软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(2):181−201 [doi: 10.13328/j.cnki.jos.004775]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于形式化方法的航空电子系统检测*

李 睿, 连 航, 马世龙, 黎 涛

(软件开发环境国家重点实验室(北京航空航天大学),北京 100191)

通讯作者: 李睿, E-mail: rui.li.buaa@gmail.com

摘 要: 随着航空型号的快速发展,航空电子系统的数字化程度越来越高,软件在其中所占的比例越来越大.对航空电子系统中的软件进行测试和检测是保证航空电子系统质量及可信运行的基础.通过分析航空电子系统软件体系结构,对航空电子系统进行形式化建模,并在此基础上,提出了一种形式化的系统级综合检测方法,从静态和动态两个方面对航空电子系统进行检测,最后通过设计并实现一个综合检测系统来验证该方法的有效性.

关键词: 模型检测;系统形式化;航空电子系统;软件配置项;有限状态自动机

中图法分类号: TP311

中文引用格式: 李睿,连航,马世龙,黎涛.基于形式化方法的航空电子系统检测.软件学报,2015,26(2):181−201. http://www.jos.org.cn/1000-9825/4775.htm

英文引用格式: Li R, Lian H, Ma SL, Li T. Avionics system testing based on formal methods. Ruan Jian Xue Bao/Journal of Software, 2015,26(2):181−201 (in Chinese). http://www.jos.org.cn/1000-9825/4775.htm

Avionics System Testing Based on Formal Methods

LI Rui, LIAN Hang, MA Shi-Long, LI Tao

(State Key Laboratory of Software Development Environment (BeiHang University), 100191, China)

Abstract: With the rapid development of aviation models, the degree of digitalization of avionics systems becomes higher and higher, and the proportion of the software in those systems becomes larger and larger. In this paper, software architecture and formal modeling of avionics systems are discussed. Further, a system level integrated testing method based on formalization for avionics system from static and dynamic aspects is proposed. At last, the effectiveness of the proposed approach is evaluated through an integrated testing system designed and implemented in this research.

Key words: model checking; system formalization; avionics system; software configuration item; finite state automaton

随着航空型号的快速发展,航空电子系统的数字化程度越来越高,软件在其中所占的比例越来越大,软件的质量已经成为新一代航空电子系统研制成败的关键所在.然而随着软件规模和复杂性的不断增加,运行环境的愈加开放、动态和多变,航空电子系统中软件实体及其之间的协同均面临着诸多变化.而航空电子系统作为一类典型的安全攸关系统,要求系统的行为及其结果是可预期的,行为的状态是可监测的,行为的结果是可评估的,行为的异常是可控制的[1,2].这也意味着航空电子系统的行为必须是可信任的.对航空电子系统中的软件进行测试和检测是保证航空电子系统质量及可信运行的基础.新型号的航空电子系统是否能够量产并投入使用很大程度上也依赖于测试和检测的结果.

本文通过分析航空电子系统软件体系结构,对航空电子系统进行形式化建模,并在此基础上提出了一种基于形式化的系统级综合检测方法,从静态检测和动态检测两个方面进行阐述,最后通过设计并实现一个综合检测系统来验证该方法的有效性.

* 基金项目: 国家自然科学基金(61003016, 61300007, 61305054); 科技部基本科研业务费重点科技创新类项目(YWF-14-JSJXY-007); 软件开发环境国家重点实验室自主探索基金(SKLSDE-2012ZX-28, SKLSDE-2014ZX-06)

收稿时间: 2014-06-23; 修改时间: 2014-10-31; 定稿时间: 2014-11-26

本文第 1 节介绍航空电子系统检测的相关研究.第 2 节对航空电子系统进行形式化建模.第 3 节给出系统配置级(静态)检测和系统运行时(动态)检测的方法.第 4 节通过原型检测系统的实现来验证本文所提出的检测方法.最后一部分是结论.

1 相关研究

目前,针对航空电子系统的检测技术主要有自动测试系统(automatic test system,简称 ATS)、机内自检(built-in test,简称 BIT)、预测与健康管理(prognostics and health management,简称 PHM)技术等.其中,ATS 是将测试所需的全部激励与测试设备集成在一起,由计算机控制高效地完成各种模式的激励以及响应信号的采集、存储与分析,自动地对被测单元(UUT)进行状态监控、性能测试和故障诊断.它包括自动测试设备(automatic test equipment,简称 ATE)、测试程序集(test program set,简称 TPS)、测试环境(test environment)、系统内自动诊断和测试系统等[3,4].国际流行的军用自动测试系统分为以美国[5]和法国[6]为代表的两个流派,技术上最为成熟.我国的 ATS 发展从全盘引进、仿制到自行研发,虽然生产规模和生成能力仅次于美、日、英、德、法,但仪器设备主要以中、低档为主.就可靠性而言,国产自动化测试设备的可靠性指标一般在 3 000h 左右,而国外所生成的同类设备能达到 10 000h 以上,平均无故障时间为 2 万~3 万小时.从整体上看,我国在技术性能、制造工艺等方面与国际先进水平还存在一定的差距.

BIT 技术由美国最早开展研究.国内外对 BIT 的定义各不相同,但是最基本的含义是大致相同的,即,BIT 指的是不依赖外部设备,系统能完成对其自身的故障检测、故障定位(对于单一的外场可更换单元)、虚假警报、错误诊断等[7].其发展历程主要分为 3 个阶段:

(1) 20 世纪 70 年代,常规 BIT 首次提出.

(2) 20 世纪 80 年代,从美国罗姆航空发展中心(RADC)提出的利用人工智能来提高机内自测的概念中发展出了初期的智能 BIT,这之后,神经网络、专家系统、模糊逻辑等智能理论和方法被不断引入 BIT 的故障诊断中.

(3) 20 世纪 90 年代中后期,分级 BIT 技术的产生[8].分级 BIT 技术是边缘扫描技术(BST)的一种延伸,它在功能板级 HIBIT 利用 IEEE 1149.1 边缘扫描技术进行测试,而系统和分系统级的测试则通过 IEEE 1149.5 测试与维修总线(test and maintenance bus)来进行.

我国对 BIT 的研究和应用始于 20 世纪 80 年代中后期,虽然在将 BIT 应用于一些国防装备系统新型号的研制实践中,我国取得了很大的进步,但是相对于国外,其中存在的差距仍不容小视.

