Consistency between Use Case, Sequence and Timing Diagram for Real Time Software Systems

Rumpa Hazra
Heritage Institute of Technology
Kolkata, India

Shouvik Dey
Cognizant Technology Solutions
Kolkata, India

ABSTRACT

Modeling of real time software systems (RTSS) consist of different components with UML 2.0 leads to a design model using various diagrams. To get a consistent model, a consistency concept for different diagrams type is needed that takes into account real time constraints. Ensuring consistency of The Unified Modeling Language (UML) model is very crucial as it is effect to the quality of UML model and directly gives impact to good implementation of Information System. Although there are increasing researches on consistency management, there is still lack of researches of consistency driven by Use Case. With this motivation, in this paper, we have proposed few consistency rules between Use Case, Sequence and Timing diagrams which focus on the establishment of timing constraints. Elements of each diagram involved in the proposed rules are formalized. Using an example, we show how the diagrams fulfill our proposed consistency rules.

Keywords

Real time software systems, UML, Consistency, Real time constraints.

1. INTRODUCTION

Real Time Systems are now omnipresent in modern societies in various domains such as avionics, control of nuclear power stations, multimedia communications, robotics, air-traffic control, process control, and embedded systems. Developing a real time embedded system is a sophisticated and complex task.

The Unified Modeling Language (UML) is a graphical modeling language for visualising, specifying, constructing and documenting the artifacts of software systems. UML is widely used to express general-purpose software design models. It encourages the use of automated tools that facilitate the development process from analysis through coding. This is particularly true for real time embedded systems, whose behavioural aspects can often be described via UML. It is therefore interesting to consider how well UML is adapted to the real time context. One important feature of UML stems from its built-in extensibility mechanisms: stereotypes, tag values and profiles. These allow adapting UML to fit the specifics of particular domains or to support a specific analysis.

Since the different stages of Real Time software life cycle model are correlated and represent common aspects there is a need to check for inconsistencies among the related models. We want to define a set of consistency rules which are necessary as well as sufficient to ensure consistency within the design models.

In general consistency within a specification is mandatory requirements because it is a prerequisite for the correct execution of the system specified in different parts. Consistency is the property that different parts of a specification are compatible with each other and not contradictory. The problem of consistency arises in cases where a specification consists of different parts, each part concentrating on a specific view of the system. Then it has to be ensured that the overall specification gained from all parts does not contain consistency errors.

Consistency is a state in which two or more overlapping elements of different software models make assertions about the aspects of the system they describe which are jointly satisfiable [3]. It is one of the attributes used in measuring the quality of Unified Modeling Language (UML) model [4]. According to [3; 5] there are three (3) main activities in model consistency management. They are consistency specification, inconsistency detection and inconsistency handling. Consistency rules which must be respected by different diagrams in order for them to be consistent are specified first. If the consistency rules are not fulfilled, inconsistencies were aroused and they should be detected and handled. Even though, there are increasing research in consistency between diagrams as reviewed by Lucas, Molina and Toval [6], there are still lacks of researches of consistency driven by Use Case. In famous system development methodologies such as ICONIX and Unified Process (UP), Use Cases provide the foundation for defining functional requirements and design throughout system development [7; 8]. The importance of Use Case can be seen in [9] as it is second ranked diagram used by UML practitioners.

UML, being visual in nature, is easy to understand and communicate. UML as a real time modeling language has some limitations. It basically provides a lot of syntax, but not enough semantics and it lacks the rigor of formal modeling languages. Formalization of UML diagrams is now a dominant area of research. In this paper we described the formalization of the elements of Use Case Sequence and Timing diagrams which represent dynamic and behavioral aspects of a system. Then, the consistency rules have been proposed by analyzing the interrelationships among those diagrams so that they together represent a coherent design.

The paper is structured as follows. In Section 2 we provide the background of related works. Section 3 provides a brief overview of the actual scope of this work. Section 4 presents formalization of the elements of Use Case, Sequence and Timing diagrams. In this section, we also propose few consistency rules. Section 5 provides a case study of UML model for ATM System. In section 6, we present our conclusions.

2. RELATED WORK

This section presents a review of some of the research works that have been done in the area of consistency of UML designs for Real Time Software Systems. An algorithmic approach to a consistency check between UML Sequence and State diagrams is described in [1] while [14] proposes a declarative approach using process algebra CSP for consistency checking between sequence and state charts. In [2] an approach for automated consistency checking named VIEWINTEGRA has been developed.

