# Novel approach to transform UML Sequence diagram to Activity diagram

**Article** · July 2021

DOI: 10.51201/JUSST/21/07300

Rajkumar N Kulkarni
Ballari Institute of Technology and Management
**20** PUBLICATIONS **29** CITATIONS

# Novel approach to transform UML Sequence diagram to Activity diagram

**Dr. R. N. Kulkarni[1], C. K. Srinivasa[2]**

[1]*Prof. & Head, Dept. of Computer Science & Engineering, BITM, VTU, Ballari, INDIA*

[2]*Assistant Prof., Dept. of Computer Science & Engineering, BITM, VTU, Ballari, INDIA*

[1]*rn_kulkarni@rediffmail.com*, [2]*srinivasck9@gmail.com*

***Abstract:*** *Unified Modelling Language (UML) is currently accepted as a defacto standard language for modeling the software in the software industry. It will allow to implement object oriented concepts to model the software system. It provides a complete pictographic representation of software. Broadly these UML diagrams are classified into two groups viz. Structural diagrams and Behavioral diagrams. The sequence diagrams and Activity diagrams belongs to the second group i.e. behavioral diagrams. The sequence diagram represents the sequence of messages flowing from one object to another and activity diagram represents the flow of activities one after the other in a system. In this paper, we are proposing an automated tool which transforms the sequence diagram (which is represented in the table format) into activity diagram. The sequence diagram which is represented in the three column table called sequence table comprises various components of sequence diagram like objects, interactions, messages, alternations, iterations, loops, etc. The proposed tool reads the sequence table and converts the entire table components into the equivalent Activity table. Further the tool reads the activity table and then transforms to its equivalent activity diagram.*

***Keywords:*** **Activity diagram, Activity table, combined fragment, sequence diagram, Sequence Table**

## 1. INTRODUCTION

UML activity diagram which represents the dynamic behavior of the software and to design any activity diagram the input is the sequence diagram and for the sequence diagram the inputs are usecase diagram and scenarios. The designed UML activity diagram are used in modelling business and to develop software systems. An Activity diagram comprises the following notations such, as Initial node, final node, activity box, decision box, synchronize bar, swim lane, and an arrow with head. These diagrams represent the workflow activities of a software system. The activities are the actions described as an execution flow or behavior of the software system. An action of an activity diagram is a single step shown in a rectangle shape with rounded corners. A control flow of an activity diagram depicts the flow of control path between actions by drawing a line with an arrowhead. A set of actions performed by an object is grouped which is called an object's activity partition or a swim lane. The concurrent or parallel execution of actions is modelled using a fork and join nodes. A fork node has one inflow edge and multiple outflow edges. A join node has multiple inflow edges and a single outflow edge that synchronizes the inflow to a single outflow edge. In this paper, we are proposing an automated tool which transforms the sequence diagram (which is represented in the table format) into activity diagram. The sequence diagram which is represented in the three column table called sequence table comprises various components of sequence diagram like objects, interactions, messages, alternations, iterations, loops, etc. The proposed tool reads the sequence table and converts the entire table components into the equivalent Activity table. Further the tool reads the activity table and then transforms to its equivalent activity diagram.

