Path generation algorithm for UML graphic modeling of aerospace test software

MingCheng Qu1,2*, XiangHu Wu1,2, YongChao Tao2, Chao Chen1,2
1School of Computer Science and Technology, Harbin Institute of Technology; Harbin 150001 Heilongjiang China
2Shenzhen Academy of Aerospace Technology, Shenzhen 518057 Guangdong China

* Corresponding author E-mail: qumingcheng@hit.edu.cn

Abstract. Aerospace traditional software testing engineers are based on their own work experience and communication with software development personnel to complete the description of the test software, manual writing test cases, time-consuming, inefficient, loopholes and more. Using the high reliability MBT tools developed by our company, the one-time modeling can automatically generate test case documents, which is efficient and accurate. UML model to describe the process accurately express the need to rely on the path is reached, the existing path generation algorithm are too simple, cannot be combined into a path and branch path with loop, or too cumbersome, too complicated arrangement generates a path is meaningless, for aerospace software testing is superfluous, I rely on our experience of ten load space, tailor developed a description of aerospace software UML graphics path generation algorithm.

1. Introduction
In the high reliability of the embedded MBT tools in the development of the project, we need to use the data flow diagram, program structure diagram, program flow diagram, state diagram, activity diagram, fault tree, causality diagram, sequence diagram, interaction diagram consists of nine types of graph modeling, graphics by quantitative expression data, process the graphic expression on the path. The test case is also based on user needs path. There are similarities in the nine kinds of path algorithm in graph primitives, there are segments linking to the source, according to the connection point and the end point can form a path; and nine kinds of graphs have many different points, such as some graphics allow loops, some graphical primitives need to appear at the same time, some need to set the connection order primitives therefore, the rationality of a robust path algorithm determines the expression of structural correctness, data correctness, logic UML modeling is correct and personification, the invention provides a set based on the path generation algorithm of UML graphics, very low time complexity, and according to the requirements of customers with graphic rules.

2. Literature Survey
At present, most of the path storage in the mature algorithm is based on the adjacency matrix [1]. Because the tools of our company need to modify the modeling graphics at any time, it will waste
unnecessary time complexity to do matrix conversion. Common tree path traversal algorithm such as Prime algorithm[2] and Kruskal algorithm[3], have different degrees of disadvantage, the Prime algorithm requires weighted transformation[4], and the concept of this scene into the right is undoubtedly redundant, and the Kruskal algorithm is based on the collection and investigation, combined with the reality of many aerospace software considerations, the algorithm of time the high complexity of the user model can be applied directly, do not need the extra conversion.

3. UML graphics modeling path generation algorithm
The expression of intention to describe detailed to, and to facilitate the reuse of the most streamlined code, complete functions, the invention by recursion, permutation and combination of knowledge, to minimize the amount of code, function block in small function in the form of writing, the pathway and loop combination (non exhaustive arrangement, the do not care about the order), the invention of the final complete test case.

3.1. determination of starting point and end point
For a path, starting point and end point must be determined after the known conditions, it is possible to calculate how many roads from starting point to end point, the following provisions in our MBT tool: the starting point and the end point is not specified in the graph, if a pixel does not point into line, is considered as the starting point; if a primitive did not point out the connection, is considered to be end point.The algorithm uses functions getStartAndEndList(Object subRootModel, List start, List end) get the start and end list of the graph, subRootModel is subgraph type, start is a list to store start nodes, end is a list to store start nodes,for example, if it is a sub data flow graph, all data source points are added to the start,all data endpoints are added to the end.

3.2. Forward pathfinding
After the getStartAndEndList function find the starting point and end point, enter the main function of the getFillProcessWithStartAndEnd(List startList, List endList, String childDiagramType, innerclass IC, String fatherNodeName), startList and endList are respectively the list starting node in S1 function and termination node list. Because the startList and endList list, corresponding to a set of starting point and a group of end point that means we have to do that when there is a one-to-one mapping of M elements startList, n elements in endList, you need to do m*n mapping.The function findroad() is the Algorithm from single starting point to single end point,this function is descendants find ancestor nodes, will involve a lot of complex loop and back.

