Global Journal of Computer Science and Technology
Volume 11 Issue 22 Version 1.0 December 2011
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

# Representing Aspect Model as Graph Transformation

By Vishal Verma, Ashok Kumar
*Kurukshetra University, Kurukshetra*

*Abstract* - In this paper we discussed a new method for representing aspect models. This method uses the basics of UML to devise a new way for specifying the model level aspects and transformations among them. The resultant model is effective from both expression and scaling point of view. The work in this paper is based on assumed transaction processing system in a bank.

*Keywords : Aspect - Oriented, Graph Transformation, UML.*

*GJCST Classification : D.2.9*

# Representing Aspect Model as Graph Transformation

Vishal Verma [α], Ashok Kumar [Ω]

*Abstract -* In this paper we discussed a new method for representing aspect models. This method uses the basics of UML to devise a new way for specifying the model level aspects and transformations among them. The resultant model is effective from both expression and scaling point of view. The work in this paper is based on assumed transaction processing system in a bank.

*Keywords : Aspect - Oriented, Graph Transformation, UML.*

## I. Introduction

In most of the software development techniques identification and presentation of aspects is done only at some specific levels which pose constraints on the designer and developers to follow a predefined pattern/steps for development process. In this method we try to develop a technique which can be used for representing and composing aspect at any level of software development. With the advent of new techniques for software development it is quite common and natural, that aspect can occur during any of the development phase i.e. requirement [1], analysis [8] and design [12]. Aspect if modularized during software modeling can leads to a clear boundary among aspects and concerns and they become more maintainable, understandable and organize-able within the model. On the other hand if aspect modules are composed with the development of base module then it helps to fully understand and analyze the model with aspects, and any ambiguity, conflicts and omissions can be avoided. Hence, the mechanism used for specification of aspect at the modeling level must be complemented with mechanism used for composition, that weave the aspect model into base model. Lack of expression and scalability are the major problems faced by the researchers for development of mechanism like this. Composition at the modeling level can be extremely rich in nature [14]. Existing models do not provide support for expressing the richness in compositions. However, increase in the degree of expressiveness can lead to the problem of scalability because a large effort is required by the developers to specify the composition. The method discussed in this paper is capable to handle the problem of scalability and expressiveness and the result of this paper is a practical technique that can be used for defining and composing aspect oriented model for best modeling purpose.

The method used in this model is based on two basic technology i.e. Role Based Meta Modeling language [2] and graph transformation [3, 5]. Role Based Meta Modeling language provides a precise, simple graphical means for specifying a model level aspect in a way that is consistent with UML [13]. It is used for modeling the structural part of security aspects [6] as well as model behavioral UML aspects [8]. The base problem faced while using RBML is that they do not scale up to marks since a lot of effort is required to specify the cross cuts among the core modes. Our discussed method shows clearly the reduction in level of effort to be done for models. Transformations using graphs have been applied in a number of problems related to the software engineering and to the problem of merging of different systems together [9], but in none of the implementations it has been categorically addressed how to apply them, in general way, to handle the aspect at any level of UML modeling. The aim of this paper is to combine together the RBML and graph transformations to achieve

a) General implementation of UML based aspect modeling and composition at any stage of abstraction.
b) To implement the proper scalability of aspect composition.

This paper illustrates the approach with an assumed transaction execution system based closely on an existing application used by banks.

## II. Model Level Aspects

Aspect oriented models are models which represent the cross cut, points cut and concern in a well arranged manner along with aspects. From the view point of problem discussed in this paper it can be defined as a model that crosscuts other model at the same level of abstraction. Here the words “same level of abstraction ” plays very important role i.e. a model is considered to be an aspect if it crosscut the other model of same interactions e.g. if requirement cut requirement model, requirement artifact cuts requirement artifact only then they are considered as aspects. In particular case a use case may not be aspect. Although a use case is suppose to always cut

*Author [α] : Department Of Computer Science and Application, Kurukshetra University Post Graduate Regional Centre, Jind.*
*E-mail : vishal.verma@kuk.ac.in*
*Author [Ω] : Department Of Computer Science and Application, Kurukshetra University, Kurukshetra.*

across multiple implantations module, it is only considered to be an aspect if it cuts across other use case.

Discussion in this paper is restricted to the definition of an aspect oriented model with a condition that a model is an aspect only if it crosscuts other model built with same perspective e.g. any model which is build for global interpretation of interaction cannot cut a model build with local interpretation of interaction and hence is not at same level of abstraction but they have different perspective- local and global. These types of models are not considered in this paper.

### a) *Representation of Aspect in Role Based Modeling Language*

Role Based Modeling Language [2] is used in this paper to represent the Aspect-Oriented Modules. This language is further complemented by France et al [10]. RBML is considered as a special case of UML Meta model in which each element of RBML is treated as a role. It is also considered that a role is a constraint of a UML Meta class with a set of optional properties that any element must possess. Because RBML is considered as a special case of UML hence each UML diagram must have a corresponding RBML diagram in which model elements are roles e.g. state roles and transitions of RBML represents a generic state that can be made concrete by assigning it to a concrete role. Proper care is to be taken that only those model elements which satisfy the property of a role should be treated as a role. RBML model defines a generic model that can be instantiated in many ways by assigning elements to its entire role [14]. Any UML model is said to conform to a RBML model if there is a valid argument of elements in the UML model to the roles of RBML model [14].