## 2. LITERATURE REVIEW

In paper [1], the author constructed a framework to metamodel UML sequence diagram to sequence table format. This paper actuated us to abstract the control flow activities from the sequence table and further, it is represented in the form of a table. In paper [2], the author described the abstraction of different types such as simple, synchronous, asynchronous messages by parsing the sequence diagram, store the abstracted messages as scenarios and translate it into user requirements. This paper helped us to parse the sequence diagram and extract the objects, messages, and interactions. The author in paper [3], discussed a strategy to interpret the UML sequence diagram to activity graph. The methodology proposed is appropriately modified and used in this work. In paper [4], the author presented a methodology to verify and analyze the consistency rules amid sequence diagram and CFG by using the OCL mapping. This paper actuated us to abstract the control flow of loop and ALT combined fragments of a sequence diagram. The author in paper [5], proposed a formal approach to interpret the activity diagrams for application-specific with semantics. This paper helped us to analyze and transform the decision node of the activity diagram to table format. In paper [6], the author developed a procedure to formalize sequence diagrams into an extended finite state machine by generating the test cases from sequence diagrams. This paper helped us to analyze and transform the ALT and LOOP of sequence diagram to activity diagram. The author in paper [7], presented a systematic approach to transform LOOP, ALT, etc. combined fragments of sequence diagrams into Z specification. This paper helped us to translate the combined fragments such as LOOP and ALT into a table format with formal specifications. In paper [8], the author introduced a procedure to formalize the sequence diagram into extended finite state machines with precise semantics. This paper provided us to construct the execution flow of combined fragment such as alt and loop of sequence diagram. The author in paper [9], discussed a methodology to generate a sequence flow chart message and a control flow graph of sequence diagram. This paper helped us to generate the flow of interactions and messages of UML sequence diagram.

## 3. METHODOLOGY

Methodology followed in the generation of activity diagram from the sequence diagram is illustrated in the following steps.
In step 1 we are taking the input as a sequence diagram which is represented in the form of a table. This abstraction of table form from the diagram is already carried out by the authors [1].
Step 2: The output of step 1 is used as an input and then the generation of the activity table and activity diagram from the sequence table is accomplished by abstracting the related information such as objects, messages ALT, and LOOP.

### 3.1 Transforming synchronous messages to an activity table

A synchronous message in a sequence diagram is initiated by the sender object and waits until the receiver or target object completes its process. This synchronous behavior is active and processes the only single message from left to right at the time order.

The given input Sequence diagram with a synchronous message shown in figure 3.1is transformed to a metamodel form called sequence table (Table 3.1). This sequence table has 3 columns namely sender, receiver, and interaction of objects represented in first-order logic.

