Research on Formal Verification Technique for Aircraft Safety-Critical Software

Yongfeng Yin, Bin Liu
Department of System Engineering of Engineering Technology, Beihang University, Beijing, China
Email: {yyf, liubin}@buaa.edu.cn

Duo Su
Center of Aviation Safety Technology, Civil Aviation Administration of China, Beijing, China
Email: suduo@263.net

Abstract—**As an important part of airborne avionics system, aircraft safety critical software (ASCS) plays an essential role to the safety of the aircraft, and to ensure its quality and reliability is one of the key problems we are facing. Formal methods have become important means for modeling and verifying safety critical software. In this paper, formal method is introduced into the ASCS verification field and the real-time extended finite state machine model (RT-EFSM) is studied, which includes the detailed real-time extension schemes and its validation methods. The verification process of ASCS based on RT-EFSM is also proposed. Furthermore, combined with the verification of an aircraft inertial/satellite navigation systems, a timed unique input/output sequence (t_UIO) is presented and the automatic generation algorithm of sub-transfer sequences based on t_UIO is given. Finally, the test adequacy criteria are discussed and a time condition coverage criterion is proposed. The actual engineering project application shows that the method proposed in this paper is of great value for the ASCS, which can be generally and effectively used in engineering.**

Index Terms—**real-time embedded software; aircraft; safety critical software; RT-EFSM; formal methods; verification; test sequence**

I. Introduction

As the development of computer technique, a wide range of complex hardware and software systems embedded in a large number of safety-critical systems, and play an irreplaceable role. In general, Safety-critical systems are computer, electronic or electromechanical systems in which failure may have severe consequences such as injury or illness, death of humans, serious environmental damage or failure of important mission. Safety-critical software, as a part of the safety-critical systems, is an important factor to safety-critical systems[1][2].

The software verification technique based on formal methods can eliminate ambiguity, enhance the accuracy and consistency of verification and improve the automation level and efficiency. Both European Space Agency (ESA) and NASA highly recommend using formal methods for the development and testing of safety-critical software[3]. Therefore, the verification technique based on formal methods is an effective way to ensure the quality of ASCS, also it guarantees the safety critical software meets the airworthiness requirements [15]. ASCS usually shows the state based behaviors wholly or partly, and one of the most commonly used formal method for testing this kind of embedded software is FSM or EFSM [10] [11] [12] [13]. Literature [4] proposed an assertion-based extended EFSM model p-EFSM for embedded software testing; literature [5] proposed a real-time finite state machine (RTFSM) model for numerical control system micro-fabrication (NCS); literature [6] presented a testing method which is based on incremental conformance; literature [7] proposed a kind of EFSM-based test input data automatically selected method; literature [8] showed an time extended finite state machine to describe the system modeling with time characteristics; literature [9] presented a testing method based on statecharts and temporal logic that oriented reactive system; literature [14] gave a EFSM model based testing method combined with hardware structure, and adopting fault injection method.

From the existing research, we can see that there is lack of a universal and effective formal verification method which has practical value in engineering for ASCS. The problems of the existing research are as fallows:

- Existing verification methods can not meet real-time requirements of ASCS, such as the lack of the temporal characteristics description during the state transition.
- The cross-linked devices, interfaces, state types and transition of ASCS are complex, while the existing methods to solve these problems is not well.
- The temporal characteristics description and time-related test adequacy criteria of existing test sequence generation method is obviously insufficient.

In this paper, grounding on abundant experience and research on real-time embedded software testing technique, we introduced formal method into the ASCS verification field and proposed a real-time extension of EFSM model, named RT-EFSM, and gives its validation methods. At the same time, we put forward a timed unique input/output sequence (t_UIO) and its automatic generation algorithm. Finally, we applied the method to the verification of an aircraft inertia/satellite navigation system, and the results proved the correctness and validity of the method.

II. RT-EFSM MODEL AND ITS VALIDATION

A. *RT-EFSM Model*

The traditional EFSM model is a six-tuple $<S, S_0, I, O, T, V>$:

S: non-empty set of finite states;
S_0: the initial state;
I: non-empty finite set of input;
O: non-empty finite set of output;
T: non-empty finite set of transition;
V: non-empty set of variables.

The traditional EFSM model is lack of the ability to describe the complexity and real time characteristic for ASCS, as a result, it can't meet the requirements of modeling for ASCS. As we consider, the ASCS model and validation theory which is based on real time extended EFSM have to solve the problems below, as it is shown in Table 1.