RBML model is used to formalize the design pattern [2] and to represent model level aspect [4]. This was extended to behavioral aspect in [8] and [7]. As per the original definition in [10] all RBML model elements must be roles i.e. they are Meta – level elements. As per [8] for representing aspects it is useful to allow object-level elements in RBML as well. The result is an extended RBML, represented by eRBML, in which an element may be Meta level or object level element [14]. Fig.1 shows the sequence of aspect in eRBML. It clearly shows that whenever the user get ack of failed transaction the HOST itself record the status in STATUS file and at the same time shut down the USER side as well. Fig 1 shows the combination of object level elements meta-level role together in one go. This type of combination is preferred since status like objects are remains unchanged and their relative updation dependent on the varying values of roles only.

i. *Instantiation: it means to assign some concrete*

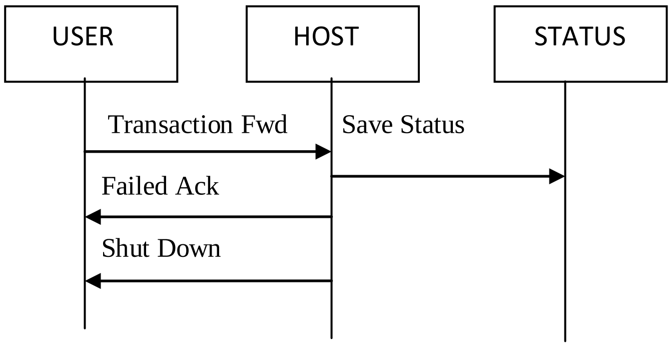

*Fig.1 :* Handling of Failed Transactions

values to the elements or one – to – one mapping from role to model elements. In context of eRBML each aspect model must be instantiated before it can be composed with a base model. Instantiation is basically used to define what the aspect should like in context of a particular application i.e. the aspect is identified and specialized to a context. Fig.2 shows another example of how aspects cross cuts each other. Sequence diagram in Fig. 2 is taken as base for further discussion and is part of our case study in coming sections. From fig. 2 it is enough to conclude that there is a controller which keeps control of accessing request from user and sending it to the server for processing. Controlling all aspect of transaction is the sole responsibility of the

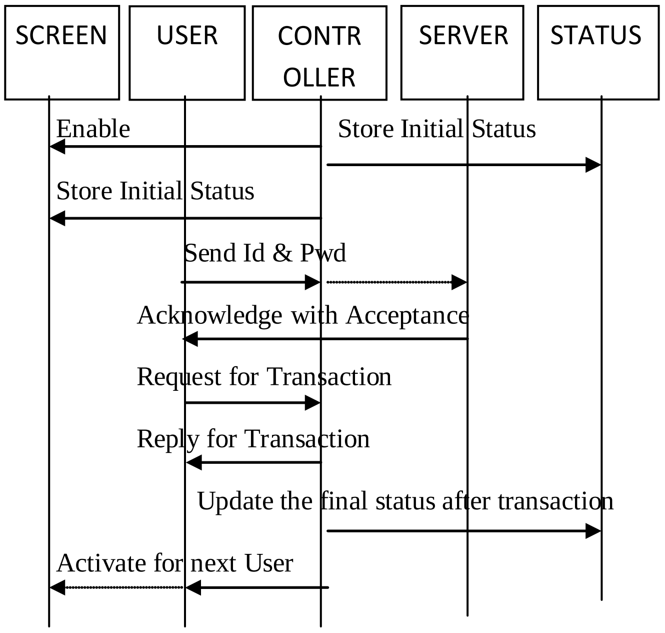

*Fig. 2 :* Base Sequence Diagram for user Transactions

controller, it also provide the necessary GUI for processing. Failure handling is not considered as part of this discussion. Instantiation is used to propose the aspect for composition with the base. In the example discussed here following instantiations are specified by the modeler: | USER -> CONTROLLER| CONTROLLER-> SERVER and failure are not considered.

Fig.3 shows the result of composition of Fig. 1 and Fig. 2 and instantiate the aspect with base modeler. Message to deal with intermediate REQ is included and is provided as an alternate execution path using UML 2.0 alt interaction fragment.

ii. *Conformance*

A UML model is said to conform w.r.t. eRBML if their exist an instantiation of eRBML model in a way that all elements of instantiated eRBML are present in UML model along with existence of constraints. The constraints are suppose to include the message ordering, sequence diagram, transition ordering and additionally specified properties of eRBML roles with respect to an UML model there may exist any number of different eRBML models that conforms dependency upon the availability of additional transaction. There may be any number of additional transactions that exist in between starting and closing transaction i.e. intermediate transactions. Hence conformance should be considered as a type of refinement.

