British Journal of Mathematics & Computer Science
16(1): 1-13, 2016, Article no.BJMCS.25374
ISSN: 2231-0851

SCIENCEDOMAIN *international*
www.sciencedomain.org

Generation of Test Cases from UML Sequence Diagram Based on Extenics Theory

Wasiur Rhmann[1*] and Vipin Saxena[1]

[1]*Department of Computer Science, B. B. Ambedkar University, Lucknow, India.*

Authors' contributions

This work was carried out in collaboration between both authors. Both authors read and approved the final manuscript.

Article Information

DOI: 10.9734/BJMCS/2016/25374
Editor(s):
(1) Junjie Chen, Department of Electrical Engineering, University of Texas at Arlington, USA.
(2) Qiang Duan, Information Sciences & Technology Department, The Pennsylvania State University, USA.
Reviewers:
(1) Lirong Wang, Virginia Tech, USA.
(2) Mohammed I. Younis, University of Baghdad, Iraq.
Complete Peer review History: http://sciencedomain.org/review-history/14195

Original Research Article

Received: 29th February 2016
Accepted: 31st March 2016
Published: 15th April 2016

Abstract

In the present work, extenics approach is used for modeling a problem of software testing. The approach is useful for generation of the test cases through the sequence diagram designed by the use of Unified Modeling Language. Authors created formalized sequence diagram by use of n-dimensional matter element property which is converted into message flow graph. Parsed data is generated through Extended Markup Language (XML) from the sequence diagram and converted the dual of message flow graph.
A traversal algorithm is designed and implemented on the dual of message flow graph for generation of valid test cases. Further test cases are minimized which satisfies path coverage criteria. The entire work is implemented on object oriented Java programming language through a case study of departure activity of aircraft.

Keywords: Software testing; test case; Extenics theory; unified modeling language; sequence diagram; XML; java.

**Corresponding author: E-mail: wasiurrhmann786@gmail.com;*

1 Introduction

Software testing is one of the important phases of the software development. Various researchers proposed the different types of software testing for enhancement of the quality of software. An excellent approach of a software testing produces the high quality of the software. Everyone knows that software testing is time consuming approach but one can minimize the valid test cases to produce high quality software. Mall [1] observed that software testing takes about 50% effort of the total development period of the software. Since test cases help for identifying the correct functioning of the software product according to the requirement of the client, therefore, It is important activity in the software development process. When the program becomes complex, then it is very difficult to select test cases especially for real time system. Craftsmen approach of testing is well explained in [2] in which test cases contain input, expected output under the certain condition.

UML is one of the important platform independent modeling language which contains various kinds of static and dynamic diagrams. These diagrams are very useful for generation of test cases and flow for finding the inconsistencies and ambiguities in the software requirements and designs. UML is used for detecting the fault at early stage of the software development which can save the development cost and time. Therefore, one can say that UML is intended to help to reduce the complexity of software size.

Extenics approach is a well defined approach for the use of formal method and transformation to solve contradictory problems. The problems are explained in such a way that goal can't attain in a given condition. In Extenics theory, things are composed of matter elements, its characteristic and measure of characteristics. In the software testing, we have to find more faults with reduced number of test cases which is a well defined contradictory problem. From the UML designs, there are numerous paths which are used as test paths to test the object-oriented software. To cover the more test paths with minimized test cases is also a contradictory problem.

In the present work these two kinds of problems are mainly considered. The said theory is used to generate minimized test cases from UML sequence diagram. A case study of aircraft simulation is considered as a real time problem. Dependence activity is modeled through UML sequence diagram for representing the dynamic behavior. Then, the dynamic diagram is converted into the XML form and further XML file is used for information extraction. The information is extracted through DOM parser for creation of Message Flow Graph (MFG). This graph is converted into dual graph then proposed traversal algorithm is used of generation of long series of test cases. This satisfies the path coverage criteria and further these test cases are minimized for the aircraft control system.

2 Related Work

Panthi and Mohaptra [3] have proposed automatic test case generation technique using sequence diagram. Here, authors extracted main features from sequence diagram and write java source code for that feature using Model junit library. By using this source code, authors generated test cases and derived different coverage criteria for generated test cases. Nayak and Samanta [4] synthesized test data from class diagram, sequence diagram and using OCL constraints. They derived attribute and constraint information from class and OCL constraint and used this information in sequence diagram. From sequence diagram, they generated Structured Sequence Graph (SCG), it is used for test sequence generation. Kumar et al. [5] presented a new technique of test case generation from UML models. They used different feature of UML 2.0 sequence diagram. Authors constructed Use Case Dependency Graph (UDG) using use case diagram and sequence diagram. It is used for constructing Concurrent Control Flow Graph (CFG) and derived test cases with full predicate coverage. Swain et al. [6] presented a technique of test cases generation using feature of UML 2.0 sequence diagram. Authors first constructed Message Dependency Graph (MDG) using sequence diagram. In MDG, they selected conditional predicate and generated test cases using slicing criteria. Samuel et al. [7] used UML communication diagram for cluster level test case generation. They first converted communication diagram into tree structure and traversed the tree in post order for selecting conditional predicates. Further, test data is generated using functional minimization technique. Generated test cases satisfy message coverage and boundary coverage criteria.

