无人机飞控软件系统建模与测试用例生成研究

吴黎明[1] 胡 军[1,3] 曹 东[2] 徐丙凤[1] 于笑丰[3,4]
(南京航空航天大学计算机科学与技术学院 南京 210016)[1]
(南京航空航天大学自动化学院 南京 210016)[2]
(南京大学计算机软件新技术国家重点实验室 南京 210093)[3] (南京大学商学院 南京 210093)[4]

摘 要 软件规模与复杂度的迅速增长已成为设计与检验现代高质量无人机飞行控制软件(FCS)系统的重要挑战。采用模型驱动工程(MDE)的框架,使用嵌入式实时系统建模语言(MARTE)建立起某型无人机飞控软件系统的模型,给出了基于时间自动机的系统动态行为的形式化模型实例;结合无人机FCS系统的应用背景,建立了基于时间自动机模型的测试用例生成方法,包括建立测试用例生成框架、测试用例生成规则以及用例生成策略等;对某型无人机飞控软件系统中的主控模块进行了建模与测试用例生成的实例分析研究。

关键词 模型驱动工程,基于模型的测试,飞行控制软件,时间自动机,无人机系统
中图法分类号 TP301 **文献标识码** A

Research on Modeling and Test Case Generation for UAV Flight Control Software System

WU Li-ming[1] HU Jun[1,3] CAO Dong[2] XU Bing-feng[1] YU Xiao-feng[3,4]
(College of Information Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[1]
(College of Automation, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)[2]
(State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)[3]
(School of Business, Nanjing University, Nanjing 210093, China)[4]

Abstract The rapid growth of software size and complexity has become an important challenge for designing and verifying modern high-quality UAV flight control software (FCS) system. Based on the architecture of Model Driven Engineering(MDE), an UAV flight control software model was established by using embedded real-time system modeling language(MARTE), and an example of formal model for system dynamic behaviors based on timed automata was given. Considering the application background of UAV FCS system, we proposed a test case generation method based on timed automata, including establishment of testing architecture, coverage rules and strategies of test case generation. Lastly, a case study of timed automata modeling and test case generation for the main control module of an UAV FCS system were provided.

Keywords Model driven engineering, Model-based testing, Flight control software, Timed automata, UAV system

无人机飞行控制软件(FCS, Flight Control Software)承担着无人机的姿态控制、自主导航、任务执行等多项功能,是确保其飞行安全、完成预定飞行任务的关键部分,也是协调、管理和控制无人机各子系统的综合控制器[1]。近年来,随着国内无人机研究与应用领域的快速发展,其机载软件系统的规模与复杂度也迅速增长,软件系统的成本与可靠性已经成为整个无人机系统开发的核心问题。如何设计、构造与检验高质量的FCS系统也成为了现在我国无人机领域中的重要挑战[2]。

以代码为中心的软件测试技术[2]是目前国内用来发现FCS系统错误、检验系统质量的传统手段,它已经不能满足目前及未来较高复杂度的FCS系统可靠性与质量的要求。现代软件工程领域中,以系统建模与分析为核心思想的模型驱动工程(MDE, Model Driven Engineering)[3-5]方法为无人机FCS系统的开发、测试与维护提供了良好的框架。本文的工作基于MDE思想,研究了某型无人机现有的FCS系统,通过分析其设计文档及源代码,研究其软件模型的建立方法,并给出无人机应用领域中基于模型的测试用例生成策略,提高了FCS系统测试用例生成的有效性以及系统的可维护性;为进一步建立与完善无人机FCS系统的各类模型,以及设计与应

到稿日期:2011-08-16 返修日期:2011-11-24 本文受江苏省研究生培养创新工程基金(CXZZ11_0218),南京航空航天大学科技创新基金(NS2010095)资助。

吴黎明(1987—),男,硕士生,主要研究方向为模型驱动的软件测试、嵌入式软件建模与分析、无人机飞行控制系统分析与验证,E-mail:kookeychen@126.com;胡 军(1973—),男,副教授,CCF会员,主要研究方向为软件分析与验证、嵌入式系统、形式化方法;曹 东(1972—),男,副研究员,主要研究方向为复杂嵌入式系统、航空控制工程、软件工程;徐丙凤(1986—),女,博士生,CCF学生会员,主要研究方向为软件工程、嵌入式软件建模与分析;于笑丰(1976—),男,讲师,主要研究方向为软件工程、模型驱动工程。

用基于模型的分析与检验方法奠定了良好的基础。

本文首先介绍了模型驱动工程(MDE)的基本思想,给出了所建立的某型无人机 FCS 系统的简要结构模型,并着重说明了本文中用来对 FCS 系统行为进行形式化建模的时间自动机模型及实例;其次结合无人机 FCS 系统的应用背景,给出了基于时间自动机模型的测试用例生成方法,包括建立测试用例生成框架、测试用例生成规则以及用例生成策略等;然后通过对某型无人机 FCS 系统中主控模块进行建模与测试用例生成的实例分析研究,验证了本文所提出的基于模型驱动的 FCS 测试用例生成方法;最后是相关工作与结束语。

1 模型驱动工程(MDE)与 FCS 系统建模

1.1 模型驱动工程(MDE)

模型驱动工程(MDE)是近几年来对象管理组织(Object Management Group,OMG)[6]在软件工程领域中提出的以软件开发过程中各层次模型为核心来构造复杂软件系统的一种方法学。MDE 强调在软件过程中必须建立多层次、多角度的抽象软件模型,并以此为核心来展开基于模型的分析、检验及模型转换工作[5,7]。通过抽象建模、分析与模型转换,MDE 方法能有效地控制和管理大规模软件系统的复杂度,增强设计、开发与测试人员对系统的理解深度,提高软件过程中的开发效率;与此同时,在 MDE 框架下,可以通过建立系统的形式化模型,严格分析系统所需满足的关键功能与性质,并构建基于模型的测试用例自动化生成方法,从而有效地提高系统可靠性、可维护性等各方面的质量。