Consistency between use case and activity diagram were proposed by Shinkawa [10], Sapna et al. [12] and Chanda et al. [13]. Shinkawa [10] specify consistency between use case, activity, sequence and state chart diagram using Colored Petri Net (CPN). He proposes that a use case may have at least an activity diagram [10]. He also defines use case, action and execution occurrences as transitions. While Sapna *et al.* [12] define elements of use case, activity and sequence diagrams using schema table. But the definitions just limited to elements of use case, actor, activity, message and object. A use case may have an activity diagram [12], each actor in use case diagram is matched to a class in activity diagram [12]. They define that each object and its messages in sequence diagram correspond to a class and its method in class diagram. They proposed two (2) consistency rules between use case and sequence diagram. OCL is used to express the consistency rules. A use case may have different flows of activities or scenarios [8]. A scenario is described by a sequence diagram. Chanda *et al.* [13] express elements of use case, activity and sequence diagram as CFG. They define an action/ activity in activity diagram as an event of a use case in use case diagram. The formal syntax of each diagram is then used to reason the rules using CFG.

Research on consistency between use case and class diagram are most interest to researchers [11; 12]. Fryz et al. [11] consider a use case diagram as user requirements and they described the diagram as a graph. They have defined consistency between use case and class diagram using graphs. Elements defined in consistency rules by Sapna et al. [12] and Chanda et al. [13] are not follow abstract syntax standardize by Object Management Group (OMG)[15]. It is an advantage to use the standard with regard to apply any proposed approach in industrial software development [6].

The semantics presented in [17] captures the consistency between sequence diagram with class diagram and state diagram. This approach may be useful to develop the model consistent checking functions in UML CASE tools and also to reason about the correctness of a design model with respect to a requirement model. With respect to timing constraints in sequence diagram, Li et al. [16] describe an algorithm based on linear programming that analyzes whether several timing constraints within a sequence diagram are consistent with each other. They extend their approach to compositions of sequence diagrams.

3. SCOPE OF WORK

In this paper we propose a formal definition of Use Case, Sequence and Timing diagrams – the three most commonly used UML 2.0 models. Based on the UML 2.0 standards, we have defined several rules to highlight the inter-diagram consistency based on the common elements present. Using an example of UML 2.0 model which consists of the three diagrams, we show how the diagrams fulfill our proposed consistency rules. Finally, the elements involved in the consistency rules are detected and formally reasoned.

4. PROPOSED WORK

In this section, we describe the formalization of the elements of Use Case, Sequence and Timing diagrams. Further, the consistency rules are shown.

Definition 1. A model (or UML model) is defined as a set
Model = {UCD, SEQD, TIMED},
Where
UCD = $\{ucd_i|1 \le i \le n\}$ is finite set of Use Case diagrams.
SEQD = $\{SeqD_i|1 \le i \le n\}$ is finite set of Sequence diagrams for Use Case.
TIMED = $\{TimeD_i|1 \le i \le n\}$ is finite set of Timing diagrams for SEQD.
Definition 1 describes a UML model that consists of at least one Use Case diagram, one Sequence diagram and one Timing diagram.

4.1 Formalization of UCD

Definition 2. Use Case Diagram (UCD) is defined as a set
UCD = {A, UC, R, CO},
Where
A is a finite set of actors where A = $\{a_i|1 \le i \le n\}$,
UC is a finite set of Use Cases where UC = $\{uC_i|1 \le i \le n\}$,
R is a finite set of relationships where R = {Assoc, Include, Extend, GenUC, GenAc},
CO is a finite set of constraints for UCD where CO = $\{CO_i|1 \le i \le n\}$,
CO_i is a finite set of constraints for a Use Case uC_i where $CO_i = \{CO_{ij}|1 \le i \le n \text{ and } 1 \le j \le n\}$.

4.2 Formalization of SEQD

Definition 3: $SeqD_{uCi}$ is defined as a finite set of Sequence diagrams corresponding to a Use Case uCi.
$SeqD_{uCi} = \{SeqD_{uCi1}, SeqD_{uCi2,..........,} SeqD_{uCin}| uC \in UC\}$
where $SeqD_{uCi} \in$ SEQD

Definition 4: $SeqD_{uCin}$ is defined as a tuple
$SeqD_{uCin} = \{P_s, E, V, l, O, C, S\}$
where

- P_s is a set of lifelines denoting participants involved in an interaction where $P_s = \{ps_i| 1\le i \le n\}$.
- E is a set of events where each event corresponds to sending or receiving a message where $E = \{e_i |1\le i \le n\}$.
- V is a finite set of edges. V is defined as a link between two P_s. So V can be represented as {(e,e') | e,e' ∈ E and e'≠ e}. $V = \{v_i| 1\le i \le n\}$.
- l is a labeling function which assigns each v ∈ V a message name m with m = l (v).
- O is a function which maps each e ∈ E to the participant it belongs to.
- C is a set of Boolean of form $t(e) – t(e') \le d$ which represents the timing constraints enforced on $SeqD_{uCin}$.
- S is a finite set of states to which participant goes where $S = \{s_i |1\le i \le n\}$.

There is an ordering relation over E in a participant. All events related to one participant are timely ordered. For any two distinct events ei and ej, let ei < ej denote that ej occurs after ei if and only if (e,e') ∈ V.

