面向DO-333的襟缝翼控制单元安全性分析

陈光颖 黄志球 陈 哲 阚双龙
(南京航空航天大学计算机科学与技术学院 南京 210016)

摘 要 DO-333是对机载软件安全性标准DO-178C关于形式化方法的补充,为机载软件开发过程中形式化方法的使用提供指导。模型检验作为一种形式化方法,可以应用于对软件需求和设计阶段制品的严格验证。基于DO-333,使用模型检验对飞控系统中襟缝翼控制单元不同阶段的软件制品进行验证与分析,判断其是否满足DO-178C的相关验证目标并提供证据支持。首先,对控制单元中襟翼与缝翼必须互斥更新的高级需求进行规约和验证;其次,对单个机翼控制逻辑的低级需求进行规约和验证。通过以上验证与分析,分别为标准中关于高级和低级需求的验证目标提供证据。文中展示了模型检验在一个机载软件认证中的应用实例,该工作将为机载软件的安全性保障和适航认证提供技术支持。

关键词 适航认证,形式化方法,模型检验,机载软件,SPIN
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2016.5.028

Safety Analysis of Slat and Flap Control Unit for DO-333

CHEN Guang-ying HUANG Zhi-qiu CHEN Zhe KAN Shuang-long
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract DO-333 is the formal supplement to the aircraft software's safety standard DO-178C, which provides guidance for the use of formal methods in the development process of aircraft software. Model checking, as a kind of formal methods, can be applied for the strict verification of the software artifacts in the requirement and design phase. Based on DO-333, this paper used model checking to formally verify the software artifacts of different development phases for a slats and flaps control unit, one component of the flight control system, aiming to justify whether the software artifact satisfies the related objectives or not in DO-178C and provide the evidence support. Firstly, model checking is applied to the specification and verification of the high_level requirements for the mutual updating operation between flap and slat. Then, it is applied to the low_level requirements for the control logic in a single wing. Based on the verification and analysis above, evidence is provided for the verification objectives about high_level and low_level requirements in DO-178C. This paper illustrated an application example of model checking in an aircraft software's certification and could provide technical support for the safety assurance and airworthiness certification of aircraft software.

Keywords Airworthiness certification, Formal methods, Model checking, Aircraft software, SPIN

1 引言

当前嵌入式软件在医疗、交通、航空航天等领域应用广泛,软件一旦失效可能会造成环境破坏、财产损失甚至人员伤亡,因此保障软件的安全性至关重要。机载软件作为一类典型的安全攸关(Safety-Critical)嵌入式软件,在投入使用前必须经过严格的安全认证。美国航空无线电委员会(The Radio Technical Commission for Aeronautics, RTCA)提出的航空适航认证标准DO-178C[1]对机载软件的安全性提出了严格要求,规定了软件开发过程中各阶段软件制品所要达到的安全目标[2]。有效验证软件制品是否满足适航认证标准中规定的目标是机载软件开发过程中的一个重要问题。

与传统的仿真、测试等验证技术相比,基于严格数学理论的形式化验证技术,在软件开发的早期就能对软件制品自动、有效地进行全路径覆盖的穷举式检查,且易于发现并发等深层次错误,极大地提高了验证的覆盖率和可信度[3]。DO-178C关于形式化方法的增补文档DO-333[4]为机载软件开发过程中形式化方法的使用提供了指导。

目前工业界和学术界已经使用形式化方法对机载软件的安全性认证进行了实践。例如,文献[2]基于系统建模语言SysML采用形式化方法验证系统静态结构模型中的安全性依赖关系与适航认证标准中所规定的目标是否一致。法国空

到稿日期:2015-04-28 返修日期:2015-08-07 本文受国家自然科学基金(61100034,61170043),中国博士后科学基金(20110491411),江苏省普通高校研究生科研创新计划资助项目,中央高校基本科研业务费专项资金(CXZZ11_0218)资助。
陈光颖(1991-),女,硕究生,主要研究方向为软件工程、形式化验证,E-mail:sx1316006@nuaa.edu.cn;黄志球(1965-),男,教授,博士生导师,CCF杰出会员,主要研究方向为软件工程;陈 哲(1981-),男,博士,副教授,主要研究方向为软件工程、形式化方法;阚双龙(1988-),男,博士生,主要研究方向为软件工程、形式化验证。

客公司将形式化方法应用于机载软件的开发，并指出形式化方法可作为传统测试的部分替代，用于 DO-178C 编码过程的目标认证[5-7]。法国 Esrerel 公司开发的 SCADE[8] 工具提供了支持适航标准 DO-178C 的认证方法和技术，在空中客车 Airbus A340 等安全关键系统中得到了部分应用[9]。文献[10]指出形式化验证技术与测试结合后可以有效应用于系统的确认和认证。文献[11]基于 DO-333，使用定理证明、模型检验与抽象解释 3 类主要的形式化验证技术分别对一个飞机导航系统开发过程不同阶段的软件制品进行验证，并为 DO-178C 提供认证证据。

模型检验与定理证明相比，自动化程度更高且不需要使用者掌握深厚的逻辑知识，与抽象解释相比，可应用于需求和设计过程的软件制品，支持对功能属性和非功能属性的验证。因此，本文基于 DO-333，使用模型检验对一个襟缝翼控制单元开发过程需求和设计阶段的软件制品进行验证与分析，检验其是否满足标准规定的目标并提供证据支持，展示模型检验在一个实际的机载软件认证中的应用。

襟缝翼控制单元(Slats and Flaps Control Unit，SFCU)是飞控系统中控制襟翼和缝翼行为的重要组件。飞行员通过驾驶舱操作杆设置襟翼和缝翼的目标状态，经由闭环反馈控制器控制襟翼和缝翼的运动，在起飞和降落过程中增加升力、提高失速临角。该单元直接影响着飞行的安全及稳定，保障其安全性至关重要。