PHM 技术是对武器系统使用的机内自检和状态(健康)监控能力的进一步拓展,其中预测指的是预计性地诊断部件或系统完成其功能的状态;健康管理是根据诊断/预测信息、可用资源和使用需求对维修活动做出适当决策[9,10].该技术借助各种算法和智能模型来识别和管理故障的发生、规划维修和供应保障,目的在于,在降低使用和保障费用的同时提高装备系统安全性、战备完好性和任务成功性,实现基于状态的维修和自主式保障.PHM 技术经历了故障诊断、故障预测、系统集成这 3 个日益完善的过程[11].

ATS,BIT 和 PHM 这 3 种检测技术被广泛地应用于航空电子系统的综合检测中,然而它们也有不足之处.例如,ATS/ATE 技术成本高、开发周期长,且一般需要完全内嵌或半内嵌在航空电子系统中.BIT 技术则要求完全内嵌在航空电子系统中,且规模一般不能超过航空电子系统的 10%,这在一定程度上限制了其功能,同时在应用过程中暴露出虚警率过高、容错能力不足等问题.PHM 技术也存在成本高、开发周期长的缺点,并且由于数据获取困难,其虚警率一直达不到理想状态.值得一提的是,由于早期航空电子系统的结构特点,这 3 种技术的着眼点都放在硬件元器件方面,而忽略了对系统中软件部分的关注.

对系统中软件部分的检测有代码检测、软件测试及形式化方法.其中,代码检测的最终结果更多地是在说明系统是正确的.我们知道,系统是否正常工作,不仅是指系统是正确的,同时还意味着系统的运行是按照预期的方式在进行.对于软件测试,因其不能被用于软件开发的早期以及测试用例覆盖范围有限等局限性,使得其在诸如航空电子系统、高铁控制系统、核安全控制系统等大规模复杂系统的可靠性检测中不能得到广泛的应

用[12].而形式化方法以数学化的程序理论为基础,作为一种思想、方法、技术,它能被应用于软件开发的每一个过程中,其中形式化建模使用具有严格数学定义语义和语法的语言来刻画软件及其性质,以及描述其行为模式,以保证软件的正确性,例如,1992 年的英国伦敦空中交通管理系统中信息显示分系统在需求阶段就采用了形式化方法,构建了 VDM 模型,与采用非形式化技术开发的其他分系统相比,其在质量及生产效率上得到了很大的提高,故障率降低至0.75个/千行代码[13].加州大学安全关键系统研究组所开发的空中交通防碰撞系统采用了基于Statecharts的需求状态机语言RSML,同样也在需求阶段就进行了形式化建模,有效地解决了开发过程中遇到的许多问题[14].Vassev 等人在文献[15]中使用自主系统规范语言 ASSL 为 NASA 任务建立形式化模型并生成了功能原型.朱岩等人建立了综合航空电子系统数据总线传输系统的随机 Petri 网系统模型,并在此基础上对综合航空电子系统进行性能分析[16];而形式化验证则在形式化建模的基础上通过模型检验、定理证明等途径来分析和验证软件是否具有所期望的性质,并符合给定的行为模式.例如,用于开发和分析形式化规范的原型证明系统 PVS 被用于航空电子微处理器 AAMP5 的描述和验证,对 209 条 AAMP5 指令中的 108 条进行了描述,验证了 11 个有代表性的微代码[17].Bieber 等人在文献[18]中用 Altarica 模型验证了空中客车 A320 液压子系统的可靠性与安全性.Eiels 等人在文献[19]中从 4 个方面对关键航空软件中形式化验证进行了综述.形式化方法不仅在航空、航天领域得到了成功的应用,也被广泛应用于其他诸如金融商业、轨道交通等安全攸关领域.

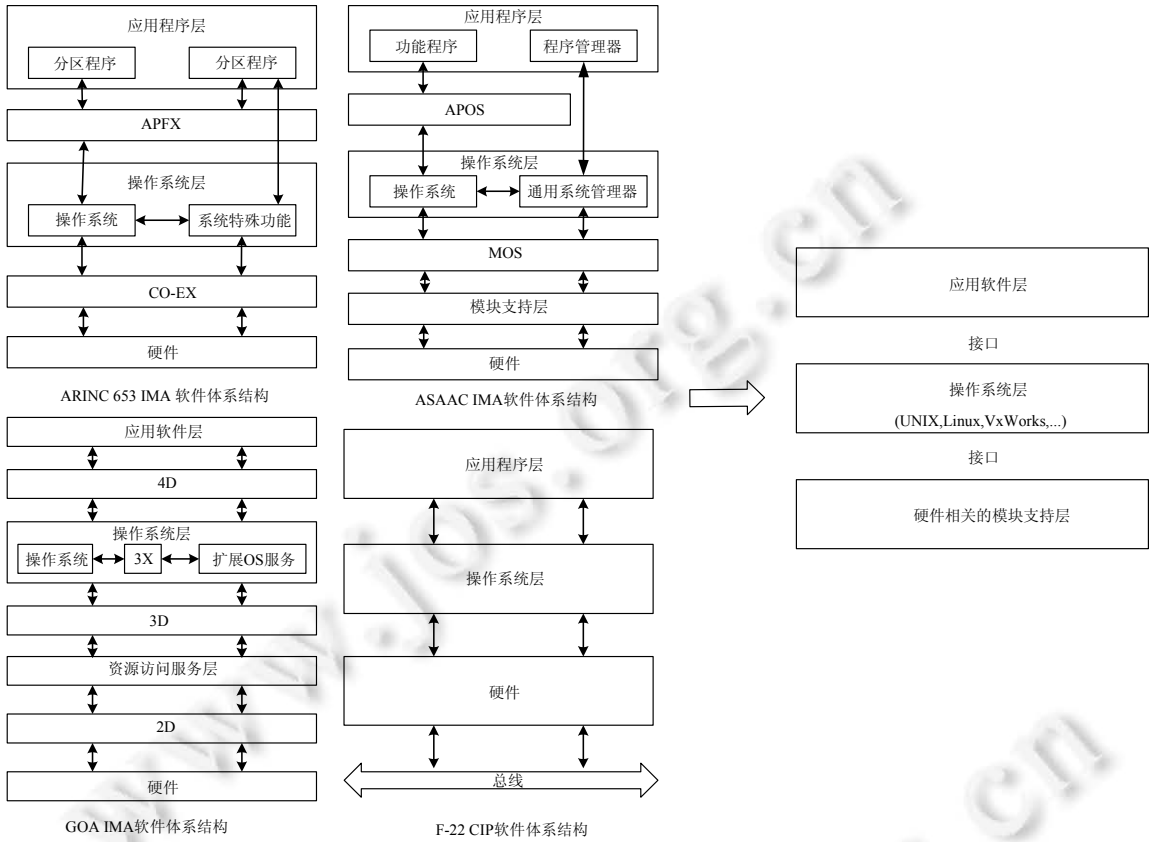
本文基于形式化方法讨论了航空电子系统自动化检测问题,定义了被检测系统的一种形式化模型——窗口树,该模型的行为可以表示被检测系统的系统级行为;定义了被检测系统中任务的状态转移图模型,该模型可以刻画被检测系统的系统级行为的正确性;基于窗口树模型和状态转移图模型,提出了一种航空电子系统的系统级综合检测方法.以往的安全攸关软件形式化方法更多地关注代码正确性分析,包括提出并使用高级抽象数学程序设计语言设计并实现安全攸关软件,适用于规模不大的嵌入式代码.本文提出的形式化方法试图分析大规模分布式安全攸关软件系统的安全检测问题,通过定义被检测系统的形式化模型和任务状态转移模型,关注安全攸关软件的系统级行为正确性分析和检测.