We define N_{SeqD} to be the set of all message names occurring in the Sequence diagram and denote $N_{SeqD,p}$ set of all message names sent or received by the participant $p \in P_s$ of the Sequence diagram. We define $N_{SeqDstate,p}$ set of all states associated with the participant $p \in P_s$ of the Sequence diagram.

We denote the event of receiving message m_i as $r(m_i)$ and the event of sending message m_i as $s(m_i)$.

4.3 Formalization of TIMED

Definition 5: $TimeD_{in}$ describes a Timing diagram corresponding to a Sequence diagram $SeqD_{uCin.}$

$TimeD_{in}$ is defined as a tuple,
$TimeD_{in} = \{P_t, M, D, S\}$,
where

- P_t is a set of lifelines denoting participants involved in an interaction where $P_t = \{pt_i \mid 1 \leq i \leq n\}$.
- M is a set of messages transferred between two pt_i where $M = \{m_i \mid 1 \leq i \leq n\}$.
- D is the finite set of constraints where $D = \{d_i \mid 1 \leq i \leq n\}$.
- S is a finite set of states in the lifeline of P_t where $S = \{s_i \mid 1 \leq i \leq n\}$.

We define N_{timeD} to be the set of all message names occurring in the Timing diagram and denote $N_{TimeD,p}$ set of all message names sent or received by the participant $p \in P_t$ of the Timing diagram. We define $N_{TimeDstate,p}$ as a set of all states associated with the participant $p \in P_t$ of the Timing diagram.

4.4 Consistency Rules between Use Case and Sequence Diagrams

Rule 1: For each Use Case there exists at least one Sequence diagram.

Rule 2: Each actor associated with a Use Case will appear as a participant in the corresponding Sequence diagram.

Rule 3: Let UCD = {A, UC, R, CO} and $SeqD_{uCin.} = \{P_s, E, V, l, O, C, S\}$ be a sequence diagram corresponding to Use Case $uC_i \in UC$. For any Use Case $uC_i \in UC$, the set of constraints $CO_i \in CO$, associated with uC_i is the subset of the set of constraints in $SeqD_{uCin}$. $CO_i \subseteq C$.

4.5 Consistency Rules between Sequence and Timing Diagrams

Rule 4: For each sequence diagram there exists one Timing diagram.

Rule 5: Participants associated with a Timing diagram are a subset of the participants which appear in the corresponding sequence diagram. $p_t \subseteq p_s$ where $p_t \in TimeD_{in}$ and $p_s \in SeqD_{uCin}$.

Rule 6: Let $SeqD_{uCin} = \{P_s, E, V, l, O, C, S\}$ be a sequence diagram and $TimeD_{in} = \{P_t, M, D, S\}$ be a Timing diagram. $\forall p \in P_t$, the set of messages names in $TimeD_{in}$ ($N_{TimeD,p}$) is a subset of the set of messages in $SeqD_{uCin}$ ($N_{SeqD,p}$). $N_{TimeD,p} \subseteq N_{SeqD,p}$.

Rule 7: Let $SeqD_{uCin} = \{P_s, E, V, l, O, C, S\}$be a sequence diagram and $TimeD_{in} = \{P_t, M, D, S\}$ be a Timing diagram. $\forall p \in P_t$, the set of states in $TimeD_{in}$ ($N_{TimeDstate,p}$) is same as the set of states in $SeqD_{uCin}$ ($N_{SeqDstate,p}$). $N_{TimeDstate,p} = N_{SeqDstate,p}$.

Definition 5: The Sequence of message events induced by a sequence diagram $SeqD_{uCin} = \{P_s, E, V, l, O, C, S\}$ related to one participant $p \in P_s$ between two events e and d, where e, d $\in$ E is the ordered sequence of message events between these two events:

Sequence (e,d) = $\langle a_i \rangle$ such that following hold

- $O(a_i) = p$, all events are associated to the participant p.
- $a_i = r(m)$ or $a_i = s(m)$ with m = label (v) for one $v \in V$, all events are send or receive events of messages of the sequence diagram.
- $e \leq a_i \leq d$ and $i \leq j$ implies $a_i \leq a_j$, the sequence is ordered.

Rule 8: Let $SeqD_{uCin} = \{P_s, E, V, l, O, C, S\}$be a sequence diagram and $TimeD_{in} = \{P_t, M, D, S\}$ be a Timing diagram. The sequence of messages generated by any participant in $TimeD_{in}$ is a subsequence of a sequence of message events related to the corresponding participant between the first and the last event of that participant in $SeqD_{uCin}$.

Rule 9: Let $SeqD_{uCin} = \{P_s, E, V, l, O, C, S\}$ be a sequence diagram and $TimeD_{in} = \{P_t, M, D, S\}$ be a Timing diagram. Suppose in $SeqD_{uCin}$, the sequence of states associated to any participant $p \in P_s$ is $\langle s_1, s_2,..,s_n \rangle$, then in $TimeD_{in}$, the same sequence of states will be related to the corresponding participant $p \in P_t$.

