Volume: 12 Issue:04 Pages: 4633-4638(2021) ISSN: 0975-0290

Ameliorated Methodology to Meta Model UML Sequence Diagram in the Table Format

Dr. R. N. Kulkarni
Prof. & Head
Department of Computer Science, BITM, Ballari, India
Email: rn_kulkarni@rediffmail.com
C. K. Srinivasa
Assistant Professor and Research Scholar, VTU Belagavi.
Department of Computer Science, BITM, Ballari, India
Email : srinivasck9@gmail.com

ABSTRACT

The unified modeling language (UML) consists of 13 diagrams. The sequence diagram is one of the UML diagrams which captures the dynamic behavior of a system. It represents how the objects communicate to the other objects using a message(s) such as simple, synchronous or asynchronous. These messages are executed in time order from top to bottom and left to right. In this work, an automated tool is proposed which meta-models by taking the UML sequence diagram as an input and then translate the diagram to XMI format using the available whitestar tool and stored in the file. We abstract the various components such as objects, messages, activation, loops (for, while), conditional statements alt (if, if-else), etc. The objects, messages, activation, loops, conditional statements alt are abstracted from the file and are represented in first-order logic, and are stored in form of a table. Finally, using the abstracted information an equivalent sequence diagram is generated.

Keywords- **first-order logic, interactions, messages, Sequence diagram, sequence table.**

Date of Submission: Jan 19, 2021 Date of Acceptance: Feb 18, 2021

1. INTRODUCTION

The Unified Modeling Language (UML) is a standard toolkit which consists of an integrated set of thirteen diagrams. These thirteen diagrams of UML are classified into static and dynamic behavioral. The UML diagrams are represented in visuals with objects, actors, actions, artifacts in order to document the information about the software systems. This UML aids the software developers to understand, develop, modify or maintain the software systems.

The sequence diagram is one of the important dynamic behavioral UML diagrams. The UML sequence diagrams enable the software architects to analyze and design the flow of logic for developing software systems. The UML sequence diagram shows the pictorial representation of objects interaction with lifelines which gives valuable information for developing software systems. The sequence diagram mainly focuses on identifying the interaction of objects within the software system.
The sequence diagram is one of the UML diagrams which captures the dynamic behavior of a usecase. The sequence diagram consists of an activation box, objects, messages, and lifelines. The objects or actors are represented in the boxes on the top of the diagram. The interaction between the objects is done with named messages the messages in the sequence diagram flow horizontal from top left to right corner. The order of the next message in the sequence diagram appears below the just one. The lifelines of the sequence diagram are the vertical dashed lines that represent the object's existence over time.

In this work, an automated tool is proposed which takes a sequence diagram in xmi format as input and abstracts the objects and messages. The tool extracts the interaction between the objects and identifies the types of messages such as synchronous, asynchronous, and self or response and combined fragment loops and alt. Further, the abstracted messages and objects are represented in first-order logic and are stored in a table format. Finally, using the abstracted information an equivalent sequence diagram is generated.

2. LITERATURE SURVEY

In paper [1], the author described the abstraction of 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.

In paper [2], the author addressed a mechanism which tracks the execution state of object interactions and formal verification of UML 2.0 sequence diagrams. This paper confines us to represent the sequence diagrams with a rich framework in first-order logic.

In paper [3], the author presented a methodology for transforming sequence diagram's combined fragments into Z language. This paper helped us to analyze combined fragments such as loop and alt of sequence diagrams and transform into Z specification.

Volume: 12 Issue:04 Pages: 4633-4638(2021) ISSN: 0975-0290

In paper [4], the author presented an automatic tool which abstracts the design information from the Java source code and constructs the UML diagrams. This paper motivated us to extract design information and generate UML sequence diagram.

In paper [5], a rule based approach was proposed by the author to derive parameter values and related information from sequence diagrams. This paper helped us to analyze the loops and alt in the sequence diagram.

In paper [6], the author discussed a framework to generate the sequence diagrams statically from an object oriented programming code by using a query refinement system. This paper helped us to abstract flow of messages between the object interactions.

In paper [7], an automated analysis was presented by the author to identify the instances of sequence diagram and represented in form of an infinite tree. This paper facilitated us to trace the sequence diagram and store in form of a table.

3. PROPOSED METHODOLOGY

3.1 The methodology proposed in this work is discussed in the following two steps:

Step 1: Extraction of Objects and Messages

In this step, the proposed tool takes the sequence diagram in XML form generated by whitestar UML as input. Every object and message in an XMI file contains a unique ID (xmi:id). These XMI IDs were used to extract the object, messages, and interactions. Figure 3.1 shows the sequence diagram rendered by whitestar UML and figure 3.2 shows the transformed sequence diagram in XMI format.

