BEHAVIOR MODELING OF SOFT REAL-TIME SYSTEM USING STEREOTYPED EXTENSION MECHANISMS

Rajendra R. Dube[1], Shantanu K. Dixit[2]

[1]*Assistant Professor, Department of Electronics and Telecommunications, Walchand Institute of Technology, Solapur, Maharashtra, India*

[2]*Professor and Head, Department of Electronics and Telecommunications, Walchand Institute of Technology, Solapur, Maharashtra, India*

Abstract

Real-time system is a special category of system which demands the correctness of a functionality to be satisfied within specified timeframe. The factors which affect the analysis and design of real-time systems are interoperability, portability, performance and reliability along with scalability as a prime concern. If the real-time systems are not correctly analyzed and designed, there will be lot many questions existing about deployment of such system. Prototyping and rapid development methodologies may not be suitable in the context of soft real-time system development since they are cost effective. In the past decade there were several modeling methodologies proposed in order satisfy the need to obtain the stable modeling paradigm. One of the popular means to model the static behavior of the system is though the application of UML profiles written for variety of domains. UML proposes use case model generation for estimating the system characteristics from the point of view of external entities expressed as actors. As UML use case models can only have structuring and extension relationships, it becomes necessary to understand the composition of the system in terms of guided relationships. This paper proposes a new UML use case stereotyping metamodel that can be used to address the need for discovering static behavior of soft real-time systems. The paper also explains the example of credit card processing which is a online transaction processing system belonging to soft real-time systems.

Keywords: *Real-time system, UML, Use Case Model, Stereotype, Metamodel*

----------***----------

1. INTRODUCTION

A real-time system is composed of tightly coupled software elements which are constrained by external interacting system. The prime characteristic of real-time system is correctness which means that the compound results of computations depend on time at which the results are produced. The real-time system has behavioral properties that stimulate the action chain in order to reach to outputs bounded by specified parameters. A real-time system is a subset of relatively complex system such as cyber-physical system. A real-time system is bound to change its states as the physical time changes. It is therefore the real-time system is required to be analyzed and developed in terms of manageable compartments. The constraint on real-time system is usually expressed as deadline that indicates the process completion to be achieved within a specified time limit adhering to performance as expected by external environment. The relative deadline is the one that is specified in relation with the task arrival time, whereas the absolute deadline is specified in relation to time zero. The real-time tasks can be classified into following categories based on the missing deadline aspect:

Hard: If the missing deadline causes catastrophic impacts on the system performance and the output is not acceptable then the system is called as hard real-time system.

Firm: If the missing deadline causes no significant impacts on the system performance and the output is of no value then the system is called as firm real-time system.

Soft: If the missing deadline causes no significant impacts on the system performance and the output is of some value is acceptable with degradation then the system is called as soft real-time system. [1]

The hard real-time system must satisfy behavioral requirements with the applied load and stress constraints within the specified deadline. The hard real-time systems are required to maintain fair policies to ascertain the robustness expressed in terms of recovery from failure situations. Fault seeding and monitoring are the techniques that can address the ways to ensure the robustness of hard real-time systems. The soft real-time systems on the other hand can miss the deadlines occasionally but still the output is considered to be valid. Clocks partitions the time space into equal sequences of durations during which an event can happen. Ideally an event is an occurrence in time that happens related to certain condition or situation in process. The event information can be further used to develop the event catalogue to analyze the efficiency of the system. The events can be periodic or aperiodic and must be carefully recognized. Following are some of the important characteristics to gauge the system dynamics:

Response Time: A hard real-time system has response time typically in milliseconds which is a case in critical systems indicating fail-safe operation and requires minimum human intervention. Soft real-time systems can take several seconds to furnish the output which is a case in on-line systems and there is no exact minimal time frame management to know the missing deadline.

Peak-load Performance: The hard real-time systems must have clearly specified peak-load management policy since the system must be able to handle explicit events which can cause the peak load and a compromise may cause a performance drift which may not be acceptable. The soft real-time system is acceptable with degraded performance due to economic issues.

Operational Efficiency: In case of hard real-time system there is possibility of multiple concurrent states occurrence that may occur during processing due to changes occurred in environment. In case of soft real-time system the states can be tuned with the external environment such that the states exhibit offered load.

Safety: The real-time system must support fault detection and recovery mechanisms to be in a consistent safe state within specified time bounds without incorporation of manual intervention.