TABLE I. THE PROBLEMS SHOULD BE SOLVED FOR RT-EFSM

No.	Description
1	How to describe the static and dynamic behavior of ASCS
2	How to describe the real time characteristic of ASCS
3	How to describe the time characteristic in transition of ASCS
4	How to solve the problem of states explosion
5	How to describe the complicated transition relationship of ASCS
6	How to ensure the correctness and consistency of the ASCS models
7	How to lay the foundation for the test cases automatic generation

Based on the analysis above, in this paper we propose the RT-EFSM model as the theoretical basis and extend the EFSM from six-tuple to eight-tuple.

$RT\text{-}EFSM=<S^*,S_0,I,O,T,V,C,E>$,and:

S^*:non-empty set of finite states, and $S^*=<S_e,S_i,S>$,and S_e is the set of states with embedded relationship; S_i is the set of states with inherited relationship; S is the set of the other states;

S_0: the initial state;

I: non-empty finite set of input;

O: non-empty finite set of output;

T: non-empty finite set of transition; and $T=<Head(t),I(t),P(t),operation,O(t),Tail(t),TM>$; and Head(t) is the start state of transition; $I(t)$ is the input message included in set I; $P(t)$ is the precondition of transition t and it could be empty; operation is the operation during transition process, usually it includes assignment statement and output statement; $O(t)$ is the output message included in set O; $Tail(t)$ is the end state of transition; TM is transition time, which is three-tuple$<t_f, t_F, t_I>$[8], t_f indicates that the transition time is a fix time, t_F indicates that the transition time is a stochastic time complies to some distribution function; t_I indicates that the transition time is a time interval;

V: non-empty set of variables, and $V=<IV, CV, OV>$; IV is the set of input variables, and they are under tester's control; OV is the set of output variables; CV is the set of environment variables, which can be local variables or global variables, and the variables that are neither input variables nor output variables can be called environment variables; both CV and OV are out of the tester's control, their values are decided by state transition operation.

C: the set of static information of each state, $C=<NAME(S^*),Variable(S^*),Action(S^*)>$, and $NAME(S^*)$ is state label, $Variable(S^*)$ is the related variable to the state ; $Action(S^*)$ is the action and activity of the state;

E: non-empty set of directed edge; it is an extension to the original EFSM edge, and connection, combination connection, selection joint are added, which can describe the system dynamic behaviors better, as it is shown in figure 1.

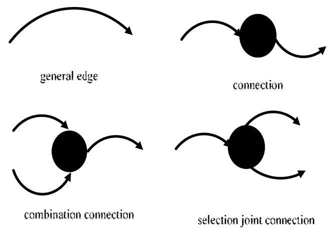

Figure 1. Extended directed edge in RT-EFSM

For the complex real-time embedded system as ASCS, the state number is usually large, which leads to great problems for testing. However, there often exists some kind of relationship between the states, making some rules for us to follow. For example, some state is embedded in another one or some state is inherited from another one. So, we extend the tuple S in EFSM to S^*. S^* is defined as the non-empty set of finite states, and S_e is the set of states with embedded relationship; S_i is the set of states with inherited relationship; S is the set of the other states. After extending, the RT-EFSM model can describe the system under test more clearly and accurately, helping the testers to understand the states and the relationships between them. Meanwhile, the extension can solve the state space explosion problem to a certain extent.

The tuple TM in T describe three different time situations when transferring from one state to another:fix time, a stochastic time complies to some distribution function and time interval. The fix time means that the time a certain transition costs each time is the same fix time, such as 50 time periods; a stochastic time complies to some distribution function means that although the time a transition costs are different each time, it follows certain rules, namely it complies to a distribution function, such as uniform distribution and index distribution; time interval means that the transition time is an interval, namely $[t_1, t_2]$, and this interval could be open, closed, or half open half closed, which can describe the delay transition.

The tuple *C* is the set of static information of each state, which can be detailed to *C=<NAME(S*),Variable(S*), Action(S*)>*. When testing some state of the ASCS, the testers can get the related variables, operation information with it easily. Meanwhile, the test platform can read related information from this state information table to support automatic generation test cases.