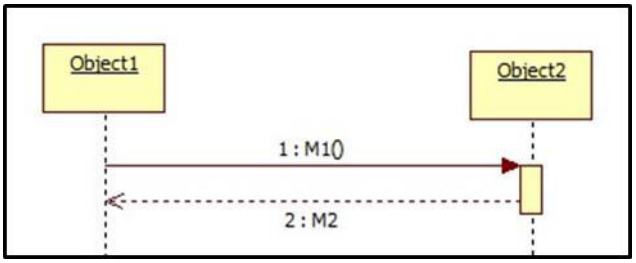


Figure 3.1: Sequence diagram with two objects

The sequence diagram shown in figure 3.1 depicts the interaction between two objects (object1 & object2) with synchronous message m1() and a response message m2(). These messages and objects are extracted from the equivalent XML form of a sequence diagram generated by whitestar tool shown in figure 3.2.

Transformed sequence diagram in xml format

```
<XMI>
  </UML:Model>
    <UML:Message xmi.id="UMLStimulus.14" name="M1" sender="UMLObject.18" receiver="UMLObject.19"
interaction="UMLInteractionInstanceSet.8">
      <UML:Message.action>
        <UML:CallAction xmi.id="UMLCallAction.15" name isAsynchronous="false" stimulus="UMLStimulus.14"/>
      </UML:Message.action>
        <UML:Message xmi.id="UMLStimulus.16" name="M2" sender="UMLObject.19" receiver="UMLObject.18"
interaction="UMLInteractionInstanceSet.8">
      <UML:Message.action>
        <UML:ReturnAction xmi.id="UMLReturnAction.17" name="" isAsynchronous="false" stimulus="UMLStimulus.16"/>
      </UML:Message.action>
    <UML:ClassifierRole xmi.id="UMLObject.18" name="Object1 message2="UMLStimulus.14" message1="UMLStimulus.16"
isRoot="false" isLeaf="false" isAbstract="false">
    </UML:ClassifierRole>
   <UML:ClassifierRole xmi.id="UMLObject.19" name="Object2" message2="UMLStimulus.16" message1="UMLStimulus.14" >
     <UML:Multiplicity xmi.id="X.25">
  <UML:Stereotype xmi.id="X.22" name="return" extendedElement="UMLStimulus.16"/>
 </UML:Model>
</XMI>
```

Figure 3.2: Transformed sequence diagram in xml format

Volume: 12 Issue:04 Pages: 4633-4638(2021) ISSN: 0975-0290

Step 2: Identifying Message types, Loops and Alt

In this step, the extracted objects and messages from the xml form are represented in the first order logic. The first-order logic of the sequence diagram is composed of objects, messages, and guard conditions such as loop and alt. Based on the interaction of objects the type of messages are identified. For example, figure 3.3 depicts the sequence diagram with two objects. Here the object1 interacts with object2 with message M1 and the object2 responds to object1 with message M2. In this case, the object1 will not interact with object2 until it receives the response from object2, i.e. the sequence of interaction is 1→2 and 2→1 such messages are referred to as synchronous message and denoted with a thick line and a solid arrow. The asynchronous messages are denoted with a line and an open arrow. Here the sender object does not wait for a response from the receiver object. This message type has an interaction 1→2 and 1→2. Each and every interaction between the sender object and receiver object are tracked and recorded in a table format as shown in table 3.1.

The loops are identified where there is a continuous interaction between objects with guard conditions. The alt works like if-else fashion. The alt is identified where there is an operation call and more response from other objects. These interactions are stored and represented in first order logic by our proposed tool along with objects and messages.

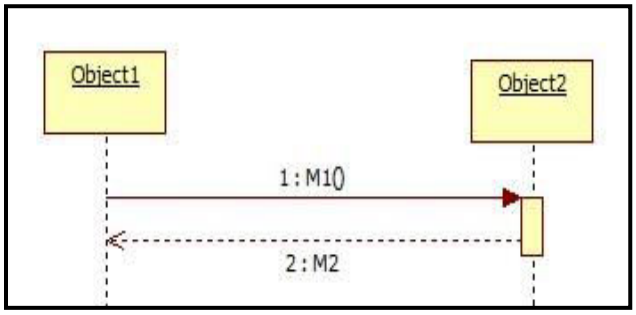


Figure 3.3: Synchronous sequence diagram with two objects

Table 3.1: First order logic for Synchronous sequence diagram with two objects

SENDER	RECEIVER	SCHEMA
1	2	[∃(Object1∧Object2) ⇒ Interaction (M1, Synchronous)]
2	1	[∃(Object2∧Object1) ⇒ Interaction (M2, Response)]