目前国内无人机 FCS 系统的设计与开发主要都是由相关的领域专家来进行的,其软件过程中缺少有效的软件工程技术与方法应用,相关的软件文档与模型也非常缺乏,这种以代码为中心的开发模式已经不能满足目前及未来高性能的无人机系统中的软件质量要求。因此,MDE 中以软件模型为核心的思想为无人机 FCS 系统的设计、开发与测试提供了一个非常好的指导框架。通过建立准确、有效的 FCS 软件模型,并在此基础上设计各类基于模型的分析、检验、测试等方法,就能在软件过程中的每个阶段上对 FCS 系统展开有效的质量保障工作,从而构建出高质量的现代无人机 FCS 系统。

1.2 基于 MARTE 的无人机 FCS 系统建模

考虑到无人机 FCS 系统是一个实时嵌入式软件系统,因此本文采用了近年来出现的"嵌入式实时系统建模与分析语言"(MARTE,Modeling and Analysis for Real-Time Embedded systems)[8]作为主要的系统建模语言。MARTE 是统一建模语言(UML)在嵌入式实时计算领域中的一个扩展(profile),尤其是提供了在系统非功能属性方面(如时间、资源等)的有效建模与分析机制。

概要述之,MARTE 中的概念主要分为基础包(MARTE Foundations)、设计模型包(MARTE Design Model)和分析模型包(MARTE Analysis Model) 3 个部分。其中,MARTE 基础包包括嵌入式实时系统中建模的基本元素,如核心元素、时间、资源、分配等和一系列核心元素,它是 MARTE 建模的最核心的部分。MARTE 设计模型包将构件、高级应用以及详细的资源建模结合起来,用来对系统的细节进行建模和设计,同时给出了实时系统的行为建模机制。MARTE 分析模型包则在上述模型的基础上,给出了基于模型的系统质量、调度以及性能等非功能特征方面的分析方法。因此,采用 MARTE 对无人机 FCS 系统进行抽象与建模,将为进一步有效地展开基于模型的分析与检验等工作提供良好的基础。

典型的 MDE 过程应该是首先根据系统应用需求建立需求分析阶段的模型并分析,然后逐步展开系统设计模型、平台级实现模型等的建模、分析、转换、验证工作。但考虑到现有的无人机应用领域中具体型号各异,且 FCS 系统相关的需求及设计文档通常内容不全、缺乏维护,难以获取系统建模的有效信息。另一方面,由于同一系列的无人机型号系统都具有类似的结构和功能,因此目前我们采取的方法是首先对某一个典型的无人机型号系统地源代码展开人工分析,进行系统的理解及其逆向抽取信息与建模工作,然后在所建模型的基础上,对同系列无人机 FCS 系统展开基于模型的分析、测试、验证等工作,并同时对所建立的模型本身进行不同型号的相应变更和检查,逐步为现有及未来的型号无人机 FCS 系统建立起准确、有效的模型集合,为有效地实施 MDE 方法提供核心的模型基础。

下面给出了两个基于 MARTE 的某型号 FCS 系统结构模型示例。图 1 是基于 MARTE 设计模型包中高级应用建模机制构建的系统总体静态结构模型,其中系统的主控制器 FlightController 使用 RtUnit 构造型定义,提供了系统行为开始执行的入口点。FlightController 控制类有 8 个 RtService 构造型定义的服务,其中 startControl()和 stopControl()是用于控制主控器服务开始和结束的两个服务,必须实时地响应要求,因此同时使用 RtService 和 RtFeature 两个构造型定义。FCS 类图还包括 5 个使用 PpUnit 构造型定义的活动类,它们定义 FlightControl 主控类的共享信息,为它提供服务并且被 FlightControl 调用。

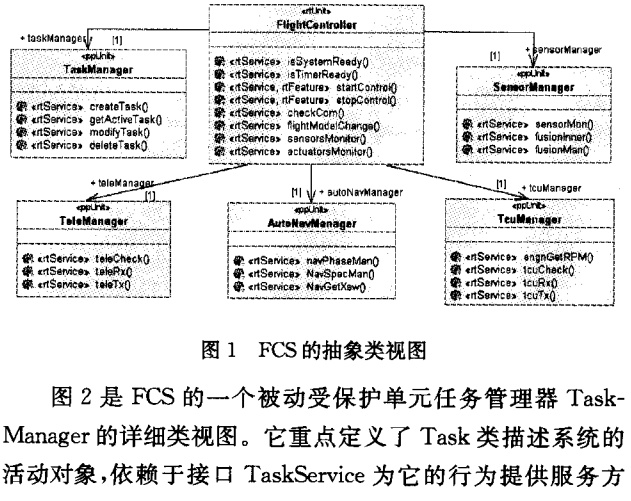


图 1 FCS 的抽象类视图

图 2 是 FCS 的一个被动受保护单元任务管理器 TaskManager 的详细类视图。它重点定义了 Task 类描述系统的活动对象,依赖于接口 TaskService 为它的行为提供服务方法,包括创建任务对象(CreateTask)、获取活动任务对象(GetActiveTask)、任务初始化(Task_Init)和删除等待任务(DelWaitTask)等服务。系统活动对象 Task 有 3 个属性,即任务名称、任务类型和任务调度器,还可以关联执行任务所需要的资源(Resource)和调度器(Scheduler)。其中一个任务调度器还可以管理多个任务,从而实现多任务并发处理。Task 活动类可以泛化成为具体的活动类 ActiveTask 和 WaitTask,