2 航空电子系统的形式化建模

航空电子系统是由各种机载信息采集设备(传感器/数据链)、信息处理设备、信息管理和显示控制设备以及相应软件组成的网络.发展至今,航空电子系统结构经历了分立式、联合式、综合式到高度综合式 4 个阶段.从航空电子系统的发展过程中可以看出,综合模块化航空电子系统(integrated modular avionics,简称 IMA)是航空电子系统的发展方向.

综合模块化航空电子系统本质上是一个高度开放的分布式实时系统,用于支持不同关键级别的航电任务程序.目前,航空、航天领域产生了 4 种典型的 IMA 软件体系结构,分别是 ARINC 653,ASAAC,GOA 以及 F-22 通用综合处理机(common integrated processor,简称 CIP)上的软件体系结构[20],如图 1(a)所示.

从这 4 种典型的 IMA 软件体系结构中可以看出,IMA 大体上可以抽象为由应用软件层、操作系统层和硬件相关的模块支持层这 3 层所组成的结构.其中每一层都相互独立,层与层之间通过定义的标准接口进行交互,如图 1(b)所示.



(a) 4 种典型的 IMA 软件体系结构 (b) 抽象软件体系结构

Fig.1 Software architecture of avionics systems

图 1 航空电子系统软件结构

2.1 航空电子系统的静态建模

如前文所述,航空电子系统是由多个分系统组成的,而每个分系统又是通过多个所属计算机软件配置项(简称 CSCI,是指满足最终使用功能的可进行单独配置管理的软件集合)的相互协同来完成相应的功能,并对外提供调用接口.本文将这些接口称为应用程序访问接口 API,并给出如下的定义:

定义 1(应用程序访问接口 API). $API=(api\_Name,I_{in},I_{out},B_{in},B_{out})$.其中,

(1) *api_Name* 表示接口的名称,作为每个 *api* 的唯一标识;

(2) $I_{in}=\{P_1,P_2,\ldots,P_m\}(m\geq 0)$表示接口 I 的输入接口,P_i 表示 I_{in} 的参数;

(3) $I_{out}=\{P_1,P_2,\ldots,P_n\}(n\geq 0)$表示接口 I 的输出接口,P_i 表示I_{out}参数;

(4) B_{in}表示输入接口I_{in}的约束集合;

(5) B_{out}表示输出接口I_{out}的约束集合.

例如,为一个网络资源的连接创建如下接口:

DWORD WNetAddConnection2(_In_LPNETRESOURCE lpNetResource,_In_LPCTSTR pPassword,
_In_LPCTSTR lpUsername,_In_DWORD dwFlags).

根据定义 1 中 API 的形式化定义,按顺序提取相应的参数,可以表示成

(WNetAddConnection2,{LPNETRESOURCE,LPCTSTR,LPCTSTR,DWORD},{DWORD},ε,ε).

航空电子系统中的软件配置项、分系统、总系统等,其运行对环境与资源都有一定的需求,如 CPU、网络

等.本文将这些需求定义为系统的性能指标.

定义 2(性能指标 PI). $PI=(r\_Name,\{(p,b)\})$.其中,

(1) r_Name 表示资源的标识;

(2) $(p.b)\in P\times B$,$P=\{p_1,p_2,p_3,\ldots,p_n\}(n>0)$表示资源的属性名集合,$B=\{b_1,b_2,b_3,\ldots,b_n\}(n>0)$表示资源属性取值的合集,$bi=[lower,upper]$,$lower$ 表示下界,$upper$ 表示上界.

例如,若一个分系统对硬件的要求是微处理器主频不小于 600MHz,内存不小于 512MB,存储容量不小于 512MB,读/写速度大于 4.5MB/s,根据定义 2 可以表示成

$\{(CPU,\{(\text{MHz},[600,\infty])\}),(Memory,\{(All,[512\text{MB},\infty])\}),(Disk,\{(capacity,[512\text{MB},\infty]),(iospeed,[4.5\text{MB/s},\infty])\})\}$.

定义 3(配置文件 CONF). $CONF=(Name^{conf},\{(a,v)\})$.其中,

(1) $Name^{conf}$ 表示文件的标识;

(2) $(a,v)\in A\times V$,集合 $A=\{a_1,a_2,\ldots,a_t\}(t>0)$,表示文件属性集合;集合 $V=\{v_1,v_2,\ldots,v_u\}(u>0)$,表示属性值的集合.

例如,大小为 2KB 的文件 *ini.conf*,是某分系统软件配置项的配置文件,它位于/*home*/*ES*/*RRM*/*conf*/下,根据定义 3 可以表示成

$(ini,\{(type,\text{“}conf\text{”}),(size,2\text{KB}),(location,\text{“}/home/ES/RRM/conf/\text{”}),(content,\{ID{:}101,IP{:}192.168.23.12,\ldots\})\})$.

软件配置项是由 *api*、支撑系统正常运行的性能指标以及配置文件组成的集合,因此通过上述的定义,可以将软件配置项 CSCI 定义为如下的四元组.