The tuple *E* is an extension of the original EFSM model general edge for the complex states transition relationship in ASCS, and connection, combination connection, selection joint are added. Connection is used to connect more than one transition or divide one transition into an array of continuous segments. Regardless the number of transition segments, they are all executed in one process from running to completion. Combination connection is a kind of connection, which connects more than one transition to an entrance ‘or’ state transition, especially several transitions triggered by different events share the same action list and/or guard condition, or transferring to the same state. Selection joint connection is a kind of connection that connects its action list before entering the next transition segment, which allows binding the actions to the first transition segment in order to be executed before the following guard expressions valued. The extension to the edge makes RT-EFSM model more suitable to model ASCS, and the test cases number can be decreased and efficiency can be enhanced when automatic generating test cases.
From the analysis above we can see that, the proposition of RT-EFSM model can solve all the problems listed in Table 1 efficiently.

B. The Verification Process of ASCS Based on RT-EFSM model

The verification process of ASCS based on RT-EFSM is shown in figure 2.

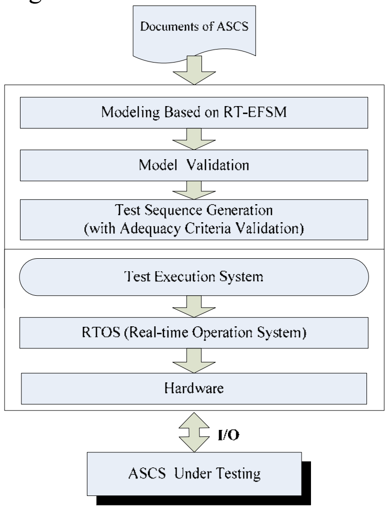
Documents of ASCS
Modeling Based on RT-EFSM
Model Validation
Test Sequence Generation (with Adequacy Criteria Validation)
Test Execution System
RTOS (Real-time Operation System)
Hardware
I/O
ASCS Under Testing

Figure 2. The verification process of ASCS based on RT-EFSM

In this paper we focus on the most important three steps in figure2, including modeling based on RT-EFSM, model validation and test sequence generation.

C. Model Validation Method

After modeling ASCS based on RT-EFSM, the RT-EFSM model needs to be validated to guarantee its consistency, determinacy (includes static determinacy and dynamic determinacy), reachability and synchronicity. The static determinacy can guarantee the RT-EFSM model is minimal; the dynamic determinacy can guarantee the RT-EFSM model is completed; the rechability can guarantee the RT-EFSM model is strong connected. We have researched on all the four aspects mentioned above and figured out corresponding algorithms.

1). Determinacy Validation

- Static Determinacy

Static determinacy for RT-EFSM model means that in any state, the target state is unique for a determinate transition. If a RT-EFSM model is not determinate, it means the system could execute different state transitions in the same situation which makes the test can’t be carried on.

The algorithm to judge indeterminacy for RT-EFSM model is:

```
Input: RT-EFSM model M=< S*，S0，I，O，T，V，C，E >
Output: return false if M is indeterminate, else return true
M_Certain(Si， ti， tj)
{
for_each Si ∈S*
for_each ti， tj ∈T(Head(ti)==Head(tj)=Si)
   if((ti.I== tj.I)&&( ti.P== tj.P)&&( ti.o== tj.o)&&( ti.O==
tj.O)&&(Tail(ti)!=Tail(tj)))
return false
return true
}
```

- Dynamic Determinacy

The tuple *V* of RT-EFSM eight tuples is a three-tuple<*IV, CV, OV*>, and *CV* refers to the set of environment variables. *CV* is out of control of testers and its value depends on the operation of state transition. As a result, if there is pre-condition in state transition and there are inner environment variables in pre-condition, the RT-EFSM model is indeterminate, that is because the next state not only depends on the current state and input, but also depends on the environment variables in the pre-condition. We call this kind of indeterminacy dynamic indeterminacy. For the RT-EFSM model whose pre-condition includes input variables, we still consider it determinate, since the input variables are under tester’s control. As long as we choose appropriate input variables values, all the pre-conditions will be TRUE, thus the pre-conditions will not impact the state transition, so this kind of RT-EFSM model is determinate.