以抽象 FCS 不同类型的飞行任务状态。

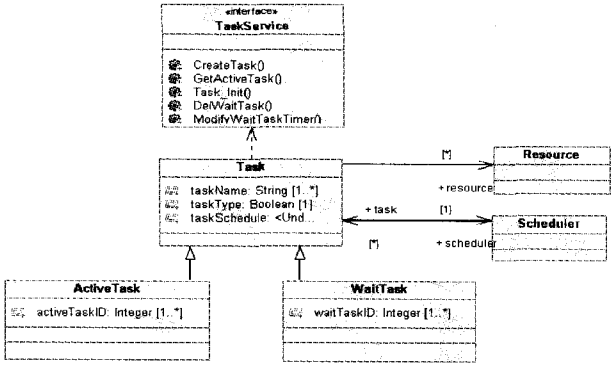


图 2　TaskManager 的详细类视图

1.3　基于时间自动机的 FCS 系统动态行为建模

MDE 的重要好处之一是可以应用基于模型的形式化方法[9]对系统进行分析、测试与验证。时间自动机[10,11]是一种专门用来对实时嵌入式系统的动态行为进行描述的形式化模型，并且目前已存在较好的时间自动机建模分析工具，如 UPPAAL[12]等。本文的主要内容即是基于时间自动机模型的无人机 FCS 系统动态行为建模，并基于时间自动机模型研究在 MDE 框架下的 FCS 系统测试用例自动生成技术。基本思想是：将测试用例生成问题转换成为时间自动机模型状态的可达性问题，基于时间自动机状态或迁移覆盖准则生成测试路径序列，并通过验证测试路径的可达性来判断生成测试用例的有效性。

为方便后续描述测试用例生成方法，在此给出时间自动机模型的简要形式化语法和语义定义[12]。

定义 1(时间自动机语法定义)　一个时间自动机 $\mathcal{A}$ 是一个六元组 (L,l_0,C,A,E,I)，其中：L 表示有穷的节点集合，$l_0\in L$ 表示初始节点，C 表示时钟集，A 表示动作集(包括动作、同步动作和内部动作)，$E\in L\times A\times B(C)\times 2^C\times L$ 表示边的集合，$B(C)$ 表示原子时间约束条件的合取集，原子约束格式为 $x\sim n$ 或 $x-y\sim n$，其中 $x,y\in C$，$\sim\in\{\leqslant,<,=,>,\geqslant\}$，$n\in\mathbb{N}$，节点之间的边带有动作、卫士条件和需要重置的时钟集，$I:L\to B(C)$ 为节点的不变式。

定义 2(时间自动机的语义定义)　假定 (L,l_0,C,A,E,I) 是一个时间自动机，其语义被定义为标签迁移系统 $(S,s_0,\to)$，其中 $S\subseteq L\times\mathbb{R}^C$，表示状态集，$s_0=(l_0,u_0)$ 表示初始状态，而变迁关系记作 $\to\subseteq S\times\{\mathbb{R}_{\geqslant 0}\cup A\}\times S$，因此

$(l,u)\xrightarrow{d}(l,u+d)$ if $\forall d':0\leqslant d'\leqslant d\Rightarrow u+d'\in I(l)$，and

$(l,u)\xrightarrow{a}(l',u')$ if there exists $e=(l,a,g,r,l')\in E$ s. t. $u\in g$，$u'=[r\mapsto 0]u$，and $u'\in I(l')$

其中，$d\in\mathbb{R}_{\geqslant 0}$，$u+d$ 表示对 C 中所有时钟 x 赋值为 $u(x)+d$，并且 $[r\mapsto 0]u$ 表示 r 中所有时钟被重置为 0，而 $C\backslash r$ 中的其他时钟满足约束条件 u。

定义 3(时间自动机网络的语义定义)　假设 $\mathcal{A}_i=(L_i,l_i^0,C,A,E_i,I_i)$ 是时间自动机网络，$\bar{l}_0=(l_1^0,\cdots,l_n^0)$ 是初始节点向量。时间自动机网络的语义被定义为 $(S,s_0,\to)$，其中 $S=(L_1\times\cdots\times L_n)\times\mathbb{R}^C$ 表示状态集，$s_0=(\bar{l}_0,u_0)$ 表示初始状态，而变迁关系 $\to\subseteq S\times S$ 定义为

$(\bar{l},u)\to(\bar{l},u+d)$ if $\forall d':0\leqslant d'\leqslant d\Rightarrow u+d'\in I(\bar{l})$

$(\bar{l},u)\to(\bar{l}[l_i'/l_i],u')$ if there exists $l_i\xrightarrow{\tau g r}l_i'$ s. t. $u\in g$，$u'=[r\mapsto 0]u$ and $u'\in I(\bar{l})$

$(\bar{l},u)\to(\bar{l}[l_i'/l_i][l_j'/l_j],u')$ if there exists $l_i\xrightarrow{c?\ g_i r_i}l_i'$ and $l_j\xrightarrow{c!\ g_j r_j}l_j'$ s. t. $u\in(g_i\wedge g_j)$，$u'=[r_i\cup r_j\mapsto 0]u$ and $u'\in I(\bar{l})$

时间自动机网络通常由多个时间自动机组成：$\mathcal{A}_i=(L_i,l_i^0,C,A,E_i,I_i)$，$1\leqslant i\leqslant n$，它们有共同的时钟和动作。每个时间自动机称为一个进程(Process)，进程间的同步通信使用输入输出动作以握手同步的方式进行，异步通信以共享全局变量的方式进行。时间自动机网络在一次同步通信中，一个同步信号发送者 $c!$ 可以和任意多个接收者 $c?$ 同步。