3 Extenics Theory

In the present scenario, concept of Extenics theory is used to solve the contradictory problems. In these kinds of problems, goal cannot be attained in the certain given conditions. Contradictory problems can be solved through formalized methods and transformation by the Extenics theory [8]. For example, in some city of a country, cars are drived through left hand steering control while in some cities of same country are derived through right hand control. When these are combined together then traffic collision occurs. There are numerous research problems which are still challenging and these are contradictory problems. Even smart researchers are not able to solve contradictory problems. Researchers develop Extenics to handle these kinds of problems by the use of three important components of extenics, which are Extenics theory, Extenics innovation method and Extension engineering. One can define the Extenics as a science which is used to develop formal model for the solution of contradictory problems [9]. For solution of these problems, we consider matter, characteristic of matter and measure and these are briefly described below:

Let us consider N is a matter, C is its characteristics then these basic three things are grouped together to form a relation R= (N, C, V). A matter can be defined by different characteristics grouped as a set C={ c_1, c_2_ _ _, c_n} with its measure set V={ v_1, v_2,_ _ , v_n}. Here n is used to represent n-dimensional matter element. V=C(N) shows the relationship between characteristic and its measurements or quality and quantity. R is given by the following

$$R = \begin{pmatrix} N & c_1 & v_1 \\ & c_2 & v_2 \\ & - & - \\ & & \\ & - & - \end{pmatrix} \tag{1}$$

When the above matters are interrelated i.e. quality and quantity then it is defined by the affair. This element is defined by I=(d,h,u) where d is a verb, h is characteristic and u is the measure. For n dimensional affair elements, this is defined below

$$I= \begin{pmatrix} d & h_1 & u_1 \\ & h_2 & u_2 \\ & - & - \\ & & \\ & h_n & u_n \end{pmatrix} \tag{2}$$

Relation element is used to describe the relation between matter and matter, affair and affair, matter and affair and given by

Q=(s, a , z)

Where s is relation name, a is characteristic name and z is measure.

Let t is a time parameter, then Q(t) shall be dynamic relation element and n dimensional relational element given by

$$Q=\begin{pmatrix} S1 & a_1 & z_1 \\ & a_2 & z_2 \\ & - & \\ & & - \\ & a_n & z_n \end{pmatrix}=(S, A, Z) \quad (3)$$

$$\text{where}\quad A=\begin{pmatrix} a_1 \\ a_2 \\ \\ - \\ a_n \end{pmatrix} \qquad Z=\begin{pmatrix} z_1 \\ z_2 \\ \\ - \\ z_n \end{pmatrix}$$

By the use of above concepts of Extenics theory, a model is to be prepared for the solution of the research problem. The model is based upon the matter element, affair element and relation element. These must describe the quantitative relation but also qualitative relation. In the present work, an UML activity model is designed for the formulation of the problem and test case generation and is represented by the following Fig. 1:

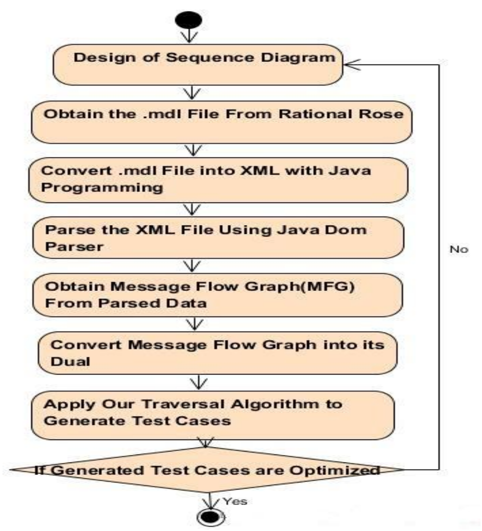

Fig. 1. Proposed test case generation UML activity model

4 Research Methodology

In the proposed work, the following steps are followed and authors have used Rational Rose tool for this purpose:

1. Construct a sequence diagram of the given problem;
2. Generate .mdl file of the sequence diagram, convert it into XML using java program. For conversion of .mdl file of Rational Rose into XML, we have to search the sequence number, object message, dir for direction of message, ordinal in .mdl file and sequence number in double quotes as attribute of sequences;

3. Generated XML file is parsed and for parsing through Java DOM parser to extract useful information like message sequence number, message, direction of message and order number of messages;
4. After step 3, we created MFG which is a directed graph whose edges are labeled with messages. In MFG, for each conditional message there will be two message outcomes;
5. Generate dual of MFG [10] whose node contains message information. This process minimizes number of edges which will reduce the number of test cases;
6. Apply the proposed traversal algorithm to generate the test cases.