Database Size: The real-time systems have small database size with accuracy and precision parameters. As the processing of the events is concerned about validating timing properties, the quick retrieval and updates are required to be performed within deadlines. Availability, integrity and security are the essential properties while designing online transaction processing systems. [2]

2. REAL- TIME SYSTEM KEY AREAS

The event-triggered and time-triggered systems depend on the type of internal triggers and not the external behavior of a real-time system. A trigger is an event that causes the start of some action or task. The triggering mechanism determines the commencement of communication and processing task. In event-triggered (ET) control, communication and processing tasks are initiated whenever an event other than the regular event of a clock tick occurs. In a time-triggered system, communication and processing tasks are initiated by progression of real time. [3]

Real systems are complex to model, analyze, and synthesize because they are composed of subsystems. The process of composition and decomposition should be iterated till the system mission is achieved. The significant impact on composition and decomposition is arising from the domain complexity and analyst's experience. Typical analysis processes which advocate Top-Down decomposition in order to structure the systems may not essentially agreeable since systems cannot be designed and analyzed in isolation. The composition and decomposition is based on the way the principles of abstraction are followed while deciding about number of components participating in subsystem design. When we structure the system elements, we integrate the components to form the subsystems at some hierarchical level and offer realization of these system parts though design correct and consistent interfaces. For later implementation stages, the guiding principle is to look into granularity and coupling between the system elements.

The composition of concurrent activities deals with modeling concurrency which is the synchronization of the processes, tasks, threads when they are required to be accessed as shared resources or exchange messages. The modeling scenario can be visualized in broad areas: structure and behavior of the system. [4]

The structure and behavior of the system are cohesive elements ordered and organized in such a way that the outcomes of system executions are stakeholder satisfying. The major issue in addressing the behavior is whether we are attempting to model static behavior or dynamic behavior of the system. In case of static behavior determination, the problem areas is required to be analyzed or synthesized depending the type of domain. The problem can seen in terms of goals to be achieved by the system, the issues concerning the stakeholders and mapping of objectives to reach to technical realization of the system. The static behavior determines the system process areas which are distinctly related to the stakeholders of the system.

Synchronous and asynchronous composition mechanisms can be used to model concurrent behavior of the real-time system. Synchronous composition is based on identification of constraints that change the state depending on the occurrence of the events at the same time or at time instants that are strictly and rigidly specified, which is a case with discrete time domain. In asynchronous composition, the execution units can start and end where the relativity between processes is independent of each other. It is therefore that the asynchronous composition does not require same type of event occurrence to start and terminate the processing units. In asynchronous systems, the interaction between the components is not exercised on large scale. This allows the system components to have minimum and required coupling with other external components. [5]

Soft real-time systems are concerned about soft aperiodic tasks. The examples of soft real-time system include online transaction processing systems. To handle the soft aperiodic tasks scheduling mechanism can be used but this degrades the performance gently. The performance enhancements can be caused by making use of server to handle these tasks. Aperiodic tasks can be scheduled in the background when there are no periodic tasks ready to execute which means that they are queued on first-come-first-served schedule. The server capacity must be designed in a way to guarantee communication and synchronization based on scheduling capabilities and response to events. The Polling server can be used when the intervals are regular and the aperiodic requests are required to be served with the server capacity. The deferrable server can be used when priorities are been

specified and the timing window allows the aperiodic requests to occur within the server capacity. The sporadic server can be used when aperiodic requests are required to be assigned with full processing load until there is a new set of aperiodic requests arrives within the server capacity.

3. USE CASE MODELING

Use cases were first proposed by Ivar Jabobson in the approach Object Oriented Software Engineering (OOSE). OOSE recommends use case based approach for analysis and design of the system. A use case expresses Functional Requirements (FR) as perceived by external entities called as actors. The actor can have multiple dimensions and expectations from the system. The actors can be a external participating entity in problem solving process such as customer, user or stakeholder. [6] [7]

The stakeholder's dimension can be exploited in terms of analysts, designers, programmers, testers, manages, administrators or simply those who are playing some organization role. [8] [9]

The Functional Requirements are explicitly stated objectives from stakeholders dimension in System Requirement Specification (SRS). The SRS need to comply with three fundamental attributes correctness, consistency and completeness. Functional Requirements are often stated as natural langue statements and there exists the problem of ambiguity. [10] A requirement is called as ambiguous it is have more than one interpretations to be derived by the stakeholders. Associating requirements to the Problem Specification is a challenge since the further development is solely based on how well the requirements are structured.