We can use the state decomposing method to convert an indeterminate RT-EFSM model to a determinate RT-EFSM model. The key point is to eliminate the state transition indeterminacy incurred by inner environment variables, namely after conversion, all the transitions are not constrained by inner environment variables. Adopting the equivalence class partition method, we divide the state whose transition is constrained by inner environment variables into sub states which are independent to each other. We use M to represent RT-EFSM model, S^* is the state set of M, $S_i \in S^*$, for $t_i \in out(S_i)$ ($out(S_i)$ is the set of transitions that started from S_i), if there is pre-condition in t_i and environment variables in this pre-condition, we divide S_i into two states. The algorithm is:

```
S* is the state set of original M=< S*, S0, I, O, T, V, C, E >,and:
    1) choose a state $S_i$ from $S^*$, and set the transition set started from $S_i$ out($S_i$);
    2) if the state transition in out($S_i$) with pre-condition and it includes environment variables, the state $S_i$ can be divided into $S_{i1}$ and $S_{i2}$ two sub states, and $S_{i1}$ for pre-condition is FALSE, and $S_{i2}$ for pre-condition is TRUE;
    3) if there are state transitions with pre-conditions still to be deal with, we continue to divide the state.
    4) if all the state transitions in out($S_i$) are done, $S^*= S^*- S_i$, if $S^*=\Phi$, end the division, otherwise go to 1).
```

2). Reachability Validation

An unreachable state means that there is no event sequence from the start state of RT-EFSM model to make the system reach the state. The reachability of RT-EFSM model refers to there is no unreachable state in it. The unreachable state is redundant and we should delete it and its related transitions. The algorithm is:

```
Input:RT-EFSM model M=< S*, S0, I, O, T, V, C, E >
Output:RT-EFSM model after deleting unreachable states M'=< S*', S0, I, O, T', V, C', E >
Remove_Unreachable_State($S_i$, $t_i$)
{
  $S_a=\{S_0\}$, $S_b=\{S_0\}$;
  while($S_b!=\Phi$)
  {
    get $S_i$ from $S_b$;
    for_each($t_i \in T$ && Head($t_i$)== $S_i$)
      if(Tail( $t_i$)! $\in S_a$)
      {
        $S_a = S_a \cup Tail(t_i)$;
        $S_b = S_b \cup Tail(t_i)$;
      }
      remove $S_i$ from $S_b$;
  }
  $S^{*\prime}= S_a$;
  $T'=\{t|\ Head(t_i) \in S^{*\prime} \cap Tail(t_i) \in S^{*\prime}\}$;
  return M';
}
```

3). Consistency Validation

The consistency of RT-EFSM model means that there is no confliction or redundancy in it. Specifically, it means that there is no equivalent state or unreachable state in it. By the premise of modeling ASCS with RT-EFSM model, we should consider the value range of state variables and if there is confliction between state transitions. Thus, the consistency of RT-EFSM model includes no state repetition or state-transition confliction.

Equivalent state refers that there are such two states that they transfer to the same target state when triggered by same event or condition. Namely, there is equivalent state in a RT-EFSM model if the following condition is satisfied.

$$\exists S_i \in S^*, \exists S_j \in S^*, t_i \in T(S_i), t_j \in T(S_j), \text{and:}$$

$$[Tail(t_i) = Tail(t_j) \cap t_i.I = t_j.I \cap t_i.P = t_j.P \cap t_i.o = t_j.o \cap t_i.O = t_j.O]$$

The algorithm to judge and delete equivalent state in a RT-EFSM model is:

```
Input:RT-EFSM model $M=< S^*,S_0,I,O,T,V,C,E >$
Output: RT-EFSM model after deleting equivalent state $M'=< S^{*'},S_0,I,O,T',V,C',E >$
Remove_ Equivalent_State($S_i$, $S_j$, $t_i$, $t_j$)
{
  $M'$=M;
  for_each $S_i$, $S_j \in S^*$
  {
    $T_1=\{t|Head(t)= S_i\}$;
    $T_2=\{t|Head(t)= S_j\}$;
    for_each $t_i \in T_1$
      if($t_j \in T_2$((Tail($t_i$)==Tail($t_j$) &&($t_i$.I== $t_j$.I)&&( $t_i$.P== $t_j$.P)&&( $t_i$.o== $t_j$.o)&&( $t_i$.O== $t_j$.O))) {
        remove $t_j$ from $T_2$
      }
        if($\Phi$==$T_2$)
        {
          $T_3=\{t|Head(t)=S_j\ ||\ Tail(t)=S_j\}$;
          remove $S_j$ from $S^{*'}$;
          remove $T_3$ from $T'$;
        }
  }
  return $M'$;
}
```