本团队已使用模型检验对控制单元编码过程的部分输出——缓冲区操作算法的源代码进行了验证[12]，但未结合相关标准提供验证证据。本文使用模型检验，针对 DO-178C 中的相关验证目标，分别对该控制单元需求过程的部分输出——襟翼与缝翼必须互斥更新的高级需求，以及设计过程的部分输出——单个机翼控制逻辑的低级需求进行验证与分析，并提供验证证据。此外，对已有工作[12]能提供的证据进行了简要补充。上述高级需求和低级需求均使用状态图进行描述，使用模型检验工具 SPIN 进行验证。

本文第 2 节介绍了 DO-333 及相关理论技术工具；第 3 节给出了基于模型检验的襟缝翼控制单元验证框架；第 4 节针对 DO-178C 高级需求的验证目标，使用 SPIN 对襟缝翼必须互斥更新的高级需求进行验证与分析；第 5 节针对 DO-178C 低级需求的验证目标，使用 SPIN 对单个机翼控制逻辑的低级需求进行验证与分析，通过以上验证分析提供相应的验证证据；最后总结全文。

2 背景知识

2.1 DO-333

DO-333 是对 DO-178C 关于形式化方法的补充，针对 DO-178C 中的具体验证目标，指出了需要形式化描述的软件制品以及从中能获得的验证证据。图 1 描述了 DO-178C 定义的 A 级软件(即软件失效会对系统安全造成灾难性影响的软件)需求、设计和编码过程涉及的生命周期数据项、开发活动和验证活动的关系。

系统需求指与系统的功能、行为、性能以及安全性等相关的需求；高级需求指从系统需求、安全相关的需求及系统设计结构开发出的，与软件的功能、行为及安全等相关的需求；低级需求指从高级需求等开发出的软件需求，该需求可由源代码直接实现[1]。其中，高级需求经过软件需求过程开发，低级需求经过软件设计过程开发。

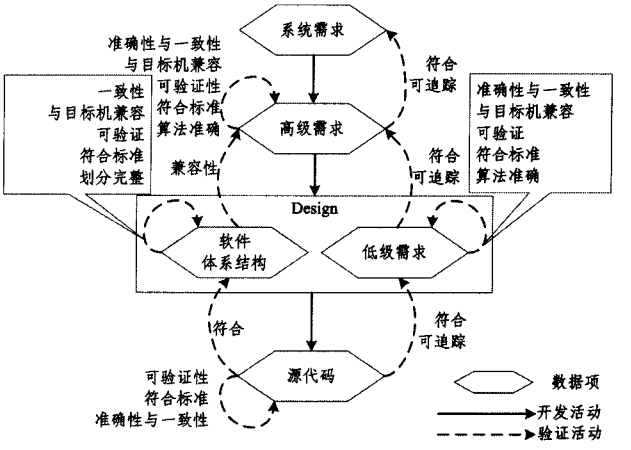


图 1 A 级软件验证过程(取自 DO-333)

验证活动检验并报告软件开发过程中引入的错误，图 1 列举了各验证活动的验证目标，包括准确性、一致性、符合性、可追踪性等。其中，一致性指数据项(需求或源代码)之间不存在逻辑冲突；符合性指数据项间的满足关系，例如，高级需求到系统需求的符合性指高级需求满足，即实现了系统需求；可追踪性指数据项间的关联，通常指双向的追踪关系，例如，高级需求和低级需求间的可追踪性，一方面指所有的高级需求都被开发出更详细的低级需求，另一方面指开发出的低级需求都能对应到某个高级需求，即该低级需求的必要性[4]。

基于对软件生命周期数据项精确无二义的形式化描述，形式化方法能够证明(或与评审等方法相结合来证明)数据项是否满足上述验证目标[4]。如文献[12]对襟缝翼控制单元缓冲区操作算法源代码的验证结果可说明该源代码满足准确、一致与可验证性，修复后的源代码符合缓冲区操作算法需求等验证目标。

2.2 SPIN

SPIN[13,14] 是一种经典的适用于并行系统的模型检验工具，通过自动穷尽搜索待验证模型的有限状态空间来检验性质是否满足，并能提供精确的反例执行路径。Promela 作为 SPIN 的输入语言，拥有类 C 语言的语法结构，允许动态创建并行进程。SPIN 使用断言、never claim 和线性时序逻辑(Linear Temporal Logic，LTL)公式来描述性质。LTL 在命题逻辑中引入线性时序连接词来表示性质随时间的变化。例如，时序算子<>(eventually)表示时序状态序列上未来某个时刻，[](always)表示序列上所有时刻，公式 pUq 在一条路径上成立表明 p 连续成立直到 q 成立且 q 在某时刻必须成立。

2.3 状态图

状态图(Statecharts)[15] 在传统状态迁移图的基础上引入了层次嵌套、并发、广播机制等概念，用于可视化地描述系统对象在其生命周期中所有可能的状态迁移。状态图包括基本、或(Exclusive or)和与(And)状态，支持条件、选择算子等或(OR)连接符及分叉、汇合等与(AND)连接符，允许事件延迟与超时，可描述并发组件。越来越多的研究将状态图与模型检验技术结合起来，用于系统安全性的分析与验证[16-18]。

3 基于模型检验的襟缝翼控制单元验证框架

本节将简单介绍基于模型检验的襟缝翼控制单元验证框架。

基于 DO-333，图 2 给出了本文基于模型检验的襟缝翼控制单元验证框架。该框架针对图 1 需求和设计过程的验证目标，对控制单元需求和设计过程输出的软件制品进行验证，并为相关的验证目标提供证据支持。

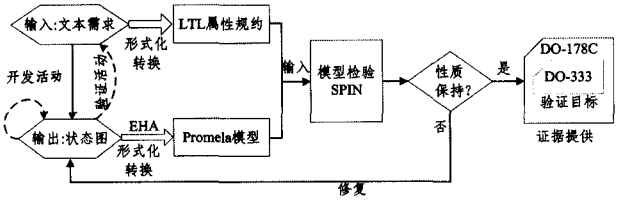


图 2　基于模型检验的襟缝翼控制单元验证框架

需求过程的输入数据项为系统需求，输出数据项为根据系统需求等开发的高级需求。设计过程的输入数据项为高级需求，输出数据项为能够满足高级需求的详细低级需求。本文需求和设计过程的输入数据项均为自然语言描述的文本需求，输出数据项则用状态图进行描述。