基于对 FCS 系统源码的分析与理解，我们使用时间自动机的建模工具 UPPAAL 对某型无人机 FCS 系统的动态行为建模。图 3、图 4 中给出了关于 FCS 内回路控制器管理的时间自动机模型示例，其中包括内回路控制(InnerController)和传感器管理(SensorMan)两个时间自动机模板，它们之间通过同步信道 fusionInner 进行通信。

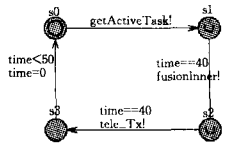


图 3　内回路控制(InnerController)时间自动机模型

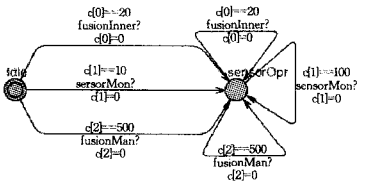


图 4　传感器管理(SensorMan)时间自动机模型

2　基于时间自动机模型的 FCS 测试用例产生

本节基于上述所建立的 FCS 系统动态行为的时间自动机模型，展开基于模型的测试用例产生框架以及具体的用例生成方法的研究。

2.1　基于时间自动机模型的测试产生框架

图 5 是在基于模型的实时系统测试框架(Architecture of Model-based test generation for Real-time system)[13-15]的基础上提出的一种基于时间自动机模型特定领域的 FCS 系统测试用例生成框架。

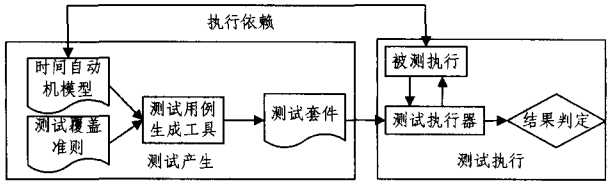


图 5　基于时间自动机模型的测试框架

整个测试框架主要分为两大部分：测试产生部分和测试执行部分。测试产生部分的主要工作包括：建立 FCS 系统的时间自动机模型，制定符合测试需求的测试覆盖标准，使用一

个实时系统测试用例生成工具 Uppaal CoVer[16,17] 来产生符合覆盖标准的测试路径；然后就可以对生成的测试路径进行序列化处理，并通过相关配置文件生成能够被测试程序识别的测试用例集(以 xml 格式的文档形式给出)。在测试执行部分，测试执行器可以读取 xml 形式的测试用例集，并和被测执行之间进行交互，从而判定测试用例的有效性，其中被测执行依赖于预先构建的系统时间自动机模型。

具体地，在测试框架中测试产生部分还需要结合无人机 FCS 系统的应用领域特征，做进一步地分析。由于无人机飞控系统是一个典型的嵌入式系统，它由很多不同功能的机载传感器(sensors)、控制器(controller)和执行机构(actuators)组成。飞控软件可以看成是一个广义的控制器，通过采集传感器的信息实时地解算出对执行机构的控制量，把解算结果输出到执行机构模块中，驱动执行机构的运作。根据这个特点，将“时间自动机模型组”进程进一步划分为控制器模型(controller model)和环境模型(environment model)两部分。控制器模型是核心控制块，它是抽取出的时间自动机模型的行为控制者；环境模型是相对于控制器模型而提出的，它为控制器模型的执行提供同步输入信号，也可以作为控制器同步输出信号的接收者，其模拟了自动机网络的执行环境。它们之间的基本关系如图 6 所示。

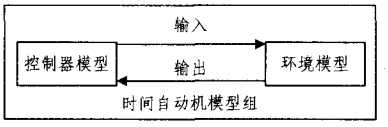


图 6　系统模型划分

此外，框架中的“测试用例生成工具”进程的作用是根据系统时间自动机模型组(已划分为控制器模型和环境模型)和测试覆盖准则，来生成由状态、延迟或离散迁移交替出现的测试路径序列组成的抽象测试用例集。本文采用 Uppaal CoVer工具来进行测试用例的生成。Uppaal CoVer 可以通过给定的时间自动机模型产生相应的测试用例集，生成的测试用例集可以进一步地编译成用来自动执行被测系统[18,19]的测试程序。具体地，在本测试框架下，使用 Uppaal CoVer 来产生测试用例的流程图如图 7 所示。

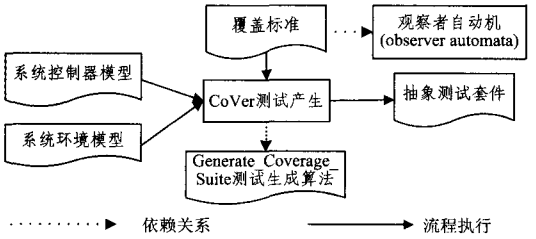


图 7　使用 CoVer 工具产生测试用例流程图

2.2　用例覆盖准则描述

在基于模型的测试方法中，生成的测试用例集也必须满足某种覆盖准则，并且测试用例集的产生过程也需要相应的覆盖准则来进行引导。覆盖准则可以看成是需要被覆盖的检查对象的一个集合，常见的有状态覆盖、迁移覆盖、数据流覆盖等[20]。以下给出本文采用的在时间自动机模型背景下的状态覆盖和迁移覆盖准则的具体表示与应用形式。