定义 4(软件配置项 CSCI). $CSCI=(Name^{csci},API,PI^{csci},CONF^{csci})$.其中,

(1) $Name^{csci}$ 表示软件配置项的名称标识;

(2) API 表示对外 *api* 的集合,$API=\{api_1,api_2,\ldots,api_m\}(m>0)$;

(3) PI 表示性能指标的集合,$PI^{csci}=\{pi_1,pi_2,\ldots,pi_n\}(n>0)$;

(4) $CONF^{csci}$ 表示软件配置项的配置文件集合,$CONF^{csci}=\{conf_1,conf_2,\ldots,conf_p\}(p>0)$.

如前文所述,航空电子系统由多个分系统组成,每个分系统又是由不同的软件配置项组成.由于软件配置项的集成,分系统包含 *csci* 集合的同时也具有独立的 *api*、性能指标以及配置文件,因此,可以将分系统 *SS* 定义为由 *csci* 集合以及独立的 *api* 集合、性能指标、配置文件集合组成的五元组.

定义 5(分系统 SS). $SS=(Name^{ss},CSCI,API,PI^{ss},CONF^{ss})$.其中,

(1) $Name^{ss}$ 表示分系统的名称标识;

(2) $CSCI$ 表示软件配置项 *csci* 的集合,$CSCI=\{csci_1,csci_2,\ldots,csci_m\}(m>0)$;

(3) API 表示 *api* 的集合,$API=\{api_1,api_2,\ldots,api_n\}(n>0)$;

(4) PI 表示性能指标的集合,$PI^{ss}=\{pi_1,pi_2,\ldots,pi_p\}(p>0)$.

(5) $CONF^{ss}$ 表示分系统的配置文件集合,$CONF^{ss}=\{conf_1,conf_2,\ldots,conf_q\}(q>0)$.

对于分系统来说,分系统是软件配置项的集成,它的性能指标来源于软件配置项,但也正是由于集成,它具有自己独立的性能要求,因此,可以认为分系统 *SS* 自身定义的指标集合 PI^{ss} 与 *csci* 的性能指标交集可以为空,也可以为非空,没有明确的包含关系.

类似地,可以如下地给出航空电子系统的定义:

定义 6(航空电子系统 ES). $ES=(Name^{ES},SS,API,PI^{ES},CONF^{ES})$,其中,

(1) $Name^{ES}$ 是航空电子系统的名称标识;

(2) SS 表示分系统 *ss* 的集合,$SS=\{ss_1,ss_2,\ldots,ss_m\}(m>0)$;

(3) API 表示 *api* 的集合,$API=\{api_1,api_2,\ldots,api_n\}(n>0)$;

(4) PI^{ES} 表示性能指标的集合,$PI^{ES}=\{pi_1,pi_2,\ldots,pi_p\}(p>0)$;

(5) $CONF^{ES}$ 表示航空电子系统的配置文件集合,$CONF^{ES}=\{conf_1,conf_2,\ldots,conf_q\}(q>0)$.

由此可见,若不考虑整个航空电子系统中任务的流转,那么整个航空电子系统可以表示成如图 2 所示的形式.

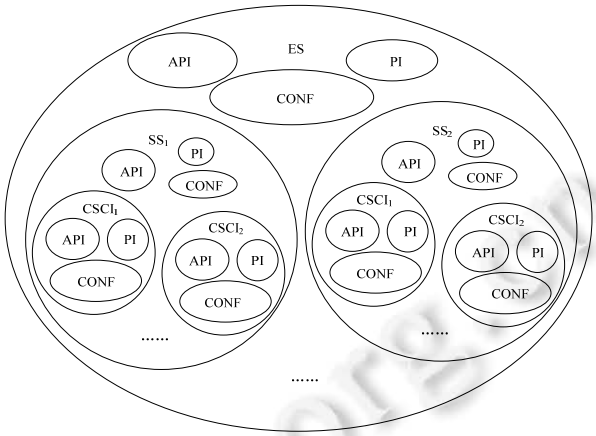


Fig.2 Formalized structure of avionics systems

图 2 航空电子系统形式化结构

2.2 航空电子系统的动态建模

2.2.1 航空电子系统窗口树建模

由于检测是对经测试后的航空电子系统做进一步验证,它不涉及对系统内部功能的检测,主要检测航空电子系统的执行环境是否能够保证系统的正常执行.因此,由于系统功能的不透明性,检测时要分析航空电子系统运行时的交互信息,而信息的交互是通过 *api* 调用来实现的.

在航空电子系统中,流程相当于 *api* 之间的有序流转,在所有的 *api* 中,由于功能的划分,会有一些关键性的 *api* 是作为某些主要功能的入口,而且可能存在多个并列的功能入口,这些 *api* 会根据输入的参数不同从而进入不同的流转,本文将并列的 *api* 集合定义为一个窗口,作为下一步操作的选择节点.如图 3 所示,其中的根节点就是一个窗口,它包含 3 个 *api*,通过这 3 个 *api* 的调用,会进入到下一个不同的窗口.

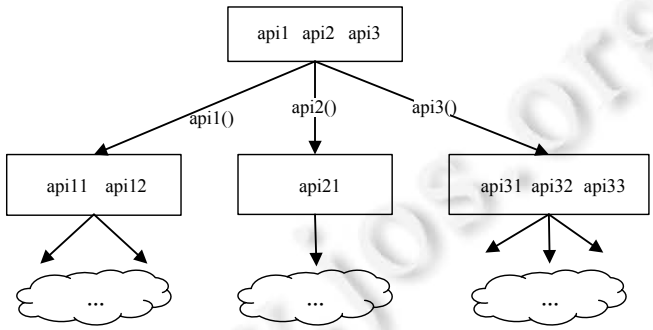


Fig.3 Generic window tree

图 3 通用窗口树

定义 7(窗口 W). $W=(API_t)$,其中,API_t 表示系统运行至 t 时间节点时可调用的 *api* 的集合,集合中 *api* 可为 0 个、1 个或者多个.