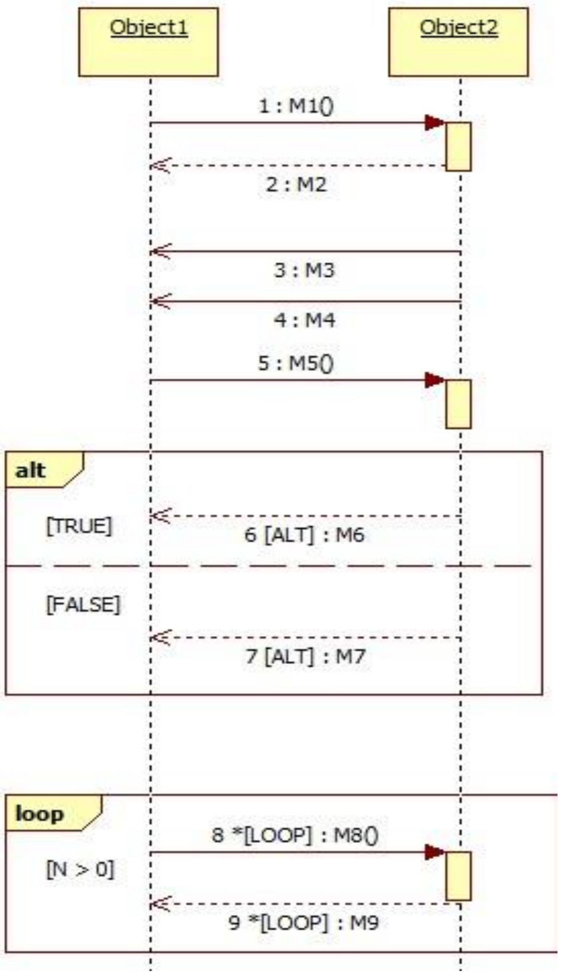


**Figure 3.1 Input Sequence diagram.**

To generate the activity diagram from the sequence table, initially, the control flow for the activity diagram is abstracted from the list of objects of the first column of the sequence table (sender column) row by row and stored in the activity table (Table 3.2). The corresponding messages and combined fragment such as ALT, LOOPS are also abstracted from the third column of the sequence table and is stored in a respective row of the activity table. The activity diagram for synchronous type message is drawn by first considering the lifeline of objects as a swim lane or object partitions. The number of swim lanes depends on the number of objects in the activity table. In this case, figure 3.1 has two objects, so two swim lanes are drawn one for Object1 and another for Object2. The first control flows from the start node and the first object of the first row of the Control flow column of the activity table. This activity along with message/call is drawn in the object1's swim lane from the initial node and message (M1). Then the second object and corresponding message (M2) are fetched from the second row of the Control flow column of the activity table and control flow is drawn accordingly in the respective object's swim lane.

### 3. 2: Transforming asynchronous messages to an activity table

An asynchronous message of a sequence diagram is initiated by the sender object and does not wait until the receiver or target object completes its process as an example shown in figure 3.1(Messages sent from OBJECT2 to OBJECT1 multiple times with messages M3, M4). This asynchronous behavior is active at any in-time order and processes multiple messages. The behavior of asynchronous message that performs the sending event from the sender object to receiver object multiple times is known as FORK node. The behavior of the fork leads to the parallel execution of activities. After the execution of all activities is accomplished, the control flow of activities converges to form a JOIN node.

**Table 3.1: Sequence Table generated by tool for figure 3.1**

| SEQUENCE TABLE | | |
|---|---|---|
| **SENDER** | **RECEIVER** | **MESSSAGE** |
| Object1 | Object2 | [∃( Object1∧ Object2) ⇒ Interaction (M1, Synchronous)] |
| Object2 | Object1 | [∃( Object2∧ Object1) ⇒ Interaction (M2,Response)] |
| Object2 | Object1 | [∃( Object1∧ Object2) ⇒ Interaction (M3, Asynchronous)] |
| Object2 | Object1 | [∃( Object1∧ Object2) ⇒ Interaction (M4, Asynchronous )] |
| Object1 | Object2 | [∃( Object2∧ Object1) ⇒ Interaction (M5, synchronous )] |
| Object2 | Object1 | [∃(Object1∧Object2)⇒Interaction(M6,Response)∧ Sequence(ALT: Guard: TRUE)] |
| Object2 | Object1 | [∃(Object2∧Object1)⇒Interaction(M7, Response )∧ Sequence (ALT: Guard: FALSE)] |
| Object1 | Object2 | [∃( Object1∧ Object2) ⇒ Interaction (M8, Synchronous)∧ Sequence (LOOP: Guard: N >0)] |
| Object2 | Object1 | [∃( Object2∧ Object1) ⇒ Interaction (M9, Response )∧ Sequence ( LOOP: Guard: N >0)] |

To draw an activity diagram for an asynchronous type of message, the swim lanes are drawn as discussed above. Later the control flows of activities are abstracted by fetching the list of objects from the first column of the generated sequence table (Table 3.1) and is stored in the activity table (Table 3.2). In this case, as shown in figure 3.1, the OBJECT2 performs multiple call events (M3, M4). Once the executions of all activities are done, the control flow is joined as there is no immediate activity with OBJECT2.

**Table 3.2: Activity Table for Sequence Table 3.1**

| ACTIVITY TABLE | |
|---|---|
| **CONTROL FLOW** | **ACTIVITY** |
| Object1 | M1 |
| Object2 | M2 |
| Object2 | M3 |
| Object2 | M4 |
| Object1 | M5 |
| Object2 | M6- A |
| Object2 | M7-A |
| Object1 | M8-L |
| Object2 | M9-L |

## 3. 3: Transforming ALT messages to an activity table

The alternative ALT is similar to the conditional IF statement but not limited to only two options. The purpose of ALT is to illustrate alternative interactions of objects. The ALT feature in a sequence diagram splits the fragment into blocks of interactions with guard condition where only the block with the true condition is executed always. The block of interactions with false conditions leads to else part.

