Description :**

**Use Case 1: Money withdrawal from ATM**

1. The use case begins when Bank Customer inserts their Bank Card.

2. Use Case: Validate User is performed.

3. The ATM displays the different alternatives that are available on this unit. [See Supporting Requirement SR-xxx for list of alternatives]. In this case the Bank Customer always selects "Withdraw Cash".

4. The ATM prompts for an account. See Supporting Requirement SR-yyy for account types that shall be supported.

5. The Bank Customer selects an account.

6. The ATM prompts for an amount.

7. The Bank Customer enters an amount.

8. Card ID, PIN, amount and account is sent to Bank as a transaction. The Bank Consortium replies with a go/no go reply telling if the transaction is ok.

9. Then money is dispensed.

10. The Bank Card is returned.

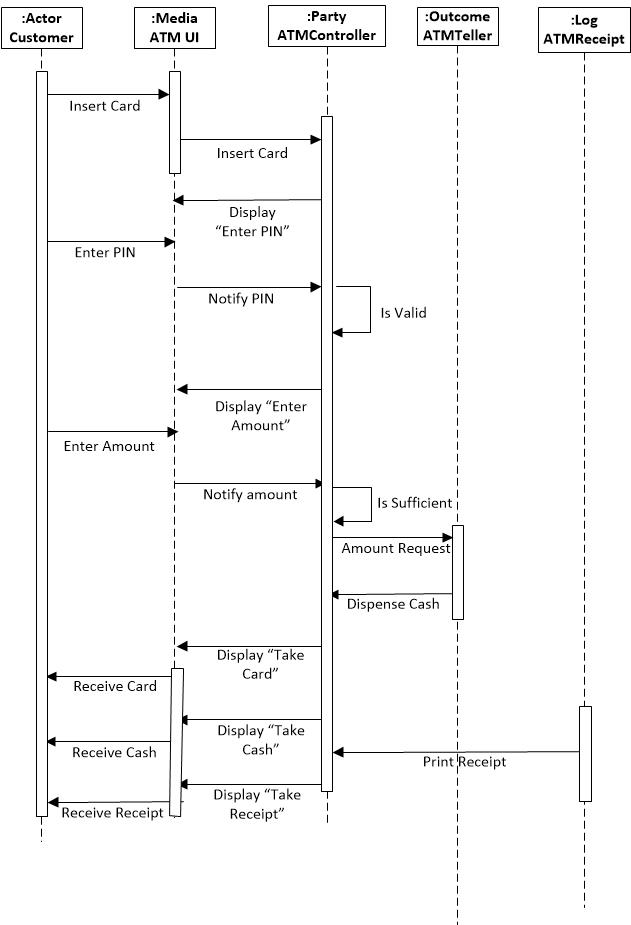
11. The receipt is printed.

12. The use case ends successfully.

**Use Case 2: Transaction Use Case**

1. Customer chooses a transaction type from a menu of options.
2. The transaction is sent to bank along with information from customer’s card and the PIN the customer enters.
3. If the bank approves the transaction, any steps needed to complete the transaction will be performed
4. A receipt of the transaction is generated.
5. If the bank reports that the customer’s PIN is invalid, Invalid PIN extension is performed and attempt is made to continue the transaction.
6. Customer can cancel a transaction by pressing a Cancel key.

**e. Sequence Diagram**



**2.2 Pattern Name: Responsiveness Stable Analysis Pattern (S)**

**2.2.1 Context**

Responsiveness is an enduring business theme which is based on how quickly a system reacts to requests and provides answers. Responsiveness is one of the most crucial elements that determines the quality of a real-time database system.

**Scenario 1**

For example let’s think of a spacecraft management system. **AnyMechanism** controls the overall operation of the spacecraft. In addition to the this it also constantly monitors the general health of the machine. It receives commands from the ground computer. **AnyTransaction** should be handled fast. **AnyData** volume may not be that high, but performance and timing attributes are very important. Not handling these attributes well causes unresponsiveness which might lead fatal results.

**Scenario 2**

This pattern can also be applied to real-time reservation systems. Airline reservation systems are used by thousands of **AnyActor**/**AnyParty**’s simultaneously. If the system becomes unresponsive due to high number of **AnyTransaction**, it may hurt the company’s reputation and cause loss of revenue.

**2.2.2 Problem**

There are some requirements that needs to be followed in order to ensure responsiveness. Ensuring responsiveness prevents application freezes which is one of the main problems of real-time applications including network management systems, real-time chat programs or spacecraft management systems.

**2.1 Functional Requirements**

This set of requirements defines what a responsive system should do. The EBTs are the ultimate goals and BOs form the concrete classes of the system.

**EBT Stable Analysis Patterns for our architecture:**

**Responsiveness:** It is the ability to react quickly to requests and provide an answer in real time**.** This systems should be responsive to provide a better user experience.

**The Related BOs are:**

**AnyActor:** Represents the users of the system and it is not in the scope of it. Actor is outside the scope of the system. They have an ability to get/update data through different kinds of requests.

**AnyParty:** Party is a collectible group who works together. It is the group which needs responsibility.

**AnyData:** It is all instances of data that is kept and transferred in the system. If it takes time to get data, systems could become unresponsive.

**AnyMechanism:** A Mechanism is natural or established process by which something takes place or is brought about. SQL and NoSQL are the primary data storage mechanisms. The most suitable one should be chosen. Each situation is different since amount and type of data is different. Each mechanism has its own advantages.

**AnyTransaction:** Any Transaction is the occurrence of certain events or procedures that results into some outcome. It depends on an event, entity and actor.

