A New Test Case Generation Method Based on Data Flow Graph

Ming-cheng QU\textsuperscript{1,2,*}, Xiang-hu WU\textsuperscript{1,2}, Yong-chao TAO\textsuperscript{1,2} and Guan-nan WANG\textsuperscript{1,2}

\textsuperscript{1}School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001 Heilongjiang China
\textsuperscript{2}Shenzhen Academy of Aerospace Technology, Shenzhen 518057 Guangdong China

*Corresponding author

Keywords: Data flow graph, Model verification, Test case generation.

Abstract. It mainly solves the test lag, is not visual and the test efficiency is low, and relies on the test design of testers, and it cannot guarantee the quality of the test cases and cannot guarantee the quality of the test. According to the steps to establish the software data flow graph model to verify the model, the artificial point of the test items for planning. Finally fill in the contents of the test items and according to the algorithm to generate the corresponding test case collection, save to the database for management.

Introduction

This paper is a research on the method of test case generation based on data flow graphs, which belongs to the field of automatic testing of software engineering, the main research will be based on the application of test case generation technology of data flow graph in embedded software testing. At present, the complexity of embedded software is getting higher and higher, and the requirements of reliability and real-time are more and more strict. But the time of listing of software products is becoming shorter and shorter. To meet these stringent requirements, efficient embedded software testing is necessary, including automatic test case planning, generation, optimization, execution and evaluation. Efficient software testing needs to solve how to produce small and complete test suite as quickly as possible under the assumption that it is as weak as possible.

In view of the rapid development of embedded software and the problems faced by traditional embedded software testing, the method of test case generation based on data flow diagram is given a better solution. Its advantages are as follows: more errors were found in the early stage of the software life cycle, because in the creation of test model, test designers will find the original software requirements or model some inconsistent or incomplete information can easily achieve requirements tracking, because it is completely based on the model, and the model can be well back, according to the test case can find more defects than the traditional test case generation coverage criteria automatically from the formal model.

The logic functions of the system modeling using a graphic way, a graphical modeling engine provides the basic data flow graph modeling primitives, primitives have related internal attributes used to describe the logic functions of the system, the size and structure of the different graphical elements are different, with distinct characteristics. Defines different parameter attributes for data flow graph elements according to system logic function, and provides different settings. Data flow graph modeling contains attributes and elements that can accurately describe the logical functions of a system. The front end provides visual display of data flow diagrams, and the back-end is stored in a standard format. After the system provides verification, the test case set is generated according to the algorithm.

Graphic Model

Graphic Definition

In the embedded software test graphics system proposed in this paper, the data flow diagram contains the following graphic elements (each element has its own attributes):

192
1) Data source: is the data source of the system, similar to the source in the network system;
2) Data end point: the data end of the system, similar to the network system in the letter;
3) Data processing node: is the implementation of data processing function unit, that is, the input data processing, and then produce the corresponding output;
4) Data flow: a set of fixed data or data items, such as data packets from the header, content, including the end of the data items, should pay attention to data decomposition and management;
5) Data storage node: is the temporary storage of data;
6) and: between the data streams with "*" said between the number of data flow between the "and" relationship, indicating that several data streams must exist at the same time to continue downstream;
7) or: between the data flow with "|" said a few data between the "or" relationship between the data flow that a few data flow as long as there is a stream can be down;
8) mutex (XOR): between the data flow with "|" said from a few data streams can only select a "mutex" relationship, indicating that these data streams can only exist a data stream Can continue to flow down.

Check Rules

The drawing of the data flow pattern must follow the principle of "from the outside to the floor, one by one", through the validation rules before the use case can be generated. The drawing of data flow graphs follows the following constraint criteria:
1) data nodes in the map cannot be connected with their own, that is, the data stream starting point and the end point cannot be the same as a point;
2) the data flow diagram has at least one data source point and a data end point;
3) Data must be configured on each data stream;
4) In addition to, or mutually exclusive, each entity must have a corresponding name and not repeatable;
5) each data processing node has only one input data stream and one output data stream;
6) When there are multiple input data streams or multiple output data streams, they must be used with, or, mutually exclusive, to clarify the relationship between these data streams;
7) The data and the output data stream of the data processing node cannot have the same data, that is, the data configured on the output data stream can only originate from the internal data of the data processing node;
8) The data stream of the data storage node can only be derived from the data configured on its input data stream and its internal configuration data;
9) The data flow relation primitive must be used between the multiple data streams of the node except for the sum, or the mutually exclusive primitive.

Use Case Generation

Generate Algorithms

Algorithm name: DFD test item test case generation algorithm
Input: DFD, legally complete DFD test item I
Output: Test set of test items for DFD test items
Process:
1) Traverse all the test path sets in the legal and complete test item I;
2) Let the simple test path set U = ∅, T = ∅;
3) For i ← 1 to W.length
4) Simplify W [i] to obtain a simple test path set S of W [i]
5) Simple test path set U = U + S;
6) For j ← 1 to U.length
7) generate the test case set M of U [j] according to the evaluation point (equivalence class, boundary, coverage criterion, etc.) of U [j]
8) \( T = T + M; \)
9) return to \( T; \)

**Generate the Process**

Test item planning and test case generation are performed according to the above process and algorithm, and finally by clicking on the corresponding button to complete. Figure 1 is from the test project planning to the test case generated by the process. The test case set is generated after the test item collection is checked by comprehensiveness and correctness.

**Application and Testing**

As shown in Figure 2, if you select "1. check coins" and "2. calculate the face value" two nodes as the original hardware check test items, the tool software will be completed, complete coin check test items as shown in Figure 2 in the shape of a solid frame. According to the data flow chart test items to determine the legitimacy of the rules, the test items are legal. Finally, to carry out some of the necessary configuration to complete the test project planning.

---

**Figure 1. Data flow chart test case generation flow chart.**

**Figure 2. Automatic vending machine data flow chart.**
Take the complete coin check test item in Figure 2 as the test object.

From a set of test cases that are automatically generated from the test case generation algorithm based on the data flow test items and a set of test cases written by the testers on the part of the function, it can be seen that the set of test cases that are automatically generated includes a set of test cases written by the testers, taking into account the testing of the internal elements of the test items, the testing of the test items, and the time constraints in the test items. The In addition, it can be seen that, in generating test cases, the tool software uses the equivalent class division method and the boundary value analysis method to ensure the quality of test cases.

Summary

In this paper, a test case generation technique based on data flow graph is proposed, which is used to describe the logical function and data flow of the system. With the establishment of the graph model of data flow graphs, the reliability and safety checks and the test items and test cases are generated by manually planning the function points. This not only saves the time and cost of the test, but also ensures the quality of the test cases. Improve the reliability and efficiency of testing under the premise of reducing the burden of testers. The set of test cases at the planning of data flow plans has the following advantages:

1) Automatically generate test cases to take into account the test items within the element of the test, test the overall test and test items in the time constraints;
2) Experience reliability / safety checks to ensure the quality of use case planning.
3) Through the test items to make the test case set of the functionality of a more consistent with the system functional requirements;

Acknowledgments

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).

References