State-transition confliction refers that when trying to transfer from one state to another, there is confliction between pre-condition and the variables, as a result the transition can't happen. Namely, there is state-transition confliction in a RT-EFSM model if the following condition is satisfied.

$$\exists S_i \in S^*, t_i \in T_i[(t_i.P \cup t_i.I) \cap S_i.V = \varnothing]$$

The algorithm to check if there is state-transition confliction in a RT-EFSM model is:

```
Input:RT-EFSM model M=< $S^*$,$S_0$,I,O,T,V,C,E >
Output:state-transition confliction couple set $ST_{cof}$={<$S_1$, $t_1$>...}
State_Transition_Conflict
{
 for_each $S_i \in S^*$
    for_each($t_i \in T$&&Head($t_i$)= $S_i$)
       if((($t_i$.P|| $t_i$.I)$\cap$$S_i$.V)= $\Phi$)
          $ST_{cof}$.add (< $S_i$, $t_i$>);
 return $ST_{cof}$;
}
```

4). Consistency Validation

For the real time embedded system modeled with RT-EFSM, there is synchronization problem between different test ports transitions. We classify the synchronization problem into the First Class synchronization problem and the Second Class synchronization problem and we discuss them separately.

- The First Class synchronization problem

Set t is a transition in M_i, S is a local state in M_j $(i\neq j)$, if there is t' (S=*head*（t'）) which has the same input as t, then S is a synchronous input state of t, and the set comprises of S is marked *Con(t)*.

Set *tx, ty* are transitions in M_i, and M_j, $(i\neq j)$, if transition sequence *tx ... ty* satisfy:

(1) *Tail(tx)* $\in$*Con(ty),*

(2) if any transition t_z $(z\neq x, y)$ is not in M_i, then the transition sequence is called the First Class synchronous input transition sequence.

We call the test sequence that includes the First Class synchronous input transition sequence existing First Class synchronization problem. The test sequence R with First Class synchronization problem may lead to test sequence R' be executed which is different from R. To avoid First Class synchronization problem, we add synchronization Lock declaration for T_i:

```
Check Lock L; // check L is 0 or not
Lock L; // L++, L is 1, synchronization Lock
// L is locked
Unlock L;// L--, L is 0
```

Lock L executes operation L++, Unlock L executes operation L--, Check Lock L is to check L is 0 or not. The operation Lock L and Unlock L are unblocked. If the synchronization Lock L is locked $(L\neq 0)$, the operation Check Lock L is blocked, otherwise it is unblocked.

The algorithm to solve the First Class synchronization problem is as below:

```
Input: local transition sequence set $E$ $(E_1, E_2, ...E_n)$ of RT-EFSM model, and $E_i$ is a continuous transition sequence;
Output: test sequence $R$;
   (1) build a synchronization Lock queue $Q$. For $t$ in non-empty set Con(t), if there is $S \in Con(t)$ traversed by $E$, then $t$ is defined as a synchronization Lock $L_i=0$;
   (2) empty all test drive $T_i$;
   (3) for each $t$ in $E$:
     ① if $t$ is a transition in $Q$, then before the declaration related to $t$ and the declaration Check Lock $L_i$;
     ② if Tail(t) $\in$ Con(t'), $t'$ is a transition in $Q$, and its corresponding synchronization Lock is $L_j$, then before the declaration related to t add the declaration Check Lock $L_j$;
     ③ if the input of transition t in $M_i$ is global input $A$, then add "Input A" in the end of queue $T_i$;
     ④ if the input of transition $t$ in $M_i$ is local input $B$ from $M_j$, then add "Receive B" in the end of queue $T_i$;
     ⑤ if the output of transition $t$ in $M_i$ is local output $C$ pointing to $M_j$, then add "Send C to $M_j$" in the end of queue $T_i$;
     ⑥ if Head(t) $\in$Con(t'), and $t'$ is a transition of $Q$, and its corresponding synchronization Lock is $L_j$, then after the declaration related to $t$ add the declaration Check Lock $L_j$.
   (4) deal with each declaration in $T$, if all the declarations corresponding to a local transition are done, then output the transition to the test sequence.
```

- The Second Class synchronization problem

Set t_x, t_y are transitions in M_i and M_j $(i\neq j)$, if transition sequence t_x ... t_y satisfy:

(1) *Head*(t_x) $\in$*Con*(t_y),

(2) if any transition t_z $(z\neq x, y)$ is not in M_j, then the transition sequence is called the Second Class synchronous input transition sequence.

