Research on automatic test case generation method of flight control system based on UML state diagram

Chunlong Fan¹ and Penghua Zou¹*

¹School of computer, Shenyang Aerospace University, Shenyang, Liaoning, LN24, China
²School of computer, Shenyang Aerospace University, Shenyang, Liaoning, LN24, China
*Corresponding author’s e-mail: zphjh1109@163.com

Abstract. In order to realize the automatic test of flight control system, the automatic generation method of flight control system test case is studied by using UML state diagram. Using UML extension mechanism, the real-time extension scheme of UML state diagram of flight control system is given. Aiming at the real-time performance of flight control system, the migration equivalence class is constructed by using the time domain equivalence partition method. After the test path is generated according to the corresponding test standard, the test case is generated automatically. Taking a simplified flight control system as an example, the feasibility and practicability of the method are verified by comparing the time of generating test cases with that of generating test cases manually.

1. Introduction
As one of the core systems of aircraft, it is necessary to fully and effectively test the flight control system in the test process [1]. In the traditional flight control system test, due to its complexity and real-time, test cases are often written manually, but this way requires a lot of manpower and material resources, and low efficiency, easy to miss some test cases [2]. Therefore, this paper proposes an automatic test case generation method for flight control system based on UML state diagram, and extends the UML state diagram in real time according to the strong real-time characteristics of flight control system. In this paper, a real-time extension scheme of UML state diagram is given. According to the time characteristics, the migration equivalence class [3] is constructed by using the method of time domain equivalence partition, and the migration test tree is generated by using the migration equivalence class. Combined with the defined test coverage criterion, the corresponding test paths and test cases are generated.

2. Test case design process based on UML state diagram
Test case is a set of operation and input data designed according to test requirements for driving system to perform a specific action [4]. Typically, a test case includes a test path and the corresponding test data. Therefore, generating test cases should include generating test cases and generating test data.

When testing the flight control system, we should not only analyze the requirements, but also refer to the interface control document (ICD), which stipulates the specific interface relationship between each subsystem, equipment or other systems, subsystems, equipment in the flight control system. Figure 1 shows the design process of flight control system test cases. Firstly, UML state diagram is used to...
generate all test paths, and then the given ICD document is used to generate test data for all input operations corresponding to the test path. Finally, test cases are generated according to a certain optimization combination strategy.

![Test case design process of flight control system](image)

Figure 1. Test case design process of flight control system.

3. Real-time extension of UML state diagram

3.1. UML extension mechanism

UML provides a large number of views for system modelling, but in order to avoid the overall complexity of UML, UML does not absorb all object-oriented modelling techniques and mechanisms, and support their own expansion and adjustment, which is the expansion mechanism of UML.

UML provides three extension mechanisms: prototype, tagged value and constraint to facilitate users to extend UML [5].

- Stereotype: Define new UML modelling elements based on existing model elements.
- Tagged value: Add specification of special information extension model elements.
- Constraints: Extending element semantics by adding new rules.

3.2. Characteristics of Flight Control System

Flight control system is the core component of aircraft, which undertakes flight control, task scheduling management and other functions. It has the characteristics of strong real-time, concurrency, high reliability, and is often cross-linked with multiple subsystems. Therefore, it is difficult to develop and test the flight control system. In the traditional flight control system test, test cases are often completed by manual input or manual script writing, but this method requires a lot of manpower and material resources, and the efficiency is low. Therefore, it is meaningful to apply UML modeling automation testing technology to large flight control system testing.

3.3. Real-time extension scheme

UML state diagram can provide very limited methods to describe the timing characteristics of the system, which can not provide a specific method to deal with the timing-related characteristics of the system and the complex of these characteristics.

In order to solve the problem that UML state diagram describes the time characteristics of flight control system, this paper extends the state diagram in real time, including:

- clock. The event can be associated with time by introducing the concept of clock, which can be divided into global clock and local clock according to the system requirements. The global clock starts from the test, and the local clock records the time when an activity or a function is called during the test execution. When the system enters each activity, the local clock will be reset once.
- time constraint. The transition of many events in flight control system is associated with time constraints. The occurrence of events in a specific time period will affect the state of the whole system, thus affecting the behaviors of the system, which is called time constraints.
time-constrained behaviors. When describing the time characteristics of the flight control system, the state diagram should be able to accurately describe some basic time-constrained behaviors, such as clock constraint migration, timeout events, time delay, etc.