以对高级需求的验证为例：①首先将由状态图描述的高级需求转换为 Promela 模型，自然语言描述的系统需求转换为 LTL 属性规约；②随后，将以上形式化描述输入模型检验工具 SPIN 进行验证，若发现错误，分析错误路径对状态图进行修复，直到性质满足；③最后，分析验证过程及结果，为 DO-178C 关于高级需求（软件需求过程的输出）的验证目标提供证据。对低级需求的验证过程与高级需求类似。

本文状态图到 Promela 的转换基于文献[17]。文献[17]使用层次自动机（Extended Hierarchical Automata，EHA）作为中间语言，提出了状态图子集到 Promela 模型的自动转换规则，涵盖了对并行、状态精化等关键概念的处理，并证明了转换的正确性。但该文献并没有涉及对条件算子连接符的处理，本文 5.1 节中第（3）点对条件算子的处理进行了补充说明。

4　对襟缝翼控制单元高级需求的验证分析

互斥是复杂并行系统中的一个安全关键问题。襟缝翼控制单元中襟翼和缝翼属于两个相互独立且控制逻辑结构一致的控制模块，二者需满足“襟翼与缝翼不能同时进行运动状态的更新（互斥更新）”的系统安全性需求。

本节使用 SPIN 对从系统安全性需求“襟翼与缝翼互斥更新”开发出的高级需求进行验证，检验该高级需求是否满足 DO-178C 关于高级需求的验证目标：①将高级需求的状态图描述转换为 Promela 模型，将襟缝翼互斥更新这一系统需求转换为 LTL 规约；②将转换结果输入 SPIN 进行验证，检验需求过程中是否引入错误，判断高级需求是否符合系统需求；③分析验证过程和结果，为相应验证目标提供证据。

4.1　高级需求的形式化规约

图 3 描述了从“襟翼与缝翼互斥更新”这一系统需求开发出的高级需求，图中襟翼与缝翼被当作一个状态图中的两个并行状态 Flap 和 Slat。

缝翼 Slat 初始时维持当前运动状态 S_cur，若发生事件 Supdate，即缝翼有更新请求，且襟翼 Flap 不处于更新状态 F_update，则产生动作 en_Supdate，进入更新状态 S_update。

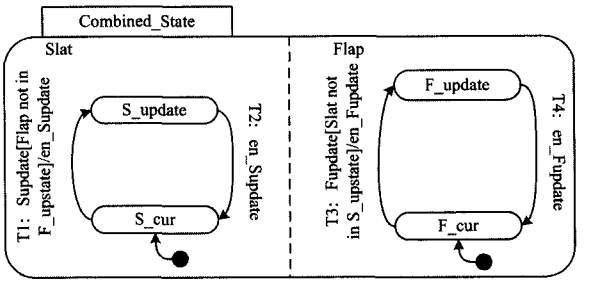


图 3　襟缝翼互斥操作的高级需求描述

状态图到 Promela 的形式化转换分为两步：状态图到层次自动机的转换；层次自动机到 Promela 的转换。

1. 状态图到层次自动机的转换

层次自动机是对状态图的抽象，由通过精化函数相关联的顺序自动机组成。精化函数将状态映射为（并行）自动机的集合。首先，顺序自动机的定义如下：

定义 1（顺序自动机）　顺序自动机 A 是一个四元组 $(\sigma_A, s_A^0, \lambda_A, \delta_A)$，$\sigma_A$ 是 A 中的有限状态集合，s_A^0 为 A 的初始状态，λ_A 为 A 中的迁移标签集合，$\delta_A \subseteq \sigma_A \times \lambda_A \times \sigma_A$ 定义了状态间的迁移关系。

λ_A 中的迁移标签表示为迁移五元组。

定义 2（迁移）　迁移 t 可表示为 (sr, ev, g, ac, tgt)，其中 sr 和 tgt 分别表示迁移的源状态和目标状态，ev 表示迁移的触发事件集合，g 表示守卫条件，ac 表示迁移的动作列表。由于可能存在状态嵌套，迁移的源状态和目标状态可以是状态的集合。

例如，状态 *Slat* 中的迁移 T1、T2 可表示为：T1(*S_cur*, *Supdate*, *Flap not in F_update*, *en_Supdate*, *S_update*) 及 T2(*S_update*, *en_Supdate*, ∅, ∅, *S_cur*)。

层次自动机由顺序自动机组成。

定义 3（层次自动机）　层次自动机是一个五元组 (F, E, Λ, f, A_0)。F 是顺序自动机的有限集合，$\forall A_1, A_2 \in F, \sigma_{A_1} \cap \sigma_{A_2} = \emptyset$；$E$ 为有限的事件集合；$\Lambda = \bigcup_{A \in F} \lambda_A$ 为层次自动机中的标签集合；精化函数 $f: \bigcup_{A \in F} \sigma_A \to 2^F$ 定义了状态到 F 幂集的树型结构映射；A_0 是唯一的根自动机。若 $|f(s)| = 1$，状态 s 被精化为一个自动机；若 $|f(s)| > 1$，状态 s 可被精化为并行的自动机组合；否则 $f(s) = \emptyset$，s 是一个基本状态。

如图 4 所示，状态 Combine_State 由两个并行状态 Slat 和 Flap 组成，精化函数将其映射为两个并行的自动机 A_1、A_2。A_1、A_2 中的状态均为基本状态，其中的迁移与图 3 相对应。

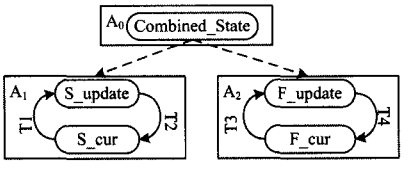


图 4　高级需求状态图的层次自动机

根自动机 A_0 有且仅有一个状态 Combine_State，无迁移。顺序自动机 A_1 的状态集合 $\sigma_{A_1} = \{S\_cur, S\_update\}$，初始状态 $s_{A_1}^0 = \{S\_cur\}$，$\lambda_{A_1} = \{T1, T2\}$。层次自动机中的事件集合 $E = \{Supdate, en\_Supdate, Fupdate, en\_Fupdate\}$。

