Advances in Engineering Research, volume 166
3rd International Conference on Automation, Mechanical Control and Computational Engineering (AMCCE 2018)

Automatic Test Case Generation for Simulation Training Software

Sun Chunsheng[a, *], Wei Xiang and Dong Yanhong
Navy Submarine Academy, Shandong Qingdao 266000, China
[a]Huang_2998@163.com

Abstract: In the virtual training simulation software, users through graphical interfaces to fullfill in-formation exchange. By adopting the method of black box testing, treating the virtual training simula-tion software as a black box and input data to drive the software running, could test the interaction of graphical interface. The test case generation method based on data flow diagram (DFD), which is characterized to form a complete set of test cases and cover the entire path of the program, could im-prove the efficiency of the test and ensure the reliability of the test results.

Keywords: data flow diagram; test case generation; simulation training software.

1. Introduction

In the equipment simulation training software, due to the large number of components in the software interface and complex interactions, operations such as target selection, tracking, motion element calculation, weapon selection and use need to be completed. There are many changes in the status, and the interface operation of the software needs to be performed through functional testing. test.

In the functional testing process, the input data drivers are run and the output of the design is judged to be correct based on the design document, without considering the internal structure of the program. The input data constitutes the test case. The design quality of the test case directly determines the effect of the functional test and is the core issue of the functional test.

At present, the test case generation methods for the graphical interface include the test case generation model based on the SOFL specification [1], the test case generation method based on the interface component association diagram [2], and the graphical interface test method based on the event relationship. These methods mainly describe the realization of software functions by describing the dynamic interaction of graphical user interface objects. The main factor considered is the interface information. It does not consider the logic implementation flow of the program, and cannot grasp the actual path of the program operation, resulting in the failure of the generated test cases. By analyzing the operation information of the software interface and constructing the data flow diagram of the operation process, it can provide a basis for analyzing and mastering the working path of the program and provide support for the construction of test cases and organization test implementation [3][4].

In this paper, the process of constructing the data flow diagram of equipment simulation software is described formally. The method of generating test cases based on the data flow diagram is presented. It lays the foundation for selecting the simulation software testing strategy and generating a complete test set.

2. Equipment Simulation training Software Data Flow Diagram Generation

2.1 Definition of Data Flow Diagram

The data flow diagram describes the flow and processing of data flow in the system and is a graphical representation of the logic system. Only five basic elements are defined in the data flow diagram,

including abstract data such as data storage, flow, use, and processing in the flow, to represent data processing in the data processing [5].
(1) Data source point, data end point: The data source point and the data end point respectively represent the external source and destination of the data, and the name of the data source point and the end point should be marked.
(2) Process/machining: Process/machining represents the operations performed on data. The input data is processed here to generate output data, and indicates the processing order, such as
(3) Data flow: It is represented in the data flow diagram by arrow segments with names and constraints. The name of the data flow is shown next to the arrow, indicating the flow of data, and the arrow indicates the flow direction. A data stream can flow from one data processing flow to another, or from data processing to data storage or from data storage to data processing. It can also flow from data source point to data processing or from data processing to data end.
(4) Data storage/files: Data storage is where data is stored, including databases and files. The data stream pointing to the data store represents write data, the data stream from the data store represents read data, and the bidirectional data stream represents modify data.
(5) Real-time connection: When the process/process is executed, the external entity and the process communicate back and forth.
The data flow diagram can be described by a quad $(N, T_D, \mathrm{C}, \mathrm{F})$, where N is the set of nodes on the data flow diagram: $N = \{P, D, D_I, D_F\}$, P is a set of data processing nodes, D is a set of data storage nodes, D_I is a set of data source nodes, and D_F is a set of data end nodes. T_D is a set of directed edges on the data flow graph, representing a collection of data flows, $\forall t \in T_D$, and $t : N \rightarrow N$. C is a constraint set for each data stream. F represents the flow relationship between nodes and data flows [6].