The other fundamental requirement set is called as Non-Functional Requirement (NFR) which is typically associated with the developers and deals with quality of the system. The commonly used NFR set consists of run-time and non run-time requirements which guides the quality of the system. [11] The NFR set may have Accuracy, Performance Reliability, Usability, Maintainability, Testability, Interoperability and Portability as prime properties applicable especially to the real-time system. The context for this is paper is limited to soft real-time system modeling by making use of extended stereotype mechanisms. With UML 2.0 becoming a standard for modeling general purpose system characteristics, there has been a consistent application of UML diagrams for system development. This not only reduces the burden on the developers to decide at the later stages the implementation strategies but also contributes significantly in deriving architecture of the system. UML 2.0 use case diagram belongs to behavioral diagrams which indicate static behavior of the system. The use case diagrams are drawn to understand the system properties derived from the Functional Requirements and assigning them as system tasks. [12]

4. SEMANTICS OF USE CASE DIAGRAM

4.1 Actor

An actor indicates the role plays a role of external entities to the subject and indicates the interaction to one of the system element indicated in terms of the use cases. The actor has a multiplicity more than one to the related use cases indicating the references that the actor can make to multiple use cases. The relationship between the actor and use cases is indicated by association which bidirectional. The arrowhead added to the association can be used as navigation medium but does not carry any significant meaning. In case of soft real-time systems, an actor has an ability to initiate multiple activities or use cases concurrently indicating the parallelism at the behavioral stage. The scope and references of the actors must be clearly stated in order to balance the consistency and correctness characteristics of the system. The actor is modeled by a stick man notation as indicated. [13] [14] [15]

4.2 Use Case

The use case indicates the usage profile of the system which is relevant to the actors identified. No use case remain in isolation which means that there cannot be an isolated functionally unused by an actor. [16]

The use case names must indicate clearly the use in terms of verbs indicating the action to be carried out and followed by a noun. The use cases are graphically represented as rounded ellipse.[17]

4.3 Stereotypes

These are the extensibility mechanisms offered by UML 2.0. The stereotype is represented as a relationship bounded by guillemets << and >>. UML 2.0 stereotypes are <<include>> and <<extend>> [18]. The proposed stereotype extension use case profile adds twelve stereotypes categorized in basic, intermediate and advance categories. The extended UML use case metamodel is depicted in Figure 1. The following section indicates stereotype categories.

4.3.1 Basic Stereotypes

- <<include>> : This relationship indicated the structural composition of use cases where the main use case includes the sub use cases.
- <<extend>> : This relationship indicates that the features of the extending use case are made available to referencing base use case.
- <<orchestrates>>: This relationship indicates that the participating use cases can collaborate to achieve desired outcome.
- <<moderates>>: This relationship indicates the limits as stated by the super use case to be fulfilled by the sub use case.
- <<precedes>>: This relationship indicates a precedence order amongst the use cases.

4.3.2 Intermediate Stereotypes

- <<invokes>>: This relationship indicates that source use case calls target use case to address the functionality expected by the actor.
- <<transcribes>>: This relationship indicates a transfer of parameters from source use case to target use case.
- <<preserves>>: This relationship indicates the parameters satisfied by the target use cases the knowledge about which is retained by the source use case.
- <<alters>>: This relationship indicates the behavioral change performed by source use case on the target use case.

4.3.3 Advanced Stereotypes

- <<governs>>: This relationship indicates that the source use case exercises certain properties determining influence over the target use case.
- <<restrains>>: This relationship restricts the target use case access to secure system use cases by performing applicability limit specification.
- <<reinforces>>: This relationship indicates extra support added by the source use case to the target use case for deriving the expected outcome.
- <<inhibits>>: This relationship indicates that the source use case hold back some of the parameters from the target use case.
- <<conceives>>: This relationship indicates the forcible control applied by source use case on target use case.

Typically, the UML 2.0 use case diagram supports <<include>> and <<extend>>. The proposed methodology makes an extensive use of the additional twelve stereotypes for better conception of the use case model. Following section introduces the application of these stereotypes to generate the detailed use case models.