First of all, understand the input parameters of findroad: start: starting point model; end: end point model; fatherRoad: storage path; fatherRoadId: storage path ID; resultRoads: path result combination; resultRoadsIds: path ID result set.

```java
id = start.getId();
if id == end.getId() // When the starting point ID and the end point ID
resultRoads.add(fatherRoad);resultRoadsIds.add(fatherRoadId);
return;
else if (start instanceof DfdAndModel || start instanceof FtaAndModel|| start instanceof CdAndModel){fatherRoad.addAll(newRoad);fatherRoadId.addAll(newRoadId);}
else{for (int i = 0; i < start.getIncomingConnections().size(); i++)
{Object line = start.getIncomingConnections().get(i);
 Object linesource = line.getSource();
 String linesourceID = linesource.getID();
 findroad(linesource, start, fatherRoad,fatherRoadId, resultRoads,
 resultRoadsIds,diagramType); }
}
if(fatherRoadId.contains(id)) {resultRoads.add(fatherRoad);// The node repeats, indicating that the path has resultRoadsIds.add(fatherRoadId);return;}
```
3.3. reverse pathfinding

function findSource(). The purpose of this function is to obtain the set of paths at which the node is the destination, and the pseudo code represents its algorithm:

First understand the input parameters of findSource: end: endpoint model; fatherRoad: Storage path; fatherRoadId: Storage path ID;

id == end.getId();

for(int i = end.getIncomingConnections().size(); i++ ){Object line = end.getIncomingConnections().get(i);String lineId = line.getId();newRoad.add(line); // save linefindSource(line.getSource(), newRoad, newRoadId);} 

findSource() is to find descendant nodes through ancestor nodes, without involving loops, so simple recursive logic is needed.

3.4. combination of path and loop

According to the above steps, we can get a key as the starting point of end point, value were both MAP resultRoads and resultRoadsIds were roadsloop and roadsIdloop, at the starting point and end point corresponds to a path set for each path is not resultRoads contains the starting point and end point, starting point and end point branch pathway contains, contains only the starting point then, combination of several branches and a path forming end point all possible moves to the starting point.

From the starting point A to the end point B has a r pathway, two branches of r1 and r2 contains starting point A, then from A to B have four roads, r, r1r, r2r, r2r1r; we use the combination of mathematical knowledge, in the graph with n loop, the final path is $c(n,0)+c(n,1)+c(n,2)+\ldots+c(n,n) = 2^n$ kinds of combinations with the unique path. This is based on mathematical formulas $c(n,m) = \frac{n!}{m!(n-m)!}$. 

4. Experimental Results

The modeling prototype diagram and path results is shown in Fig1:

![Fig1](image)

The blue path loops, due to our use of this algorithm does not care about the order, so the final path digital 4, if we consider the path order, the final results should be 5, only the algorithm adjusted slightly, then do not discuss.

5. Conclusion

Through the algorithm provided by this tool, find the path and loop, and then make the combination of the two, you can accurately find the original modeling diagram. The findSource() function is used to assist the findroad() function in finding a path with a node as the destination. In fact, the findroad() function is a positive lookup, and the findSource() function is a reverse lookup, and the path generation problem can be handled in two ways, in the invention, the function findSource() is only
responsible for finding the data flow graph, causality diagram and fault tree and the node under the future path, the function is clear, clear thinking, without complicated logic to achieve a simple function, the use of this tool is the premise of correct modeling, without considering the correctness of modeling, if we consider model checking, model checking rules with our MBT tools, if the check on the starting point and the end point of the model through the case of forced through the generated path will cause the path and expectations do not match.

Acknowledgements
This work is funded by National natural Science Foundation of China (No.61402131); China Postdoctoral Science Foundation Special Fund (2016T90293); China Postdoctoral Science Foundation Item (No.2014M551245); Heilongjiang Provincial Postdoctoral Science Foundation Item (No.LBH-Z13105); Central University Basic Scientific Research Business Expence Special Fund (No.HIT.NSRIF.201651); Harbin Science and Technology Innovator Special Fund (2015RAQXJ047). Science Innovation Committee technology development project (NO.CXZZ20150504150427057)

References
[1] Adiga C, Rakshith B R, So W. On the mixed adjacency matrix of a mixed graph[J]. Linear Algebra & Its Applications, 2016.
[2] PRIME interoperability tests and results from field . I. Berganza,A. Sendin,A. Arzuaga,M. Sharma,B. Varadarajan. IEEE International Conference on Smart Grid Communications (SmartGridComm) . 2010.
[3] Performance-driven simultaneous place and route for island style FPGAs. Nag S K,Rutenbar R A. Proc IEEE ACM ICCAD . 2013.
[4] Mohamed Aissa,Adel Ben Mnaouer,Rion Murray,Abdelfettah Belghith. New Strategies and Extensions in Kruskal’s Algorithm in Multicast Routing[J]. International Journal of Business Data Communications and Networking (IJBDCN).2011.