**AnyMedia:** The outcomes and entities reside on a medium i.e. the means of communication.

**AnyEntity:** Entity is a participant of the complete system that the pattern represents.

**AnyEvent:** Anything that takes place that is about responsiveness.

**AnyLog**: They are text files that stores database transaction information. They should be saved.

**2.2 Non-Functional Requirements**

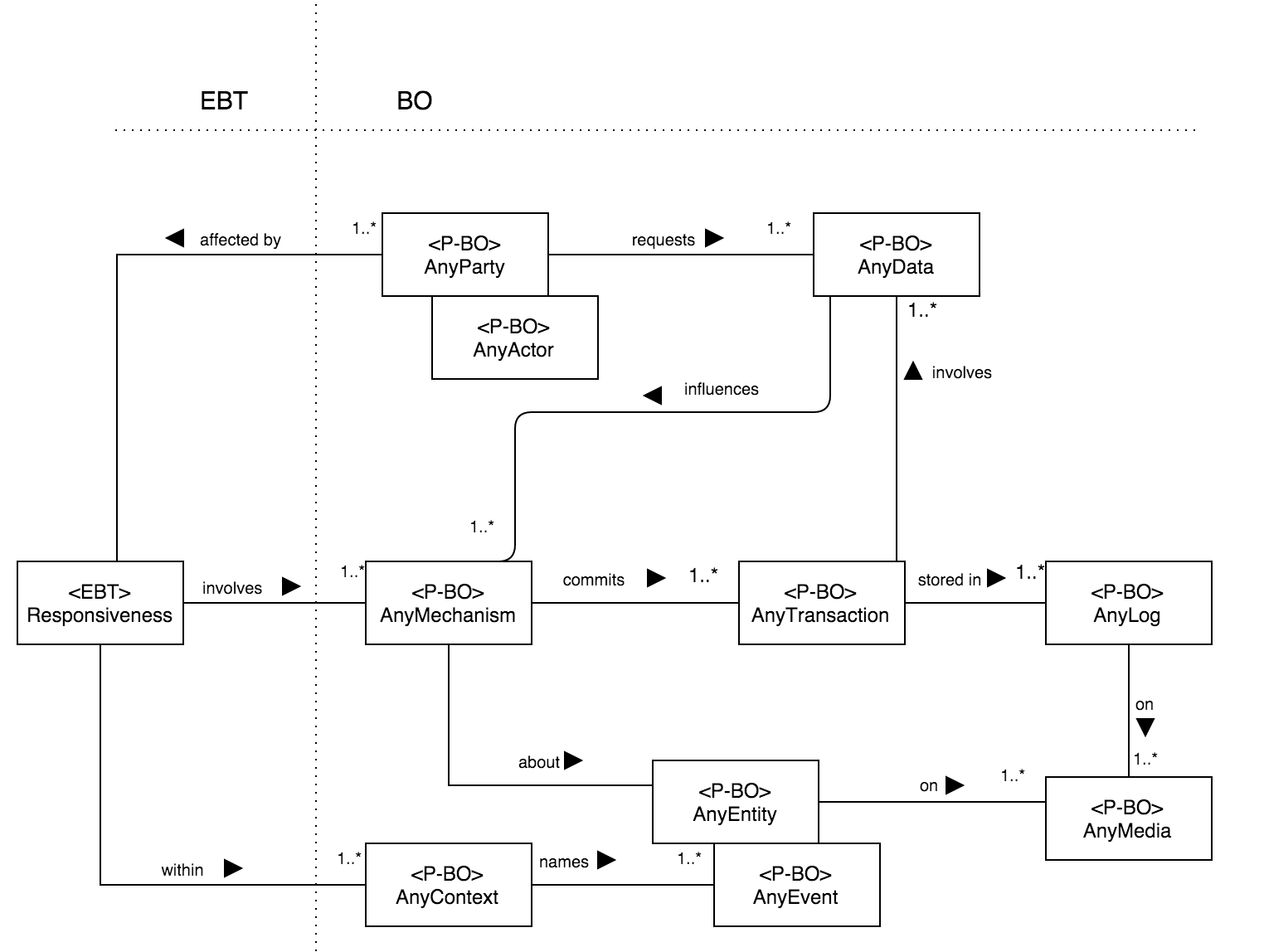
**Reliability:** It is the quality of being reliable. Reliability deals with consistency and quality of responsive systems.

**Completeness:** The pattern should be able to present the state and rules according to the system in which it is applied.

**Speed:** It is expected that the system is fast enough for the result to come back and be acted on right away.

**2.2.3 Solution**

The following model represents a representation of the solution that deals with the Responsiveness theme:



**Figure 2:** Responsiveness Stable Analysis Pattern.

**Responsiveness Stable Design Pattern:**

|  |  |
| --- | --- |
| <P-EBT> | Patterns Enduring Business Theme |
| <P-BO> | Patterns Business Objects |
| <EBT> | Enduring Business Themes |
| <BO> | Business Objects |

**Description:**

**1**. Responsiveness refers to the ability to react quickly to requests and provide answers in real time.

**2**. Responsiveness affects AnyActor/AnyParty.

**3**. Responsiveness involves AnyMechanism.

**4.** Responsiveness is within AnyContext.

*Participants:*

* *AnyParty/AnyActor:* The authorized participants who request or sends data. They are directly effected by responsiveness.
* *AnyData*: An information piece that is requested by AnyParty/AnyActor. Carried by AnyTransaction.
* *AnyMechanism*: Responsiveness involves AnyMechanism and AnyData influences AnyMechanism. It is about AnyEntity/AnyEvent.
* *AnyContext*: Responsiveness is within AnyContext. AnyContext names AnyEntity/AnyEvent.
* *AnyEntity*: Is on AnyMedia. It represents anything that exists on the database.
* *AnyEvent*: Is on AnyMedia. It represents a database event that takes place.
* *AnyMedia*: Is any kind of medium that data is sent or received. AnyEntity/AnyEvent and AnyLog is kept on AnyMedia.
* *AnyLog*: Is on AnyMedia. It saves information about AnyTransaction.
* *AnyTransaction*: AnyTransaction information is kept in AnyLog. It has data to be processed.