4. Test case generation of flight control system based on UML state diagram

4.1. Constructing Transfer Equivalent Classes
In the UML state diagram, time constraints divide the clock range into several time zones, each of which has different effects on the number of migration trigger events. To facilitate discussion, some conceptual definitions are given:

4.1.1. Time zone
T is a set of clocks, t is a clock in X with a range of (0, +∞). The time constraint σ of T divides the value range of the clock into multiple continuous intervals, that is, the time domain division, which is represented by symbol D, and each interval becomes the time domain. Specific definitions are given below:

- Definition 3.1 Time zone
  Assuming that σ is a time constraint, the time points contained in its expression are as follows:
  \( G_1, G_2, ..., G_k \) are positive integers, and \( G_1 < G_2 < ... < G_k \), then the time of σ is divided into:
  \( \{ (0, G_1)^*, (G_1, G_2)^*, ..., (G_k, +\infty)^* \} \), \( (0, G_1)^*, (G_1, G_2)^*, ..., (G_k-1, G_k)^*, (G_k, +\infty) \) is a time zone in the partition, expressed as \( d_1, d_2, ..., d_k \), with \( , \ast \) denotes [ or ).

- Definition 3.2 Effective time region
  Let E be an event set, \( e \in E \), D be a time-constrained partition of e, \( d \in \mathbb{Z} \). If the event e can trigger migration within d, then d is called an effective time region of e.

- Definition 3.3 Invalid time region
  Let E be an event set, \( e \in E \), D be a time-constrained partition of e, \( d \in \mathbb{Z} \), if the event e cannot trigger migration within d, then d is called an invalid time region of e.

4.1.2. Equivalent division of clock region
This method corresponds the number of clock variables to the dimension of coordinate axis, describes the different situations corresponding to different clock numbers, and divides the corresponding clock region into three types of intervals: point interval, line interval and open interval, as shown in figure 2 and figure 3. In figure 2, there is only one clock, the time region is divided by a one-dimensional coordinate axis. In figure 3, there are two clocks, the time region is divided by a two-dimensional coordinate axis. There are two clocks a, b, and their clock constraints are (0,3). In the figure, \( \{ a=2, b=2 \} \) is a point interval, \( \{ 1< a<2, b=2 \} \) is a line interval, and \( \{ 1< a<2, 1 <b<3 \} \) is an open interval. In this paper, there is a single clock and two clocks coexist.

4.1.3. Constructing Transfer Equivalent Classes
Events trigger different migrations in different time regions. In this paper, the migration equivalence class is constructed by using the method of clock region equivalence partition. The equivalence class includes a state set and three action sets, which are necessary, possible, and not. In this article, a migration equivalence class contains four parts: number, event, time constraint, and subsequent state.
Because of the coexistence of multiple clocks in the flight control system, when describing this kind of situation, the time region is equivalently divided, and all possible time regions are combined. As there are more clocks, the dimension of division is higher. This paper only discusses one and two clocks. The method description of constructing migration equivalence class is given as follows:

- firstly, the migration set ES of the trigger event of state S is generated. Events in ES trigger state migration.
- select the events in the collection ES in turn, set to e, and follow the following steps:
  - the migration set of event e is generated, and its time constraint set is constructed by the migration set, and the time domain division of event e is generated.
  - the migration equivalence class is generated according to the following rules: If there is only one clock, then each time domain corresponds to a migration equivalence class. If the time domain corresponds to the time constraint of a state migration, the next state of the migration equivalence class is the target state of the migration. If the state transition is not triggered, the next state is NULL. If there are two clock variables, the time regions of the two clock variables are combined to generate a new time region. If the new time region corresponds to a certain migration time constraint, the next state of the migration equivalence class is the target state of the migration. If no state migration is triggered, it is NULL.