The interactions, messages, and guard condition of the sequence diagram's alternatives are abstracted and are stored in the activity table. All the messages that are associated with ALT are labeled as M#-A, (where # is a message/call number) that is M6-A, M7-A, etc., and are stored in the activity table. In activity table 3.2, the messages that are labeled with A and corresponding objects (OBJECT1, OBJECT2) are said to be in ALT. To depict alternative interactions, the messages labeled with A, and the respective object interactions along with control flow are fetched from the activity table. A decision node is used to represent an ALT fragment. Here the control flows from OBJECT1 to Decision node with message M5, Decision node to OBJECT2 with message M6 if guard condition is satisfied. Else the control flows from OBJECT1 to Decision node, Decision node to OBJECT2 with message M7, and Decision node to the final node.

### 3. 4: Transforming ALT messages to an activity table

A loop consists of a sequence of interactions or tasks which is specified once and is executed several times. Loops in sequence diagram are used to recur the set of interactions between the objects. The loops in the sequence diagram actuate and terminate the iteration with an interaction operand or with a guard condition. The loop is executed several times until the guard condition is satisfied. The interactions, messages, and guard condition of the sequence diagram's loop are abstracted from the sequence table and are stored in the activity table. All the messages that are associated with loops are labeled as M#-L, (that is M8-L, M9-L, etc.), and are stored in the activity table. In activity table 3.2, the messages that are labeled with L and corresponding objects (OBJECT1, OBJECT2) are said to be in a loop.
To generate an equivalent activity diagram, the messages labeled with L, and the respective object interactions along with control flow are fetched from the activity table. Depending upon the pre or post-loop, the decision node is placed at the appropriate position. In this case, the control flows from OBJECT1 with message M8 to OBJECT2 with message M9, OBJECT2 with message M9 to Decision node, Decision node to OBJECT1 if the guard condition is satisfied. Else the control flows from OBJECT1 with message M8 to OBJECT2 with message M9, OBJECT2 with message M9 to Decision node, Decision node to final node.