Rule 10: $SeqD_{uCin} = \{P_s, E, V, l, O, C, S\}$ be a sequence diagram and $TimeD_{in} = \{P_t, M, D, S\}$ be a Timing diagram. In $SeqD_{uCin}$, if participant A enters into state S1 at time TA1 and remains in that state until time TA2 then in the corresponding $TimeD_{in}$, participant A changes its state to S1 at time TA1and duration of that state will be (TA2-TA1) time unit. Sometimes, this duration is specified by some constraints.

Rule 11: Let $SeqD_{uCin} = \{P_s, E, V, l, O, C, S\}$ be a sequence diagram and $TimeD_{in} = \{P_t, M, D, S\}$ be a Timing diagram. If any two messages in $SeqD_{uCin}$ are associated with duration constraint such as d, then in $TimeD_{in}$, the corresponding messages must be related to same duration constraint (such as d). Suppose in $SeqD_{uCin}$, the sequence of states associated to any participant $p \in P_s$ between these two messages are $\langle s_1, s_2,..,s_n \rangle$, then in $TimeD_{in}$, same sequence of states will be related to the corresponding participant $p \in P_t$.

5. RESULT AND DISCUSSION

This section shows an example of UML model for ATM System. The model consists of UML 2.0 Use Case, Sequence and Timing Diagrams. The elements of each diagram are described and how the diagrams fulfilled our proposed consistency rules are also shown.

5.1 UML Model for ATM System

The requirements of ATM system are captured and visualized using a Use Case diagram as shown in Figure 1. The functionalities of each Use Case in Figure 1 are then modelled using at least one Sequence diagram. In order to show how UML diagrams fulfilled our proposed consistency rules, we just show one Sequence (Figure 2) and one Timing diagram (Figure 3) for Withdrawal Use Case.

Based on **Definition 1**, we get

$Model_{ATM} = \{ucd_{ATM}, SeqD_{withdrawal0}, TimeD_{withdrawal0}\}$.

5.2 UML Use Case Diagram for ATM System

Figure 1 shows a Use Case diagram for ATM System.

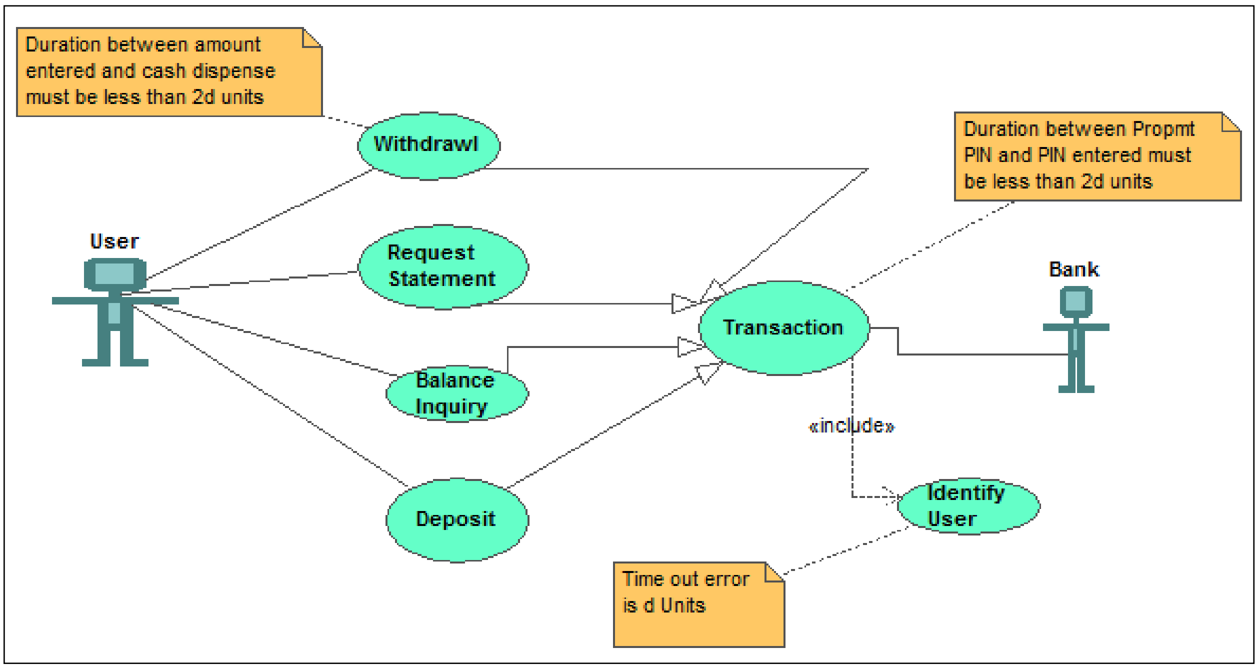
Duration between amount entered and cash dispense must be less than 2d units
Withdrawl
Duration between Propmt PIN and PIN entered must be less than 2d units
User
Request Statement
Bank
Transaction
Balance Inquiry
«include»
Deposit
Identify User
Time out error is d Units

Fig 1: Use Case Diagram for ATM System.