2. 层次自动机到 Promela 的转换

（1）模拟事件、环境和状态

将事件定义为 int 型常量集合；每个事件都对应一个 bit 型的环境变量，表示该事件是否待处理；状态定义为 bit 型变量，表示当前是否处于该状态。

例如对图 4 层次自动机的变量定义：

```
1. #define Supdate 1/*定义事件*/
   …
2. #define en_Fupdate 4
3. bit S_update,S_cur; /*定义 A1 中的状态变量*/
   …
4. bit Q_S,Q_ens,Q_F,Q_enf;/*定义事件环境变量*/
5. int Ev;/*存储当前事件*/
```

其中，将事件 *Supdate* 定义为常量 1；将 A_1 的状态定义为 bit 型变量；给出各事件相应的环境变量定义；使用 int 型变量 *Ev* 存储当前被选择的事件。

(2)模拟迁移步骤

使用进程 STEP 模拟迁移步骤：首先选择事件，随后执行满足迁移条件的迁移，进程体为一个 do 循环。事件选择函数定义如下：

```
/*事件选择函数*/
6. inline event_select()
7. { if ::Q_S==1->Ev=Supdate;Q_S=0;
   …
8.     ::Q_enf==1->Ev=en_Fupdate;Q_enf=0;
       ::else->skip;
9. fi;}
```

函数 *event_select*()从值为 1 的环境变量中选择事件，随后选择并执行层次自动机中的迁移。迁移 t 的可执行条件为表达式(sr&&ev&&g)为真，即处于源状态、事件发生且守卫条件满足。

```
10. proctype STEP()
11. {do ::atomic{event_select();/*选择事件*/
  /*模拟 A1 中迁移*/
12. if ::(S_cur==1)&&(Ev==Supdate)&&(F_update==0)->
    S_cur=0;S_update=1;Q_ens=1;/*模拟 T1*/
13. ::(S_update==1)&&(Ev==en_Supdate)->
  S_update=0;S_cur=1; /*模拟 T2*/
14. ::else->skip;/*A1 中无可执行迁移*/
15. fi;
/*模拟 A2 中迁移*/
…}
16. od;}
```

自动机 A_0 中无迁移，A_1、A_2 相并行，因此使用 if 语句分别模拟 A_1 和 A_2 中的迁移。例如 12—15 行对顺序自动机 A_1 中迁移的模拟，其中对于迁移 $T1$，若表达式(S_cur==1&&Ev==*Supdate*&&F_update==0)为真，迁移可执行，则产生迁移动作 *en_Supdate*，即将其对应的环境变量 Q_ens 置 1，此时缝翼进入更新状态，离开当前状态，即 S_update 置 1，S_cur 置 0。

函数 *init* 初始化状态、环境变量，运行进程。如将 A_1、A_2 中初始的状态变量置 1。

```
init
{S_cur=1;F_cur=1;…/*初始化*/
run STEP();}
```

若将事件集定义为先进先出的队列，则用信道 chan 进行模拟，顺序的 Promela 模型可相应转换为并行结构[17]。

4.2 验证与分析

系统安全性需求“襟翼与缝翼不能同时进行运动状态的更新”的 LTL 规约为公式 P：

$$P\{[]\ !((S\_update==1)\&\&(F\_update==1))\}$$

使用逻辑公式!((*S_update*==1)&&(*F_update*==1))表示襟翼与缝翼不会同时处于更新状态，S_update、F_update 为已定义的状态变量。时序算子[]表示状态序列上的所有时刻该逻辑公式都为真。

使用 SPIN 验证高级需求(见图 3)的 Promela 模型未发现错误，添加系统需求的 LTL 属性后继续验证，未发现错误，表明需求过程输出的高级需求满足襟缝翼互斥更新的系统需求。

针对 DO-178C 中高级需求的验证目标，表 1 总结了本节控制单元高级需求的满足情况，并提供了具体的证据说明。

此外，DO-333 对形式化方法的可靠性、需求规约的正确性、分析过程及结果的正确性等提出了要求。本节高级需求到 Promela 模型、系统需求到 LTL 属性形式化转换的正确性、分析过程及结果的正确性等通过评审确认。

表 1　SFCU 高级需求验证目标满足情况

目标	满足程度	证据说明
A-3.1 软件高级需求符合系统需求	■	SPIN 验证 LTL 属性未发现错误，表明 SFCU 的高级需求(见图 3)的 Promela 模型满足系统需求；襟缝翼互斥更新的 LTL 属性说明了该高级需求符合系统需求。
A-3.2 高级需求是准确的和一致的	■	使用状态图描述高级需求并将其转换成形式化的 Promela 模型说明了高级需求的准确性；转换后的 Promela 模型在 SPIN 中验证无错，即不存在逻辑冲突，证明了高级需求的一致性。
A-3.3 高级需求与目标机兼容		本节对高级需求的验证未涉及目标机，无法说明。
A-3.4 高级需求是可验证的	■	高级需求的状态图描述转换为 SPIN 可验证的 Promela 模型表明了高级需求的可验证性。
A-3.5 高级需求符合标准	□	本节验证工作仅能说明图 3 高级需求状态图的 Promela 模型符合 Promela 的语法规范。
A-3.6 高级需求可追踪到系统需求	□	高级需求的 Promela 模型满足系统互斥需求的 LTL 属性，说明该系统需求被开发成更详细的高级需求；本节验证工作未能说明高级需求可追踪到系统需求，即高级需求的必要性，这通过评审说明。
A-3.7 算法准确	■	使用状态图建模互斥更新的高级需求，并通过模型检验证明了高级需求实现了襟缝翼互斥更新的系统需求，说明算法设计的准确性。

注：■：满足　□：部分满足

5　对襟缝翼控制单元低级需求的验证分析

襟翼与缝翼的控制逻辑结构一致，包括翼面位置控制模块、电机转速控制模块、测试维护模块等。本节关注与控制逻辑相关的高级安全性需求，如机翼的展开限制、故障的有效性检测及处理等。