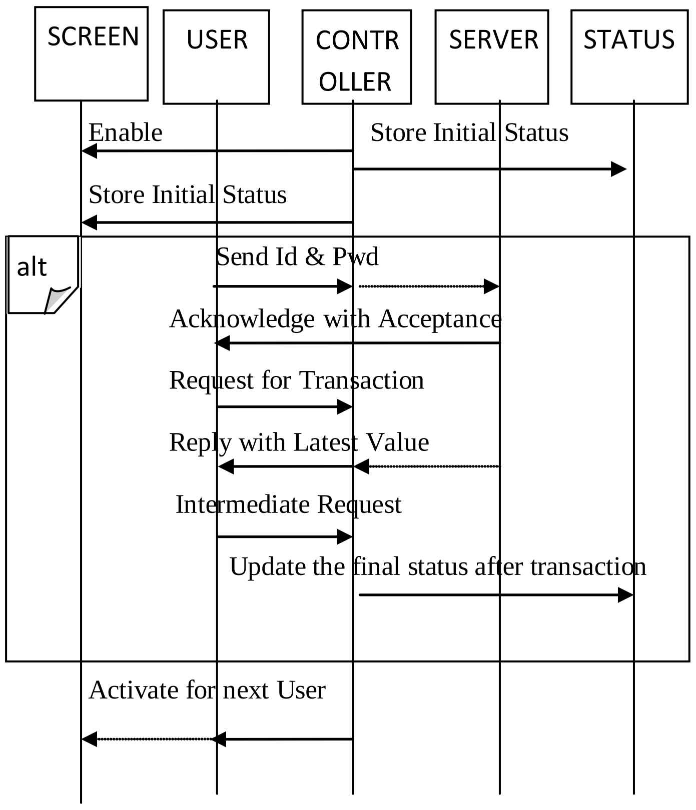

*Fig. 3 :* Composite of Fig.1 and Fig. 2

*b) Aspect Composition*

Model level aspect can be specified in a well defined way in eRBML with respect to the aspects of any UML diagram. As it is necessary to represent the model aspect in a modular fashion, composing the model aspect with base model is also important. Here we are comparing the composition approaches of France et al [11] and Whittle [8] to identify the limitations of existing method to model. Fist one out of these two approaches use templates to represent aspects. Instantiation of eRBML aspect before composition is mandatory in both of the approaches, though both of the approaches [11] and [8] have different way of implantation of composition.

We have discussed a single method for composition in Fig. 3. This figure though uses simple technique, yet there are many alternates by using which composition could be done. In fig.3 the intermediate transaction is introduced which can be placed at different level of execution and can produce different compositions accordingly. Although it is simple in nature, it may be not be suitable in many cases. Common limitations of this method is that it is not able to specify the fact that how the aspect messages should be interleaved with the base model, or to specify that the aspect messages define a sequence executed in parallel with the base model message. As an alternative there are many possible ways that composition could occur. Challenging part to find a way for specifying composition that admits a high degree of expressiveness, with minimum effort to be applied for modeling. To find the response on expressiveness and scalability below we compare the techniques discussed in [11] and [8].

The method discussed in [11] allows the modeler to describe the composition directive that finally tailors the tailor algorithm. These composition directives permit the user to specify the aspect message interleaved with base or as an alternative or to run in parallel of it. "Addition", "deletion", and "move", statements are supposed to be used as directives to make the composed model. To merge the base and instantiated model first of all their elements with the same name are merged together. On completion of merging of elements with same name in first go directives are tailored to drive the exact form of composition. This all method of tailoring demands a lot pressure on modeler. Manual composition in this way demands a composition to be implemented by first applying the directive in each and every model's element also in the base model. This type of procedure can't be scaled at all.

In contrast to this the method discussed in [8] is at higher level of abstraction. In this method the composition operators are used instead of composition directives. Specifically AND, OR and IN operators are used. AND operator is used to interleave the base model with aspect model. OR can be used to provide the alternative sequence among base and aspect model. IN has some special use and it is used to insert the aspect message in any base sequence. Operators used in this approach offer a high level view of composing aspect models. This approach is more suitable and easy then the one discussed in [11] since user is not required to work with element by element manner. The only drawback of this approach is limited

Global Journal of Computer Science and Technology Volume XI Issue XXII Version I December 2011

Global Journal of Computer Science and Technology Volume XI Issue XXII Version I December 2011

number of operators for performing all desired tasks.

## III. New Method of Composition

Techniques discussed in sec 2 have limitations with respect to scalability. In that technique it is the sole responsibility of modeler to specify a set of role instantiations for each aspect and for each base model that is cross cut by aspect. It is obvious that for large aspect more number of instantiations is required to be supplied. From fig. 3 it is clear that all the instantiations are given in non graphical and in low level format that are time consuming to understand and ultimately make the maintenance of the model more difficult for the user. In comparison the method discussed in this paper provide a clean and clear way of separation of aspects and the base model. The newly discussed technique provides a new way of representing and composing model aspects in a way that maintains aspect modularity along with scalability. Basically graph transformation rules are used for representing composition and is represented by a rule (L -> R) bearing left hand side and right hand side. Left side is responsible for keeping points where the aspect should be applied and right side keeps the aspects in it.

### *a) Aspect as Graph Transformation*

A graph transformation discussed in [5] is a rule represented by r and has L as left hand side and R as right hand side. Rule r is supposed to be applied on a graph G and the process of applying r finds a graph homomorphism, h, [5] from L to G and replacing h (L) in G with h(R). To avoid any kind of unreferenced edges i.e. edges with missing resources or target node - L(R) is applied into G in such a way that all edges connected to a removed node in h(L) are reconnected to a replacement node in h(R).