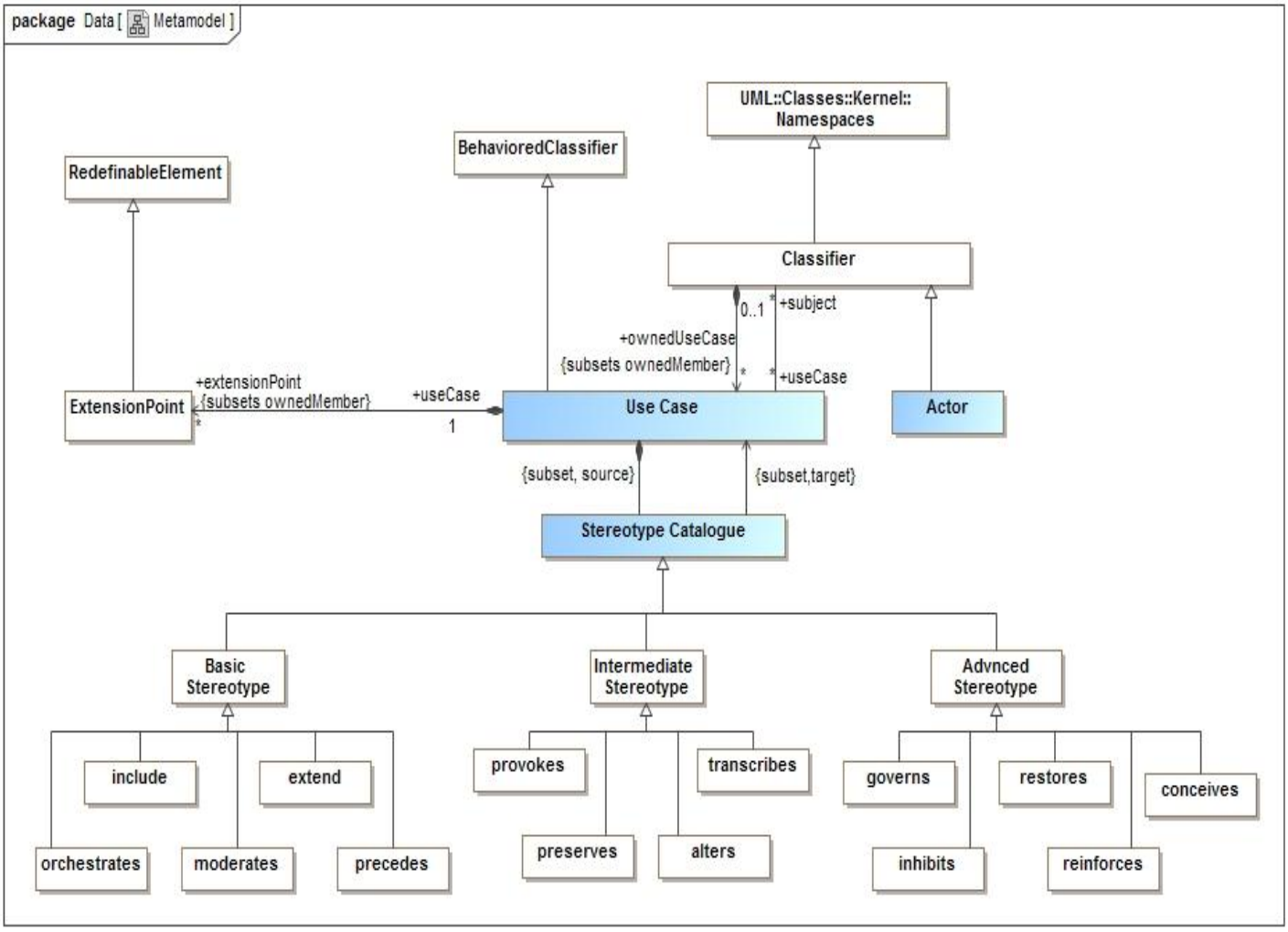

Fig -1: Extended UML Use Case Metamodel

5. MODELING SOFT REL-TIME SYSTEMS

Online transaction processing systems are the example of soft real-time systems. We have considered the Credit Card Processing system as an example. We introduce four scenarios: Card/PIN issuance, Merchant Account Management, Acquiring Bank Processing and Payment Gateway Management. The following section discusses in detail these four scenarios. [19] [20] [21]

5.1 Card/PIN Issuance

Financial institutions are required to deal with fraud-related risks when issuing credit cards either managed by self or contracted to third parties. Fraudulent customer transactions may result due to ineffective card and PIN issuance procedures. The access to customer account and PIN information can be potential threat towards misuse of security policies crafted in support of customer. Backup mechanisms can be an asset for performing status registration of the cards issued which can benefit the employees to retrieve the card information against losses. Adequate controls must also exist for captured cards and rejected cards with sufficient information about the card configuration.

The mail management with the customers should be able to deploy disbursed and returned cards procedures separately. The disbursement controls should be released based on random distribution system such that the card delivery could be validated. The PIN generation should be incorporated by controlling active PIN information along with encryption information. The PIN generation process should have configuration profile which can be updated by a panel of card moderators. The intimation regarding the PIN information should not appear in any of the online transactions performed with customer profile management.

Adequate security provisions must be guaranteed to restrict the accesses to customer account through direct online channel.

The PIN information must be regenerated on demand with update information which in turn can be an additional asset for customer service management. A card security code (CSC), alternatively called as card verification number (CVN), card verification value (CVV) or card verification code (CVC2) is a security feature for "card not present" payment card transactions instituted to reduce the incidence of credit card fraud. The card issuer should generate unique CVN for customer accounts for verification of customer records. The PIN should not be printed or embedded on the card but is required to be entered by the cardholder during a point-of-sale transactions. Tokenization which is a process for an artificial account number (token) should be stored or transmitted in place of the true account number. Figure 2 indicates the use case diagram drawn by using <<include>> and <<extend>> stereotypes whereas Figure 3 indicates application of extended stereotypes.