本节使用 SPIN 对从襟/缝翼控制逻辑相关的高级需求开发出的低级需求进行验证，检验该低级需求是否满足 DO-178C 关于低级需求的验证目标：①将实现控制逻辑的低级需求状态图转换为 Promela 模型，将与机翼控制逻辑相关的高级安全性需求规约为 LTL 属性；②将转换结果输入 SPIN 进行验证，检验设计过程中是否引入错误，判断低级需求是否符

合高级安全性需求；③分析验证过程和结果，为相应验证目标提供证据。

5.1 低级需求的形式化规约

图 5 描述了从襟/缝控制逻辑相关的高级需求开发出的低级需求，图中包含多个并行状态。其中：

a)翼面控制模块通过手动开关(Mulset)自动(F_Auto)或手动(F_Manual)实现机翼的运动；根据期望运动指令(F_Mcmd/F_Acmd)设置转速(Mspd)；设置的转速控制风扇开关(FanSwitch)；机翼状态(Mstate)根据期望指令进行更新。

b)电机转速控制模块根据手动/自动状态下的期望指令、转速设置、禁用状态(InhibitTest)及主控开关(MDTest)判断电机的驱动电压输出(Mdrv)和制动指令(Mbrk)。

c)测试维护模块包括传感器检测、主控判断和禁用判断。传感器检测通过传感器实现风扇转速(FanSwitch. on)、处理器温度(Cpu_Temp)、电机温度(F_MTemp)的故障判断及处理；主控判断实现对转速控制器(SpdCtrlTest)的故障检测，设置主控开关(MDTest)的工作模式；禁用判断基于风扇转速、处理器温度、翼面偏差(Posioverbias)及电机驱动输出检测(Mdb_Test)的状态判断禁用是否有效。

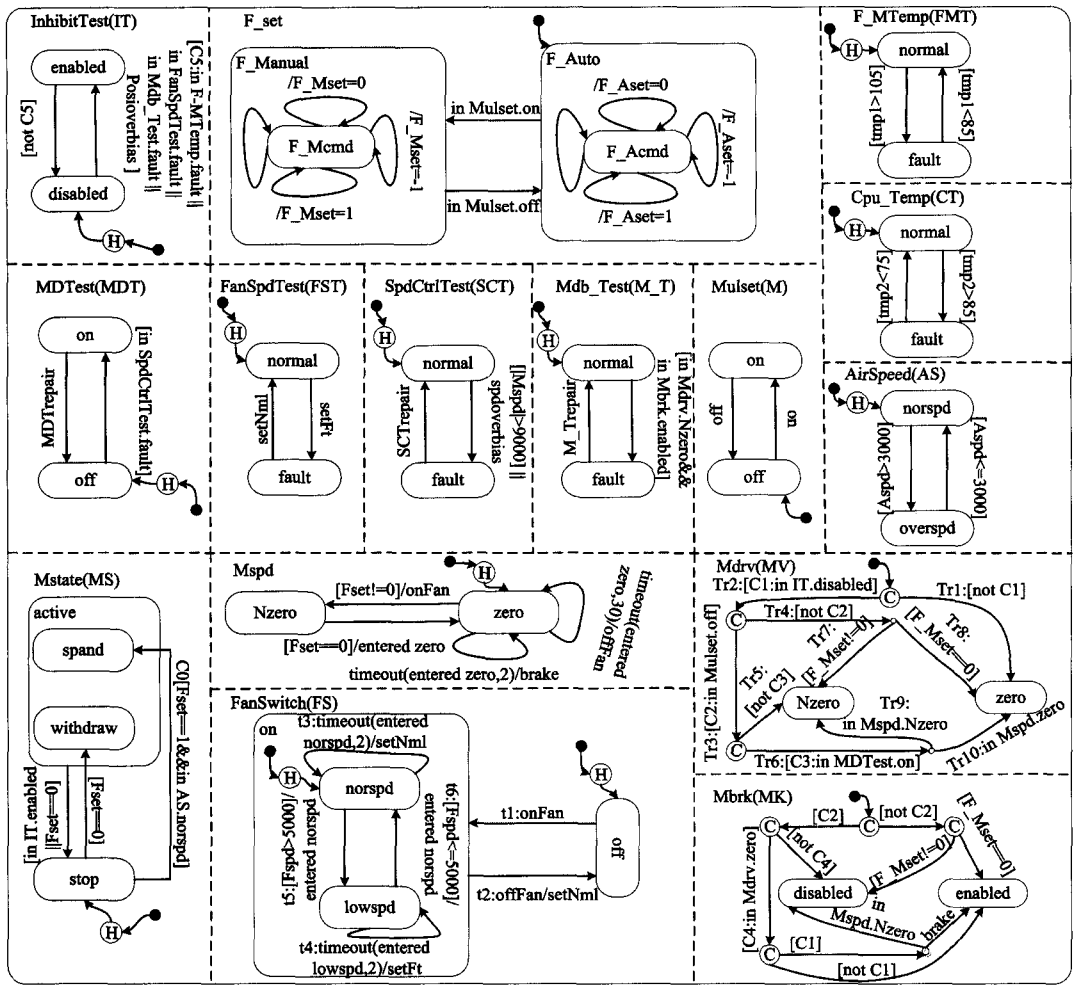


图 5 襟/缝翼控制逻辑的低级需求描述

(1)根据 4.1 节定义 3，图 5 中的各并行状态可映射为多个并行自动机。图 4 中自动机 A_1 和 A_2 中的状态均为基本状态，无法精化，A_1、A_2 相并行。而图 5 FanSwitch 中的状态 on、Mstate 中的状态 active 含有子状态，可继续精化。例如，FanSwitch 中状态 on 包含转速正常 norspd 和过低 lowspd 两个异或的基本状态，继续将 on 映射为一个顺序自动机。

图 6 描述了状态 FanSwitch 的层次自动机，图中各迁移与图 5 相对应，A_1 为 A_0 的子自动机。

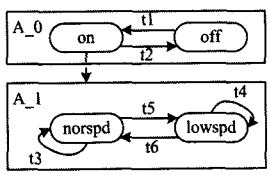


图 6 状态 FanSwitch 的层次自动机

图 6 对应的 Promela 代码如下：

```
/*模拟 A_0 迁移*/
1. if ::select(t1)->fire(t1); /*t1*/
2.    ::select(t2)->fire(t2);/*t2*/
3.    ::!select(t2)&&(select(t3) || select(t4) ||
      select(t5) || select(t6))-> /*模拟 A_1 迁移*/
4.      if ::select(t3)->fire(t3);/*t3*/
            …/*t4,t5,t6*/
5.      ::else->skip;
6.      fi;
7. ::else->skip
8. fi;
```