4.2. Definition of test coverage criterion

When constructing test cases, certain test coverage criteria should be followed. After studying the white-box test coverage criterion and graph test coverage criterion, this paper defines the following test coverage criterion combined with the extended UML state diagram:

- state coverage criterion. When analyzing the UML state diagram to generate test cases, it is required that the generated test case set can fully cover all in UML state diagram.
- migration coverage criterion. The migration coverage criterion requires that the test case set generated based on UML state diagram can traverse all migrations in the state diagram.
- complete determination of coverage criteria. The complete criterion in real-time systems such as flight control system is the adequacy test for the time domain determined by time constraints. Due to the time constraints of the real-time system, in different time domains, the state of the system will have different migrations for the same input event. Therefore, it is necessary to test all possible time domains of the system in the test, so as to ensure that the description of time characteristics is reasonably realized.

4.3. Test paths generation

The above migration equivalence class is used as a node to generate a migration test tree. Combined with the test coverage criterion, all test paths can be obtained by using the depth traversal algorithm. The following is the description of migration test tree generation algorithm as table 1:

| Algorithm: Migration test tree algorithm |
|-----------------------------------------|
| Input: Real-time extended UML state diagram |
| Output: Migration test tree |
| 1) Initialization: Let nstate point to the initial state of the state diagram, the migration test tree is T, T contains a root node root, the target state of root is nstate, and let node point to the root node. |
| 2) If node points to no child node |
|   - If the destination state of the node that the node points to is NULL Then |
|   - If node points to the root node Then |
|   | | | Continued table 1. A slightly more complex table with a narrow caption. |
| Algorithm: Migration test tree algorithm |
| end |
| Else |
|   Node points to the parent node of the current node and goes to step 2). |
Else
   nstate points to the next state of the node and constructs the migration equivalence class of nstate
   For each migration equivalent class, backtracking the superior node
      If migration equivalence class does not appear on the traceback path Then
         Set the child node of the nstate node node node
      If nstate points to the node without child nodes
         Node points to the parent node of the current node and goes to step 2).
      Else
         Node points to the first child node of the current node and goes to step 2).
   (3) If node points to a child node Then
      For each child node of a node
         If the node has not been accessed Then
            node points to the node and marks the node as accessed and goes to step 2).
         If each node has been accessed and the node points to the root node Then
            End
      Else
         Node points to the parent node of the current node and goes to step 2).
      After the algorithm is completed, a migration test tree is built. Use depth traversal to get all possible test paths from root to child nodes.

4.4. Test data generation
Test input is mainly obtained from ICD documents, from the following aspects to describe the input data:
   • data types: In addition to the common int, long, double, and bool data types in the flight control system, there are also a large number of composite data, such as structural body data. A uniform, standardized definition of such composite data is required to facilitate automatic generation of test cases.
   • value range: Deterministic input variables can be selected directly according to the demand. Uncertainty variables, such as variables with upper and lower value boundary, the selection is more complex. In order to find as many errors as possible, it is necessary to use a variety of test strategies such as equivalence class division, boundary value and special value.
   • time constraint: Due to the time characteristics of the flight control system, the test input should also meet certain time constraints. Effective input data in time constraints and abnormal input data not in time constraints.

4.5. Test cases generation
Through traversing all the test paths, the control flow, data flow and branch conditions in the execution process can be obtained, the input test data and state conditions required for each path can be determined, and the corresponding test cases can be generated.

5. Case analysis
Taking a simplified flight control system as an example, the method of automatically generating test cases based on UML state diagram is given.

The UML state diagram of the flight control system is mainly described as follows. When the system is in the program control stage, it starts to enter the ground stage. In the ground state, there are two sub-state navigation states and hydraulic system operation. The two sub-states are controlled by two clocks Timer1 and Timer2. If the hydraulic system failure within 5 minutes into the warning and protection, exit after 2 minutes. If the hydraulic system is good and the navigation algorithm is completed within 3 minutes, it enters the takeoff stage. If a termination take-off order occurs, take-off from taxiing to termination take-off, at the end of take-off, if the speed is less than 5km/h, in the shutdown phase, exit after 4 minutes. If the speed is above 5 km/h within 3 min when the take-off is terminated, it enters the warning and protection state entered, and the forced with draw after 2 minutes.
5.1. UML state diagram establishment

For the convenience of subsequent description, the UML state diagram of flight control system after real-time expansion is given, which eliminates the formal description after hierarchy, as shown in Figure 4; Table 2 illustrates the meaning of state and migration.