**2.2.4 Application Mapping**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EBT** | **BO** | **Network Management System - IO** | **Spacecraft Management System** | **Messaging Application** | **Airline Reservation** | **Real-Time Stock Software** |
| **Responsiveness** | **AnyActor/AnyParty** | Network Administrator, Internet User | Scientist | Smartphone User | Customer | Banker |
| **Responsiveness** | **AnyTransaction** | Getting Latency Data | Sensor Data Gathering | Message Delivery | Reservation Confirmation | Data Request |
| **Responsiveness** | **AnyContext** | Internet Networks | Spacecraft Management | Messaging Systems | Airline Reservation Systems | Real-Time Stock Systems |
| **Responsiveness** | **AnyMechanism** | NetworkManagement | Spacecraft Management | Messaging Management | Reservation Management | Stock Management |
| **Responsiveness** | **AnyData** | Bandwidth Info | Activity Data | Text | Reservation Info | Stock Detail |
| **Responsiveness** | **AnyLog** | Packet Record | Machine Health Record | TextRecord | Transaction Record | Transaction Record |
| **Responsiveness** | **AnyEntity** | Router | Spacecraft | Message | Reservation | Stock |
| **Responsiveness** | **AnyEvent** | Monitoring | Monitoring | Sending Message | Searching | Updating |

**2.3 Pattern Name: Duration Stable Design Pattern (S)**

**2.3.1 Context**

Measurement is the assignment of a number to a characteristic of an object or event, which can be used to be compared with other objects or events. Measurement is a cornerstone of [trade](https://en.wikipedia.org/wiki/Trade), [science](https://en.wikipedia.org/wiki/Science), [technology](https://en.wikipedia.org/wiki/Technology), and [quantitative research](https://en.wikipedia.org/wiki/Quantitative_research) in many disciplines. Historically, many [measurement systems](https://en.wikipedia.org/wiki/System_of_measurement) existed for the varied fields of human existence to facilitate comparisons in these fields. Often these were achieved by local agreements between trading partners or collaborators.

In Computer Science some common units of measurement are, bit, which is the smallest unit of information used by computers to transfer and process data. Byte, which is the base unit of information for every computation. In software engineering, function point is a unit of measurement to express the amount of business functionality an [information system](https://en.wikipedia.org/wiki/Information_system) (as a product) provides to a user. Another unit of measurement is second or hours, which is used to measure a system development duration or time of execution.

**2.3.2 Problem**

Software quality has become very important in the last decades given that system are more and more complex every day. That’s why different techniques and standards have appear to measure it and they are still under development to improve measurement techniques.

**2.1 Functional Requirements**

**Specification:** a good system specification is a key to measure software quality.

**Transaction**: every operation in the system should be consider as a transaction, so it can be identified.

**Log:** the system should log every transaction, so it can be traceable.

**2.2 Non-Functional Requirements**

**Quantifiability:** the system should be quantifiable in terms of time, resources, lines of code, etc.

**Traceability:** every transaction should be stored. It allows to trace and measure every operation that is made on the system.

**Quality**: system should pursue excellence. Every measurement should be made in pursue of quality.

**2.3.3 Solution**

The following model represents a representation of the solution that deals with the modernity theme:



**Figure 3:** Duration Stable Design Pattern.

*Participants:*

* *AnyParty/AnyActor:* any user can measure software, using *AnyMechanism*, in terms of time, function points or resources employed.
* *Quality*: Measurement determines the quality of a system and *Quality* involves *AnyMechanism*.
* *AnyMechanism*: it initiates *AnyTransaction* and it is about *AnyEntity/AnyEvent*.
* *Duration*: Measurement measures the duration of a system during any given period of time.
* *AnyTime*: any given time period that involves *AnyType*.
* *AnyMechanism*: *AnyParty/AnyUser* uses *AnyMechanism* and it is about *AnyEntity/AnyEvent*.
* *AnyType*: any type names *AnyEntity/AnyEvent*.
* *AnyEntity/AnyEvent*: it resides on *AnyMedia*.
* *AnyTransaction*: it records *AnyLog*.
* *AnyLog*: it resides on AnyMedia.
* *AnyMedia*: it is any resource used to store system data.

**2.3.4 Application Mapping**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EBT** | **BO** | E-commerce Application | Social Media Application | Operative System | Airline Reservation | Real-Time Stock Software |
| **Measurement** | **Quality** | Number of requests per second | Availability | Throughput | Relative Efficiency | Scalability |
| **Measurement** | **Duration** | Time of response per request | Latency | Cpu processes per hour | Ticket reservation process time. | New Stock registration average time |
| **Measurement** | **Size** | Catalog size | Social media users | Files created | Tickets issued | Stock registrations |
| **Measurement** | **Weight** | Bandwidth load | Power consumption | Compression ratio | Performance per watt | Environmental impact |

**2.4 Pattern Name: Criticality Stable Analysis Pattern (M)**

**2.4.1 Context**

The **Goal** of this article is to understand the blissful way of designing a model using concepts of the knowledge map or the stability model. They are made of EBT’s (Enduring business themes), BO’s (Business objects) and IO’s (Industrial Objects).

Thus, the motivations behind this article are as below:

· Security

· Reliability

· Effectiveness

In a Real-Time Database System Criticality of all the data is important. So we try to come up with a Stable Analysis Pattern for **Criticality.** This would be applicable to all the things which can be critical and that is why it is called Stable Analysis Pattern. The EBT described in this report is **Criticality** which is a Stable Analysis Pattern.

**Scenario 1:**

Consider the situation of Fukushima Daichi Nuclear disaster. The Government is a **Party**(BO) and the people or citizen are **Actor**(BO). The government tried to a nuclear test using any **Mechanism**(BO) but it failed. The **Data**(BO) was not complete and the **Rules**(BO) for carrying out the test were not well defined. Hence missing **Criticality**(EBT) lead to Nuclear Disaster.