当从一个窗口流转到另一个窗口时,会调用多个 *api*,本文将对 API 的调用称为事件.事件是窗口流转的基本行为,给出如下定义:

定义 8(事件 E). *E*=(*Event_Name*,*Precondition*,*API*,*Parameter*),其中,

(1) *Event_Name* 表示该事件名称,唯一标识一个事件;

(2) *Precondition* 表示事件的前置条件,事件前置条件为一布尔表达式,当表达式值为真实时,该事件可以被触发;

(3) *API* 表示该事件触发时的操作集合,可以为空;

(4) *Parameter* 表示调用 *api* 时的实参.

一个或多个事件组成的有限序列称为事件序列,下文中用事件 e 指代一个事件序列 $e_1;e_2;\ldots;e_n$.

通常情况下,航空电子系统执行任务的过程可以看作是大量窗口的有序组成.从一个窗口经过一个或者多个事件流转到另一个窗口,在任务的流转中,由于参数的不同,而调用不同的 *API*,进而产生不同的流转方向,因此可以用一棵树来表示航空电子系统中任务的流转,本文将其称为窗口树.其中,窗口是窗口树的节点,事件则是窗口树的边.

定义 9(窗口树 WT). $WT=(W,w_0,E,R_{W,E})$,其中,

(1) W 是窗口的集合,窗口 $w\in W$ 称为一个节点;

(2) $w_0\in W$ 为初始窗口,即被测系统的启动初始窗口;

(3) E 为事件的集合,即被测系统的窗口之间流转的事件集合;

(4) $R_{W,E}\subseteq W\times W\times E$ 表示窗口间的流转关系,如果存在 $(w_i,w_j,e)\in R_{W,E}$,则表示窗口节点 w_i 与窗口节点 w_j 之间有父子关系,且 w_i 为父节点,e 为窗口 w_i 流转到窗口 w_j 的事件.

以民航航空电子系统中气象探测任务的流转过程为例,通过分析过程中窗口的流转关系来构建相应的窗口树.气象探测任务是飞机在飞行中用来对前方航路上的危险气象区域进行实时地探测,以便选择安全的航路来保障飞行的舒适和安全.其所有功能都是围绕雷达分系统进行的,主要的工作流程是启动雷达、设置雷达参数、扫描以及对扫描到的目标的判断与处理,如图 4 所示.

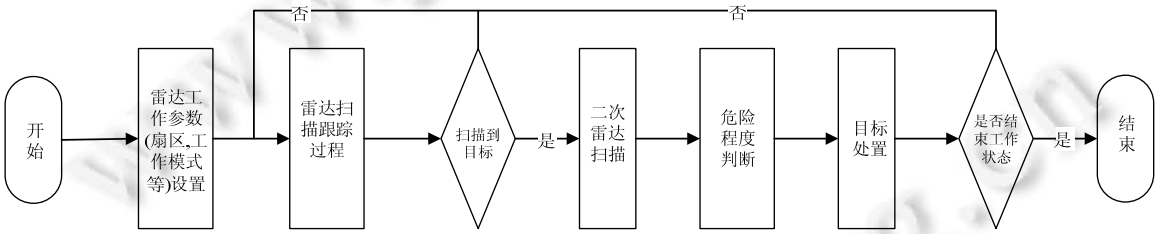


Fig.4 Meteorological observation mission process of avionics systems

图 4 航空电子系统气象探测任务流程

根据程序分片分析思想,可以将气象探测任务的工作过程归类为 4 个阶段:日常搜索阶段、系统响应阶段、气象研判阶段、应用处置阶段,如图 5 所示.

在日常搜索阶段,系统启动雷达后进行持续的扫描,主要功能由雷达分系统来完成,一旦雷达在系统设置的搜索扇区扫描到设置的目标(降水、湍流、风切变等),就进入到系统响应阶段,主要功能由雷达分系统和气象识别分系统完成,此时系统根据相关的计算模型,从一次、二次雷达搜索信息中计算检测到的目标点迹和航迹,将一次点迹、一次航迹、二次点迹、二次航迹和综合航迹发送给指控分系统,接下来系统进入气象研判阶段,根据搜集到的情报进行综合,判断目标的威胁程度,主要功能由指控分系统完成,在指控分系统将研判的结果发送至显控台后,就进入到应用处置阶段,这时操作员会根据系统计算出的目标威胁程度进行相应的处置(例如,将目标发送给地面指挥中心、选取对应的方案等).这 4 个阶段具有时序关系,分工明确,且每个阶段都有不同的分系统参与完成.图 6 所示的是构建的气象探测任务窗口树.

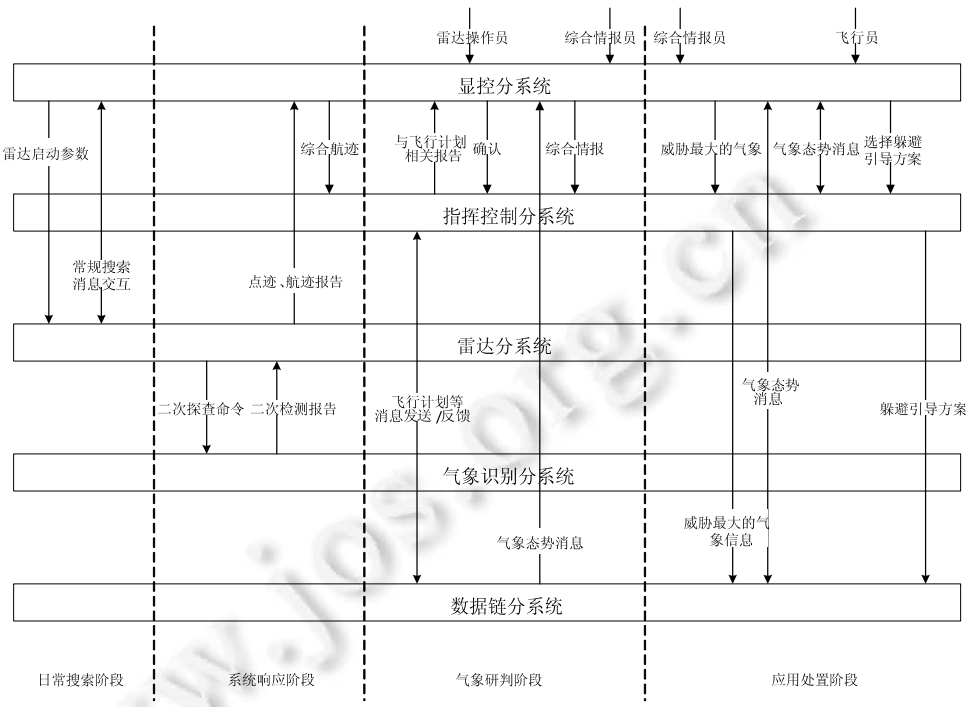


Fig.5 Meteorological observation mission phase analysis diagram of avionics systems

图 5 航空电子系统气象探测任务阶段分析图

图 6 中,日常搜索阶段包括雷达工作参数设置(搜索扇区,任务剖面,工作模式等),这里以常用的 4 种工作模式为例,即合成孔径雷达(SAR)模式、信标模式、气象模式、扫描跟踪模式(TWS).其中,SAR、信标和气象工作模式主要工作内容都在这一阶段完成,且主要是与显控进行数据的交互;在扫描跟踪工作模式,当没有搜索到目标的时候,仅仅是与显控分系统进行常规的数据交互,一旦雷达搜索到目标,任务流程就会转入系统响应阶段.