- Based on **Definition 2**, we have the following
  $ucd_{ATM} = (A_{ATM}, UC_{ATM}, R_{ATM}, CO_{ATM})$
  $ucd_{ATM} \in UCD_{ATM}$
- Two actors are present in Use Case ATM. They are customer and Bank, i.e.,
  Based on **Definition 2**, we have the following
  $A_{ATM} = \{a_{Customer}, a_{Bank}\}$
- Six Use Cases named Withdrawal, Request Statement, Balance Inquiry, Deposit, Transaction, Identify User, i.e.,
  Based on **Definition 2**, we have the following
  $UC_{ATM} = \{useCase_{Withdrawal}, useCase_{Request\ Statement}, useCase_{Balance\ Inquiry}, useCase_{Deposit}, useCase_{Transaction}, useCase_{Identify\ User}\}$
- Based on **Definition 2**, we have the following
  $R_{ATM} = \{Assoc_{ATM}, Include_{ATM}, GenUC_{ATM}\}$.
  In the diagram, there is no generalization of actors and <<extend>> relationship between Use Cases.
- Based on **Definition 2**, we have the following
  CO_{ATM} = {Duration between amount entered and cash dispense must be less than 2d units, Duration between Prompt PIN and PIN entered must be less than 2d units, Time out error is d unit}.

5.3 UML Sequence Diagram for ATM System

Figure 2 shows a Sequence diagram ($SeqD_{Withdrawal}$) for Withdrawal Use Case depicted in figure 1.

- Based on **Definition 3**, we have the following
  $SeqD_{uCWithdrawal} = \{SeqD_{Withdrawal}\}$
- Based on **Definition 4**, we have the following
  $SeqD_{Withdrawal} = \{P_{sWithdrawal}, E_{Withdrawal}, V_{Withdrawal}, l_{Withdrawal}, C_{Withdrawal}, S_{Withdrawal}\}$
- Three participants are present in Sequence diagram $SeqD_{Withdrawal}$
  $P_{sWithdrawal}$ = {User, ATM_Sys, Bank}.
- Some elements of $E_{Withdrawal}$ are
  s(Insert Card), r(Insert Card), s(Enter PIN), r(Enter PIN), s(Verify Account), r(Verify Account) etc.
- Some elements of $C_{Withdrawal}$ are
  {t(r(PIN Prompt)) – t(s(Enter PIN)) <d}, {t(s(Enter Amount)) - t(r(Dispense Cash)) <2d} etc.
- $S_{Withdrawal}$ = {Idle, Processing, Waiting}.
- Some messages are
  Insert Card, Enter PIN, PIN Prompt, Verify Account, Enter Amount, Dispense Cash etc.