**Scenario 2:**

Consider the case of UK air traffic control failure. The Computer **System**(BO) failure caused cancelled flights in UK. National Air Traffic Services i.e. **Party**(BO) stated that the problems occurred as more workstations were being brought online i.e. **Mechanism**(BO) to deal with traffic increase. Computer failure made it impossible to access **Data**(BO) . All the departing flights were delayed for 35 minutes So this is how you get to know the importance of **Criticality**(EBT) or how important and sensitive every information is.

**2.4.2 Problem**

Problem includes designing a stable pattern which can be used by all the entities which require a real time database system for critical data to achieve their desired outcome. The suggested pattern should be dynamic and versatile. What's more, it can be utilized on any structural engineering, idea or media. The proposed example ought to have the capacity to apply in any application regardless of any space which is the thing that makes the example steady and viable. The aim of developing such a pattern is to resolve the core issue of defining criticality for any kind of trading.

There are 2 types of requirements:

**Functional Requirements**

This set of requirements defines what a system is supposed to do. The EBTs come at the very core of the system and represent the core knowledge of the system. EBTs being the ultimate business goals, we want to keep them constant even if we offer business in different ways over time. It also describes the BOs involved in the Stability Model. BOs form the concrete classes of the system and generally referred as “Workhorses” of the system.

**EBT Stable Analysis Patterns for our architecture:**

**Criticality:** Criticality is the quality, state, or degree of being of the highest importance. It is the state of being urgent, an earnest and insistent necessity.The process includes tracking the importance of all the data in the system. Also, knowing what type of system needs what kind of data is important too.

**The Related BOs are:**

**Any Actor:** Actor is outside the scope of the system. Actor is representing the users in system. It has four types: Software, Hardware, People, and Creature.

**Any Party:** Party is a collectible group who works in collaboration. It is the group which needs critical data.

**Any Data:** It is all instances of data which can be critical in our system.

**Any Mechanism:** A Mechanism is natural or established process by which something takes place or is brought about. It can be considered as group of parts having a particular function.

**Any Transaction:** Any Transaction is the occurrence of certain events or procedures that results into some outcome. It depends on an event, entity and actor.

**Any Media:** The outcomes and entities reside on a medium i.e. the means of communication.

**Any System:** All kinds of scenarios we can think about that are critical .

**Any Entity:** Entity is a participant of the complete system that the pattern represents.

**Any Event:** Any Event that poses exposure to threat.

**Any Rule:** A set of rules defined in the system .

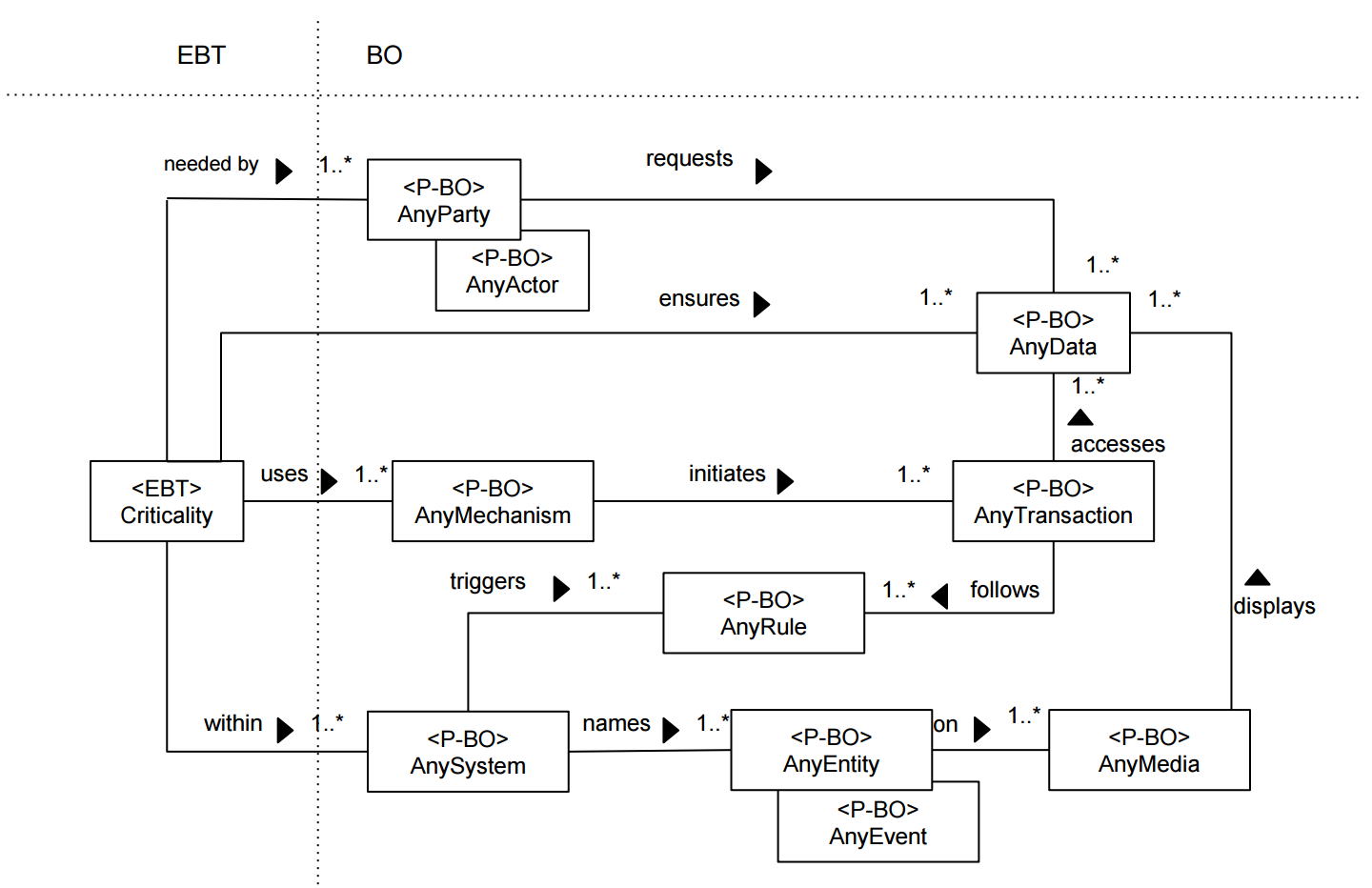
**Non-Functional Requirements**

**Security:** Anything that is critical should be secure as per the extent. We need to measure how secure the critical data is to define its capabilities. For example, if we know how secure a critical data is, then we can use that in a Bank or in any business firm.