状态覆盖准则应用：当一条测试序列在时间自动机模型中执行时，路径覆盖了时间自动机模型的所有节点，则这条测试序列满足了节点覆盖标准[20]。具体地，为了产生满足节点覆盖的测试用例，我们为每一个状态节点 $l_i(1\leqslant i\leqslant n)$ 引入布尔类型的辅助变量 $a_i(1\leqslant i\leqslant n)$，除去初始节点，每个节点对应的辅助变量都初始化为 false。遍历时间自动机模型时，对于目标节点为 $l_i: l'\xrightarrow{g,a,u} l_i$ 的一次迁移满足的情况下，我们将 l_i 对应的辅助变量 a_i 进行赋值 $a_i:=$true，对于满足系统状态可达性要求在所有的 a_i 值都为 true 时(即表达式 $a_1==\text{true}\wedge a_2==\text{true}\wedge\cdots\wedge a_i==\text{true}, 1\leqslant i\leqslant n$ 为真)，测试路径搜索结束。

迁移覆盖准则应用：当一条测试序列在时间自动机模型中执行时，路径覆盖了时间自动机模型所有的边，则这条测试序列满足了迁移覆盖标准[20]。迁移覆盖也可以类似状态覆盖在时间自动机的边上增加可达性属性来实现。具体做法如下：为每一条边 $e_i(1\leqslant i\leqslant n)$ 引入布尔类型的辅助变量 $b_i(1\leqslant i\leqslant n)$，然后将每一条边对应的辅助变量值都初始化为 false。遍历时间自动机模型时，对于任何一次迁移 $e: l'\xrightarrow{g,a,u} l''$ 发生时(满足迁移条件 g 且动作 a 发生)，我们将 e_i 对应的辅助变量(可达性属性)b_i 进行赋值 $b_i:=$true，对于满足系统迁移可达性要求的所有 b_i 都为 true 时(即表达式 $b_1==\text{true}\wedge b_2==\text{true}\wedge\cdots\wedge b_i==\text{true}, 1\leqslant i\leqslant n$，为真)，测试路径搜索结束，产生了满足迁移覆盖标准的测试用例集。

为了将上述用例覆盖准则结合形式化的时间自动机模型，我们使用 Uppaal CoVer 中所提供的 observer 自动机语言[21]来具体地表达状态覆盖准则和迁移覆盖准则。具体如图 8、图 9 所示。

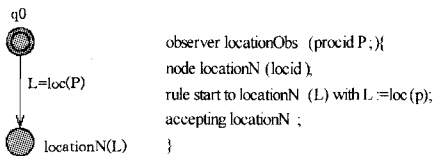


图 8　状态覆盖 observer 自动机图和规则描述

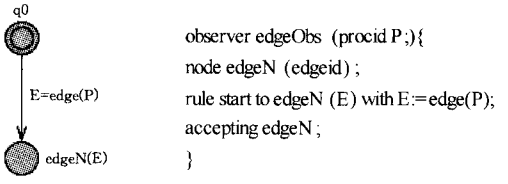


图 9　迁移覆盖 observer 自动机图和规则描述

图 8 和图 9 所示的两个 observer 自动机 locationObs 和 edgeObs 分别表示状态覆盖准则和迁移覆盖准则的观察者自动机规则描述，其作为 Uppaal CoVer 工具产生测试用例的覆盖准则。locationObs 自动机有一个开始状态 q_0 和一个接受状态 locationN，从 q_0 迁移到 locationN 节点需执行 $L=$loc(P)操作，表示获取时间自动机 P 的所有状态；edgeObs 自动机也有一个开始状态 q_0 和一个接受状态 edgeN，从 q_0 迁移到 edgeN 节点需执行 $E=$edge(P)，表示获取时间自动机 P 的所有边迁移。

2.3 测试用例生成方法描述

本小节在前两小节的基础上，给出了针对无人机 FCS 系统特定应用领域的基于时间自动机模型的测试用例生成方法。考虑到在某型无人机系统中，飞控软件的主要运行方式是由主控模块在系统时钟 time 的约束下周期性地调用、执行相关的控制任务，使得在所构建的时间自动机模型中存在以下描述形式：两个可达状态之间可能存在若干条同步动作(synchronization action)相同但卫士条件(guard condition)不同的迁移组(具体的模型视图可参见第 3 节)。这样有可能使得在进行模型分析时所需遍历状态空间增大，搜索路径序列的复杂度提高，生成测试用例集不太合理。为此，在下面测试用例生成方法的具体描述中，还包括了对主控模块的时间自动机模型的预处理，以便生成易于理解的测试用例，提高生成效率。

具体而言，本文的测试用例集生成方法是基于文献[22]中提出的全局覆盖算法，此处只给出了生成路径覆盖序列的方法细节(见图 10)。其基本思想是：首先对主控模块时间自动机模型进行预处理，将任何两个可达状态之间具有相同同步动作和不同卫士条件的迁移组进行合并(比如状态迁移 $s1\rightarrow s2$，迁移组为：time=20，a!；time=40，a!；time=200，a! 合并为 time==20 ∧ time==40 ∧ time==200，a! 的形式)，使用 passed 和 waiting 两个数据结构来表示已经检测过的状态集合和等待检测的状态集合，passed 初始化为空，waiting 初始化为一个三元组 (s_0, c_0, ε)，s_0 表示模型的初始状态，c_0 表示状态 s_0 的覆盖项，ε 表示一条空路径，使用变量 generatedSuite 和 coverage 表示产生的测试用例集集合和算法检测到的覆盖项(状态或者迁移)的集合；然后选取 waiting 中的一个状态三元组 (s,c,ω)，并把它加入 passed 集合，对于该状态的所有后继状态 $(s',c',\omega'):(s,c,\omega)\overset{g,a}{\Rightarrow}(s',c',\omega')$ 进行判断处理。如果新的覆盖项 c' 不包含于 coverage，则将路径对 (ω',c') 插到套件集合 generatedSuite 中；如果在状态集合 passed 或者状态集合 waiting 中不存在状态 (s_i,c_i,ω_i) 满足 $s_i=s'$，则将 (s',c',ω') 插到状态集合 waiting 中。直到状态集合 waiting 为空时，算法结束。