3.2 Block diagram to generate sequence diagram from the specification Table

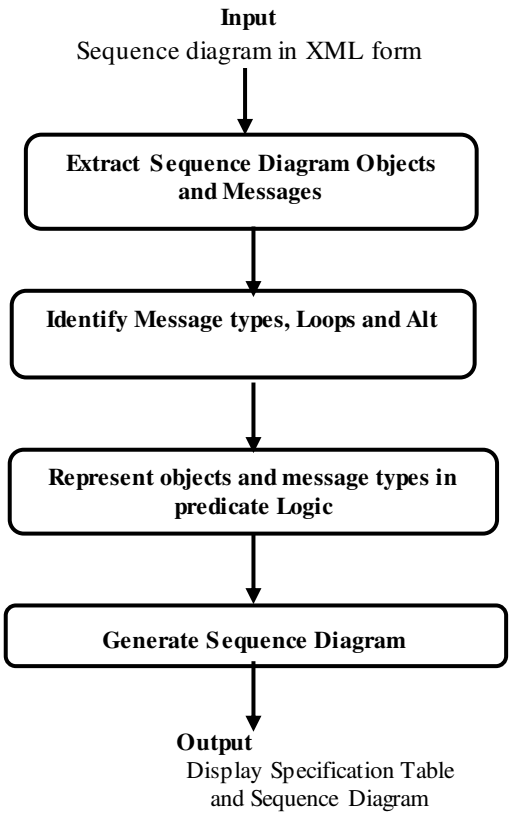


Figure 3.4: Block diagram to generate sequence diagram and Specification Table

The above block diagram figure 3.4 depicts the methodology to generate the sequence diagram and specification table. Firstly, the objects and messages are extracted from the sequence diagram in xml form. For each interaction between the objects, the tool identifies the types of message such as synchronous, asynchronous, and simple. The tool automatically identifies the loops and alt conditions and are transformed into amenable first-order logic called specification table. Later sequence diagram is generated as an output.

3.3 Procedure to generate sequence diagram and amenable First-Order Logic of a sequence diagram

The following stages are carried out to generate a sequence diagram and amenable First-Order Logic of sequence diagram

Stage 1: In this stage, a generic tool whitestar UML is used to draw the UML sequence diagram and is exported to XMI format. Now our proposed tool scans the generated XMI and abstracts the related information such as objects, messages, and guard conditions.

Stage 2: In this stage based on interactions of objects our proposed tool identifies the types of messages.

For synchronous message types, the interactions are (sender → receiver) and (receiver → sender). For

Volume: 12 Issue:04 Pages: 4633-4638(2021) ISSN: 0975-0290

asynchronous message type the interactions are (sender → receiver), (sender → receiver). For self-message type the interaction is (sender → sender). For response message type the interaction is (receiver → sender). For loops the interactions is N times where N>0 [(sender → receiver) and (receiver → sender)]. or is [(sender → receiver), (sender → receiver)].

For alts the interactions is [(sender → receiver) and (receiver → sender), (receiver → sender)].

Stage 3: In this stage, our proposed automated tool writes the abstracted objects and messages. For simple message in the first order logic as ∃ (Sender ∧ Receiver) → Interaction (Message, message type) and for loops and alt as ∃ (Sender ∧ Receiver) → Interaction (Message, message type) ∧ Sequence (Loop, Guard) and for alt as ∃ (Sender ∧ Receiver) → Interaction (Message, message type) ∧ Sequence (ALT, Guard). Finally, the automated tool abstracts the needful information from the specification table and generates the sequence diagram as an output. The tool also provides a feature to save the abstracted information in a Microsoft Excel sheet.

4. Results and Discussion

In this section, we illustrate with examples about the generation of the sequence table and sequence diagram by our proposed tool. Figure 4.1 shows the sequence diagram with three objects (User, DataControl, DataSource) interacting with each other with messages. Here the object DataControl interacts N times, (N>0) synchronously with object DataSource. The object DataSource responds to object DataControl until the guard condition is reached. The proposed tool automatically identifies the above order of messages, looping objects with guard conditions, and generates an amenable sequence table with a sequence diagram. The XMI of figure 4.1 is the input and table 4.1 is the output generated by our tool.