We call the test sequence that includes the Second Class synchronous input transition sequence existing Second Class synchronization problem. The test sequence R with Second Class synchronization problem may lead to test sequence R' be executed which is different from R.

Input: local transition sequence set *E (E₁, E₂, …Eₙ)* of RT-EFSM model, and E_i is a continuous transition sequence;
Output: test sequence *R*;
(1) as the same as the steps (1), (2), (3), (4);
(2) if there is the Second Class synchronous input transition sequence t_x … t_y in the test sequence generated above, then mark the transition t_y, thus the input of this transition will not be generated when generating corresponding input sequence.

III. TEST SEQUENCE GENERATION

A. *Test Sequence Generation Method*

With the validation of the model, we got a minimal, completed and strong connected RT-EFSM model. Figure 3 is a schematic diagram of an aircraft inertial/satellite navigation systems to transfer from alignment state to navigation state. After simplifying the RT-EFSM model shown in figure 4 is obtained. And S_0 is alignment state, S_1 is input course alignment state, S_2 is memory course alignment state, S_3 is GC alignment state, S_4 is navigation state, t_1 and t_2 is time interval (0, 120ms], t_3 and t_4 are fix time, 10ms and 50ms respectively. This RT-EFSM model state transition is shown in Table 2. We will take it for example to explain how to automatically generate test sequences.

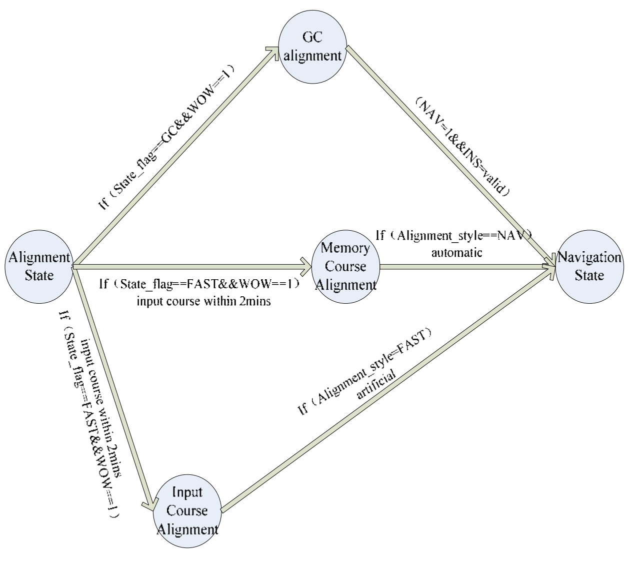
GC alignment
Alignment State
Memory Course Alignment
Navigation State
Input Course Alignment
If（State_flag==GC&&WOW==1）
（NAV=1&&INS=valid）
If（Alignment_style==NAV）automatic
If（State_flag==FAST&&WOW==1）input course within 2mins
If（State_flag==FAST&&WOW==1）input course within 2mins
If（Alignment_style=FAST）artificial

Figure 3. A schematic diagram of an aircraft inertial / satellite navigation systems

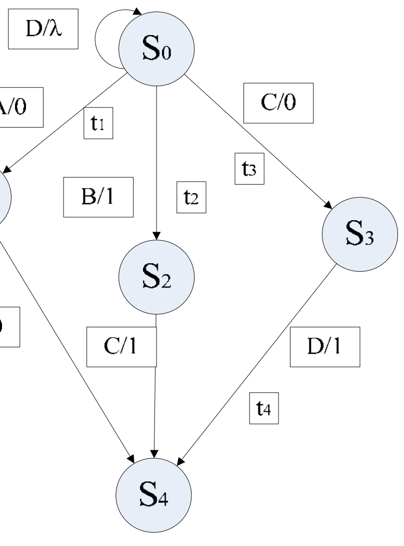

Figure 4. RT-EFSM model for the aircraft inertial / satellite navigation systems

TABLE II THE STATE TRANSITIONS OF FIGURE 4

	A(t) B(t) C(t) D(t)	A B C D
Start_state	Output	End_state
S_0	0(t_1) 1(t_2) 0(t_3) λ	S_1 S_2 S_3 S_0
S_1	/ 0 / /	/ S_4 / /
S_2	/ / 1 /	/ / S_4 /
S_3	/ / / 1(t_4)	/ / / S_4
S_4	/ / / /	/ / / /

Firstly, we add start state and end state to the model and we get figure 5.