算法 Generate_Coverage_Suite

输入：飞控软件主控模块时间自动机模型组(xml 格式模型文件)

输出：满足迁移覆盖准则的测试路径序列

```
Algorithm Begin
  For all s_i --g_k,a_k--> s_j, 1≤i,j≤n,k≥1 in TA do
    If k>2 ∧ g_1≠g_2≠…≠g_k ∧ a_1 a_2 … a_k then
      mergeEdge(s_i, s_j)
    End If
  End For
  For all e_i: l' --g,a,u--> l'', 1≤i≤n in EdgeMergedTA do
      b_i:=flase
  End For
  passed:=∅, waiting:={(s_0, c_0, ε)}, generatedSuite:=∅, coverage:=c_0
  While wating≠∅ do
    Fetch (s,c,ω) from waiting, add (s,c,ω) to passed
    For all (s',c',ω'):(s,c,ω) ⇒(g,a) (s',c',ω') do
      If c' ⊄ coverag then
        Add (ω',c') to generatedSuite, coverage:=coverage ∪ c'
      End If
      If ¬∃(s_i,c_i,ω_i):(s_i,c_i,ω_i)∈passed∪waiting ∧ s_i=s' then
        Add (s',c',ω') to waiting
      End If
    End For
  End While
  return generatedSuite //返回测试用例集
Algorithm End
```

图 10 满足迁移覆盖准则的测试路径生成算法

基于类似的思想，我们同样可以建立满足状态覆盖准则的基于时间自动机的测试用例生成方法，在此不再详述。

3 实例应用与分析

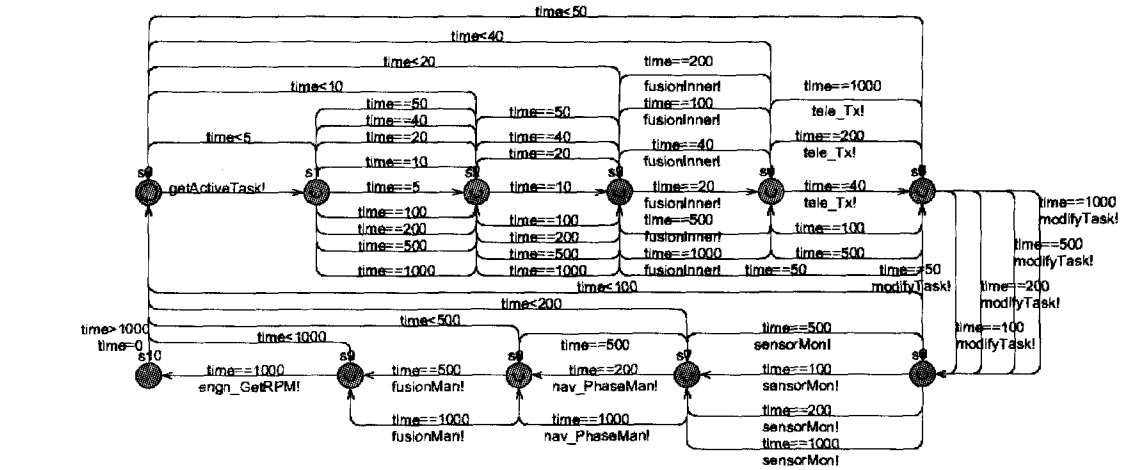
本节将上述的基于时间自动机模型的测试用例产生框架在某型无人机 FCS 系统中的主控模块中展开具体的实例应用，生成满足迁移覆盖的测试用例集。

经过对系统源码的分析，主控模块的执行过程可概要描述为：系统上电后开始运行，然后对系统中各类硬件、软件和外围设备等参数进行初始化设置，准备接受传感器串口数据信息；系统进入主控模块处理流程后，任务调度模块将启动一个准备好的任务准备执行；然后在系统时钟的约束下，周期性地执行相关各类飞行任务。基于前述的时间自动机建模理论，以下给出主控模块的时间自动机模型组，包括控制器模型和环境模型，如图 11 所示。

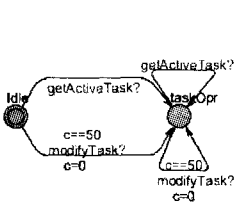
其中，控制器模型为图 11(a)，环境模型为图 11(b)－(f)，它们之间共用一个全局时钟 time，通过定义全局同步信道进行通信。同步信道的含义为：getActiveTask 表示获取一个活动任务，fusionInner 表示内回路传感器信息管理信号，tele_Tx 表示遥测数据回传信号，modifyTask 表示修改等待任务队列时间，sensorMon 表示传感器监控信号，nav_PhaseMan 表示自主导航飞行管理，fusionMan 表示根据指令设置信息源状态字，engn_GetRPM 表示发动机转速采集信号。

主控模块的行为由主控制器时间自动机来控制执行，在全局时钟 time 的约束下，通过同步信道发送或者接受同步信号，周期性地执行相关飞行任务操作。比如在 20ms 控制周期内，主控制器先后发出了两个信号 getActiveTask! 和 fusionInner!，任务管理自动机和传感器管理自动机会同步响应接受 getActiveTask? 和 fusionInner?，并根据信号类别执行相关任务。