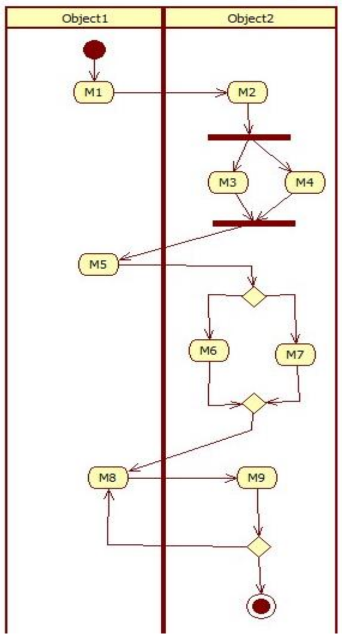
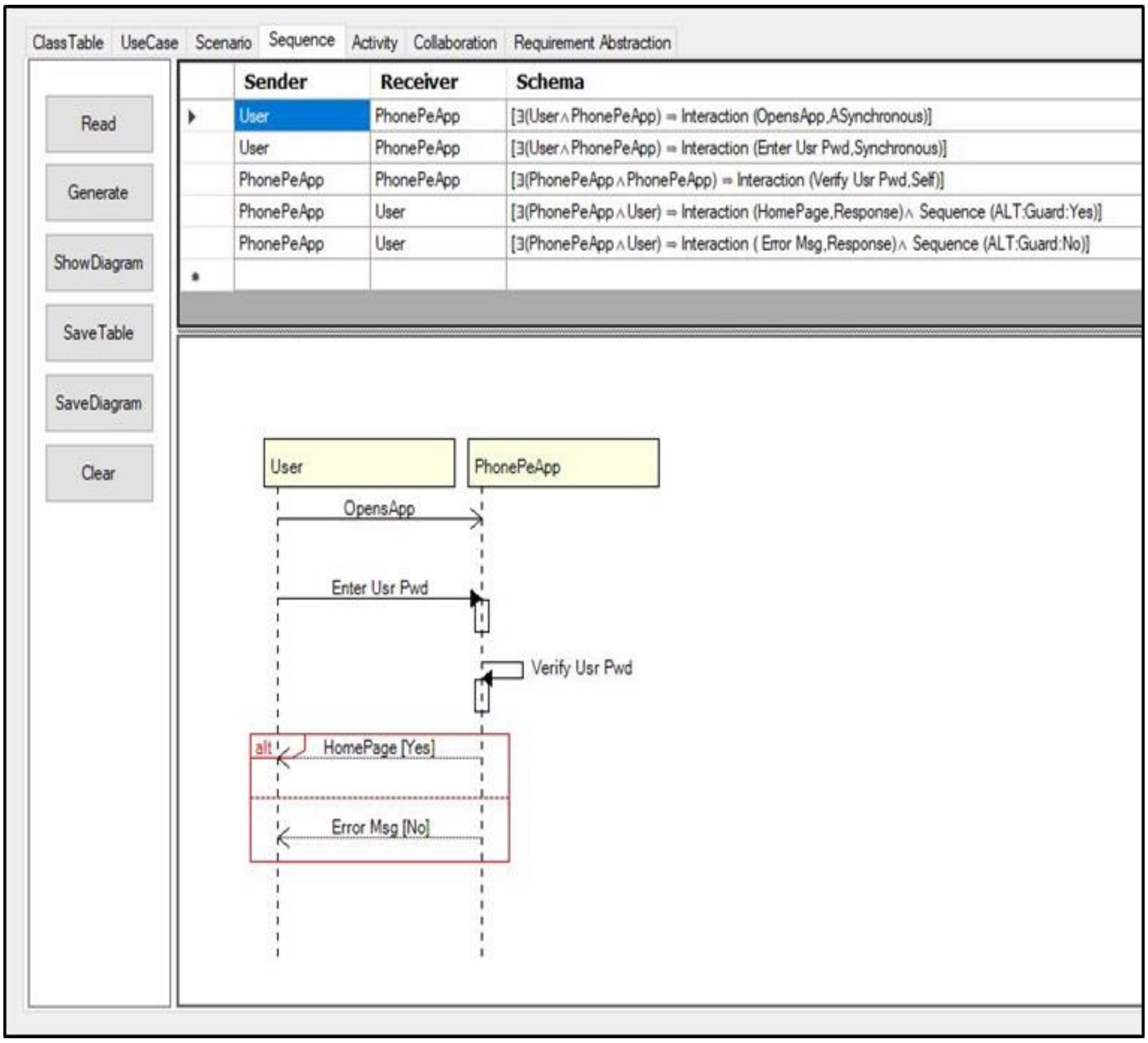