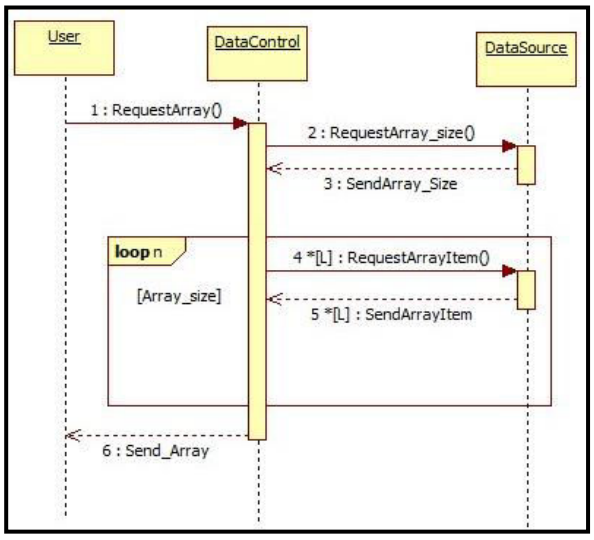
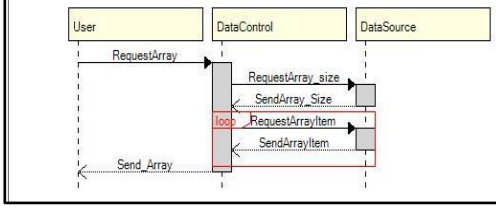


Figure 4.1: Sequence diagram with 3 objects interacting synchronously in loop

Table 4.1 First order logic for Sequence diagram with 3 objects interacting synchronously in loop

Sender	Receiver	Schema
User	DataControl	[∃(User ∧ DataControl) ⇒ Interaction (RequestArray,Synchronous)]
DataControl	DataSource	[∃(DataControl ∧ DataSource) ⇒ Interaction (RequestArray_size,Synchronous)]
DataSource	DataControl	[∃(DataSource ∧ DataControl) ⇒ Interaction (SendArray_Size,Response)]
DataControl	DataSource	[∃(DataControl ∧ DataSource) ⇒ Interaction (RequestArrayItem,Synchronous) ∧ Sequence (Loop:Guard:)]
DataSource	DataControl	[∃(DataSource ∧ DataControl) ⇒ Interaction (SendArrayItem,Response) ∧ Sequence (Loop:Guard:)]
DataControl	User	[∃(DataControl ∧ User) ⇒ Interaction (Send_Array,Response)]



In some possible situations, the objects continuously interact with each other either synchronously or asynchronously. This continuous interaction of objects is said to be in the loop. Figure 4.2 shows the sequence diagram with two objects (Object1, Object2) interacting with messages (M1, M2, M3, M4, M5, M6) synchronously. Our proposed tool identifies these type of continuous interactions ([1→2 2→1], [1→2 2→1], [1→2 2→1]). and renders as a loop as shown in table 4.2.

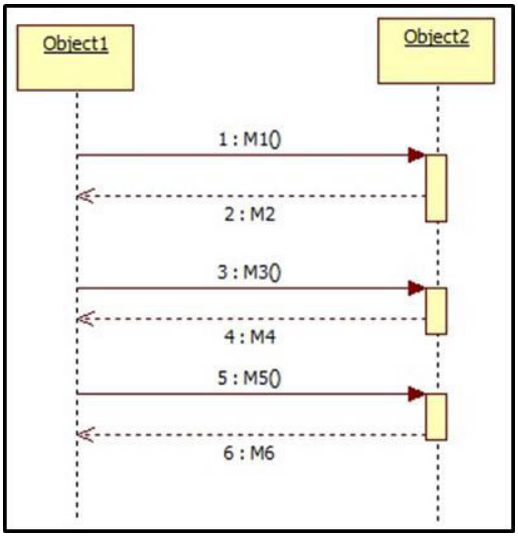
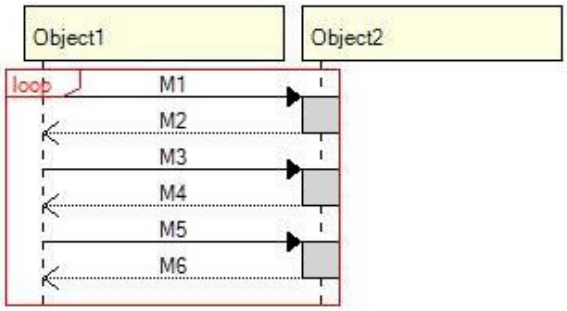


Figure 4.2: Sequence diagram with 2 objects interacting continually with synchronous messages

Volume: 12 Issue:04 Pages: 4633-4638(2021) ISSN: 0975-0290

Table4.2: First order logic for Sequence diagram with 2 objects interacting synchronous