In DFS, root is selected and reached at the end of node either left child or right child while in proposed algorithm, root is selected; find its closest neighbor whose distance is minimum. The proposed steps are given below:

Input Graph G, start node, end node
Output Test sequences

```
VisitednodeList=startnode, NodeList=0;
Algo(G, VisitednodeList)
NodeList= getadjacentnode of lastnode in VisitednodeList;
for each node in NodeList;
      If node is in VisitednodeList ;
   continue;
      end if
 If node is end node;
   Add node in VisitednodeList;
         print Vistednodelist;
         remove Lastnode from VisitednodeList;
     break;
 end if
 end for
 for each  node in NodeList;
 if VisitednodeList contains node or node is end node;
 continue;
 end if;
         add node in VisitednodeList;
         Algo( G, VistednodeList);
remove Lastnode from VisitednodeList;
end for
```

The above algorithm is implemented on the case study of aircraft departure activity.

5 A Case Study: Aircraft Departure Activity

The UML is a de facto standard modeling language used to design and analysis of an object-oriented system. It provides a blueprint of the system [11]. There are several diagrams which describe structural, behavioral and implementation view of the system. Sequence diagram is one of the important diagrams of UML to describe the dynamic aspect of the system. A sequence diagram illustrates the objects and messages that passes between them over time and it supports dynamic view of the evolving system. It shows explicit sequence of messages that are passed between objects. Different elements of sequence diagrams are as follows:

5.1 Object

Objects are entity which exchange messages to communicate with each other in time order.

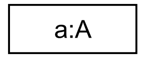

Fig. 2. Representation of an object

5.2 Message

Objects interact with each other using different type of messages. Different types of messages are represented as below:

Fig. 3. Representation of exchange of messages

In this section, we illustrated our approach on a case study of aircraft departure activity of aircraft control system. Airtcraft control system co-ordinates aircraft for departure. With the huge number of aircraft, there is need to co-ordinate departure activity of aircraft to avoid collision of aircraft [12].

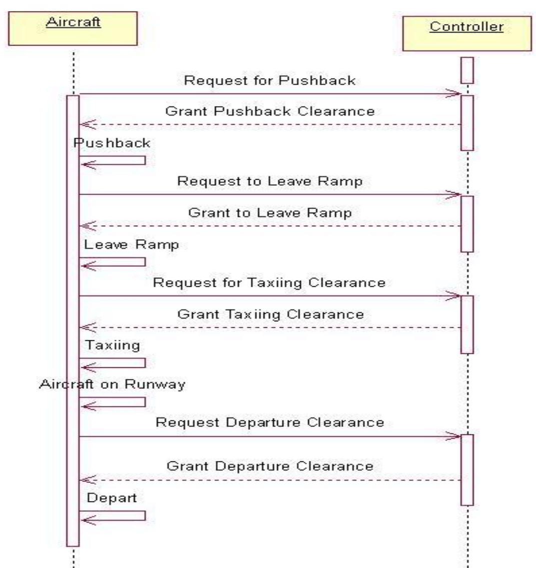

Fig. 4. UML sequence diagram of aircraft departure activity

Fig. 4 shows a sequence diagram of departure activity of aircraft. In this diagram, there are two objects i.e. Aircraft and Controller. Messages are exchanged between these two objects. Firstly object Aircraft sends a message of request for pushback clearance. The procedure of push backward away of aircraft from airport gate is called pushback. In pushback procedure, aircraft moves backward with the external power. If Controller object grants pushback request to Aircraft then Aircraft pushback. Secondly message from Aircraft to Controller is request to Leave Ramp. If Controller grants Leave then Aircraft Leaves the Ramp. Ramp is area where Aircraft maintenance activities are performed. Now, the Aircraft request for Taxiing Clearance. Taxiing is the procedure in which Aircraft moves on its own power. If permission of taxiing is granted then Aircraft moves on runway. Runway is a rectangular area used for landing and takeoff of the aircraft. Next aircraft requests for departure clearance from Controller. If Controller grants departure clearance. Then Aircraft departs. The presented sequence diagram is represented using n-dimensional matter element SD= (N, C_m, V_m) where N represents sequence diagram, C_m's are characteristics of different type of messages and V_m's are N's measure about characteristics. Objects Aircraft (A) and Controller (C) are represented as affair elements I= (d, h, u). Exchange of messages between objects are represented as relation element Q=(s, a, z). Authors used Rational Rose software for drawing the sequence diagram of aircraft departure activity. We generated a Petal file of sequence diagram and this Petal file contains entity and relationship between these entities. This file is further parsed to generate XML file from model file. For this, we searched string and use left and right parenthesis methods to generate XML file.