**Figure 3.2: An equivalent Activity diagram for figure 3.1**

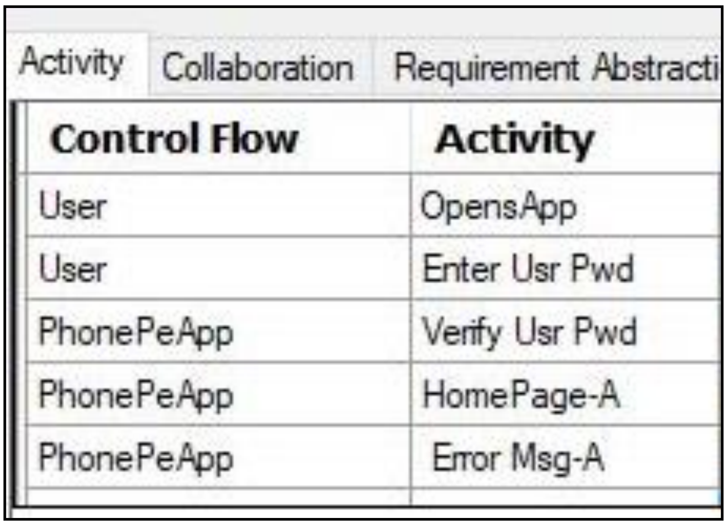
## 4. RESULTS AND DISCUSSIONS

To demonstrate the transformation of sequence diagram to an activity diagram a case study of simplified PhonePe login system sequence diagram is given as an input for our proposed automated tool. The given input sequence diagram figure 4.1 consist of two objects namely (user, phonepe) interacting synchronously and asynchronously. It also has a self-message and combined fragment ALT which represents the alternative interaction between the objects based on the guard conditions. Here in figure the object user interacts asynchronously with PhonePeApp to open an App. The object user calls a behavior “Enter Usr pwd “, the object PhonePeApp validates the input password. The PhonePeApp opens the “Homepage” when the guard condition is “YES” and displays an “Error Msg” if the condition is “NO”. The tool processes the given input sequence diagram, abstracts the required information and generates the sequence table as shown in figure 4.1. The tool further abstracts the information by scanning the sequence table and generates a new table called activity table (figure 4.2). The tool renders the activity diagram by reading the contents of activity table as an output as shown in figure 4.3.



| Sender | Receiver | Schema |
|---|---|---|
| User | PhonePeApp | [∃(User∧PhonePeApp) ⇒ Interaction (OpensApp,ASynchronous)] |
| User | PhonePeApp | [∃(User∧PhonePeApp) ⇒ Interaction (Enter Usr Pwd,Synchronous)] |
| PhonePeApp | PhonePeApp | [∃(PhonePeApp∧PhonePeApp) ⇒ Interaction (Verify Usr Pwd,Self)] |
| PhonePeApp | User | [∃(PhonePeApp∧User) ⇒ Interaction (HomePage,Response)∧ Sequence (ALT:Guard:Yes)] |
| PhonePeApp | User | [∃(PhonePeApp∧User) ⇒ Interaction ( Error Msg,Response)∧ Sequence (ALT:Guard:No)] |

**Figure 4.1: Input Sequence diagram and sequence table**


Activity
Collaboration
Requirement Abstracti

| Control Flow | Activity |
|---|---|
| User | OpensApp |
| User | Enter Usr Pwd |
| PhonePeApp | Verify Usr Pwd |
| PhonePeApp | HomePage-A |
| PhonePeApp | Error Msg-A |