2.2 Data flow diagram generation for equipment simulation training software

In the equipment simulation training software, the graphical interface consists of menus, buttons, text boxes, list boxes, and other components. There are complex interactions among the components, and the components themselves also have state transitions. The training personnel operate through the graphical interface. The data processing process is hidden behind the interface. If an appropriate test case is to be constructed, analysis must be performed on the software to establish a data flow diagram for the program execution path.
The graphical interface of the equipment simulation training software can be described as a quad $(N, T_D, \mathrm{C}, \mathrm{F})$, where:
N is a collection of nodes on the data flow graph: $N = \{C_I, C_A, P, \mathrm{L}, D, D_I, D_F\}$. C_I is a collection of information components on the graphical user interface. The information component includes a text box, a list box, etc. The input and output are data information, which is identified by the page and component name of the component. C_A is a collection of behavioral components on the graphical user interface. The behavior components include menus, buttons, and other locations. The input is control information. The user operates the behavior components to generate a trigger event and the driver program runs. The behavior component is identified by the page where the component is located and the component name. P is a collection of data processing. L is a set of logical nodes. D is a collection of data storage nodes. D_I is a collection of data source point nodes. D_F the set of data end nodes.
T_D is a collection of data flows. $\forall t \in T_D$, The data flow from the node n_1 to the node n_2 is recorded as $t = (n_1, n_2)$, and n_1 is recorded as $head(t)$, and n_2 recorded as $tail(t)$.
C is a constraint set for each data stream.
F indicates the flow relationship between nodes and data flows.

Take a target selection and tracking solution interface in an equipment simulation training software as an example. The main operation flow is: the user selects the target to track the attack target through a pull-down menu, and selects a method for solving a motion element to establish a tracking solution for the target. . When the elements of the target solution are stable and can meet the attack conditions, go to the attack page and select the appropriate torpedo or missile weapon to attack. If the attack condition cannot be met, a prompt dialog box is given at a predetermined time to remind the user to take other tactics in time.

3. Test Case Generation Method Based On Data Flow Graph

After the data flow diagram is established, the program's running path is clearly represented and can be used to construct test cases. Test cases refer to external settings, operation instructions, and input data and collections required by the driver to execute from one state to another. Test cases typically include a specific test path and a set of input data, including operation commands, input data, And initialize the setting data and so on. The test path describes a specific execution flow of the system. The input data refers to the specific input value given to the interactive operation that needs to be input in the test path. The design of the test case usually consists of two parts, namely the design of the test path based on the data flow diagram and the design of the input data.

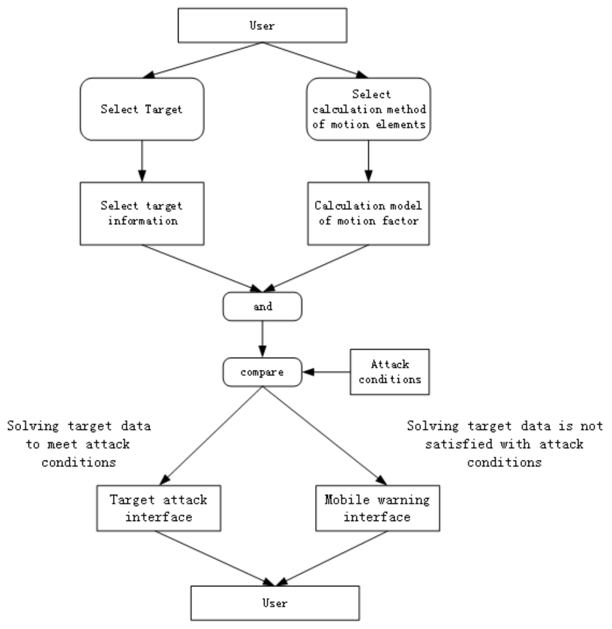
User
Select Target
Select calculation method of motion elements
Select target information
Calculation model of motion factor
and
compare
Attack conditions
Solving target data to meet attack conditions
Solving target data is not satisfied with attack conditions
Target attack interface
Mobile warning interface
User

Fig 1 Target Selection And Tracking Data Flow Diagram

3.1 Based On The Data Flow Diagram To Generate Equipment Simulation Training Software Test Path

After modeling with the data stream diagram equipment simulation training software, a test path can be proposed based on the model.