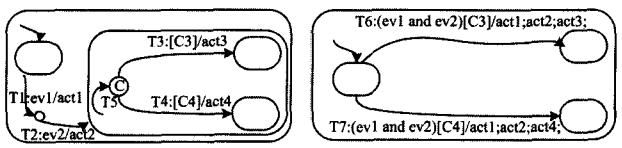
使用嵌套的 if 模拟 A_0 及 A_1 中的迁移。其中，*select*(*ti*)表示迁移 *ti* 的可执行条件：$sr(ti)\&\&ev(ti)\&\&g(ti)$，*fire*(*ti*)表示产生迁移动作 $ac(ti)$，状态转换为 $tgt(ti)$。先模

拟A_0中迁移。由于A_1是对A_0中状态on的精化，A_1中迁移的源状态包含状态on，例如，此时$t5$的源状态为{*norspd*、*on*}。

(2)传感器检测模块根据传感器反馈的外界环境变量取值判断故障发生与否，本文使用非确定的条件赋值语句来随机模拟其所有可能的取值情况。例如，对空速状态AirSpeed：

```
9. if /*随机赋值*/
10.  ::Aspd=1;/*模拟条件[Aspd>3000]为真*/
11.  ::Aspd=0;/*模拟条件[Aspd<=3000]为真*/
12.  fi;
```

(3)图5中驱动电压输出状态Mdrv和制动指令状态Mbrk均含有条件算子连接符，已有算法[17]并没有给出对条件算子连接符的处理方法。由于连接符仅作为一种中间状态存在，无法构成一个完整的迁移，如图7(a)带条件算子的状态图，其中的迁移$T1$、$T2$、$T5$并没有导致实际的状态转换。



(a)带条件算子的状态图　　(b)转换后无条件算子的状态图

图7　含条件算子状态图的等价转换

基于状态图的语义，带条件算子的状态图均可以转换为不含条件算子的状态图且转换前后状态图的语义保持一致[15]。例如，图7(a)可转换为不含条件算子的状态图7(b)，图7(b)中迁移$T6$、$T7$等价于图7(a)中的有序迁移集合{$T1$，$T2$，$T5$，$T3$}和{$T1$，$T2$，$T5$，$T4$}，其中的触发事件、守卫条件分别为集合中各迁移对应部分的逻辑与。

据此，对于Mdrv，分别将有序的迁移集合：{$Tr1$}、{$Tr2$，$Tr4$，$Tr7$}、{$Tr2$，$Tr4$，$Tr8$}、{$Tr2$，$Tr3$，$Tr5$}、{$Tr2$，$Tr3$，$Tr6$，$Tr9$}及{$Tr2$，$Tr3$，$Tr6$，$Tr10$}视为一个完整迁移，相应的转换如下：

```
/*模拟Mdrv中迁移*/
13. if ::MV_Nzero&&IT_enabled->MV_zero=1;
14. MV_Nzero=0; /*模拟迁移{T1}*/
    ...
15. ::MV_zero&&/*模拟{Tr2,Tr4,Tr7}*/
    (IT_disabled&&M_on&&F_Mcmd!=0)->
16. MV_Nzero=1;MV_zero=0;
17. ::else->skip;
18. fi;
```

5.2　验证与分析

使用LTL公式规约与控制逻辑相关的高级安全性需求，部分需求描述及LTL属性如下。

需求1：当空速过大时，对机翼的展开操作进行限制。

P1{[]((*AS_norspd*==0) || ((*F_Acmd*==1 || *F_Mcmd*==1)&&*MS_spand*==0->((*MS_spand*==0)U(*AS_norspd*==1))))}

时序算子U(until)表示当有展开请求时，机翼仍处于非展开状态直到空速正常。

需求2：转速设置为0且持续0.5分钟后，关闭风扇开关。Mspd中超时动作timeout(entered zero,30)/offFan实现该需求。动作offFan发生，风扇关闭。

P2{[]((*FS_on*&&*Ev*==*offFan*)->()(*FS_off*==1))}

时序算子()表示最终风扇会被关闭。

需求3：禁用有效，无论处于手动或自动状态，电机驱动输出为0。

P3{[]((*IT_enabled*==1)->()(*MV_zero*==1)))}

一旦禁用(InhibitTest)处于有效状态(enabled)，电机驱动输出(Mdrv)最终被置为零(MV_zero)。

需求4：禁用有效，无论处于手动或自动状态，制动指令有效。

P4{[]((*IT_enabled*==1)->()(*MK_enabled*==1)))}

一旦禁用(InhibitTest)处于有效状态(enabled)，制动指令(Mbrk)最会被置为有效(enabled)。

需求5：驱动非零，而制动指令有效，电机输出故障。

P5{[]((*MV_Nzero*==1&&*MK_enabled*==1)->()(*MT_fault*==1))}

一旦驱动输出(Mdrv)非零(Nzero)且制动(Mbrk)有效，电机输出检测(Mdb_Test)最终能够检测出该故障。

使用SPIN验证低级需求的Promela模型未发现错误，添加LTL属性后继续验证，表2给出了各属性的验证结果统计，其中仅验证需求4的属性P4时发现了错误。

表2　各需求LTL属性验证数据统计

LTL	状态	内存(MB)	时间(s)	错误
P1	1626745	230.280	22.9	无
P2	1626745	230.280	23.1	无
P3	1655721	232.038	23.1	无
P4	7	130.866	<1	有
P5	1626745	230.280	22.9	无

图8(a)、图8(b)分别给出了属性P4、P2的验证结果。图8(a)表明，在搜索深度为249步时，发现了错误。

```
full statespace search for:
    never claim              +(p4)
    assertion violations     +(if within scope of claim)
    acceptance cycles        +(fairness enabled)
    invalid end states       -(disabled by never claim)
State-vector 44 byte,depth reached 249,errors:1
        7 states,stored(11 visited)
        3 states,matched
        14 transitions(= visited+matched)
        305 atomic steps
hash conflicts:0(resolved)
```