**Figure 4.2: Activity table generated as output by our proposed tool**

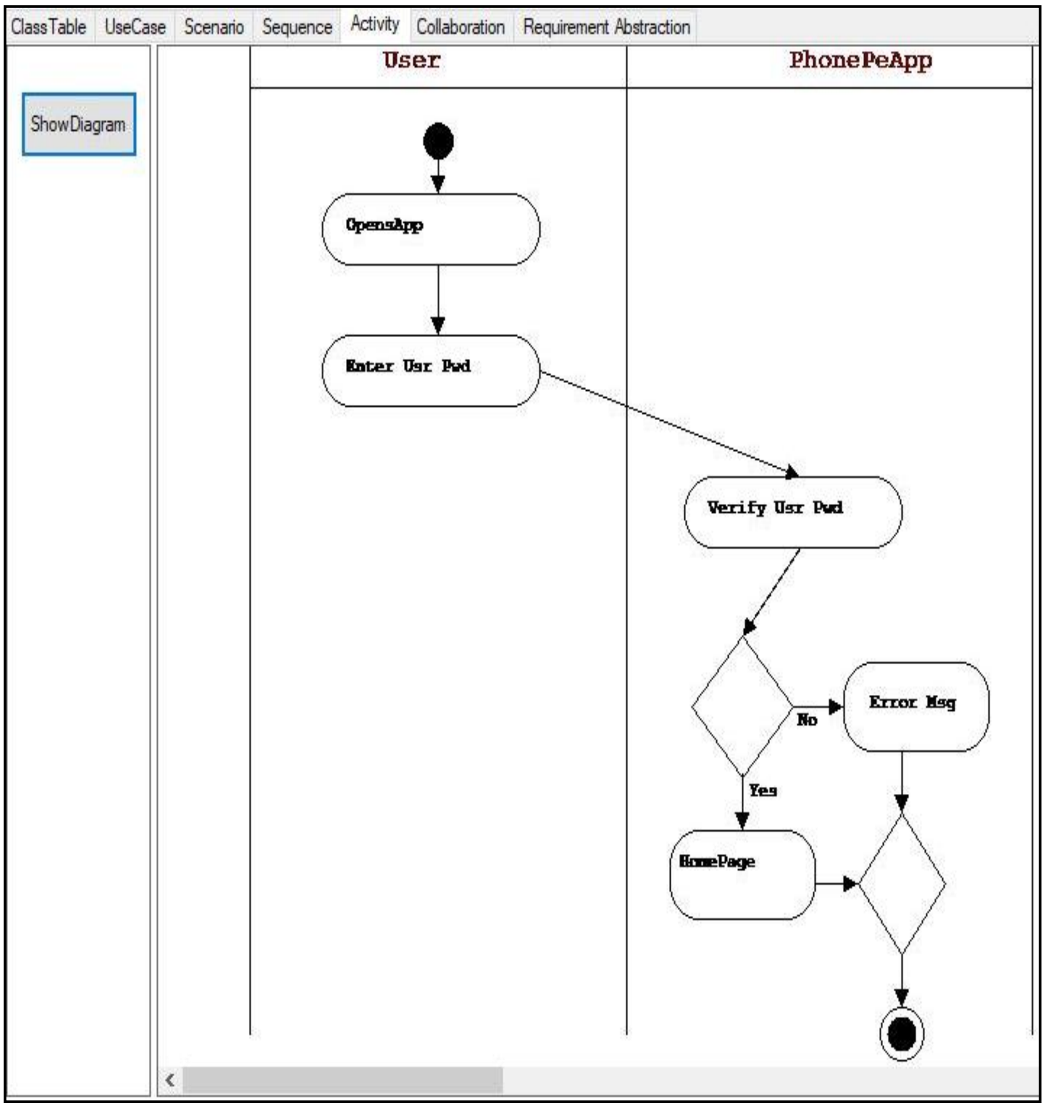


**Figure 4.3: Activity diagram generated by our proposed tool**

## 5. CONCLUSIONS

In this paper, we presented and discussed about automated tool which extracts the objects, messages, interactions, loops and ALT from the sequence diagram which is represented in the form of a sequence table. The process of generating the Activity table from the sequence table is discussed and further the extracted information is stored in the activity table. The information related to the control flow activities are extracted from the activity table and then the activity diagram is generated automatically. The generated activity diagram helps the user to analyze transitions of the activities of a software system's dynamic behavior. The proposed tool is tested for its correctness and completeness by taking various sequence diagrams.

## REFERENCES

[1] Dr. R. N. Kulkarni, Srinivasa C. K. Ameliorated Methodology to Meta Model UML Sequence Diagram in the Table Format, International Journal of Advanced Networking and Applications Volume: 12 Issue: 04 Pages: 4633-4638(2021) ISSN: 0975-0290 Feb -2021

[2] Dr. R. N Kulkarni, Singri Swathi, Afsana, Soumya N M, Heena Kousar, Reverse Engineering of UML sequence diagram for the Abstraction of Requirements, International Journal of Combined Research & Development (IJCRD)eISSN: 2321-225X; pISSN: 2321-2241 Volume:4; Issue: 4; April -2015.

[3] Santosh Kumar Swain, Durga Prasad Mohapatra and Rajib Mall, Test Case Generation Based on Use case and Sequence Diagram, Int.J. of Software Engineering, IJSE Vol.3 No.2 July 2010.

[4] Vahid Garousi, Lionel Briand, Yvan Labiche, Control Flow Analysis of UML 2.0 Sequence Diagrams, first European Conference, ECMDA-FA 2005, Nuremberg, Germany Procedings Springer 2005.