UML diagram can be represented in the form of a graph because it is defined by the UML meta-model which is a graph where the nodes are Meta classes and the edges are meta-relationships [13]. Hence it is possible to represent transformation over UML model as graph transformation. Particularly we see composition of an aspect model with a base UML model as a graph transformation LHS and RHS both are eRBML models. As above L side specifies the points where the aspect should be applied and the R side specifies the crosscutting structure/behavior that should be inserted at those points.

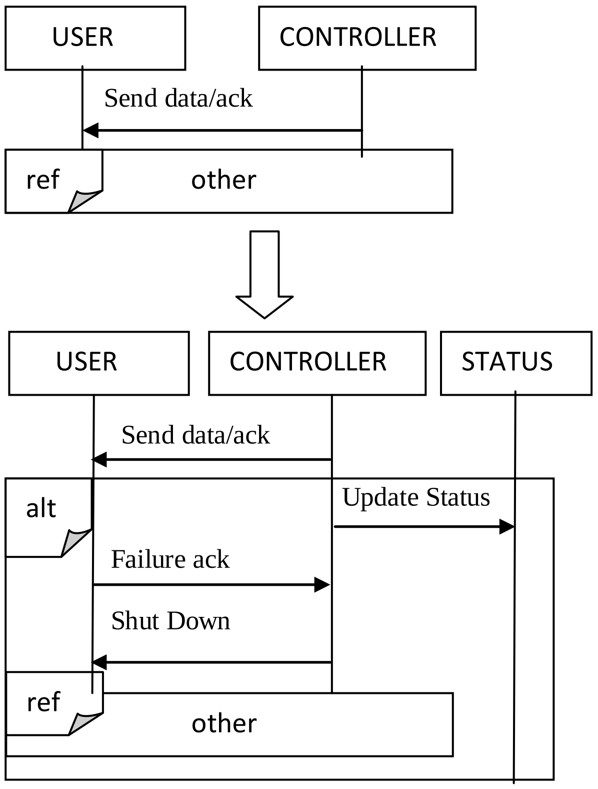

*Fig 4 :* Composition as Graph Transformation and handling Failure Aspects

Fig 4 represents how aspects from fig. 1 can be represented by using graph transformation. It shows two parts of aspects definition. Left side defines the aspects itself and Right side defines the composition strategy. On applying composition it would become possible to deal with message for future that is inserted as an alternative sequence after all instances of send data/ack, a message sent from CONTROLLER to USER. The approach used here helps to define the expressiveness and scalability related to composition in an easy way. It becomes clear from fig. 4 that it become possible to keep a complete separation of the aspects and its composition strategy. It helps to reuse of the aspects and application of the same aspects with different purpose and different composition strategy. This technique is a fully expressive way of defining composition strategy – as one is shown in Fig. 4(it is one other alternative may be used). This strategy uses the number of instantiations required to design a model. In the example discussed in Fig . 4 only one instantiation is required to be provided by the modeler i.e. failure ack. Rest all roles can be instantiated by graph matching against a base mode, the left side of the graph transformation is required to be matched with base model instantiating USER, SERVER , send data/ack (only failure ack is required to be instantiated). In fig.4 the UML 2.0 ref fragment is used to specify the placeholder for a sequence of messages in the base. This is an easy way to match against a message sequence whose position in the composed model can then be specified exactly on the R.H.S of the transformation. At the time of defining the graph transformation w.r.t eRBML, the base definition must be modified. When matching against the Left side of the transformation rule r, it is mandatory to discuss the instantiation for the role elements. In this context the base definition is modified as the graph transformation applies to a UML model if and only if the Left side of transformation has a graph match i.e. module conformance exist there. The method in terms of scalability and expressiveness can be defined as:

i. *Scalability*

Main limitation of the scalability of all aspect approaches based on RBML is that the modeler must instantiate the role elements for each base model crosscut by the aspects. Use of graph transformations reduce this effort because instantiating the role elements become automated to some extent. Instantiation place a need to find a base model over which graph transformation can be applied- i.e. finding a match for left hand side of the transformation rule. As per above discussion we apply the module conformance while applying an aspect i.e. while working with the eRBML model, R (rule), match a UML model say U, modulo conformance if and only if there is an instantiation of the role element Ø, such that Ø(R) conforms to model U. as is clear from Fig. 4 it has modulo conformance with the base model and hence problem of scalability is managed well by graph transformations.

ii. *Expressiveness*

As shown in Fig. 4 and sec 3.1 the Right side of the graph transformation rule, r, defines the manner in which the aspect cross cuts a base model. Since Right side is a model in itself, it completely reflects the expressiveness and how the cross cutting is defined. Here the aspect messages can be defined as an alternative to a base message or messages, as interleaved with the base message, accessing in parallel with the base message or any other combination of above discussed alternatives. The composition operator discussed in sec 2.2.2 can be defined as special case but graph transformation allows any combination of these operators to be specified, or needed for new operators to be specified. The composition directive in sec 2.2.1 are subsumed by the graph transformation approach because there is no longer any need to tailor the aspect composition algorithm to add, delete or remove elements - these modifications are rather defined explicitly in the Right side of the transformation.