In Fig. 5 .mdl file of sequence diagram is shown which is generated from Rational Rose we used notepad to open this file here different tags are Object Message, dir, sequence, ordinal.

```
(object Message "Request for Pushback"
        quid            "49FA979B00DA"
        frequency       "Aperiodic"
        synchronization "Simple"
        dir             "FromClientToSupplier"
        sequence        "1"
        ordinal         0
        quidu           "000000000000"
        creation        FALSE)
    (object Message "Grant Pushback Clearance"
        quid            "49FA97C002DE"
        frequency       "Aperiodic"
        synchronization "Return"
        dir             "ToClientFromSupplier"
        sequence        "1.1"
        ordinal         1
        quidu
FALSE)

        creation
```

Fig. 5. Some part of .mdl file

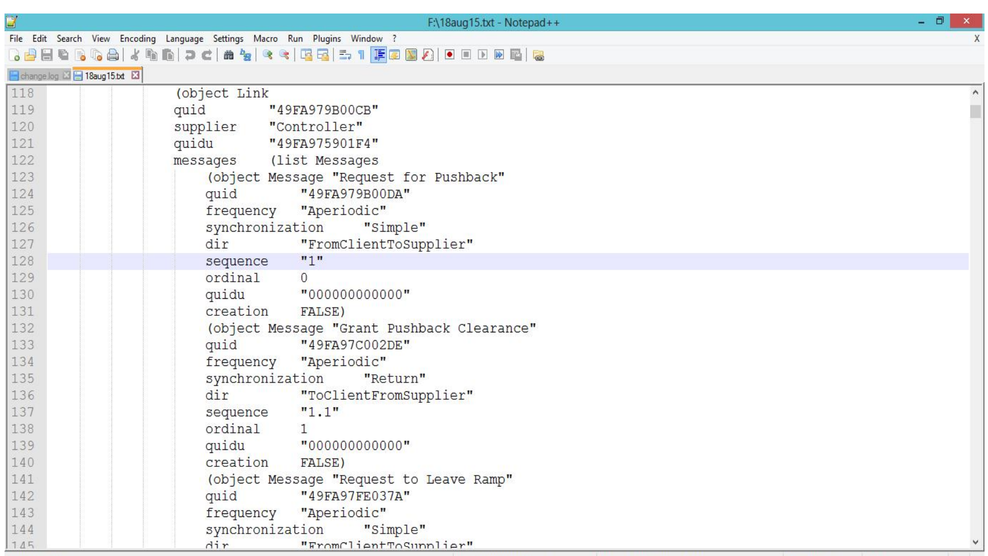

Fig. 6. .mdl file of sequence diagram generated from rational rose

In Fig. 8 we generated XML of .mdl file and output is shown on notepad++. XML contains sequence numbers as attributes and messages, and directions and order number as ordinal.

```
<sequence id="1">
<Object Message>"Request for Pushback</Object Message>
<Direction>"FromClientToSupplier"</Direction>
<Ordinal>0</Ordinal>
</sequence>

<sequence id="1.1">
<Object Message>"Grant Pushback Clearance</Object Message>
<Direction>"ToClientFromSupplier"</Direction>
<Ordinal>1</Ordinal>
</sequence>
```