在系统响应阶段中,首先,雷达会将扫描到的目标信息组织成命令发送给二次雷达进行消息确认和航迹点迹跟踪;之后,二次雷达将进一步搜集到的目标发送给雷达分系统,雷达分系统将两次收集到的一次点迹、二次点迹、一次航迹、二次航迹进行计算,同时形成综合航迹;最后雷达分系统将所有数据打包发送给指控分系统.

在接下来的气象研判阶段中,指控分系统首先将判断数据与飞行计划的相关性,若与飞行计划无关,则判断该目标无威胁,并将结果直接推送显控分系统显示给操作人员;若与飞行计划相关,就进入到下一步的威胁等级判定,即进入图 6 所示的情报综合窗口,收集所有能够获得的情报(反射的强弱、接收回波信号频率等)进行目标威胁程度判定,然后将判定结果推送显示控制分系统,供操作员进行下一步的操作.

在应用处置阶段,操作员根据目标的威胁程度进行航迹选择,发送给地面中心或其他相关单位,以形成统一的气象态势图,同时接收其他单位送来的气象态势报告,系统综合给出可选处置方案,操作员通过研判后选取合适的躲避引导方案,同时将所选方案发送给相关的单位以更新整个气象态势图.

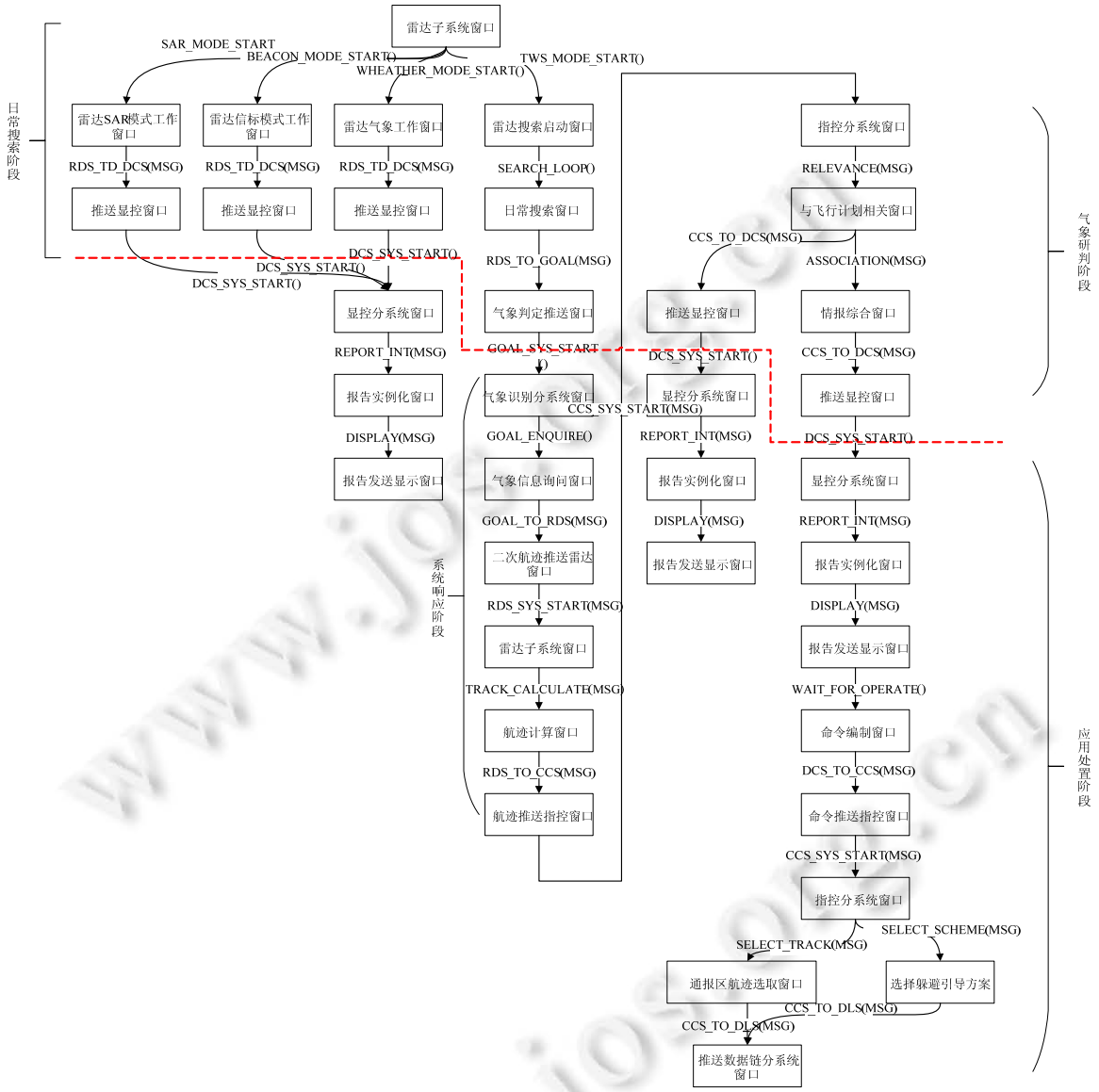


Fig.6 Meteorological observation mission window tree of avionics systems

图 6 航空电子系统气象探测任务窗口树

2.2.2 航空电子系统执行过程状态图模型

航空电子系统的执行过程状态图是对航空电子系统完成某一任务的整个过程进行分析,刻画过程中的正确行为.系统的状态一般用系统行为主体的状态来表示,主体指的是事务的主要部分,也指实践活动和认识活动的承担者.因此给出如下定义:

定义 10(行为主体 Subject). 行为主体(Subject)指的是系统运行过程中的所有计算、存储、传输过程的承担者.