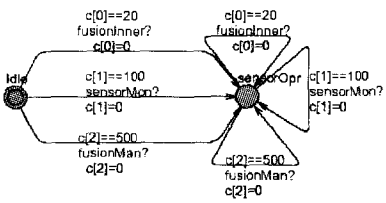
基于图 5、图 7 给出的基于时间自动机模型的测试用例产生框架及方法所产生的满足迁移覆盖准则的测试用例集如表 1 所列。在 2.3 节提出的测试用例生成算法中，使用了二元组 (ω',c') 作为测试用例集的返回类型，其中 ω' 表示算法构建的中间路径，c' 表示满足某种覆盖要求的覆盖项集合(这里表示迁移集合)。



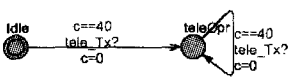
(a) 主控制器自动机模型



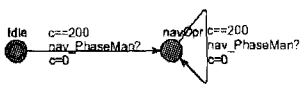
(b) 任务管理自动机模型



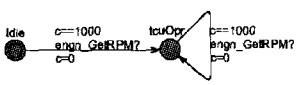
(c) 传感器管理自动机模型



(d) 遥控遥测自动机模型



(e) 自主导航自动机模型



(f) 发动机管理自动机模型

图 11　FCS 系统主控模块时间自动机模型

表 1　某型无人机 FCS 主控模块执行的测试用例

用例编号	算法 Generate_Coverage_Suite 生成的测试路径(ω')	迁移覆盖项(c')
tc1	getActiveTask,delay 4	e_{01},e_{10}
tc2	getActiveTask,delay 5,null	e_{01},e_{12},e_{20}
tc3	getActiveTask,delay 10,null,null	e_{01},e_{12},e_{23},e_{30}
tc4	getActiveTask,delay 20,null,null,fusionInner	e_{01},e_{12},e_{23},e_{34},e_{40}
tc5	getActiveTask, delay 40, null, null, fusionInner,tele_Tx	e_{01},e_{12},e_{23},e_{34},e_{45},e_{50}
tc6	getActiveTask, delay 50, null, null, modifyTask	e_{01},e_{12},e_{23},e_{35},e_{56},e_{60}
tc7	getActiveTask, delay100, null, null, fusionInner,null,modifyTask,sensorMon	e_{01},e_{12},e_{23},e_{34},e_{45},e_{56},e_{67},e_{70}
tc8	getActiveTask, delay 200,null, null, fusionInner, tele_Tx, modifyTask, sensorMon, nav_PhaseMan	e_{01},e_{12},e_{23},e_{34},e_{45},e_{56},e_{67},e_{78},e_{80}
tc9	getActiveTask, delay500, null, null, fusionInner,null,modifyTask,sensorMon,null,fusionMan	e_{01},e_{12},e_{23},e_{34},e_{45},e_{56},e_{67},e_{78},e_{89},e_{90}
tc10	getActiveTask,delay1000,null,null,fusionInner,tele_Tx,modifyTask,sensorMon,nav_PhaseMan,fusionMan,engn_GetRPM	e_{01},e_{12},e_{23},e_{34},e_{45},e_{56},e_{67},e_{78},e_{89},$e_{9,10}$,$e_{10,0}$

注:null 表示状态迁移发生时不做任何动作,其中 e_{ij} 表示状态节点 $s_i \rightarrow s_j$ 的迁移。

从表 1 可以看出,测试路径 ω'和迁移覆盖项集合 c'在测试生成算法 Generate_Coverage_Suite 的执行过程中不断地增长和扩大,算法各个阶段的迁移覆盖项都不尽相同,即每次生成的路径序列都满足新的迁移覆盖要求,而所有的迁移覆盖项 c'的合集 $\bigcup_{i=1}^{10} c_i'$恰好覆盖主控模型自动机的所有有效迁移,因此,测试路径集合 $\bigcup_{j=1}^{10} \omega_j'$满足算法的迁移覆盖要求。可以得出结论,测试用例集合 $tc1-tc10$ 满足迁移覆盖要求。

4　相关研究工作

目前国内关于无人机飞控软件的相关测试与分析工作大多是从代码的角度对系统进行研究,而采用基于模型驱动工程(MDE)的思想和 MARTE 来对无人机系统建模,并为系统设计、实现及测试等提供模型级的机制与方法的研究工作非常少。如文献[23]给出了基于高安全应用开发环境 SCADE 的无人机飞控软件设计方法,其对飞控软件的各个功能模块进行了初步建模与分析;文献[1]对无人机飞控软件的测试技术做了一定的研究,包括单元测试和集成测试,但是大部分的测试工作都是针对代码的测试,并没有建立基于模型的软件测试方法所需的系统结构和行为模型。

此外,在其他应用领域中已经存在一些基于实时嵌入式软件模型的测试用例生成方面的研究。如文献[24]提出了基于时间扩展有限状态机 t_EFSM 的实时嵌入式软件测试用例生成方法,并给出了案例分析,但是没有对生成的测试用例进行完备性分析;文献[21]使用 Uppaal 工具对实时系统的离线测试(Offline Testing)和在线测试(Online Testing)进行研究,给出了离线和在线测试生成的框架和策略描述,但并没有给出具体的案例分析来验证测试生成;文献[22]给出了一个基于模型的测试用例集生成的全局算法,并且使用 Uppaal CoVer 工具对该算法进行了实现,但没有给出满足一定覆盖标准的测试用例生成算法。

结束语　本文的主要工作包括:基于 MDE 思想,对某型

无人机现有的 FCS 系统展开研究，通过分析其设计文档及源代码，采用 MARTE 建立起 FCS 系统的软件模型，并给出了基于时间自动机的系统动态行为的形式化模型实例；结合无人机 FCS 系统的应用背景，给出了基于时间自动机模型的测试用例生成方法，包括建立测试用例生成框架、测试用例生成规则以及用例生成策略等。然后，对某型无人机 FCS 系统中主控模块进行建模与测试用例生成的实例分析研究。实例模型的应用分析表明，所提基于时间自动机模型的测试用例集生成方法是可行的，对于提高现有及未来无人机 FCS 系统的测试的有效性具有很好的指导意义。