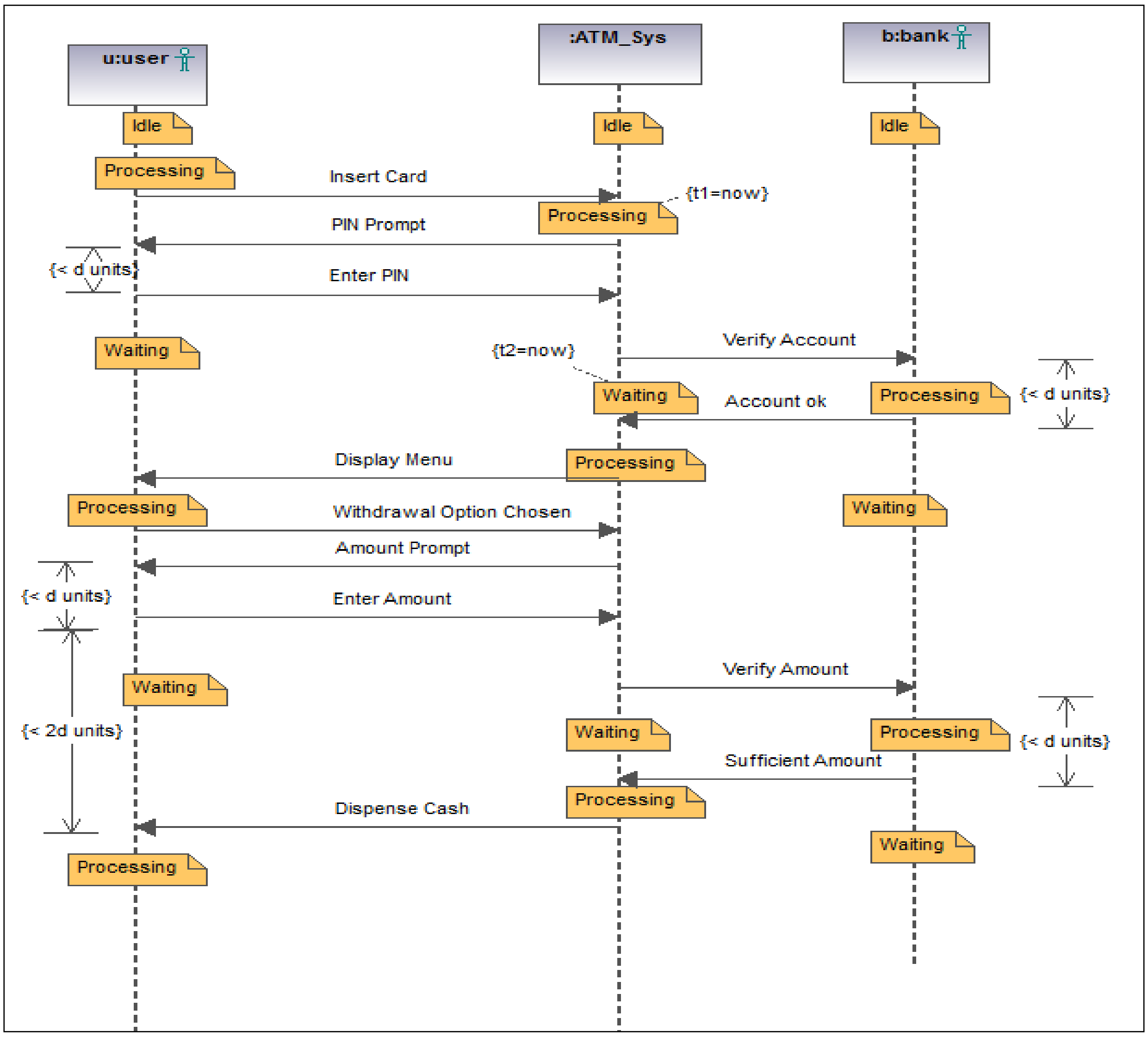

Fig 2: Sequence Diagram for Withdrawal Use Case

5.4 UML Timing Diagram for ATM System

Figure 3 shows a Timing diagram (TimeD$_{Withdrawal}$) for Sequence diagram SeqD$_{Withdrawal}$.

- Based on **Definition 5**, we have the following TimeD$_{Withdrawal}$ = {P$_{tWithdrawal}$, M$_{Withdrawal}$, D$_{Withdrawal}$, S$_{Withdrawal}$}
- Three participants are present in Timing diagram TimeD$_{Withdrawal}$
  P$_{tWithdrawal}$ = {User, ATM_Sys, Bank}.
- Some elements of M$_{Withdrawal}$ are
- Insert Card, Enter PIN, PIN Prompt, Verify Account, Enter Amount, Dispense Cash etc.
- Some elements of D$_{Withdrawal}$ are
  {t(r(PIN Prompt)) – t(s(Enter PIN)) <d}, {t(s(Enter Amount)) - t(r(Dispense Cash)) <2d} etc.
- S$_{Withdrawal}$ = {Idle, Processing, Waiting}.

5.5 Consistency Rules between Use Case and Sequence Diagrams

a. For Withdrawal Use Case in Fig 1, there is a Sequence diagram in Figure 2. Fig 1 and 2 satisfy the **Rule 1.**

b. According to **Rule 2,** User and bank of use case diagram in Fig 1 appear as participants in Sequence diagram in Fig 2.

c. Withdrawal Use Case in Fig 1 is associated with some constraints which are reflected in Fig 2.