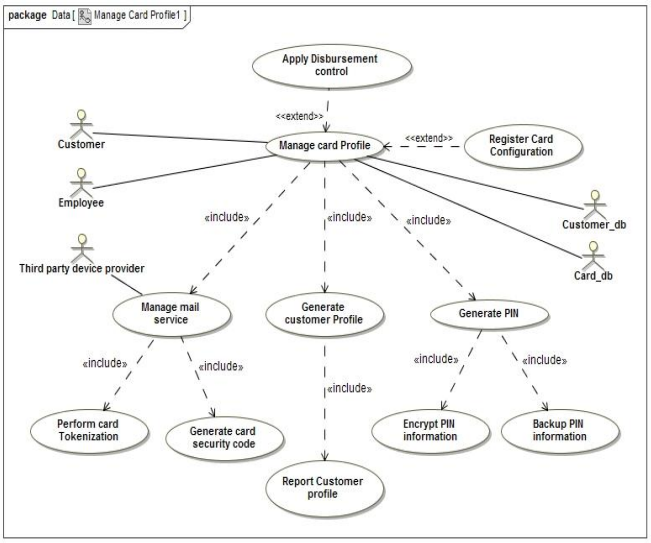

Fig -2: Use Case Model with <<include>> and <<extend>>

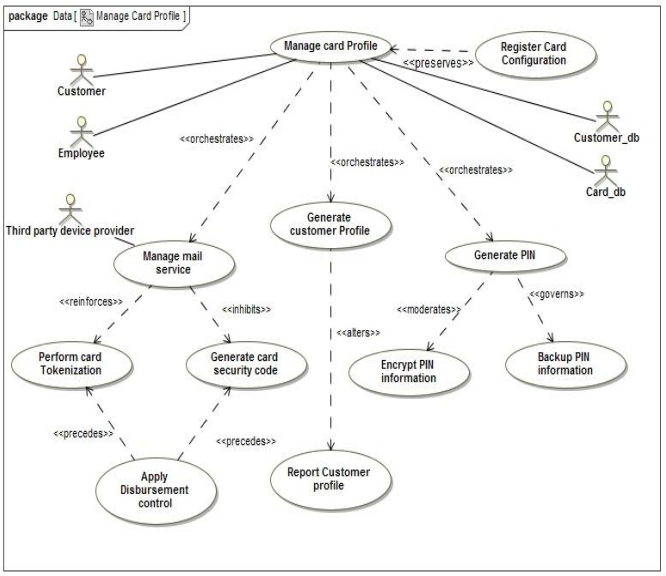

Fig -3: Use Case Model: Extended Stereotypes Approach

5.2 Merchant Account Management

It refers to the acquiring bank or the independent sales organization that represents an organization that the merchant deals with. A group of companies called Independent Sales Organizations (or ISOs) provide technical support, transaction processing, bear the risk of chargeback and set the price of the services. A Merchant Account Provider charges for the Merchant Account with certain periodic or item-based process.

The pricing method would be a 6-Tier pricing method as described below:

Authorization fee: It is charged whenever a transaction is forwarded to the card-issuing bank for the purpose verifying authorization controls.

Transaction fee: It is charged when the customer accepts the authorization when authorization is successful carrying no errors.

Statement fee: This fee is paid by the merchants monthly in order to receive the settlements performed during a month. This monthly statement indicates the statistics of processing performed by merchants within a month and related charge for it.

Monthly fee: This fee is charged for the maintenance of merchant account and the merchants are required to pay the surcharge if adequate balance does not exist.

Batch fee: It is charges when the merchant follows settlement process in order to inform the acquiring bank about completed transactions.

Figure 4 indicates the use case diagram drawn by using <<include>> and <<extend>> stereotypes whereas Figure 5 indicates application of extended stereotypes.