**Effectiveness:** All the critical information should be effective. This defines how effective it is to produce the desired outcome. The main focus is on the effectiveness and the outcome of the transaction. For example in a spacecraft management system whenever we demand critically or need any result of our query, then the result should be effective and produce the correct result, i.e. it should be effective.

**Reliability:** Anything that is critical should always be reliable. For example, suppose in a Bank there are lot of customers and a lot of transactions takes place daily for every customer. So every transaction has data that is critical to them, hence should be stored in a reliable database.

**2.4.3 Solution**



**Figure 4:** Criticality Stable Analysis Pattern.

**Criticality Stable Design Pattern:**

|  |  |
| --- | --- |
| <P-EBT> | Patterns Enduring Business Theme |
| <P-BO> | Patterns Business Objects |
| <EBT> | Enduring Business Themes |
| <BO> | Business Objects |

**Description:**

**1**. Criticality refers to the importance of AnyData for processing AnyTransaction.

**2**. Criticality is needed by AnyActor/AnyParty.

**3**. It uses AnyMechanism which initiates AnyTransaction and ensures AnyData.

**4.** It is within AnySystem (MySql systems, NoSql systems, etc.).

*Participants:*

* *AnyParty/AnyActor:* The authorized participants who request or sends data. Criticality is needed by them for data ensurity.
* *AnyData*: An information piece that is requested by AnyParty/AnyActor. AnyMechanism ensures validity and consistency of AnyData.
* *AnyRule*: It is one of a set of principles that controls database tasks. AnyRule is triggered by the system and followed by AnyTransaction.
* *AnyMechanism*: Criticality uses AnyMechanism to ensure validity and consistency. AnyMechanism initiates AnyTransaction process and ensures AnyData.
* *AnySystem*: Criticality is within AnySystem. It triggers rules to ensure AnySystem names AnyEntity/AnyEvent.
* *AnyEntity*: Is on AnyMedia. It represents anything that exists on the database.
* *AnyEvent*: Is on AnyMedia. It represents a database event that takes place.
* *AnyMedia*: AnyMedia displays AnyData. It refers to any kind of medium that data is sent or received. AnyEntity/AnyEvent is kept on AnyMedia.
* *AnyTransaction*: AnyTransaction accesses AnyData to process. It follows AnyRule that is defined for consistency.

**2.4.4 Application Mapping**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **EBT** | **BO** | **Nuclear Measurement System - IO** | **Flight Management System - IO** | **Earthquake Early Warning System - IO** | **Heart Surgery System - IO** | **Driverless Car System - IO** |
| **Criticality** | **AnyActor/AnyParty** | Scientist | Pilot, Engineer | Scientist, Government | Surgeon | User |
| **Criticality** | **AnyTransaction** | Nuclear Data Gathering | Flight Plan Calculation | Wave Tracking | Heart Rate Calculation | Distance Calculation |
| **Criticality** | **AnySystem** | SQLServer | Oracle8I | MySQL | SQLServer | NoSQL System |
| **Criticality** | **AnyMechanism** | Detector Hardware | GPS Sensor | Wave Sensor | Heart Rate Sensor | Detector Hardware |
| **Criticality** | **AnyData** | Nuclear Measurement Data | Waypoint Info | WaveInfo | PatientData | Trafficlight Info |
| **Criticality** | **AnyRule** | Radiation Limit | Distance Limit Between Planes | Disregard Earthquakes(Low magnitude) | Medical Alert | Positioning Alert |
| **Criticality** | **AnyEntity** | Energy | Plane | Wave | Heart | Car |
| **Criticality** | **AnyEvent** | Enery Conversion | Flight | Earthquake | Surgery | Driving |
| **Criticality** | **AnyMedia** | Measurement Software | Management Software | Alert Machine | Device Screen | Device Screen |

**2.4.5 Case Studies**

**Case Study: Fukushima Daichi Nuclear disaster**

The Fukushima Daichi Nuclear disaster is still an ongoing energy accident at the Fukushima I Nuclear Power Plant initiated by the tsunami on 11 March, 2011. The damage caused by the tsunami produced [equipment failures](https://en.wikipedia.org/wiki/Nuclear_and_radiation_accidents#Equipment_failure), and without this equipment a [loss-of-coolant accident](https://en.wikipedia.org/wiki/Loss-of-coolant_accident) followed with three [nuclear meltdowns](https://en.wikipedia.org/wiki/Nuclear_meltdown) and releases of radioactive materials beginning on 12 March. It is the largest nuclear disaster since the [Chernobyl disaster](https://en.wikipedia.org/wiki/Chernobyl_disaster) of 1986 and the second disaster (after Chernobyl) to be given the Level 7 event classification of the [International Nuclear Event Scale](https://en.wikipedia.org/wiki/International_Nuclear_Event_Scale).