Fig. 7. Some part of XML generated

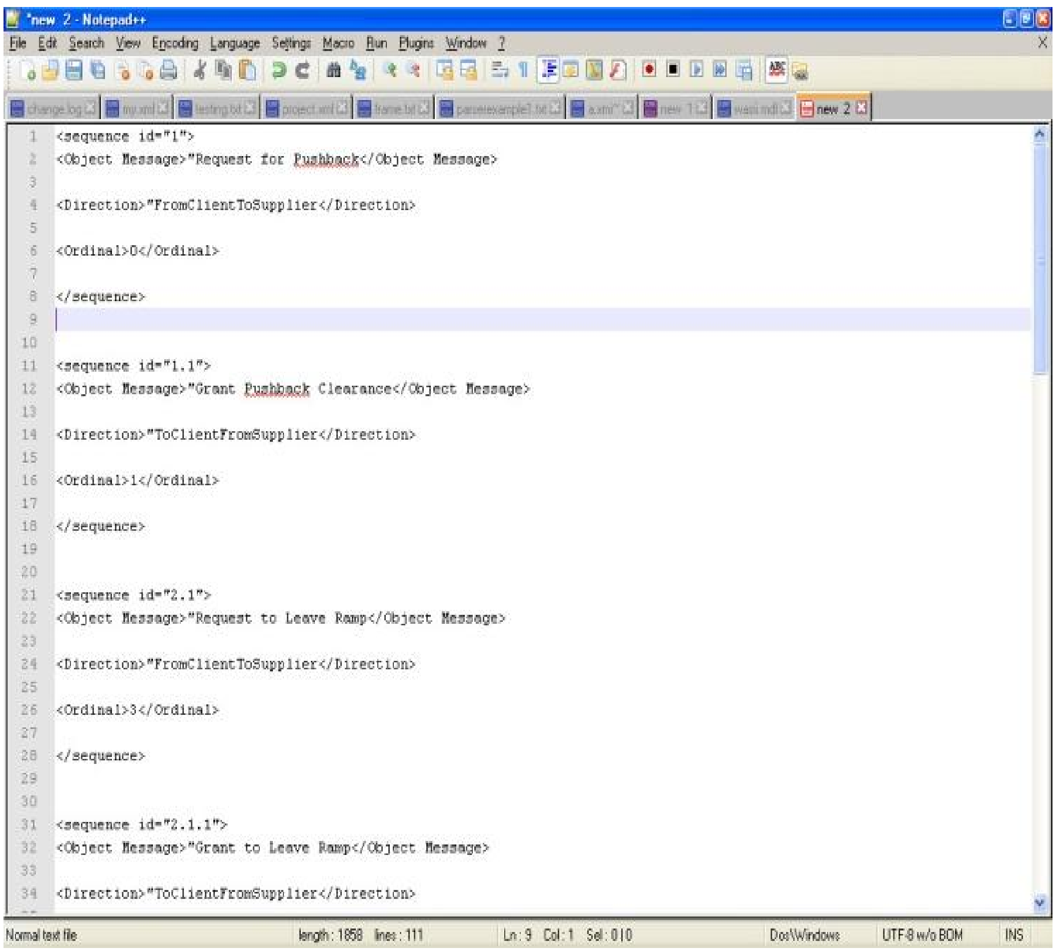

Fig. 8. XML generated from .mdl file of sequence diagram

In Fig. 9 we presented screen shot of java program in which we have taken .mdl file of sequence diagram and generated the XML. Netbeans IDE is used for java programming. Then DOM parser of java is used to extract useful information for constructing message flow graph which is presented in Fig. 10. In DOM parsing we used sequence to getElementsbyTagName. Parsed data contains information about sequence numbers, messages, direction etc.

Graphviz [13] is used to visualize the graph which takes input a dot file. Fig. 11 Is Message flow graph constructed from sequence diagram. In message flow graph there is a directed edge from Request Pushback Clearance to Pushback Granted and Permission not Granted.