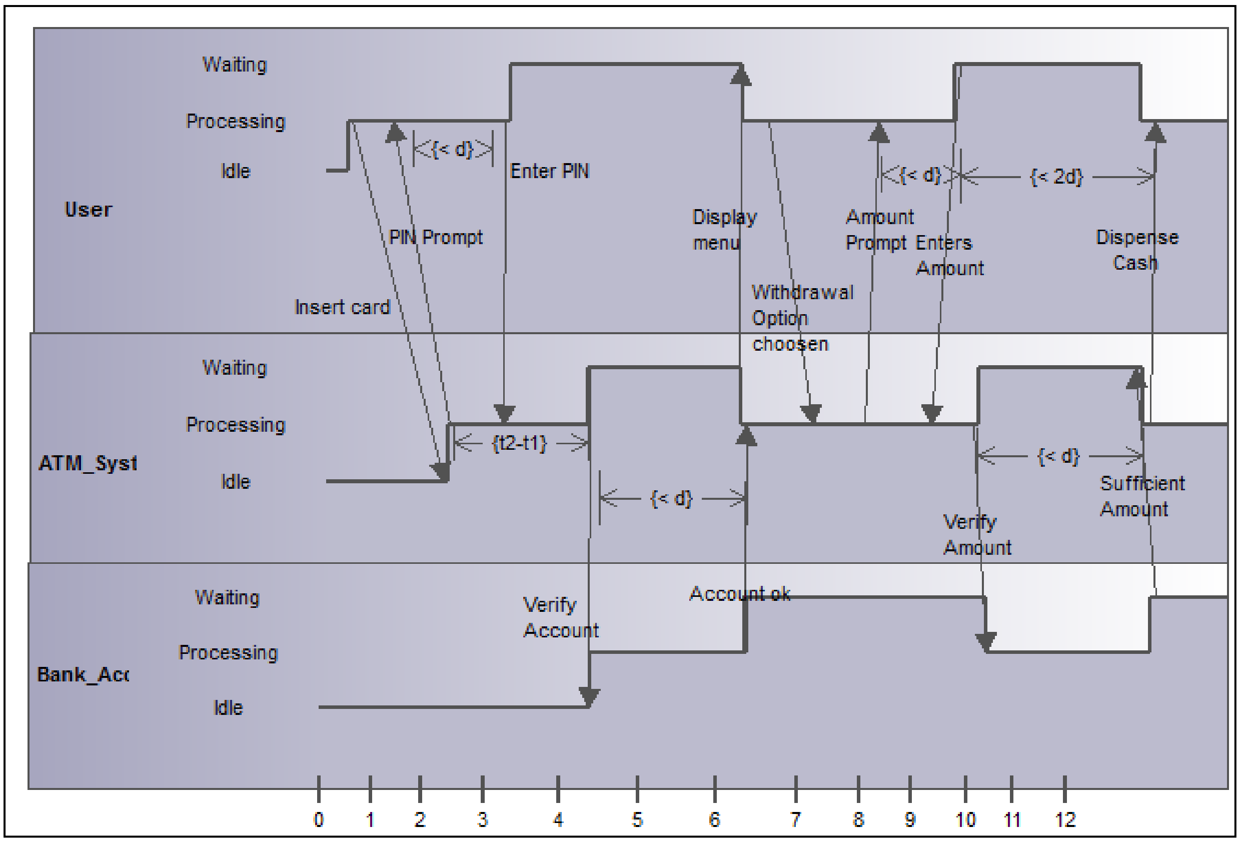

Fig 3: Timing Diagram corresponding to the Sequence Diagram for Withdrawal Use Case

5.6 Consistency Rules between Sequence and Timing Diagrams

d. Fig 2 and 3 satisfy the **Rule 4**.

e. According to **Rule 5**, we require that the number of participants in a Timing diagram is a subset of participants appears in corresponding sequence diagram. With respect to our example, both diagrams (Fig 1 and Fig 2) contain three participants.

f. According to **Rule 6**, we require that the set of message names associated to any participant of the Timing Diagram to be a subset of the set of messages related to the corresponding participant in the Sequence diagram.

With respect to our example, the set of message names of the Timing diagram associated to participant bank in Fig 3 is {Verify Account, Account Ok, Verify Amount, Sufficient Amount} which is same as the set of messages of the corresponding participant in the Sequence diagram.

g. According to **Rule 7**, we require that the set of states associated to any participant of the Timing Diagram is same as the set of states related to the corresponding participant in the Sequence diagram.

With respect to our example, the set of states of the Timing diagram associated to participant User in Fig 3 is {Idle, Processing, Waiting} which is same as the set of states of the corresponding participant in the Sequence diagram.

h. According to **Rule 8**, we require that the sequence of messages generated by any participant in TimeDin is a subsequence of a sequence of message events generated by the corresponding participant in SeqDuCin. We can say that sequence of messages is same in both diagrams.

In Fig 2, sequence of messages of the Sequence diagram related to participant User is <s(Insert Card), r(PIN Prompt), s(Enter PIN), r(Display Menu), s(Withdrawal Option Chosen), r(Amount Prompt), s(Enter Amout), r(Dispense Cash)>. In Figure 3, it is noticed that User is associated with the same sequence of messages.

i. According to **Rule 9**, we require that sequence of states is same in both diagrams.

With respect to our example, the sequence of states of the Timing diagram associated to participant ATM_Sys in Fig 3 is {Idle, Processing, Waiting, Processing, Waiting, Processing} which is same as the sequence of states of the corresponding participant in the Sequence diagram.

j. **Rule 10** requires that if any participant p ∈ Ps enters into state S1 at time T1 and exits that state at time T2 then in the Timing diagram, corresponding participant p ∈ Pt changes its state to S1 at time T1and duration of that state will be (T2-T1) time unit.

In Fig 2, participant ATM_Sys enters into the Processing state at time t1 and exits that state at time t2. We use the new metaclass in UML 2.0, TimeObservationAction, to know when a participant changes its state. A time observation action is an action that, when executed, returns the current value of time in the context in which it is executing. It is depicted with the keyword "now". In Figure 3, corresponding participant remains in that state (Processing) for the duration (t2-t1) time unit.

k. According to **Rule 11**, first we require that any two messages are associated with same duration constraint in both the diagrams. Next we require that the sequence of states related to any participant is the same in both diagrams between these two messages.