The plant comprised six separate [boiling water reactors](https://en.wikipedia.org/wiki/Boiling_water_reactor). At the time of the earthquake, reactors 4, [5 and 6](https://en.wikipedia.org/wiki/Fukushima_Daiichi_Nuclear_Power_Plant#Nuclear_disaster_of_March_2011) were [shut down](https://en.wikipedia.org/wiki/Shutdown_(nuclear_reactor)) in preparation for re-fueling. However, their [spent fuel pools](https://en.wikipedia.org/wiki/Spent_fuel_pool) still required cooling. Immediately after the earthquake, the electricity producing reactors 1, 2 and 3 automatically shut down their sustained [fission reactions](https://en.wikipedia.org/wiki/Fission_reaction), inserting [control rods](https://en.wikipedia.org/wiki/Control_rod) in what is termed a [SCRAM](https://en.wikipedia.org/wiki/SCRAM). Following this legally mandated "safety precaution" which ceases the reactors' normal running conditions, the reactors were unable to generate power to run their own coolant pumps. Emergency diesel generators came online, as designed, to power electronics and coolant systems, all of which operated right up until the tsunami destroyed the generators for reactors 1–5 due to their location in unhardened low-lying areas. The two generators cooling reactor 6 were undamaged and were sufficient to be pressed into service to cool the neighboring reactor 5 along with their own reactor, averting the overheating issues that reactor 4 suffered.

There have been no [fatalities linked to short term overexposure to radiation](https://en.wikipedia.org/wiki/Acute_radiation_syndrome) reported due to the Fukushima accident, while approximately 18,500 people died due to the earthquake and tsunami. However approximately 610 are estimated to have died due to workers' exposure and the evacuation of residents near the power plant. Estimates of the total human fatalities caused by the nuclear accident are up to 10,000, maximum cancer mortality and morbidity is calculated to be 1500 respectively 1800. In Addition to that the rates of mental illnesses among evacuated people rose fivefold compared to the Japanese average.

On 5 July 2012, the [Japanese National Diet](https://en.wikipedia.org/wiki/National_Diet) appointed [The Fukushima Nuclear Accident Independent Investigation Commission](https://en.wikipedia.org/wiki/Fukushima_Daiichi_nuclear_disaster#Fukushima_Nuclear_Accident_Independent_Investigation_Commission) (NAIIC) submitted its inquiry report to the Japanese Diet.The Commission found the nuclear disaster was "manmade", that the direct causes of the accident were all foreseeable prior to 11 March 2011. The report also found that the Fukushima Daiichi Nuclear Power Plant was incapable of withstanding the earthquake and tsunami.

Following the [1999 Tokaimura criticality accident](https://en.wikipedia.org/wiki/Tokaimura_nuclear_accident#In_1999), there was interest in Japan for developing radiation-resistant robots for use in the event of nuclear accidents. The Japanese government budgeted 3 billion yen (US $38 million) for [research and development](https://en.wikipedia.org/wiki/Research_and_development). Several companies produced [state of the art](https://en.wikipedia.org/wiki/State_of_the_art) prototypes in 2001, which were tested and deemed technical successes. In December 2002, a task force (which included TEPCO executives) further concluded that the robots were unnecessary: the possibility of Chernobyl-scale disasters was completely discounted and it was thus assumed that human employees- compared to whom the robots had limited speed and range- would still be able to operate in the event of an accident. The program halted, and the prototypes remained in storage until March 2006; some were subsequently donated to [Tohoku University](https://en.wikipedia.org/wiki/Tohoku_University). The termination of the program left Japan without functional radiation-resistant robots to send into Fukushima when the crisis began.

Three investigations into the Fukushima disaster showed the man-made nature of the catastrophe and its roots in [regulatory capture](https://en.wikipedia.org/wiki/Regulatory_capture) associated with a "network of corruption, collusion, and nepotism."Regulatory capture refers to the " situation where regulators charged with promoting the public interest defer to the wishes and advance the agenda of the industry or sector they ostensibly regulate." Those with a vested interest in specific policy or regulatory outcomes lobby regulators and influence their choices and actions. Regulatory capture explains why some of the risks of operating nuclear power reactors in Japan were systematically downplayed and mismanaged so as to compromise operational safety.

**a. Case Studies:**

The government and nuclear company were criticized for :

1. Insufficient prevision to natural disasters, specially tsunamis and earthquakes

2. A network of corruption between regulators and nuclear plant company, where no serious control was made to ensure nuclear facilities safety.

3. The company under control of the Nuclear plant didn’t take stronger measures to prevent disasters for fear of inviting lawsuits or protests against its nuclear plants.

**b. Class diagram**



**c. Use Cases:**

1. Take enough measures to prevent natural disasters

Actor: government, regulators, company

2. Lack of a good emergency plan.

Actor: government, company.

3. Not enough investment in nuclear plant safety research.

Actor: company.

4. Lack of strict control over nuclear plants.

Actor: regulator, company.