定义 11(客体 Object). 客体(Object)指的是在系统运行过程中被主体操作的所有对象.

定义 12(主体行为 Behavior). 主体行为(Behavior)指的是在系统运行过程中主体对客体的操作行为.

在软件系统中,用由主体行为过程抽象出来的的行为状态指代整个系统的状态s.通常在触发某个事件e后,

任务的状态会发生改变,因此,可以将状态转变看成一个函数$\delta: s\times E\to s$.每次触发一个事件 e 后,窗口状态由 s_i 转换成 s_j,即 $s_j=\delta(s_i,e)$,其中,s_i,s_j 表示窗口状态.

定义 13(事件 e 可执行). 对于事件 e 和状态 s,若事件 e 的前置条件成立,则称事件 e 在状态 s 下可执行.

定义 14(有效后继事件). 对于事件 e_i 和 e_j,若对状态 $s_j=\delta(s_i,e_j)$,e_j 的前置条件成立,即 e_j 可执行,则称 e_j 是 e_i 的有效后继事件.

定义 15(有效事件). 事件序列 $e_1;e_2;\ldots;e_n$,称为有效事件序列,当且仅当 e_{i+1} 为 e_i 的有效后继事件,其中,$1\leqslant i<n$.若事件 e 是一个有效事件序列 $e_1;e_2;\ldots;e_n$,则称 e 为有效事件.

在进一步给出定义之前,给出假设:检测用例 $case=e_1;e_2;\ldots;e_n$,对于状态 s_0,e_1 的前置条件成立,其中,s_0 是被测系统的启动初始状态,并且 $e_1;e_2;\ldots;e_n$ 为有效事件序列.

定义 16(检测). 给定检测用例 $e_1;e_2;\ldots;e_n$,若当前状态为 s_0,$s_1=\delta(s_0,e_1),\ldots,s_n=\delta(s_{n-1},e_n)$,则称 s_0 为检测用例 $e_1;e_2;\ldots;e_n$ 的输入,s_n 为检测用例 $e_1;e_2;\ldots;e_n$ 的输出(检测结果),可以表示为 $s_n=\delta(s_0,e_1;e_2;\ldots;e_n)$,也可以记为 $s_0{:}e_1;e_2;\ldots;e_ns_n$.

定义 17(状态序列). 在用例执行中,若事件 e_i 的执行产生中间状态 s_i,则称由一个或多个这样的中间状态所组成的有限序列 $s_1;s_2;\ldots;s_n(n\geqslant 1)$为一个状态序列.

根据上述定义,本文给出状态流转图的定义.

定义 18(状态流转图 STG). $STG=(S,S_0,E,R_{S,E},F)$,其中,

(1) S 表示状态流转图中状态的有限集合;

(2) S_0 表示开始状态(也叫做初始状态),$S_0\in S$;

(3) E 是事件或者事件序列集合(非空有限集合);

(4) $R_{S,E}$ 是状态转移函数集合:$\delta: S\times E\to S$;

(5) F 是最终状态的集合,S 的子集(可能为空).

接下来同样以航空电子系统中气象探测任务的流转过程为例,通过抽象系统任务,定义系统任务状态来构建航空电子系统气象探测任务的状态流转图.

任务是气象探测任务中的主体,由于气象探测任务具有强时间约束和时序的特点,这在任务运行过程中表现为数据在固定有序的流程中进行流转,若将流转的数据统称为报告(report),则可以将任务及任务状态定义为

任务::=(流程,报告);任务状态::=(流程状态,报告状态).

流程是系统基于时间顺序的消息流转逻辑以及计算逻辑实例化后(任务数据正式加入流转)的具体流程,因此可以将流程定义为

流程::=事件序列.

于是,对于流程的操作就可以看作是对事件或事件序列的触发.结合前文的分析,对于气象探测任务,它的流程操作可以表示为

流程操作::=雷达工作参数设置|日常搜索|气象判定|雷达航迹计算|指控情报综合|
显控显示|显控命令输入|指控选取航迹|指控选取躲避引导方案.

若用流程到达的分系统标识来表示任务中流程的状态,则气象探测任务中的流程状态集可以表示为

流程状态集::={null,显控,雷达,目标,指控,数据链,结束}.

类似地,可以给出气象探测任务中的报告、报告操作、报告状态集以及任务状态集的表示:

报告::=(一次雷达景,二次雷达景,一次点迹,一次航迹,二次点迹,二次航迹,综合航迹,
系统航迹,人工命令,躲避引导方案),
报告操作::=一次雷达景编制|二次雷达景编制|一次航迹计算|二次航迹计算|综合航迹计算|
系统航迹计算|人工命令传达|系统航迹选取|躲避引导方案制定,
报告状态集::={null,拟制,实例化,编辑,传输},
任务状态集::={(显控,null),(雷达,拟制),(气象,编辑),(雷达,编辑),(指控,编辑),

(显控,实例化),(指控,拟制),(数据链,传输),(结束,null)}.

基于前文中对航空电子系统的阶段行为状态、事件的定义以及任务流转阶段分析,可以构建出航空电子系统气象探测任务的流转状态图,如图 7 所示.

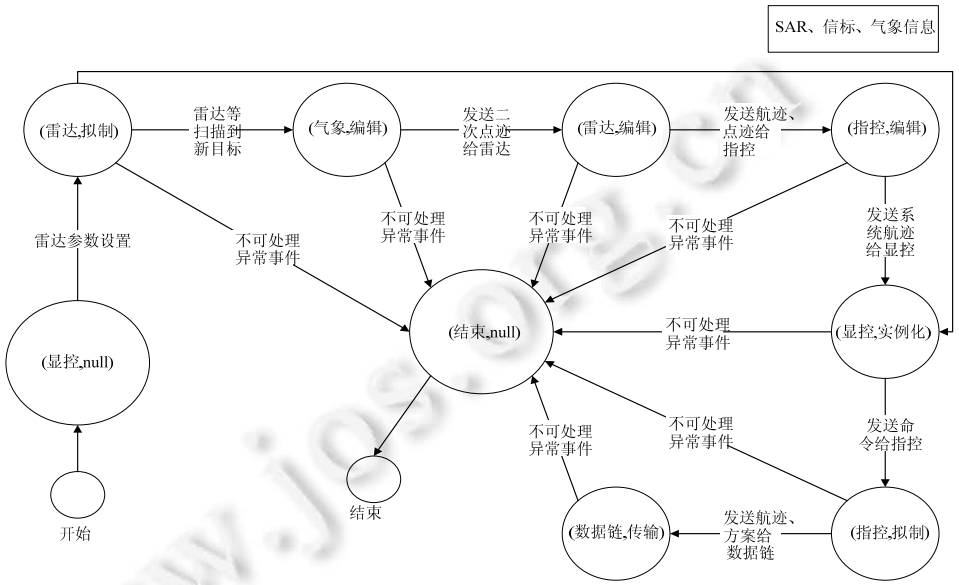


Fig.7 Meteorological observation mission transfer state diagram of avionics systems