Sender	Receiver	Schema
Object1	Object2	[∃(Object1∧Object2) ⇒ Interaction (M1,Synchronous)∧ Sequence (Loop1:Guard:N=6)]
Object2	Object1	[∃(Object2∧Object1) ⇒ Interaction (M2,Response)∧ Sequence (Loop1:Guard:N=6)]
Object1	Object2	[∃(Object1∧Object2) ⇒ Interaction (M3,Synchronous)∧ Sequence (Loop1:Guard:N=6)]
Object2	Object1	[∃(Object2∧Object1) ⇒ Interaction (M4,Response)∧ Sequence (Loop1:Guard:N=6)]
Object1	Object2	[∃(Object1∧Object2) ⇒ Interaction (M5,Synchronous)∧ Sequence (Loop1:Guard:N=6)]
Object2	Object1	[∃(Object2∧Object1) ⇒ Interaction (M6,Response)∧ Sequence (Loop1:Guard:N=6)]



5. CONCLUSION

In this work, we have presented a semi-automatic tool which takes the UML sequence diagram in xmi format as an input and then generates a sequence diagram with a specification table as an output. the proposed tool extracts the objects, messages, interaction, and few combined fragments such as loops and alt. the message types and combined fragments are identified by our tool based on the interaction of objects. the tool then stores the abstracted information and represents in first-order logic in a table format. Finally, the tool renders the sequence diagram.

6. REFERENCES

[1] Dr. R. N Kulkarni et.al, 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.*

[2] V.Lima, C. Talhi, D. Mouheb, M. Debbabi, and L. Wang," Formal Verification and Validation of UML 2.0 Sequence Diagrams using Source and Destination of Messages, Elsever *Electronic Notes in Theoretical Computer Science 254 (2009) 143–160*

[3] Nazir Ahmad Zafar, Formal Specification and Verification of Few Combined Fragments of UML Sequence Diagram, *Arab J SciEng (2016) 41:2975–2986, DOI 10.1007/s13369-015-1999-9.*

[4] Dr. R. N Kulkarni et.al, Abstraction of Uml Diagrams From Java Code, *International Journal of Combined Research & Development (IJCRD) eISSN:2321-225X; pISSN:2321-2241 Volume: 2; Issue: 4; April-2014.*

[5] Preeti Satish, Arinjita Paul, Krishnan Rangarajan. Extracting the Combinatorial Test Parameters and Values from UML Sequence Diagrams, *2014 IEEE International Conference on Software Testing, Verification, and Validation Workshops.*

[6] Chris Alvin · Brian Peterson · Supratik Mukhopadhyay. Static generation of UML sequence diagrams, *springer 2019 International Journal on Software Tools for Technology Transfer.*

[7] Seung Mo Cho, Hyung Ho Kim, Sung Deok Cha, Doo Hwan Bae. A semantics of sequence diagrams, *Elsevier Information Processing Letters 84 (2002) 125–130.*

[8] Mubarak Albarka Umar, Chen Zhanfang. A Comparative study of Dynamic Software Testing

Volume: 12 Issue:04 Pages: 4633-4638(2021) ISSN: 0975-0290

Techniques. *Int. J. Advanced Networking and Applications Volume: 12 Issue: 03 Pages: 4575-4584(2020) ISSN: 0975-0290.*

[9] Dr Vipin Saxena, Deepa Raj. Local Area Network Performance Using UML. *Int. J. of Advanced Networking and Applications Volume: 02, Issue: 02, Pages:614-620 (2010).*

[10] Martina Seidl "UML@ Classroom", *Springer International Publishing Switzerland 2015, DOI 10.1007/978-3-3/-19-12742-2.*

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

[12] B Rumpe, Modeling with UML, *Springer Imternational Publishing Switzerland 2016, DOI 10.1007/978-3-319-33933-7-2.*

AUTHORS PROFILE

Dr. R. N. Kulkarni holds a Ph.D in software Engineering from Visvesvaraya Technological University, Belagavi, Karnataka, India. He has more than 31 years of teaching experience and published nearly 70 publications in International journals/conferences and also national conferences. His area of research work are Software Engineering, DBMS, Soft Computing and Object Technology. He is presently working as Prof. & Head of Computer Science & Engineering Department at Ballari Institute of Technology & Management, Ballari,Karnataka, India.

C. K. Srinivasa has completed M.Tech in Computers Science and Engineering from Kuvempu University Karnataka, India. His teaching interests are in the areas of Computer Programming, computer graphics and Object-Oriented Analysis and Design. His research interests and work are in the areas of Software Engineering. He is Currently working as Associate Prof. in Computer Science & Engineering Department at Ballari Institute of Technology & Management, Ballari, Karnataka, India.