According to the data flow diagram shown in Fig 1, you can analyze and establish a test path, which can be expressed as:

Let S and be two nodes, then:

(1) If there is a data stream $t=(m,\mathrm{n})$ from m to n, then the data stream t is a path;

(2) If a, b is a path, and $tail(a)=head(b),(a,b)$ is also a path, the implementation of (a,b) needs to perform a and b;

(3) If , a ,b is a path, and $tail(a)=tail(b)$ or $head(a)=head(b)$, $(a\ and\ b)$ is also a path, the implementation of $(a\ and\ b)$ needs to implement a and b;

(4) If , a , b is a path, and $tail(a) = tail(b)$ or $head(a) = head(b)$, $(a\ or\ b)$ represents two paths, the execution of $(a\ or\ b)$ needs to execute a and b respectively;

(5) If , a , b is a path, and $tail(a) = tail(b)$ or $head(a) = head(b)$, $(a\ xor\ b)$ represents two paths, the execution of $(a\ xor\ b)$ needs to be performed separately one of a b ;

To test the program path shown in Figure 1, describe the path set of the data flow graph from the data source point to the end point as: (data source point, ((target selection drop-down box *and* motion element solution drop-down box) *and* Attack condition) , Compare (target attack page maneuver dialog box), *xor* data end point). The path set contains the following paths:

(1) (Data source point, ((Object selection drop-down box *and* Motion component solving method drop-down box)) *and* Attack condition), Comparison, Target attack page, Data end point);

(2) (Data source point, ((Object selection drop-down box *and* Motion element solving method drop-down box)) *and* Attack condition), Comparison, Maneuvering dialog box, Data end point);

The test of the interface needs to execute these two paths in sequence.

3.2 Generate Test Cases From The Test Path

Each interaction input that appears in turn in the test path is defined, including trigger events, value fields, constraints, etc. A determination table is established to arrange conditions and behaviors, and test cases are designed for each rule. For the range and constraints, combine the input data with the combination analysis method and the boundary value method to form a data set to ensure the test quality [6]. In the actual test work, the main use of design documents, the use of PICT and other combination of tools for combination design, to cover the process of program execution.

4. Conclusions

This paper aims at the test requirements of the graphical interface of equipment simulation training software, uses the method based on data flow diagram to extract the test path, and generates test cases based on this. The method can comprehensively test various paths of program operation, and can also facilitate the process of management testing combined with determination table design. It can quickly help testers find the location of the error, and is very suitable for small and medium-sized equipment simulation training software.

References

[1] A Jefferson Offutt, Liu Shaoying. Generating testing data from SOFL specification, J. The Joumal of Systems and Software.1999.49(1) 49-62

[2] Du Shuzhu, Tan Jianrong, Lu Guodong. Software Functional Testing Technology Based on Interface Component Relation Graph, J. Journal of Computer Research and Development: 2002, 39(12) 148-152

[3] Ward P.T, The transformation schema: An extension of data flow diagram to represent control and timing, J. IEEE Transactions on Software Engineering, 1986(2) 198-210

[4] Mary Jean Harrold, Testing: a roadmap, J. ICSE '00 Pro-ceedings of the Conference on The Future of Software En-gineering, 2000 61-72

[5] Jeff Offutt, Shaoying Liu, Aynur Abdurazik, Paul Ammann, Generating test data from state-based specifications, J. Software Testing, Verification and Reliability, 2003 13(1. 25-53

[6] Romli R., Automatic programming assessment and test data generation a review on its approaches, J. Information Technology (ITSim), 2010 (3). 1186 - 1192