[5] Hans Gronniger Dirk Rei, Bernhard Rumpe, Towards a Semantics of Activity Diagrams with Semantic Variation Points, Model Driven Engineering Languages and Systems, Proceedings of MODELS 2010, Oslo, Norway. Lecture Notes in Computer Science, Springer, 2010.

[6] Mauricio Rocha, Adenilso Simao, Thigo sousa, Model based test case generation from UML sequence diagrams using extended finite state machines, © Springer Science+ Business Media, LLC, part of Springer Nature 2021, https://doi.org/10.1007/s11219-020-09531-0.

[7] Nazir Ahmad Zafar, Formal Specification and Verification of Few Combined Fragments of UML Sequence Diagram, Arab J SciEng (2016) 41:2975–2986, © King Fahd University of Petroleum & Minerals, Springer 2016DOI 10.1007/s13369-015-1999-9.

[8] Prateeva Mahali, Durga Prasad Mohapatra "Model based test case prioritization using UML behavioural diagrams and association rule mining", International Journal of System Assurance Engineering and Management 2018, volume 9, issue 5 issn 1063-1079 DOI: 10.1007/s13198-018-0736-7, Springer India.

[9] Ajay Kumar Jena, Santosh Kumar Swain, Durga Prasad Mohapatra, "Test Case Creation from UML Sequence Diagram: A Soft Computing Approach", January 2015, DOI: 10.1007/978-81-322-2012-1_13.

[10] Pariksha Jain, Dinesh Soni, "A Survey on Generation of Test Cases using UML diagrams "2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), DOI: 10.1109/ic-ETITE47903.2020.395.

[11] Atif Mashkoor , Alexander Egyed, Evaluating the alignment of sequence diagrams with system behavior, International conference on industry 4.0 and smart manufacturing, Procedia Computer Science 180 (2021) 502–506 Elsevier.

[12] Meiliana, Irwandhi Septain, Ricky Setaiwan Alianto, Daniel, Ford Lubman Gaol et al, Automated Test Case Generation from UML Activity Diagram and Sequence Diagram using Depth First Search Algorithm, 2nd International Conference on Computer Science and Computational Intelligence 2017, 13-14 October 2017, Bali, Indonesia, Procedia Computer Science 116 (2017) 629–637, Elsevier.

[13] Van-Doc Vua,b, Trong-Bach Nguyena, and Quyet-Thang Huynha,1, Formal Transformation from UML Sequence Diagrams to Queueing Petri Nets, Advancing Technology Industrialization Through Intelligent Software Methodologies, Tools and Techniques H. Fujita and A. Selamat (Eds.) IOS Press, 2019, doi:10.3233/FAIA190082.

[14] Felician Campean and Unal Yildirim, Enhanced Sequence Diagram for Function Modelling of Complex Systems, Procedia CIRP 60 ( 2017 ) 273 – 278, he scientific committee of the 27th CIRP Design Conference doi: 10.1016/j.procir.2017.01.053, 2212-8271 © 2017, Elsevier.

[15] Haoqing Zhang, An Approach for Extracting UML Diagram from Object-Oriented ProgramBased on J2X, Advances in Engineering Research, volume 113, International Forum on Mechanical, Control and Automation IFMCA 2016.

[16] OMG Unified Modeling Language TM (OMG UML), superstructure version 2.2", http://www.omg.org/spec/UML/2.2/ Superstructure.

[17] B. Rumpe, "Modeling with UML", Springer International Publishing Switzerland 2016, DOI 10.1007/978-3- 319- 33933-7_2.

[18] Martina Seidl "UML @ Classroom", Springer International Publishing Switzerland 2015, DOI 10.1007/978-3-319-12742-2.

[19] James Rumbaugh, Ivar Jacobson, Grady Booch, "Unified Modeling Language Reference Manual", 2nd Edition, ISBN-13 9780321245625, Rational Software Corporation ©2005 Addison-Wesley Professional.

[20] Jim Arlow UML 2 AND THE UNIFIED PROCESS: Practical Object-oriented Analysis and Design, 2nd Edition, pearson publishers 2015.