## IV. Case Study

For the purpose of doing the case study of the expected system to be developed, we assume a system in general which is responsible for processing of transactions raised by user in terms of bank transactions. Every user of bank is supposed to execute a set of queries (may be predefined) for completion of desired tasks. We assume a system for study in which each user is required to first authenticate him/herself for executing other transactions. After authentication use is provided with a GUI by using which rest all requests can be processed. Some of the simplest form of query is deposit and withdraw of amount and to get a balance or mini statement from the bank. In all of this type of queries a proper integrity among user interface window and ATM machine is mandatory i.e. any transaction which affect the balance in the account must be effective at all place and do the final status change at some common location. These type of queries are expected to be executed form ATM machine, from online based banking system, mobile based banking system or from a window in a bank's office where a bank officer is supposed to execute desired on verification of credentials from user. Important among all these alternatives of query execution is that they must do final status change at common location which is accessed by all means of query execution and all the time latest updated value must be available at that location i.e. SERVER. Data integrity is clearly an important issue to be maintained in design of this type of systems. Any user who is permitted to use his account by a number of means is dependent on one central location i.e. SERVER for latest updated values. The design of this system is done by using the two – phase commit protocol for maintaining serializability among the transactions to keep the commonly used values updated at all the time. The stress here is on the application implementation of protocol not on its practical details and is embedded in the working of CONTROLLER. Following we are showing the embedding of protocol in the CONTROLLER (core) functionality and how the protocol is implemented via aspects. This implementation is easily readable and hence any desired changes in integrity are easily implementable. The design is done in UML by keeping the dynamic nature of the design.

Importantly two situations demands the close look upon the updated data i.e. first when a user is accessing the account by bank window and at the same time accessing via the mobile banking services and second when transaction through ATM is under process and at the same time mobile banking transaction is executing. Both of the situations demand the very proper execution of two phase commit protocol.

Fig. 5 and Fig. 6 shows the base sequence execution of transaction models corresponding to the execution of query's from WINDOW and MOBILE at a time and from ATM and MOBILE at a time. In second discussed scene the execution of query and updation in final value may be delayed for some time because updation done through ATM may take some time for final updation in the system. In fig. 6 we are introducing a new syntax (all) used for processing the number of transactions together finally at the common server location. Transfer of amount among the inter bank accounts processed in this way cause delayed execution. The intermediate results may get stored in CONTROLLER and are finally updated to the SERVER. Here the two phase commit protocol is not modeled as part of the core

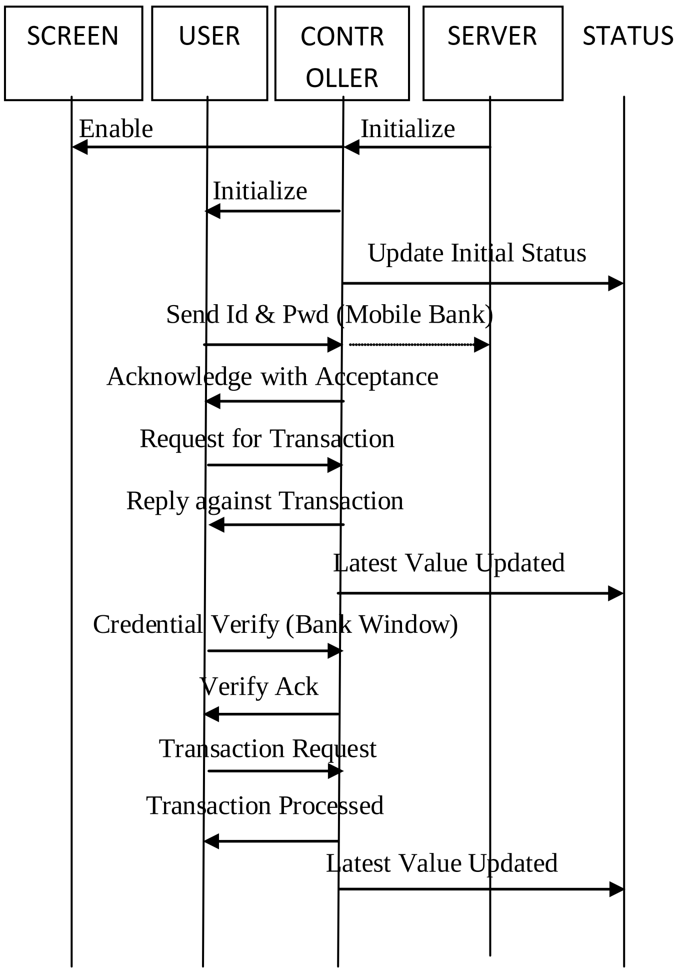

*Fig. 5 :* Maintain Serializaibility among parallel execution of Window and Mobile banking (single user)

functionality. Rather it is modeled separately for easy modification if needed. In Fig.5 and Fig. 6 every time the trigger of transaction is initialized by initializing both the server as well as client by CONTROLLER. Then first of all data (initial) is updated at sever and first GUI is provided to the client. In steps proceeding further the ID and PWD is submitted from USER to the SEVER and on receiving the ACK (POSITIVE) further transactions are processed.