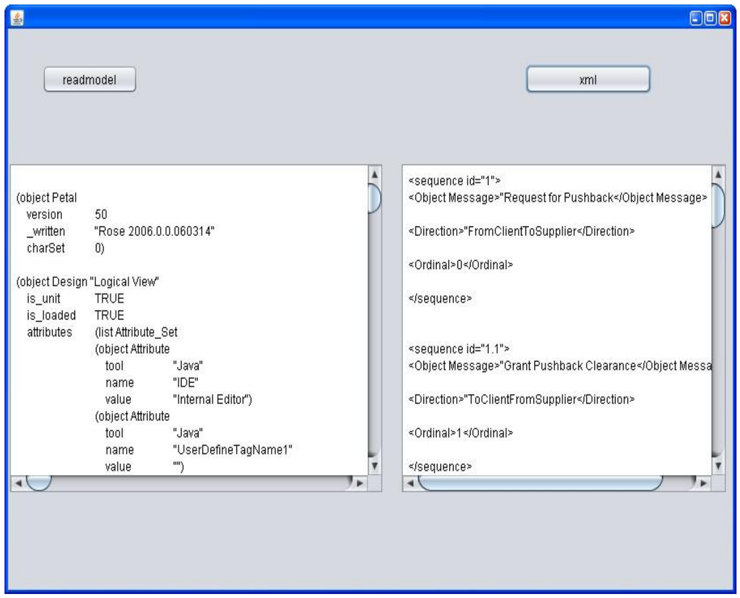

Fig. 9. XML generated from .mdl file of sequence diagram

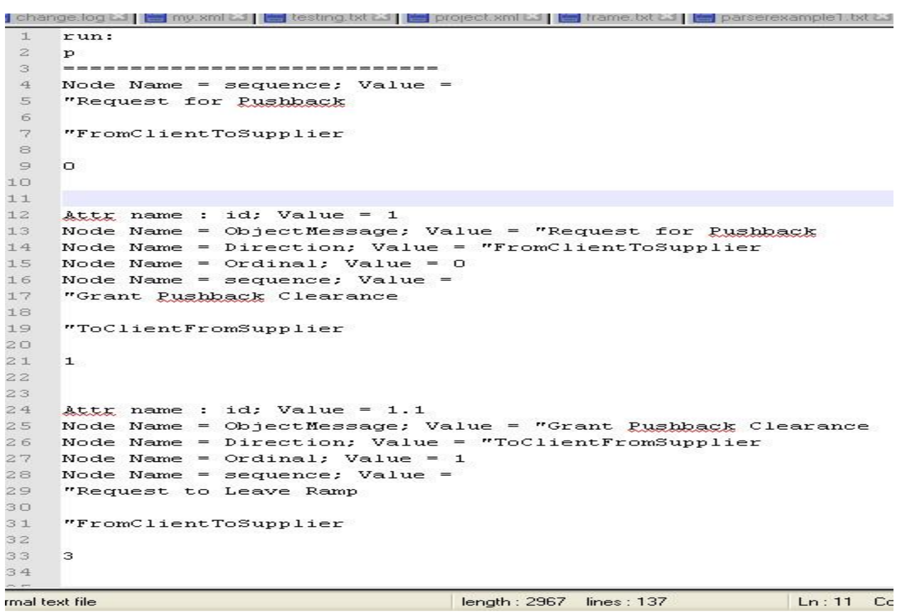

Fig. 10. Information extraction using java DOM parser for generating message flow graph