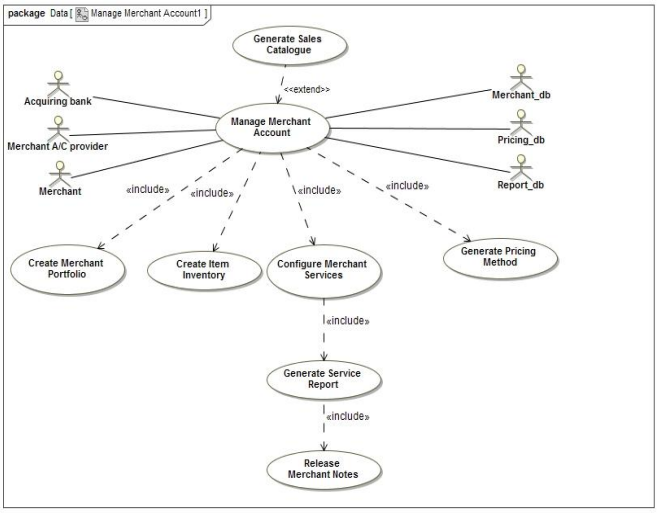

Fig -4: Use Case Model with <<include>> and <<extend>>

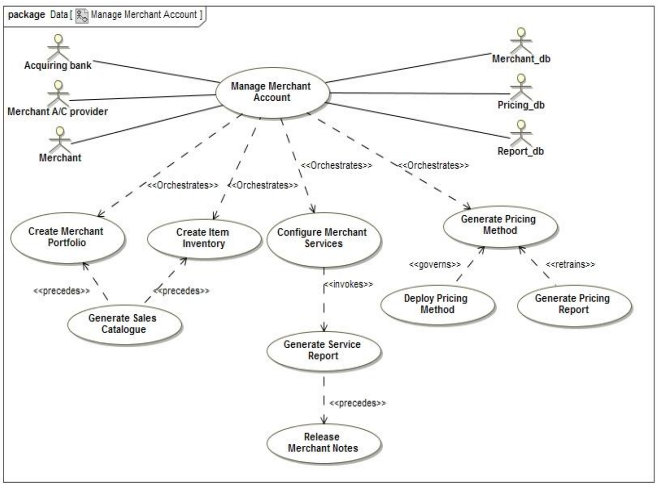

Fig -5: Use Case Model: Extended Stereotypes Approach

5.3 Acquiring Bank Processing

Acquiring banks are responsible for deploying the payment system used by merchants and third-party service providers. Acquiring banks are required to check that merchants and third-party service providers adhere to Payment Card Industry Data Security Standards (PCI DSS). Acquiring banks are responsible for maintaining credit information of merchants on a regular basis. Acquiring banks should elicit the profile generation for new merchants in coordination with a third party such as an ISO.

Acquiring banks should check the background of merchants, review credit history, and validate periodic sales record. In case of online merchants, the acquiring banks may review merchant's website and collect the information for chargeback review. In case the merchant does not pay chargeback, the acquiring bank will be responsible to pay the sum to the issuing bank.

Acquiring banks should be able to maintain the merchant portfolio and perform monitoring functions over merchant's account. Acquiring banks should generate reports for merchant status, transactions committed by merchants, scale of transactions by merchants within a time band, periodic average turnover. Acquiring banks can deal with fraudulent merchant accounts by deferred funding and incorporating fraud detection panel. The merchant account would be liable to accept the decisions of acquiring banks in terms of credit reversals and insurance management.

Figure 6 indicates the use case diagram drawn by using <<include>> and <<extend>> stereotypes whereas Figure 7 indicates application of extended stereotypes.

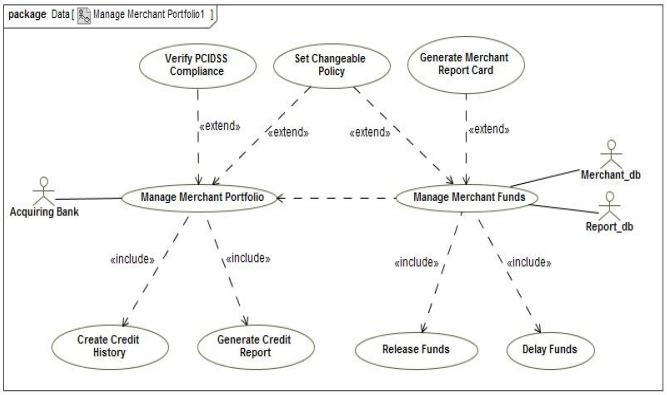

Fig -6: Use Case Model with <<include>> and <<extend>>

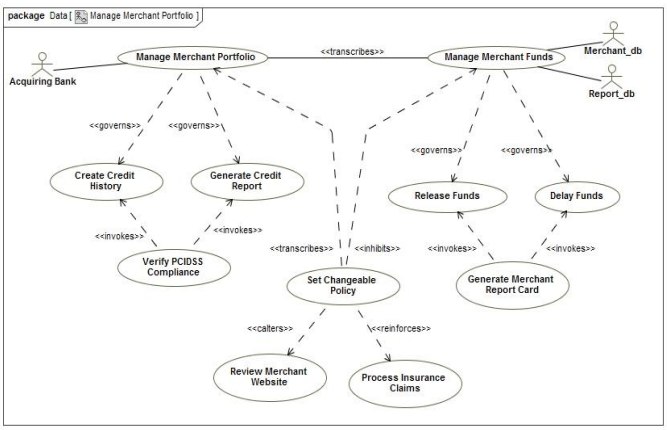

Fig -7: Use Case Model: Extended Stereotypes Approach

5.4 Payment Gateway Management

A payment gateway is a Service Provider used for authorization of payments related to credit cards, retailers supporting information transfer for customer transactions. The transactions can be performed with recognizable services as supported by the service provider such as mobile phone or website. The customer places order for an item from a web-enabled card processing unit.