(a)性质P4的验证结果

```
state-vector 44 byte,depth reached 37530,errors:0
  1626745 states,stored
  4794600 states,matched
  6421345 transitions(= stored+matched)
1.429698e+08 atomic steps
hash conflicts:102796(resolved)
```

(b)性质P2的验证结果

图8

分析错误执行路径，当禁用有效，控制单元处于手动状

态，手动指令 *F_Aset* 为 1(展开)或－1(收回)时，制动指令无效，因而不满足需求“禁用有效，制动有效”。分析图 5 的 Mbrk 状态，当系统处于手动状态，即 *C*2 为假时，直接根据手动指令 F_Mset 判断制动指令是否有效，未考虑禁用是否有效，这一设计失误导致了验证出错。图 9 给出了修复后的状态图，图中增加了手动状态下，即 *C*2 为假时对禁用是否有效的判断 *C*1。

验证修复后的状态图未发现错误，表明修复后的低级需求满足与控制逻辑相关的高级安全性需求。

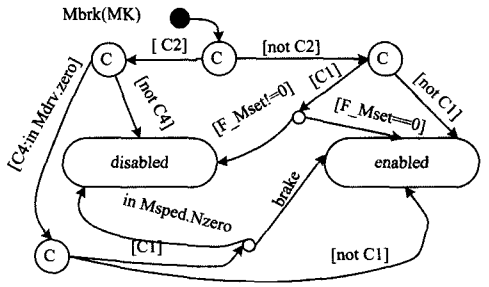


图 9　修复后的 Mbrk 状态

表 3　SFCU 低级需求验证目标满足情况

目标	满足程度	证据说明
A-4.1 低级需求符合高级需求	■	SPIN 验证 LTL 属性未发现错误，表明 SFCU 的低级需求(图 5)的 Promela 模型满足控制逻辑相关的高级安全性需求：机翼的展开限制、故障的有效检测和处理等的 LTL 属性，说明该低级需求符合高级需求。
A-4.2 低级需求是准确的和一致的	■	使用状态图描述低级需求并将其转换成形式化的 Promela 模型说明了低级需求的准确性；转换后的 Promela 模型在 SPIN 中验证无错，即不存在逻辑冲突证明了低级需求的一致性。
A-4.3 低级需求与目标机兼容		本节对低级需求的验证未涉及目标机，无法说明。
A-4.4 低级需求是可验证的	■	低级需求的状态图描述转换为 SPIN 可验证的 Promela 模型表明了低级需求的可验证性。
A-4.5 低级需求符合标准	□	本节验证工作仅能说明图 5 低级需求状态图的 Promela 模型符合 Promela 的语法规范。
A-4.6 低级需求可追踪到高级需求	□	低级需求的 Promela 模型满足与控制逻辑相关的高级安全性需求的 LTL 属性，说明该高级需求被开发成低级需求；本节验证工作未能说明低级需求可追踪到高级需求，即低级需求对某高级需求的必要性，这通过评审说明。
A-4.7 算法准确	■	使用状态图建模控制逻辑的低级需求，并通过模型检验证明了低级需求实现了与控制逻辑相关的高级需求，说明了算法设计的准确性。

注：■：满足　□：部分满足

针对 DO-178C 中低级需求的验证目标，表 3 总结了本节修复后的控制单元低级需求的满足情况，并提供了具体的证据说明。

本节低级需求到 Promela 模型、高级需求到 LTL 属性转换的正确性、分析过程及结果的正确性等通过评审确认。

结束语　有效验证机载软件制品是否满足适航认证目标是软件开发过程中的一个重要问题。针对该问题，本文基于适航标准 DO-178C 的形式化方法补充文档 DO-333，结合飞控系统中的襟缝翼控制单元，提出了面向 DO-333 的襟缝翼控制单元验证框架，展示了模型检验如何应用于一个实际的机载软件开发过程不同阶段软件制品的验证以及从中能够获得的认证证据，为机载软件的安全性保障和适航认证提供技术支持；其次，验证过程中对含有条件算子状态图的处理进行了补充；最后，发现了设计过程中引入的错误并进行了修复。本文展示了模型检验在一个实际的机载软件认证中的应用，表明模型检验可有效应用于机载软件的验证以保障安全，并能为 DO-178C 的认证目标提供证据支持。

参 考 文 献

[1] RTCA. DO-178C, Software Considerations in Airborne Software[S]. December 2011

[2] Xu B F, Huang Z Q, Hu J, et al. Model-Driven safety dependence verification for component-based airborne software supporting airworthiness certification[J]. Acta Aeronautica et Astronautica Sinica, 2012, 33(5): 796-808(in Chinese)
徐丙凤，黄志球，胡军，等. 面向适航认证的模型驱动机载软件构件的安全性验证[J]. 航空学报，2012，33(5)：796-808

[3] Huang Zhi-qiu, Xu Bing-feng, Kan Shuang-long, et al. Survey on Embedded Software Safety Analysis Standards, Methods and Tools for Airborne System[J]. Journal of Software, 2014, 25(2): 200-218(in Chinese)
黄志球，徐丙凤，阚双龙，等. 嵌入式机载软件安全性分析标准，方法及工具研究综述[J]. 软件学报，2014，25(2)：200-218

[4] RTCA. DO-333, Formal Methods Supplement to DO-178C and DO-278A[S]. December, 2011

[5] Souyris J, Wiels V, Delmas D, et al. Formal verification of avionics software products[M]// FM 2009: Formal Methods. Springer Berlin Heidelberg, 2009: 532-546

[6] Moy Y, Ledinot E, Delseny H, et al. Testing or Formal Verification: DO-178C Alternatives and Industrial Experience[J]. Software, IEEE, 2013, 30(3): 50-57

[7] Miller S P, Whalen M W, Cofer D D. Software model checking takes off[J]. Communications of the ACM, 2010, 53(2): 58-64

[8] The Scade Website [OL]. http://www.esterel-technologies.com/products/scade-suite