Two phase commit protocol is modeled and shown in Fig. 7. It is build by considering the aspectual view of transaction and keeping them in sequence in an eRBML. Aspects are used to define a general pattern of communication to be used by

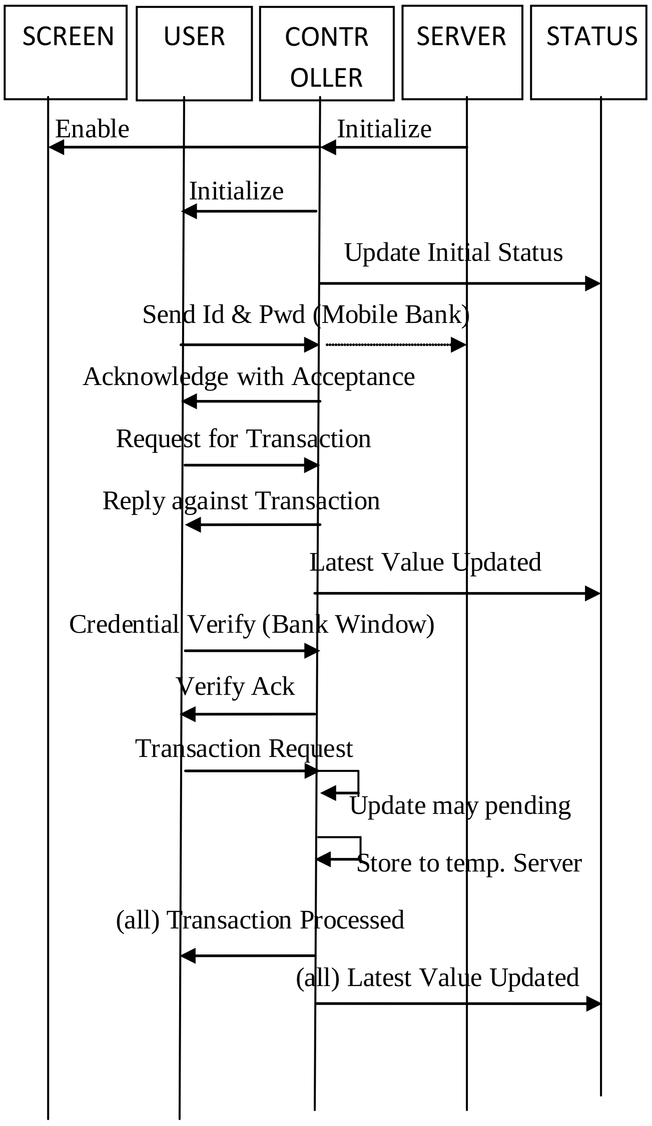

*Fig. 6 :* Maintain Serializaibility among parallel execution of Window and Mobile banking (single user) (Updated Fig. 5)

protocols and are easy to modify for reusability at any stage of application development. Two identified elements in Fig. 7 are USER and COMMIT SERVER. Interaction among the two is given in the form of message role so that it can be instantiated whenever required with any specific message names. Important implication of Fig. 7 is that it commit any of the transaction only if both the USER as well as COMMIT SEVER agrees on the transaction. This all is modified and is shown in Fig. 8

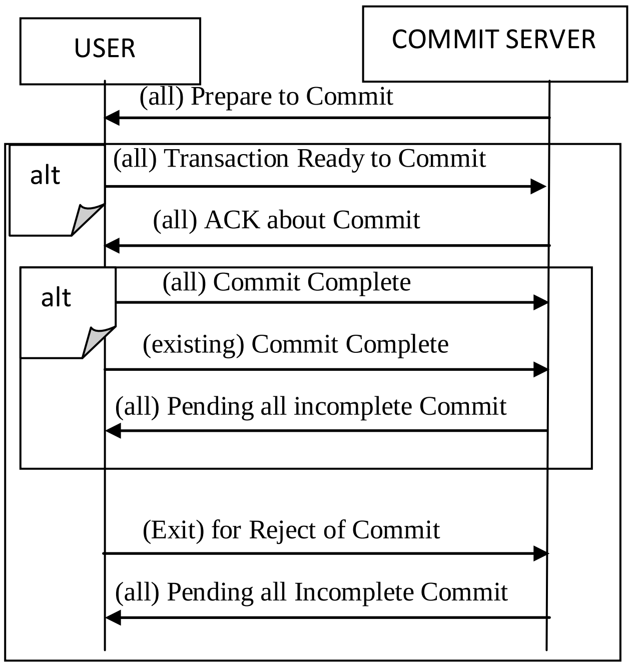
USER
COMMIT SERVER
(all) Prepare to Commit
alt
(all) Transaction Ready to Commit
(all) ACK about Commit
alt
(all) Commit Complete
(existing) Commit Complete
(all) Pending all incomplete Commit
(Exit) for Reject of Commit
(all) Pending all Incomplete Commit