The web browser will encrypt the item related transaction before forwarding it to merchant's web server. The payment gateway ensures that the encrypted information is complaint with the merchant's Payment Card Industry Data Security Standard. The merchant web server forwards the details of the transactions to the payment gateway. The payment processor receives an alert for the transactions and the acquiring bank is informed about the transaction. The payment processor checks the transaction validity by using the card information and intimates approval or rejection of payment to payment gateway.

The issuing bank receives the transaction request and validates the card information for fraudulent transactions. The issuing bank forwards a response code to the payment processor in case of transaction leading to an unsuccessful status due to insufficient funds. The payment processor forwards the authorization information to the payment gateway which in turn directs this information to a web response handing unit. The merchant responds the order by committing to it and the transaction is updated with clear response with a message forwarded to issuing bank for settling the transaction. The merchant selects the settlement options and acquiring bank is informed about the settlement option. The acquiring bank settles the payment and the customer is informed about the transaction payment.

Figure 8 indicates the use case diagram drawn by using <<include>> and <<extend>> stereotypes whereas Figure 9 indicates application of extended stereotypes.

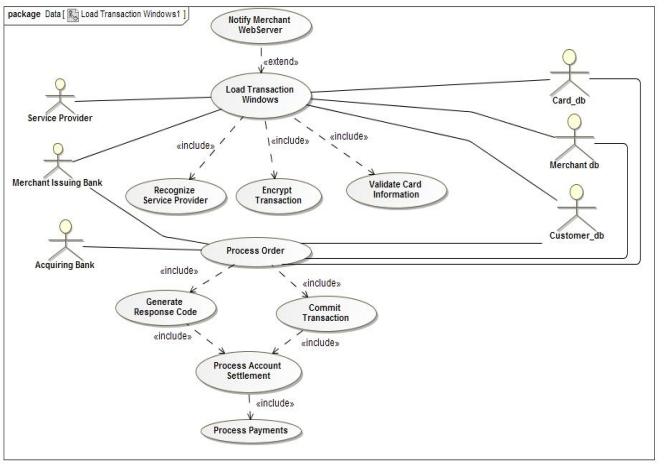

Fig -6: Use Case Model with <<include>> and <<extend>>

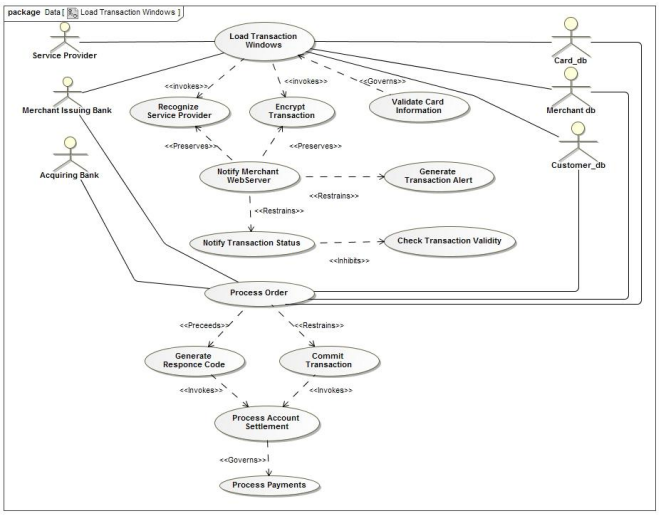

Fig -7: Use Case Model: Extended Stereotypes Approach

6. CONCLUSION

The paper presents an approach to extend the use case modeling using extended stereotypes for soft real-time systems. It can be identified from the use case diagrams that the use of <<include>> and <<extend>> stereotypes limits the system hierarchy to be perceived in a monotonous way. The modeler can garb better control over organizing the use cases by using applicable relationships defined into basic, intermediate and advanced stereotypes. It can be observed that the use of basic stereotypes can be made to organize the system elements in such a way that the cooperative behavior can be realized while modeling the design. The intermediate stereotypes can be used when there is a transformation between source and target use case. The advanced stereotypes are about mapping the constraints as applicable to the use cases. Overall, it can be observed that the extended UML use case stereotype extensions enhance the Use Case Model richness.