**d. Use Cases description:**

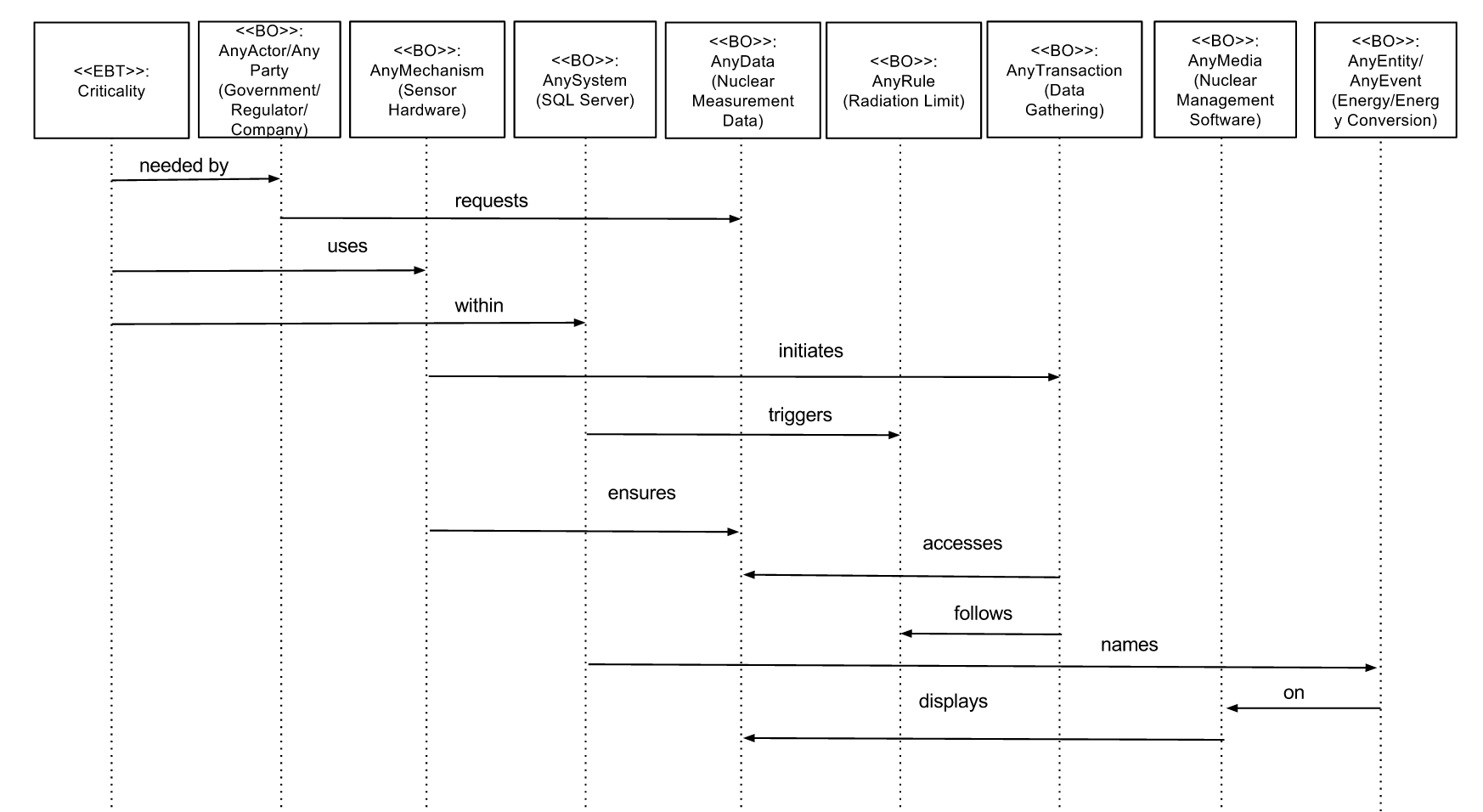
**1. Take enough measures to prevent natural disasters:** the government didn’t put much effort to demand companies to invest more on safety. The regulators simply didn’t put much effort to supervise nuclear facilities and the companies just relaxed the measures to prevent natural disasters.

**2. Lack of good emergency plan:** when the disaster occurred there wasn’t a good emergency plan. The robot’s rescue plan was dismantled a few years beforeand many workers were exposed to radiation when they tried to fix the problem.

**3. Not enough investment in nuclear plant safety research:** Tepco, the company that ruled the nuclear plant, stopped investing resources in safety plans because they considered it wasn’t necessary by that time.

**4. Lack of strict control over nuclear plants:** the companies had a strong influence on regulators promising them good future jobs in the industry while not affecting company’s reputation.

**e. Sequence Diagram:**



**2.4.5 Case Studies**

**Case Study: UK air traffic control failure**

An unprecedented computer systems failure set off the mayhem that caused delays and cancelled flights for thousands of passengers in December, 2013 in the United Kingdom.

In its first detailed explanation of the computer glitch that led to flight disruptions, National Air Traffic Services, which operates the UK’s main air traffic control centre at Swanwick, said the problems occurred as more workstations were being brought on line to deal with an increase in traffic.

“In normal operations the number of workstations in use versus in standby fluctuates with the demands of the traffic being controlled,” Nats said in a statement. “In this instance a transition between the two states caused a failure in the system which has not been seen before.”

The computer failure made it impossible for the controllers to access data regarding individual flight plans. Aircraft were prevented from taking off, but those in the air and close to airports were allowed to land during the shutdown that lasted about 35 minutes.

Computer software experts have said the problem appears to lie in the age of the systems – some of which date to the 1960s.

The Swanwick centre controls aircraft travelling through 200,000 square miles of airspace over England and Wales and handles 5,000 flights in every 24-hour period. It has been beset with problems since it first opened 12 years ago. The last major outage led to almost 300 flights being cancelled and 1,400 delayed over a two-day period.

That failure in December 2013 cost Nats £7.3m and led to the top executives [losing](http://www.ft.com/cms/s/0/3fc6eb6a-fde3-11e3-bd0e-00144feab7de.html) part of their annual bonus.

**a. Case Studies:**

The National Air Traffic service was criticized for :

1. A small problem costed millions and millions to airlanes.

2. It took two days to the controller to restore the service.

3. The systems employed were too old to be on operation for such a critical service.

**b. Class diagram**



**c. Use Cases:**

1. Lack of contingency plan

Actor: government, NAT

2. Very old systems employed

Actor: NAT.

3. It took so long to detect the problem

Actor: NAT.

4. Many problems since agency creation

Actor: NAT.

**d. Use Cases description:**

**1. Lack of contingency plan:** when the problem occurred, there were no contingency plan to quickly resolve the situation. The authority just landed all the flights and cancelled the ones that were going to departure.

**2. Very old systems employed:** the authority employed very old legacy systems to control the United Kingdom airspace and over 5000 flights, 24 hours a day. Which such a dynamic place like an airport, newer technologies should have been employed to prevent this failure.

**3. It took so long to detect the problem:** it took two days to detect where the problem was and restore the flights. It is a considerable time for a critical service like flight transportation, specially in the crowded United Kingdom airspace.