目前正在继续开展的工作包括：研究基于更多覆盖准则（如数据流覆盖等）的测试用例生成方法，并且对某型无人机飞控软件的其他模块，如控制律解算模块、导航模块、地面检测模块、遥控遥测模块等进行建模和测试用例生成，逐步构建一个完整的无人机 FCS 系统的测试生成、检查及维护的方案，全面提高 FCS 系统测试用例生成的有效性以及系统的可维护性；同时为进一步建立与完善无人机 FCS 系统的各类模型，深入展开无人机应用领域中基于模型的分析与检验方法奠定良好的基础。

参 考 文 献

[1] 杨一波，陈欣. 无人机飞行控制软件测试技术研究[D]. 南京：南京航空航天大学，2008

[2] 王鑫，陈欣，张民. 基于 SCADE 的无人机飞行控制系统软件设计[D]. 南京：南京航空航天大学，2008

[3] Mussa M, Ouchani S, Sammane W A, et al. A Survey of Model-driven Testing Techniques[C]// 2009 Ninth International Conference on Quality Software. 2009: 167-172

[4] Heckel R, Lohmann M. Towards Model-Driven Testing[J]. Electronic Notes in Theoretical Computer Science, 2003, 82(6): 33-43

[5] 张天，张岩，于笑丰，等. 基于 MDA 的设计模式建模与模型转换[J]. 软件学报，2008，19(9)：2203-2217

[6] Object Management Group, OMG[OL]. http://www.omg.org/

[7] 张天，李宜东，Jouault F，等. 基于 MDE 的异构模型转换：从 MARTE 模型到 FIACRE 模型[J]. 软件学报，2009：214-233

[8] OMG. UML Profile for MARTE, Beta 2[EB/OL]. http://www.omg.org/cgi-bin/doc? ptc/2008-06-08, 2008

[9] Clarke E M, Wing J M. Formal methods: state of the art and future directions[J]. ACM Computing Surveys, 1996, 28(4): 626-643

[10] Herber P, Fellmuth J, Glesner S. Model checking SystemC designs using timed automata[C]// Proceedings of the 6th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. 2008: 131-136

[11] Lampka K, Perathoner S, Thiele L. Analytic Real-time Analysis and Timed Automata: A Hybrid Methodology for the Performance Analysis of Embedded Real-time Systems[J]. Design Automation for embedded systems, 2010: 193-227

[12] Behrmann G, David A, Larsen K G. A Tutorial on Uppaal[Z]. Formal Methods for the Design of Real-Time Systems, 2004

[13] Hessel A. Model-Based Test Case Generation for Real-time Systems[R]. 1214.51. Faculty of Science and Technology, 2007

[14] 颜炯，王戟，陈火旺. 基于模型的软件测试综述[J]. 计算机科学，2004，27：184-187

[15] Gutierrez J J, Escalona M J, Mejias M, et al. An approach for Model-driven test generation[C]// Research Challenges in Information Science, 2009. RCIS2009. Third International Conference on. April 2009: 303-312

[16] Uppaal CoVer[EB/OL]. http://www.uppaal.org/cover/, 2005

[17] Hessel A, Pettersson P. CoVer-A Test Case Generation Tool for Real-time Systems[C]// FATES07. Tallinn, Estonia, 2007

[18] Nielsen B, Skou A. Automated test generation from timed automata [J]. International Journal on Software Tools for Technology Transfer, 2003(5): 59-77

[19] Hessel A, Larsen K G, Nielsen B, et al. Time-optimal real-time test case generation using Uppaal[C]// Lecture Notes in Computer Science. 2004: 136-151

[20] Myers G. The Art of Software Testing[M]. Wiley-Interscience, 1979

[21] Hessel A, Larsen K G, Nielsen B, et al. Testing real-time systems using Uppaal[C]// Formal Methods and Testing, LNCS 4949. 2008: 77-117

[22] Hessel A, Pettersson P. A global algorithm for model-based test suite generation[C]// Third Workshop on Model-Based Testing. Braga, Portugal, 2007

[23] 王鑫，陈欣，张民. 基于 SCADE 的无人机飞行控制系统软件设计[D]. 南京：南京航空航天大学，2008

[24] Yin Yong-feng, Li Zhen, Liu Bin. Real-time Embedded Software Test Case Generation Based on Time-extended EFSM: A Case Study[C]// WASE International Conference on Information Engineering. 2010: 272-275

（上接第 86 页）

[8] 焦李成，杜海峰. 人工免疫系统进展与展望[J]. 电子学报，2003，31(10)：1540-1549

[9] 杨咚咚，焦李成，公茂果，等. 求解偏好多目标优化的克隆选择算法[J]. 软件学报，2010，21(1)：14-33

[10] Leclerc B. Minimum spanning trees for tree metrics: abridgements and adjustments[J]. Journal of Classification, 1995, 12: 207-241

[11] kdd cup99 dataset[OL]. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999

[12] Catlett J. On changing continuous attributes into ordered discrete attributes [C]// EWSL' 91. 1991: 164-178

[13] Ma Li, Jiao Li-cheng, Bai Lin, et al. Polyclonal Clustering Algorithm and its Convergence[J]. The Journal of China Universities of Posts and Telecommunications, 2008, 15(3): 110-117

[14] 罗敏，王丽娜，张焕国. 基于无监督聚类的入侵检测方法[J]. 电子学报，2003，11(11)：1714-1716

[15] 杨草原，刘大有，杨博，等. 聚类集成方法研究[J]. 计算机科学，2011，38(2)：166-170