   In Fig 2, there is a duration constraint {< d units} between two messages Enter Amount and Dispense Cash. In Sequence diagram, participant ATM_Sys is associated with sequence of states {Processing, Waiting, Processing} between these two messages.

   In Timing diagram same scenario is repeated.

6. CONCLUSION

The increasing complexity of now-a-days ubiquitous Real Time Systems requires an adequate modeling language. UML, which is a widely used visual object oriented modeling language, has proved to be effective and suitable for Real Time Systems. However UML is semiformal in nature and hence ambiguities may arise in design specifications among models that represent overlapping but different aspects of the same system. Consistency between the diagrams is important for the successful implementation of a model, especially when existing components have to be integrated. Ensuring consistency among different models representing different phases of its life cycle are of utmost importance.

In this paper we propose formal definitions for UML 2.0 Use Case, Sequence & Timing diagrams, the three widely used models which represent dynamic and behavioral aspects. In this paper, a set of consistency rules are defined which focuses on timing aspects of Real Time Software Systems. We have considered ATM system as our example and our approach has been applied to this case study and our proposed consistency rules are satisfied.

7. REFERENCES

[1] Boris Litvak, S. T. and Yehudai. A Behavioral consistency validation of uml diagrams. First International Conference on Software Engineering and Formal Methods (SEFM'03), Brisbane, Australia, page pp 118, September 22-27 2003.

[2] Egyed, A. Scalable consistency checking between diagrams-the view integra approach. 16th IEEE International Conference on Automated Software Engineering (ASE'01), San Diego, California, November 26-29 2001.

[3] G. Spanoudakis, & A. Zisman. Inconsistency Management in Software Engineering: Survey and Open Research Issues, Handbook of Software Engineering and Knowledge Engineering, ed., World Scientific Pub. Co, New Jersey, 2001.

[4] Nugroho, & M. R. V. Chaudron. A survey into the rigor of UML use and its perceived impact on quality and productivity, in: Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, 2008, pp. 90-99.

[5] Hubaux, A. Cleve, P.-Y. Schobbens, A. Keller, O. Muliawan, S. Castro, et al. (2009). Towards a Unifying Conceptual Framework for Inconsistency Management A F. J. Lucas, F. Molina, A. Toval.: A Systematic Review of UML Model Consistency Management, Information and Software Technology, 51, 2009, 1631-1645.

[6] F. J. Lucas, F. Molina, A. Toval. A Systematic Review of UML Model Consistency Management, Information and Software Technology, 51, 2009, 1631-1645.

[7] D. Rosenberg, M. Stephens. Use Case Driven Object Modeling with UML: Theory and Practice, A press, 2007.

[8] J. W. Satzinger, R. B. Jackson, S. D. Burd. Object-Oriented Analysis and Design with the Unified Process, Thomson Course Technology, 2005.

[9] B. Dobing, J. Parsons. Dimensions of UML Diagram Use: A Survey of practitioners, Journal of Database Management, 19, (1), 2008, 18.

[10] Y. Shinkawa.: Inter-Model Consistency in UML Based on CPN Formalism, in: 13th Asia Pacific Software Engineering Conference (APSEC '06) 2006, pp. 414-418.

[11] L. Fryz, & L. Kotulski. Assurance of System Consistency during Independent Creation of UML Diagrams, in: 2nd International Conference on Dependability of Computer Systems, 2007 (DepCoS-RELCOMEX '07), 2007, pp. 51-58.

[12] P. G. Sapna, H. Mohanty. Ensuring Consistency in Relational Repository of UML Models, in: 10th International Conference on Information Technology (ICIT 2007), 2007, pp. 217-222.

[13] J. Chanda, A. Kanjilal, S. Sengupta, S. Bhattacharya.Traceability of Requirements and Consistency Verification of UML UseCase, Activity and Class diagram: A Formal Approach, in: International Conference on Methods and Models in Computer Science 2009 (ICM2CS), 2009, pp. 1-4.

[14] Küster, J. M. ,Stehr, J. Towards explicit behavioral consistency concepts in the uml. Second International Workshop on Scenarios and State Machines: Models, Algorithms and Tools, Portland,Oregon, USA, May 3 2003.

[15] Object Management Group (OMG), OMG Unified Modeling LanguageTM (OMG UML), Superstructure Version 2.3. Retrieved from: <http://www.omg.org/spec/UML/2.3>, 2010.

[16] Xuandong Li , Johan Lilius.: Timing analysis of UML sequence diagrams. In Robert France and Bernhard Rumpe, editors, UML'99 - The Unified Modeling Language. Beyond the Standard. Second International Conference, Fort Collins, CO, USA, October 28-30, 1999, Proceedings, volume 1723 of LNCS, Pages 661-674, Springer, 1999.

[17] Xiaoshan Li, Zhiming Liu, He Jifeng.: A Formal Semantics of UML Sequence Diagram. Australian Software Engineering Conference (ASWEC'04), Melbourne, Australia, April 13-16, 2004.