**4. Many problems since agency creation:** since the agency was created many problems occurred, but not real solutions were taken or implemented.

**e. Sequence diagram:**



**3. Consequences**

In this paper the concepts of Responsiveness , Criticality, Transaction and Duration have been analyzed and mapped for Real-Time Database Systems. For short-size patterns the context and the problem is defined, solution diagram is given and five different applications are mapped. For mid-size patterns two detailed case studies are also provided. This design will provide system architects many advantages, since the patterns are independent to any domain and scenario. Instead of working on a specific scenario we focused on the ultimate goal of the problem so our approach is applicable to any Real-Time Database System scenario that relies heavily on Duration, Criticality, Transaction and Responsiveness.

**4. APPENDIX**

**Knowledge Map of Real-Time Database Systems**

**1. KM Name: Real-time Database Systems (RTDBS)**

**2. KM Nickname: None**

**3. KM Domain/Subject/Topic Description:** RTDBS is a database system that processes data in real time to handle tasks whose state is always changing. This system is different than traditional databases in which the data is unaffected most of the time unlike real-time systems. Yielding reliable responses is a must to keep RTDBS effective.

**4. EBTs/Goals:**

Table 1: EBTs of “**RTDBS**”

|  |  |
| --- | --- |
| EBTs/Goals | Description |
| Accuracy | The requirement that the data values stored for an object are the correct values. |
| Actuality | All the data stored in the database for this system should be real time or actual. |
| Consistency | The requirement that any data written to the database must be valid according to constraints or other rules. If a transaction introduces an inconsistent data, an error is returned. |
| Criticality | The database state of being of the highest importance. Critical database systems inclined to find errors. |
| Control | Refers to determining the behaviour of the database. |
| Efficiency | The state of being able to accomplish a database task with the least waste of effort and time. |
| Predictability | Refers to making it possible to know in advance about database events. |
| Reliability | The system should be reliable. It should be fault tolerant and it should guarantee data integrity. |
| Responsiveness | This systems should be responsive. They should react quickly to requests and provide an answer in real time. |
| Sensitivity | State of responsiveness and awareness to database challenges and demands. |
| Timing | The control or choice of when database task should be done. |

**5. BOs/Properties:**

Table 2: BOs of “**RTDBS**”

|  |  |
| --- | --- |
| BOs/Capabilities | Description |
| AnyBaseline | A starting point used for measurement and comparisons. |
| AnyBenchmark | A standard or point of reference against which things are compared. |
| AnyConstraint | A specified rule that defined for the database in a database table. Data must comply with constraints. |
| AnyContext | It defines which elements are part of the problem under responsiveness. |
| AnyData | An information piece that is kept in the database. There are many types of data such as integer, float, string etc. |
| AnyDeadline | The latest time by which database transaction or task should be completed. |
| AnyDuration | Instant at which data is demanded. |
| AnyEntity | Represents anything that exists. An existence. |
| AnyEvent | Represents something that takes place. An occurrence. |
| AnyImpact | Let’s us know how the impact would be of the criteria. |
| AnyLog | A text file that stores database transaction information. |
| AnyMechanism | Any process or way in which data is processed. |
| AnyMedia | Any kind of media or medium data is sent or received. |
| AnyParty/Any Actor | Any party, any organization, any individual who sends or receives data. |
| AnyPerformance | A database process or action of carrying out or accomplishing a database task. |
| AnyRequest | A way in which the database receives/responds to queries. |
| AnyRule | One of a set of regulations that governs database tasks. |
| AnySchedule | A model to show execution of transactions in the database system. |
| AnySystem | An application that interacts with the user and the database to capture and analyze data. |
| AnyTraffic | Trading and movement of the data along the database network. |
| AnyTransaction | A unit of work that has sequence of operations. |
| AnyType | Represents all the instances of the criteria. |

**6. Knowledge Map (Core Knowledge):**

Table 3: Knowledge Map of “**RTDBS**”

|  |  |
| --- | --- |
| EBTs | BOs |
| Accuracy | AnyData, AnyParty/AnyActor, AnyMechanism, AnyType, AnyEntity/AnyEvent, AnyMedia, AnySystem |
| Actuality | AnyData ,AnyActor/Any Party, AnyTime, AnyMedia, AnyType, AnyEntity, AnyEvent, AnyMechanism |
| Consistency | AnyActor/AnyParty, AnyData, AnyTransaction, AnyConstraint, AnyType, AnyEntity, AnyEvent |
| Control | AnyParty/AnyActor, AnyMechanism, AnyImpact, AnyPolicy |
| Criticality | AnyParty/AnyActor, AnyMechanism, AnyRule, AnyData, AnyTransaction, AnySystem, AnyEntity/AnyEvent, AnyMedia |
| Efficiency | AnyParty/AnyActor, AnyData, AnySystem, AnyMechanism, AnyType, AnyEntity/AnyEvent, AnyMedia |
| Predictability | AnyParty/AnyActor, AnyData, AnySystem, AnyOutcome, AnyTransaction |
| Reliability | AnyRequest, AnyData, AnyActor/AnyParty,AnyImpact, AnyType, AnyMedia, AnyMechanism, AnyEntity, AnyEvent |
| Responsiveness | AnyRequest, AnyData, AnyActor/Any Party, Any Media, Any Mechanism, Any Event,AnyLog, AnyContext, AnyTransaction |
| Sensitivity | AnyParty/AnyActor, AnyData, AnyTime, AnyEntity/AnyEvent, AnyType, AnyMedia, AnyDta |
| Timing | AnyActor/Any Party, AnyData, AnyTransaction, AnyConstraint, AnyType, AnyEntity, AnyEvent, AnySchedule, AnyDuration, AnyDeadline |

References:

[1] http://www.cs.helsinki.fi/u/jplindst/papers/rtds.pdf