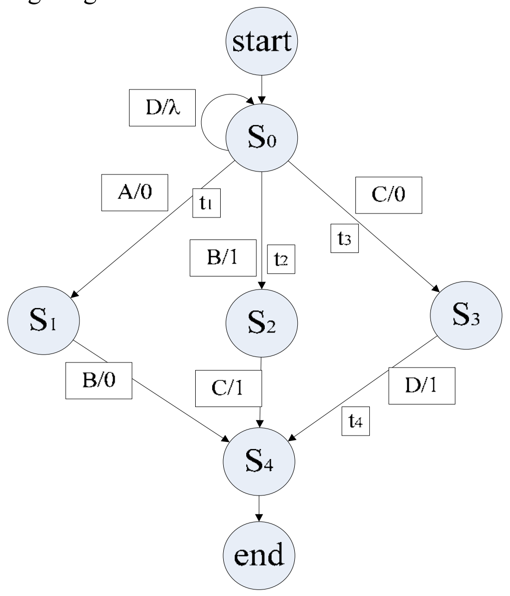

Figure 5. RT-EFSM model after adding start state and end state

Then we adopt depth-first search algorithm to traversal figure5, forming Depth-First Search Tree (DST). Adopting depth-first search algorithm we can not only traversal the whole model, but also get the complete transition of each branch.

Considering the real time characteristic of ASCS, in this paper we propose a time extended UIO sequence t_UIO:

t_UIO=[t]_?I / !O, and the '[ ]' refers to optional, '?' refers to input and '!' refers to output.

According to the search algorithm above and *t_UIO* expression, we generate the *t_UIO* sequences as it is shown in Table 3.

TABLE III T_UIO SEQUENCE FOR FIGURE 4

State	t_UIO
S_0	t_1_?D / !λ
S_1	_?B / !0
S_2	_?C / !1
S_3	t_4_?D / !1
S_4	Φ
S_0	t_1_?D / !λ
S_1	_?B / !0
S_2	_?C / !1
S_3	t_4_?D / !1
S_4	Φ

For each transition between states we generate sub test sequence following the steps below:

- Reset RT-EFSM (input *r*) to get back to start state.
- Find out the shortest path from S_0 to S_i;
- Input the symbol that makes the state transfer from S_i to S_j;
- Input the *t_UIO* sequence of S_j.

Following the 4 steps above we can get each transition sequence:

r D D…………………S_0->S_0

r A(t_1) *B*………………S_0->S_1

r B(t_2) *C*………………S_0->S_2

r C(t_3) *D*(t_4)……………S_0->S_3

r A(t_1) *B*………………S_1->S_4

r B(t_2) *C*………………S_2->S_4

r C(t_3) *D*(t_4)……………S_3->S_4

In order to achieve a minimum cost to complete all of the state traversal, merging the sub test sequence above, optimization, that is, to remove repetitive sequences to available integrated test sequence:

$$r\,D\,D\,r\,A(t_1)\,B\,r\,B(t_2)\,C\,r\,C(t_3)\,D(t_4)$$

Based on the sub test sequences and integrated sequence above, combining with testing strategy, instancing corresponding state variables, test cases are generated.

B. Test Coverage Adequacy Criteria

In EFSM-based testing method, the coverage criteria that are commonly used and more mature nature is state coverage, transition edge coverage and full predicate coverage [16].

In this paper, considering the real time characteristic of ASCS, we propose time condition coverage sufficiency criteria besides the criteria mentioned above to analyze the generated test sequences:

$$\exists S_i \in S^*, \exists S_j \in S^*, \forall t \in T\{S_i \to S_j\}, \text{ and}$$

$$t = \{t \mid t = t_f \parallel t \le t_F \parallel t \in t_I\}$$

According to the criteria, all the boolean conditions related to time of transitions of RT-EFSM model are valued true or false once. There are test sequences for both satisfying the time constrains and not satisfying the time constrains in order to ensure they satisfy sufficiency requirement.

IV. CONCLUSIONS

In this paper, grounding on abundant experience and research on real-time embedded software testing technique, we introduced formal method into the ASCS verification field and proposed a real-time extension of EFSM model, named RT-EFSM, and gives its validation methods. At the same time, we put forward a timed unique input/output sequence (t_UIO) and its automatic generation algorithm. Finally, we applied the method to the verification of an aircraft inertia/satellite navigation system, and the results proved the correctness and validity of the method.