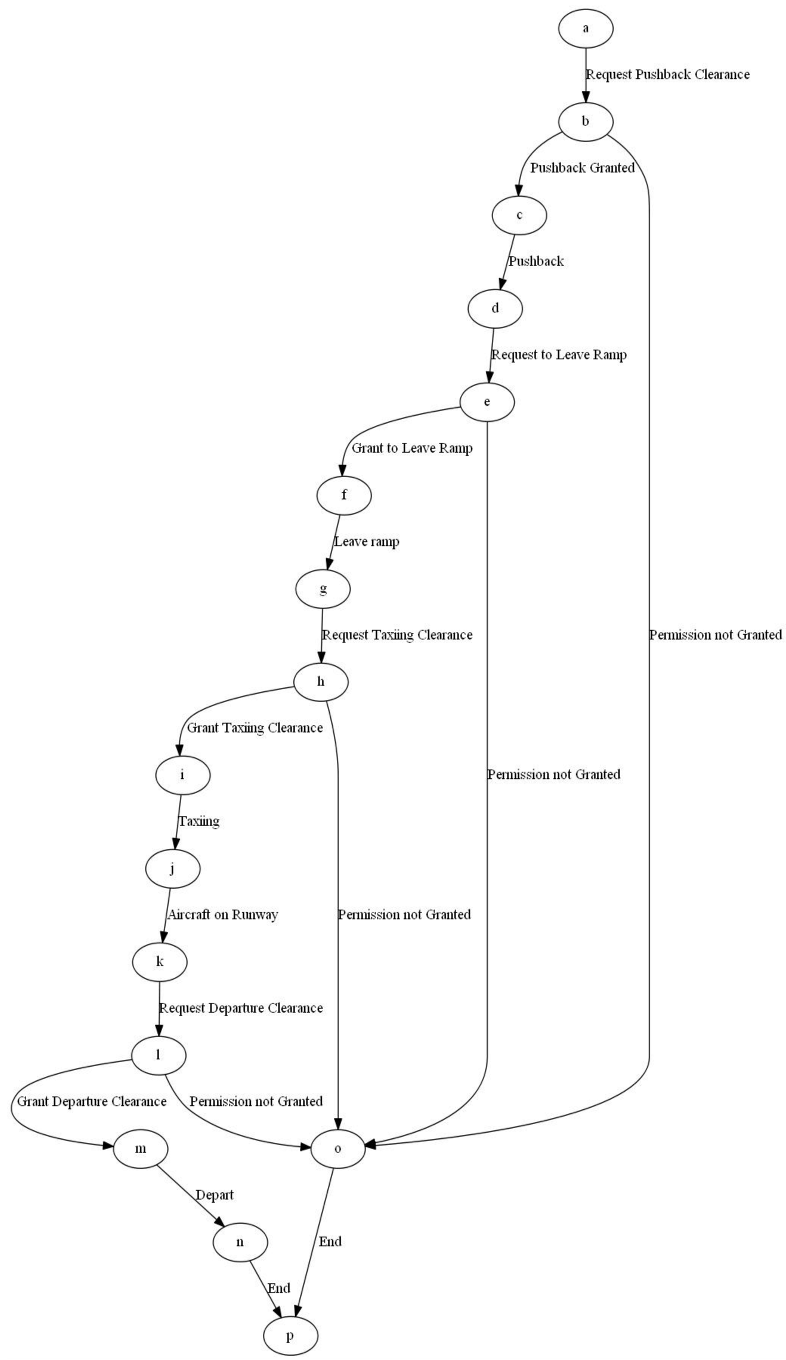

Fig. 11. Message flow graph (AFG) from UML sequence diagram

If Pushback Granted then aircraft Pushback and it goes for Request to Leave Ramp. If controller grants to Leave Ramp area aircraft Leave Ramp. Then aircraft Request for Taxiing Clearance. From here Taxiing clearance may be granted or Not. If taxing clearance is obtained then aircraft moves on runway. Next aircraft request for departure clearance. If permission granted from Controller then aircraft depart.

In Fig. 11 there is an edge incoming Request Pushback Clearance to node b and outgoing edge from b is Pushback Granted. In dual graph these edges are converted into nodes and there is an edge from node Request Pushback Clearance to Pushback Granted node.

Similarly other nodes are created from message flow graph. In Message flow graph there are multiple edges of Permission not Granted which are converted into single node of Permission not Granted node.

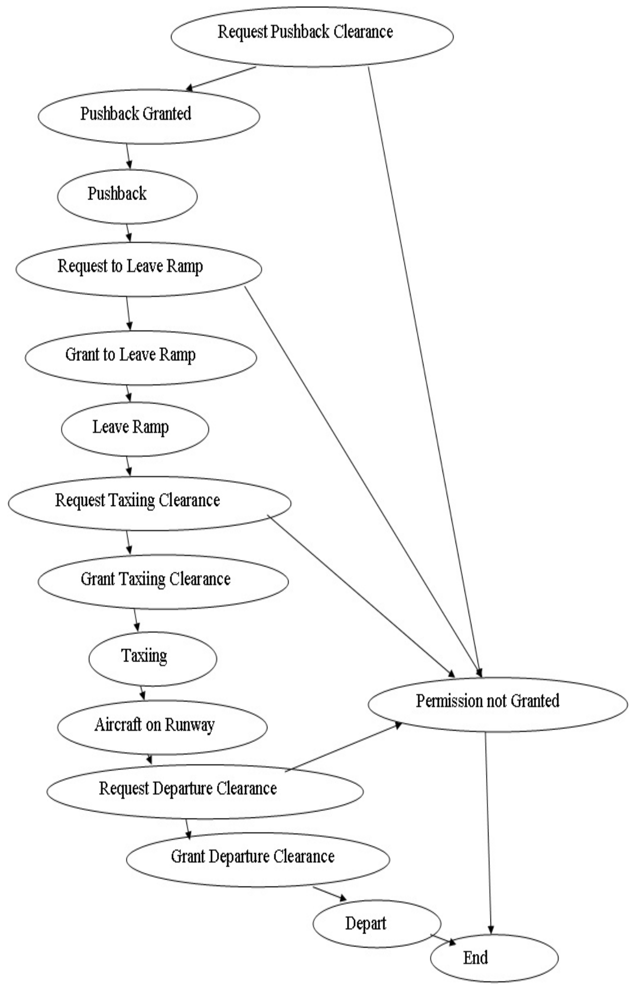

Fig. 12. Dual of message flow graph (AFG)

6 Results and Discussion

Test cases: There are five paths generated from the traversal of the above graph using above algorithm which gives us five test cases. Screenshot of test cases is presented in Fig. 13.

Test case1: Request Pushback Clearance-> Pushback Granted-> Pushback-> Request to Leave Ramp-> Grant to Leave Ramp -> Leave Ramp->Request Taxiing Clearance->Grant Taxiing Clearance-> Taxiing->Aircraft on Runway-> Request Departure Clearance->Grant Departure Clearance->Depart->End

Test case 2: Request Pushback Clearance-> Pushback Granted-> Pushback-> Request to Leave Ramp-> Grant to Leave Ramp -> Leave Ramp->Request Taxiing Clearance->Grant Taxiing Clearance->Taxiing->Aircraft on Runway-> Request Departure Clearance->Permission not Granted-> End

Test case 3: Request Pushback Clearance-> Pushback Granted-> Pushback-> Request to Leave Ramp->Grant to Leave Ramp-> Leave Ramp -> Request Taxiing Clearance-> Permission not Granted-> End