*Fig. 7 :* Two Phase Commit Protocol

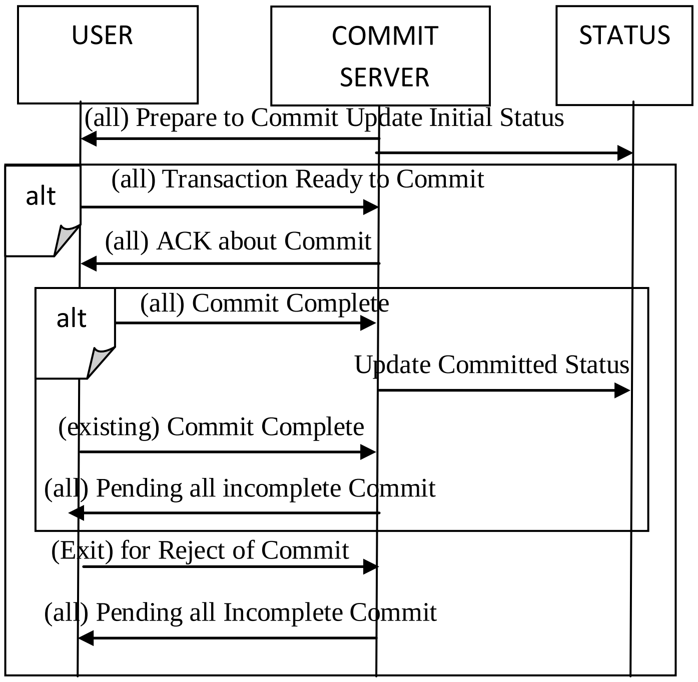
USER
COMMIT SERVER
STATUS
(all) Prepare to Commit
Update Initial Status
alt
(all) Transaction Ready to Commit
(all) ACK about Commit
alt
(all) Commit Complete
Update Committed Status
(existing) Commit Complete
(all) Pending all incomplete Commit
(Exit) for Reject of Commit
(all) Pending all Incomplete Commit

*Fig. 8 :* Updated Two Phase Commit Protocol

Fig. 9 and Fig. 10 shows the left side and Right side of graph transformation for refined aspect discussed in Fig. 6. Important to note here is that the Left side says that we have to apply the aspect at the points at which prepare for commit will appear and the same should preceded by Initialize message. The enable is true for both of first and second scene. Here it is possible to process the step by step manner or to execute a separate algorithm for execution of transactions. Right side of graph is shown in Fig. 10. In this fig other messages are included to take into consideration all or any kind of transaction which not be used in general by all user but is expected to execute in some special case only. The messages are supposed to be executed only if the reply from the two phase commit protocol is true. Two phase commit protocol is able to reply true or false depending on the execution of transactions. The base and aspect model are composed in such a way that match for all other messages is done only after point of successful commitment. The use of existing method discussed in [11] and [8] are not able to specify the conditions. The method presented by [8] may allow the weaving in the way which we want to describe. In method discussed in [11] it is needed to specify a list of composition directives that give the instructions to composition algorithm where to placethe messages matching with other messages. Hence the messages discussed in [11] and [8] are not appropriate for presenting the directives in easy way and are time consuming and error- prone too. In comparison to these two methods the graph transformation is an easy graphical method to specify the directives. In the method suggested in this paper it is very easy to place any additional messages anywhere in the Right side of the graph transformation rule. Along with is also possible to specify a different composition way to simplify|modifying the Right side of the Rule.

## V. Conclusion and Related Work

All of the existing approaches used to identify, compose and represent aspect at various level of software development faces a number of limitations especially the problem of scalability. The approach discussed in this paper for representing aspect at any level of software development using the UML methodology based on role modeling language. Various level of hierarchy are used to structure aspects and their possible instances. The problem of scalability is sorted out in this method since graph transformation allow the matching at any level of development and it automatically compose aspects along with the problem of expressiveness is also sorted out as use