The future work is to improve the RT-EFSM model extension scheme, research on how to generate test cases automatically to make this theory and method guide project practice better.

ACKNOWLEDGMENT

We owe our thanks to the anonymous reviewers and Department of System Engineering of Engineering Technology of Beihang University.

REFERENCES

[1] Qin Zhidong , Lei Hang , Sang Nan, Xiong Guangze , Gu Youpeng. Study on the Reliability Demonstration Testing Method for Safety-critical Software[J]. Chinese Journal of Aeronautics.2005, Vol.26 No.3,pp.334-339.

[2] Lin Bin, Gao Xiaopeng, Lu Mingyan, Ruan Lian, Study on Reliability Simulation Soft-ware System for Embedded Software[J], The Journal of Beijing University of Aero-nautics and Astronautaics, 2000, Vol.26,No.4, pp.490-493.

[3] NASA software safety guidebook[C], NASA-GB-8719.13, NASA, 2004.

[4] A.Guerrouat, H.Richter. A formal approach for analysis and testing of reliable embedded systems[J]. Electronic Notes in Theoretical Computer Science, 2005, Vol.141, No.3, pp.91-106.

[5] Yao Xinhua, Pan Xuezeng, Fu Jianzhong, Chen Zichen. Research on real-time FSM modeling for microfabrication nc system[J]. Journal of zhejiang university (engineering science). 2005, vol.39, no.12, pp.1965-1968.

[6] Wu Fenlan, Hu Chaoju, SongYu.Study on software test method based on finite state machine[J]. Journal of North China Electric Power University.2005, Vol.32,No.6, pp.60-63.

[7] ZhangYong, Qian Leqiu Wang Yuanfeng.Automatic Testing Data Generation in the Testing Based on EFSM[J].Chinese Journal of computers. 2003, Vol.26, No.1O.pp.1295-1303.

[8] Mercedes G. Merayo, Manuel Nu′ n˜ez, Ismael Rodrı′ Guez. Formal testing from timed finite state machines[J].2008, Computer networks, 52(2008), pp.432–460.
[9] Li Shuhao, Wang Ji, Dong Wei, Qi Zhichang. A framework of property-oriented testing of reactive systems[J]. CHINESE JOURNAL OF ELECTRONICS. 2004, Vol.32, No.12A, pp.222-225.
[10] A. V. Aho, A. T. Dahbura, D. Lee, and M.U. Uyar. An optimisation technique for protocol conformance test generation based on UIO sequences and Rural Chinese Postman Tours[J]. In S. Aggarwal and K. Sabnani, editors, Protocol Specification, Testing, and Verification, New Jersey, 1988.
[11] S. Fujiwara, G.v. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi. Test selection based on finite state models[J]. IEEE transaction on Software Engineering. 1991, Vol.17, No.6, pp.591-603.
[12] M. Krichen, S. Tripakis, An expressive and implementable formal framework for testing real-time systems[C], in:17th International Conference on Testing of Communicating Systems, TestCom'05, LNCS 3502, Springer, 2005, pp.209–225.
[13] A. Khoumsi, A method for testing the conformance of realtime systems, in:7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems[J], FTRTFT'02, LNCS 2469, Springer, 2002, pp. 331–354.
[14] Franco,Fummi Cristina Marconcini Graziano Pravadelli. Functional Verification based on the EFSM Model[C]. Ninth IEEE International High-Level Design Validation and Test Workshop, HLDVT'04. 2004, pp.69-74.
[15] RTCA/DO-178B, Software considerations in airborne systems and equipment certification, Requirements and Technical Concepts for Aviation[M], Washington, D.C., December 1992. EUROCAE ED12B in Europe.
[16] Zhan Xuede. Research on software testing method based on UML Startcharts[M]. PH.D dissertation. Shanghai University.2005.7.

Yongfeng Yin was born in 1978. Now he is a lecturer and Ph. D. candidate. He received the B. S. and M.S. degrees from Beihang University, in 2000 and 2003. His research interests include system engineering, real-time embedded software testing and software reliability. E-mail: yyf@buaa.edu.cn

Bin Liu is a Ph. D. advisor and professor. His research interests include system engineering, software testing and software reliability. E-mail: liubin@buaa.edu.cn

Duo Su is a Ph. D. His research interests include System security, airworthiness. E-mail: suduo@263.net