[9] Bochot T, Virelizier P, Waeselynck H, et al. Model checking flight control systems: The Airbus experience[C]// 31st International Conference on ICSE Companion 2009. 2009: 18-27

[10] Laurent O. Using formal methods and testability concepts in the avionics systems validation and verification(v&v) process[C]// 2010 Third International Conference on Software Testing, Verification and Validation(ICST). IEEE, 2010: 1-10

[11] Cofer D, Miller S. DO-333 Certification Case Studies [M]// NASA Formal Methods. Springer International Publishing, 2014: 1-15

[12] Chen Z, Gu Y, Huang Z, et al. Model checking aircraft controller software: A case study[J]. Software: Practice and Experience, 2013, 45(7)

(下转第 161 页)

2008,7(3):1-53

[3] Lv M,Guan N,Zhang Y,et al. A Survey of WCET Analysis of Real-Time Operating Systems[C]//Proceedings of the 2009 International Conference on Embedded Software and Systems. 2009:65-72

[4] Metzlaff S, Ungerer T. Impact of Instruction Cache and Different Instruction Scratchpads on the WCET Estimate[C]//2011 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS). 2012:1442-1449

[5] Ni F,Long X,Wan H,et al. Using Basic Block Based Instruction Prefetching to Optimize WCET Analysis for Real-Time Applications[C]//Proceedings of the 2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies. 2012:459-466

[6] Yoo J, Lee J, Hong S. Petri Net-Based FTL Architecture for Parametric WCET Estimation via FTL Operation Sequence Derivation[J]. IEEE Transactions on Computers, 2013, 62(11): 2238-2251

[7] Puschner P, Prokesch D, Huber B, et al. The T-CREST approach of compiler and WCET-analysis integration[C]//2013 IEEE 16th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC). 2013:1-8

[8] Ji Meng-luo,Qi Zhi-chang. An Overview of Worst Case Execution Time (WCET) Analysis[J]. Computer Science, 2006, 33(10):238-241(in Chinese)
姬孟洛,齐治昌. 实时系统程序最差情况执行时间(WCET)的分析[J]. 计算机科学,2006,33(10):238-241

[9] Zhang Bao-min,Wu Guo-wei,Yao Lin. Program worst-case execution time extreme value statistics estimation method[J]. Computer Engineering and Applications,2010,46(26):67-71(in Chinese)
张保民,吴国伟,姚琳. 程序最坏执行时间极值统计方法[J]. 计算机工程与应用,2010,46(26):67-71

[10] Marref A,Betts A. Accurate Measurement-Based WCET Analysis in the Absence of Source and Binary Code[C]//Proceedings of the 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing. 2011:127-135

[11] Hu Ming-hua,Tang Ming-duan. Worst case execution time estimation based on distribution function[J]. Computer Engineering and Design,2006,27(16):3045-3047(in Chinese)
胡明华,汤铭端. 基于分布函数的程序执行时间的静态预估[J]. 计算机工程与设计,2006,27(16):3045-3047

[12] Lisper B, Santos M. Model Identification for WCET Analysis[C]//Proceedings of the 2009 15th IEEE Symposium on Real-Time and Embedded Technology and Applications. 2009:55-64

[13] Bygde S, Ermedahl A, Lisper B. An Efficient Algorithm for Parametric WCET Calculation[C]//Proceedings of the 2009 15th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications. 2009:13-21

[14] Leveque T,Borde E,Marref A,et al. Hierarchical Composition of Parametric WCET in a Component Based Approach[C]//Proceedings of the 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing. 2011:261-268

[15] Lv Ming-song,Guan Nan,Wang Yi. Survey of Cache Analysis for Worst-Case Execution Time Estimation[J]. Journal of Software,2014,25(2):179-199(in Chinese)
吕鸣松,关楠,王义. 面向 WCET 估计的 Cache 分析研究综述[J]. 软件学报,2014,25(2):179-199

[16] Wu Guo-wei, Li Zhang. Precise program worst-case execution time analysis method[J]. Computer Engineering and Applications,2010,46(18):60-64(in Chinese)
吴国伟,李张. 一种精确程序最坏执行时间分析方法[J]. 计算机工程与应用,2010,46(18):60-64

[17] Bernat G, Colin A, Petters S. pWCET: a Tool for Probabilistic Worst-Case Execution Time Analysis of Real-Time Systems: YCS-2003-353[R]. England,UK:University of York,2003

[18] Petters SM. How much worst case is needed in wcet estimation[C]//International Workshop on Worst Case Execution Time Analysis. 2002

[19] http://wiki.mbalib.com/wiki/PERT

[20] Aho V A,Sethi R,Ullman J D. 编译原理[M]. 北京:机械工业出版社,2008

[21] T. I. TMS320C6000 CPU and instruction set reference guide[Z]. 2000

[22] Wu Su,Luo Yan-sheng. Application of a New Module of PERT in Project Management[J]. Construction Technology, 2007, 36(12):50-53(in Chinese)
吴苏,罗延生. 一种新 PERT 模型在项目管理中的应用[J]. 施工技术,2007,36(12):50-53

(上接第 156 页)

[13] Holzmann G J. The SPIN model checker: Primer and reference manual[M]. Reading: Addison-Wesley,2004

[14] Baier C,Katoen J P. Principles of model checking[M]. Cambridge: MIT press,2008

[15] Harel D. Statecharts: A visual formalism for complex systems[J]. Science of Computer Programming,1987,8(3):231-274

[16] Mikk E,Lakhnech Y,Siegel M,et al. Implementing statecharts in PROMELA/SPIN[C]//Proceedings 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques, 1998. IEEE,1998:90-101

[17] Latella D,Majzik I,Massink M. Automatic verification of a behavioural subset of UML statechart diagrams using the SPIN model-checker[J]. Formal Aspects of Computing,1999,11(6): 637-664

[18] Dong Wei,Wang Ji,Qi Zhi-chang. Slicing UML Statecharts for Model Checking[J]. Acta Electronica Sinica, 2002, 30(12): 2084-2089(in Chinese)
董威,王戟,齐治昌. UML Statecharts 的切片模型检验方法[J]. 电子学报,2002,30(12):2084-2089