REFERENCES

[1] Mario Bravetti, M. Bernardo and F. Corradini, Real Time and Stochastic Time, SFM-RT 2004, LNCS 3185, pp. 132–180, Springer-Verlag 2004

[2] Diletta R. Cacciagrano, Flavio Corradini ,Expressiveness of Timed Events and Timed Languages, SFM-RT 2004, LNCS 3185, pp. 98–131, 2004.,Springer-Verlag 2004

[3] V. Gruhn, R. Laue , Patterns for Timed Property Specifications, Elsevier, Electronic Notes in Theoretical Computer Science 153 (2006), pp. 117–133

[4] Shixin Zhuang,• Christos G. Cassandras, Optimal Control of Discrete Event Systems with Weakly Hard Real-Time Constraints, Discrete Event Dynamic System (2009) pp. 19:67–89, Springer Science + Business Media, LLC 2008

[5] Francis Cottet, Joelle Delacroix, Claude Kaiser, Zoubir Mammeri, Scheduling in Real-Time Systems, John Wiley & Sons Ltd, The Atrium,

Southern Gate, Chichester, West Sussex PO19 8SQ, England, 2002, ISBN 0-470-84766-2

[6] Kotonya G, Sommerville I., Requirements engineering with viewpoints, Cooperative Systems Engineering Group Technical Report CSEG/10/1995, University of Lancaster, UK

[7] Dahlstedt, Å.G., Persson, A., Requirements interdependencies: state of the art and future challenges, Engineering and Managing Software Requirements, Springer, Berlin 2005, pp 95–116

[8] Mylopoulos J, Chung L, Yu E, Nixon B., Representing and using non-functional requirements: a process-oriented approach, IEEE Trans Software Eng 1992; Vol. 18(6), pp 483–497

[9] Meyer, B., Nawrocki, J.R., Walter, B. (eds.), Balancing Agility and Formalism in Software Engineering (CEE-SET 2007), Lecture Notes in Computer Science, vol. 5082, Springer, Berlin, 2008 pp 267–278

[10] Moreira, A. Rashid, and J. Araújo (2005). Multi-Dimensional Separation of Concerns in Requirements Engineering. Proc. 13th IEEE International Requirements Engineering Conference (RE'05). 285-296.

[11] Antje von Knethen, A Trace Model for System Requirements Changes on Embedded Systems, IWPSE 2001, Vienna Austria, ACM 2002 1-58113-508 -4/02/006 pp 17-26

[12] Simone Röttger, Steffen Zschaler, Tool support for refinement of non-functional specifications, Software System Model (2007) 6:pp. 185–204, Springer-Verlag 2006Pietro Colombo, Matteo Pradella, Matteo Rossi, A UML 2compatible language and tool for formal modeling realtime system architectures, SAC'06 April 2327, 2006, Dijon, France, pp. 1785-1790

[13] Gregor Engels, Jochen M. Kuster, Reiko Heckel, A Methodology for Specifying and Analyzing Consistency of Object-Oriented Behavioral Models, ESEC/FSE 2001, Vienna, Austria, ACM 2001 1-58113-390-1/01/09, pp 186-195

[14] Petri Kukkala, Jouni Riihimäki, Marko Hännikäinen1, Timo D. Hämäläinen1, and Klaus Kronlöf, UML 2.0 Profile for Embedded System Design, Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05), 1530-1591/05

[15] Shane Sendall, Alfred Strohmeier, Specifying Concurrent System Behavior and Timing Constraints Using OCL and UML, UML 2001, LNCS 2185, pp. 391−405, 2001, Springer-Verlag 2001

[16] Hooman J, Kugler H, Ober I, Votintseva A, Yushtein Y., Supporting UML-based development of embedded systems by formal techniques, Software System Model 7(2), 2008,pp 131–155

[17] Ahmed W, D Myers, Faster exploration of high level design alternatives using UML for better partitions, Proceedings of the conference on design, automation and test in Europe: Proceedings, DATE '06, European Design and Automation Association, Belgium, 2006, pp 579–580.

[18] OMG UML profile for MARTE, beta 1, ptc/07-08-04. http:// www.omg.org/cgi-bin/doc?ptc/ 2007 2007-08-04

[19] K. Kathirvel, Credit Card Frauds and Measures to Detect and Prevent Them, International Journal of Marketing, Financial Services & Management Research, Vol.2, No. 3, March (2013), pp 172- 179

[20] Sujit Chakravorti, Theory of Credit Card Networks: A Survey of the Literature, Review of Network Economics, Vol.2, Issue 2 – June 2003, pp 50-68

[21] Adilah Abd Razak Parker Hood, Allocating Unauthorised Credit Card Payment Losses: The Credit Card Guidelines and Consumer Protection, Int. Journal of Economics and Management 3(2): 248 – 261 (2009), ISSN 1823 - 836X

BIOGRAPHIES

R.R. Dube has completed ME in Electronics from TKIT, Warananagar, India in 1998.Currently he is working as Associate Professor in Department of Electronics and Telecommunication at Walchand Institute of Technology, Solapur. Presently he is perusing his Ph.D. from University of Solapur, Maharashtra, India, in Real Time System Modeling. His areas of interest are Embedded Systems and Real-time Systems.

Dr.S. K. Dixit has received Ph.D. in Electronics from Shivaji University, Kolhapur in 2002. Currently he is working as Head and Professor in Department of Electronics and Telecommunication at Walchand College of Engineering, Solapur.