![Figure 4. Formal description of flight control system UML state diagram.](image)

| State ID | State name                | Migration ID | Migration information                          |
|----------|---------------------------|--------------|-----------------------------------------------|
| S0       | program control           | e1           | navigation algorithm completed [Timer1>3]     |
| S1       | navigation state          | e2           | hydraulic system works well [Timer2>5]        |
| S2       | operation of hydraulic system | e3       | operation failure of hydraulic system [0<Timer2<5] |
| S3       | skate take-off stage      | e4           | take-off is not allowed allow=0               |
| S4       | end take-off stage        | e5           | Gspeed>5km/h [Timer1>3]                       |
| S5       | shutdown stage            | e6           | Gspeed<5km/h [Timer<3]                        |
| S6       | warning and protection    | e7           | wait [Timer1>4]                               |
| S7       | exit                      | e8           | wait [Timer1>2]                               |

5.2. UML state diagram establishment

According to the real-time extended UML state diagram, the migration equivalence class is constructed, and the migration test tree is established by using the migration test tree generation algorithm, as shown in figure 5.

![Figure 5. Migration test tree.](image)

5.3. UML state diagram establishment

A deep traversal method is used to traverse the migration test tree and generate the corresponding test path, as shown in table 3.

| No | Test path                                                                 |
|----|----------------------------------------------------------------------------|
| P1 | S0→S1∨S2→e1∨e2{run(5, +∞)}→S3→e4 | take-off is not allowed allow=0 → S5 → e6 {wait(0,3), speed<5km/h} → S5 → e7 {wait(0,4)} → NULL |
Each test path combines the actual test data and migration conditions to generate the corresponding test cases. The set of test cases corresponding to all paths is the final test case set.

6. Conclusion
The test case generation method based on UML state diagram is applied to flight control system test, and the real-time extension scheme of UML state diagram is given. Using the method of time zone partition, the UML state diagram is transformed into a migration test tree with the migration equivalence class as the node, the coverage criteria are defined, the test path is generated, and the test cases are automatically generated through certain transformation rules. In the actual project, compared with the time of generating test cases manually, the time of generating test cases in the original 30 days is shortened to 5 days after using this method. This method can effectively improve the test efficiency of flight control system and reduce the error of test cases caused by human factors.

Acknowledgments
This work was part of the Program on Youth Science Foundation Project funded under the National Natural Science Foundation of China, grant number 61902260. The authors are grateful to all study participants. We also thank Professor Fan for his help and support in the experiment.

References
[1] Yang S B, Zhang J, Qin X X, et al.(2020)Study on the Flight Test Risk Assessment of Civil Aircraft Flight Control System Failure. Journal of Aeronautics, Astronautics and Aviation, 52(2 Vol.52 No.2):171-181.
[2] Liscouet-Hanke S, Jahanara H, Bauduin J L.(2020)A Model-Based Systems Engineering Approach for the Efficient Specification of Test Rig Architectures for Flight Control Computers. IEEE Systems Journal. I.S. Jacobs and C.P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G.T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271-350.
[3] Dappiaggi C, Drago N, Longhi R.(2020)On Maxwell's Equations on Globally Hyperbolic Spacetimes with Timelike Boundary[J]. Annales Henri Poincaré, 21(7):2367-2409.
[4] Madeja M, J Porubn, M Baíková, et al.(2021)Automating Test Case Identification in Open Source Projects on GitHub. Computing and Informatics, Vol.32:1001-1031
[5] WANG PanLIU Guang-zhong.(2019)Research on Fault-tolerant Clock Synchronization Based on Passive Clustering [J]. Computer Technology and Development, (001):159-162.