Test case 4: Request Pushback Clearance-> Pushback Granted-> Pushback-> Request to Leave Ramp->Permission not Granted-> End

Test case 5: Request Pushback Clearance-> Permission not Granted-> End

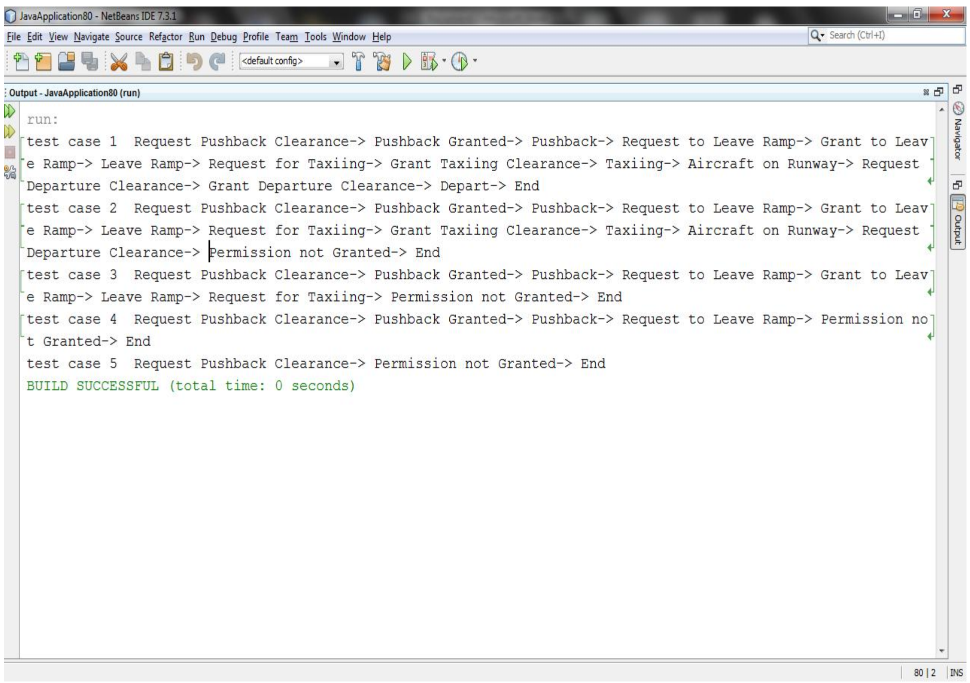

Fig. 13. Generated test cases of aircraft departure activity

7 Conclusions

In software testing test case generation is an important activity. In this paper we proposed an approach of test case generation based on extenics theory. Here we considered sequence diagram and transformed this into message flow graph and obtained its dual graph. From this dual graph we generated test cases. In dual of message flow graph redundant edges are converted into single node which reduces the number of test cases generated. Test cases generated from our approach can detect integration faults and are suitable for functional testing. Our technique achieves important message path coverage. Moreover in future we may use other diagram of UML to generate test cases. In future we can optimize the test cases by using different UML diagrams.

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Mall R. Fundamental of software engineering, 2nd edition, Prentice Hall; 2003.

[2] Jorgensen PS. Software testing: A craft's man approach, 4th edition, CRC presses Taylor and Francis Group; 2013.

[3] Panthi V, Mohapatra DP. Automatic test case generation using sequence diagram. International Journal of Applied Information System. 2012;2(4):22-29.

[4] Nayak A, Samanta D. Automatic test data synthesis using UML sequence diagrams. Journal of Object Technology. 2010;9(2):75-104.

[5] Swain SK, Mohpatra DP, Mall R. Test case generation based on use case and sequence diagram. International Journal of Applied Information System. 2010;3(2):21-52.

[6] Swain RK, Panthi V, Behera PK, Mohapatra DP. Sliced based test case generation using UML 2.0 sequence diagram. International Journal of Computational Intelligence Studies. 2014;3(2):221-250.

[7] Samuel P, Mall R, Kant P. Automatic test case generation from UML communication diagrams. Information and Software Technology. 2007;49(2):158-171.

[8] Wen C. The extension set and non compatiable problems. Science Explorer. 1983;3(1):83-97.

[9] Available:http://extenics.gdut.edu.cn/Introduction to Extenics

[10] Available:http://en.wikipedia.org/wiki/Dual_graph

[11] Booch G, Rambaugh G, Jacobson I. Unified modeling language user guide. 1st edition, Addision Wesley; 1998.

[12] Available:https://en.wikipedia.org/wiki/Air_traffic_control

[13] Available:http://www.graphviz.org/Download.php

Peer-review history:
The peer review history for this paper can be accessed here (Please copy paste the total link in your browser address bar)
http://sciencedomain.org/review-history/14195