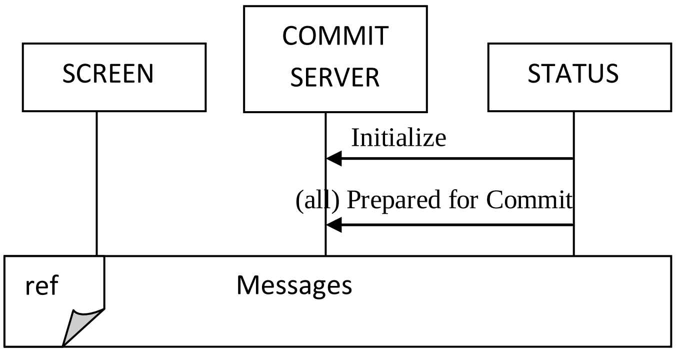

*Fig. 9 :* Left Hand Side of Graph Transformation

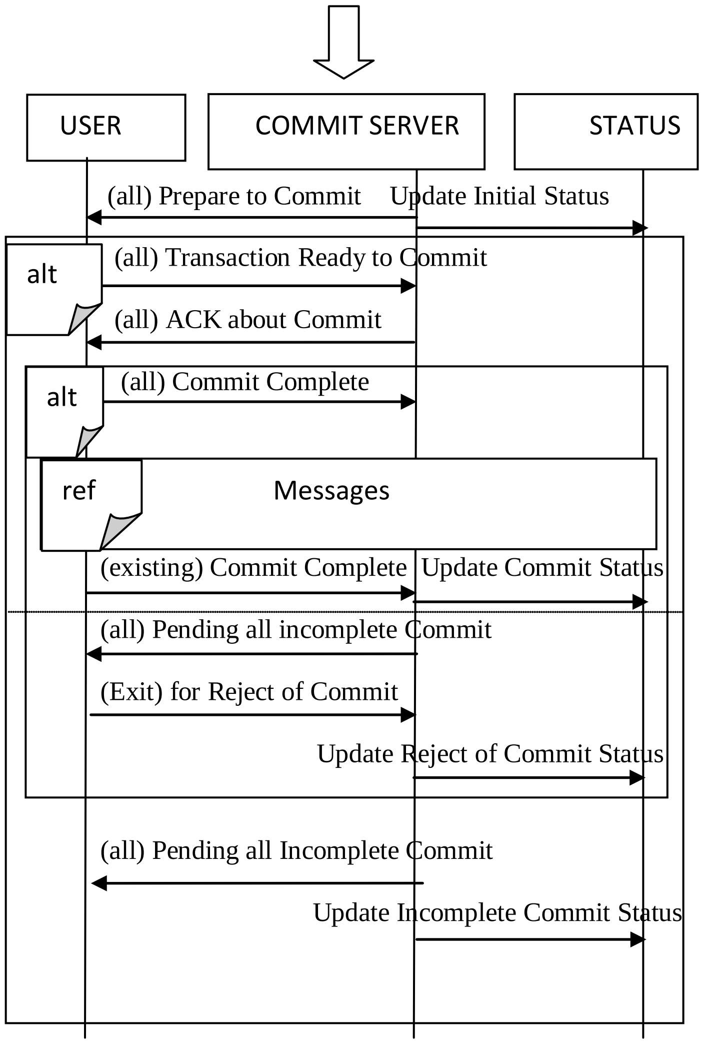
USER
COMMIT SERVER
STATUS
(all) Prepare to Commit
Update Initial Status
alt
(all) Transaction Ready to Commit
(all) ACK about Commit
alt
(all) Commit Complete
ref
Messages
(existing) Commit Complete
Update Commit Status
(all) Pending all incomplete Commit
(Exit) for Reject of Commit
Update Reject of Commit Status
(all) Pending all Incomplete Commit
Update Incomplete Commit Status

*Fig. 10 :* Right Hand Side of Graph Transformation

of graphical method in terms of graph transformation expresses all implementations. The validity of approach is reflected through its use on bank's transactions.

The approach discussed in this method is more close to syntactic implementations a lot of modifications can be done in terms of syntax related issues so that immediate implementations in programming language can be done. The modification to resolve the conflict among the aspects can also be done. The matching process discussed in this paper is also open to be modified. The modification in terms of forward and backward movements on matching at any level of transformation can be done.

## References Références Referencias

1. A. Rashid, A. Moreira and J. Araujo, (2003) “Modularisation and Composition of Aspectual Requirements”. AOSD 2003 Boston, USA, ACM Press, pp 11-20, March 2003.
2. Dae-Kyoo Kim, Robert France, Sudipto Ghosh and Eunjee Song, “A Role- Based Metamodeling Approah to Sepecifying Design Patterns”, COMPSAC 2003, Dallas, Texas, 2003.
3. D-K. Kim, “A Metamodeling Approach to Sepcifying Patterns”, PhD Thesis, Colorado State Univreisity, 2004.
4. G. Georg and R. France, “UML Aspect Specification using Role Modles”,OOIS,France, Lecture Notes in Computer Science, Springer, Vol. 2425, pp 186-191, September 2002.
5. G. Rozenberg, editor. “Handbook of Graph Grammers and Computing by Graph Transformation, Volume 1: Foundations.” World Scientific, 1997.
6. I.Ray, N. Li, R. B. France and D-K.Kim, “Using UML to Visualize Role-based Access Control Constraints.” SACMAT 2004: 115-124.
7. J. Araujo, J. Whittle and D. K,Kim, “Modeling and Computing Scenario- Based Requirements with Aspects.” RE 2004,Kyoto, Japan, IEEE CS Press, 2004.
8. J. Whittle and J. Araujo, “Scenario Modeling with Aspects”, IEEE Proceedings Software,Vol 151(4) pp. 157-172,2004.
9. M. Goedicke,B. Enders, T. Meyer, G. Taentzer, “ViewPoint-Oriented Software Development : Tool Support for Integrating Mutiple Prespective by Distributed Graph Transformation.” TACAS 2000; 43-47.
10. R. France, /D-K,Kim,S.Ghosh andE. Song. “A UML-Based Pattern Sepcification Technique.” IEEE Transactions on Software Engineering, Vol 30(3), pp 193-206,2004.
11. R. France I. Ray,G. Georg and S. Ghosh, “An Aspect-Oriented Approach to Design Modeling”, IEEE Proceedings Software, Vol 151(4), pp 174-186, August 2004.
12. S. Clarke and R. J. Walker, “Composition Patterns : An Approach to Designing Reusable Aspects”. ICSE 2001, Toronto, Canada, IEEE CS Press, pp 5-14,2001.
13. UML version 2.0 Available from the Object Management Group, 2005, http://www.omg.org.
14. Whittle. Jon, Araujo. Joao, Moreira. Ana, “Composing Aspect Models with Graph Transformations”, EA 06, Shanghai, China. 2006.