图 7 航空电子系统气象探测任务流转状态图

航空电子系统的所有人机交互操作都是从显控台开始的,因此初始状态为(显控,null);当发生雷达参数设置事件后,系统进入(雷达,拟制状态),表示即将开始任务的流转,报告初拟制;当雷达搜索到目标后,发送二次雷达景参数给气象识别分系统,系统进入(气象,编辑)状态;气象识别分系统收集到更多的目标信息后,将信息发送给雷达进行一次航迹、二次航迹和综合航迹的计算,系统进入(雷达,编辑)状态;雷达将综合航迹等数据发送给指控,对目标的信息进一步处理,系统进入了(指控,编辑)状态;指控最终将航空电子系统扫描到的信息发送到显控台供操作人员查看操作,系统处于(显控,实例化)状态;操作人员通过查看目标信息,做出相应的操作,并将命令发送给指控,系统进入(指控,拟制)状态,表示重新进入下一报告的拟制(雷达搜索目标信息整理报告完成);指控对命令做出响应,将选择的航迹和方案发送给数据链,由数据链发送给相应的操作人员,此时的系统处于(数据链,传输)状态,表示一次的搜索目标信息处理流程完成.若过程中出现一些不符合流程的事件或状态,则到达(结束,null)状态.

图 4 中描述的航空电子系统的任务的运行流程可以映射到如图 5 所示的 4 个阶段中,而贯穿于这 4 个阶段的其中一次任务流转可以在图 6 所示的窗口树中找到与之对应的一条窗口树路径.这就表明了被检测系统与其形式化模型行为的一致性.图 4 和图 5 是从系统级的角度来描述航空电子系统的行为,图 6 中给出的窗口树是对航空电子系统行为的模拟,其中窗口树中的一条路径就是对航空电子系统的一次运行行为的模拟.从前文的示例分析中可以看出,本文所构建的形式化模型从系统级的角度来看是能够较为完备地模拟实际被测系统.

3 形式化的系统级综合检测方法

针对前文构建的航空电子系统静态模型和动态模型,本文将对航空电子系统的检测分为两类:系统配置级(静态)检测和系统运行时(动态)检测.

系统配置级(静态)检测主要针对系统配置环境出现错误而导致的一系列问题,包括以下几类:

(1) 一致性.系统模块相关安装文件、执行文件等被篡改,或者模块更新时版本号与版本不匹配引起的一致

性问题.

(2) 适配性.系统模块和其他相关模块、系统环境不适配,或者接口错误、部署平台信息错误等引起的适配性问题.

系统运行时(动态)检测主要针对系统运行时由于模块失效、路径错误、数据出错、接口失效等导致任务失败的问题.按照问题发生的系统等级分为以下几类:

(1) 软件模块可用性.实现某功能的功能模块的动态运行环境可能受到相关模块和系统运行时的影响,如果该功能模块的动态运行环境不能满足它对运行条件的要求,则可能导致该模块不能实现预期效果.

(2) 分系统可用性问题.分系统在其动态运行环境支持下执行,该环境可能影响其执行路径(模块内程序执行路径、模块间数据流转路径等),若路径异常,则可能导致分系统部分功能不可用;支撑分系统功能的若干硬件单元可能存在部分失效,也将导致分系统部分功能不可用.

(3) 系统可用性问题.系统级功能通常由若干分系统协同实现,如果分系统协同失效(如交互数据异常、接口异常、时序异常等),则可能影响该功能正常实现.

航空电子系统的综合检测可以看作是对系统静态问题和动态问题进行检测和排除,验证航空电子系统运行环境的正确性和系统运行时的消息流转时序正确性.

3.1 系统配置级(静态)检测方法

静态检测是为动态检测做准备,为动态检测提供一个可信的静态环境,以确保动态检测的结果的有效性与正确性.

对于航空电子系统而言,环境提供的是各类资源,包括基础支撑和中间件等,系统是通过各种 *api* 调用来使用资源.接下来给出资源和环境的形式化定义.

定义 19(系统资源 R). $R=\{api_1,api_2,\ldots,api_n\}$.

定义 20(资源状态 R-s). $R\text{-}s=(r\_Name,\{(p,v)\},F)$,其中,

(1) *r_Name* 表示资源的标识;

(2) $(p,v)\in P\times V$,$P=\{p_1,p_2,p_3,\ldots,p_n\}$表示资源的属性集合,$V=\{v_1,v_2,v_3,\ldots,v_m\}$表示资源的属性值集合;

(3) *F* 表示资源对外提供的 *API* 集合,$F=\{api_1,api_2,\ldots,api_s\}$.

例如,可以如下表示内存资源的随机的状态:

(内存资源,{(总量,4.0G),(可用,3.9G),(剩余,22%),(已用,78%)},{(*new*,{*size*},{*return*},{*size_t*},{{*void**,*NULL*}}),(*malloc*,{*size*},{*return*},{*size_t*},{{*void**,*NULL*}})},(*relloc*,{*mem_address*,*newsize*},{*return*},{*void**,*unsigned int*},{{*void**,*NULL*}}),(*free*,…),…)

定义 21(系统环境 OS). 系统环境 OS 是由多个资源组成的,因此可以表示为 $OS=\{r_1,r_2,r_3,\ldots,r_n\}$,$r_i\in R$.

定义 22(系统环境状态 OS-S). 系统环境的状态 OS-S 可以表示为组成系统的所有资源的状态元组,即 $OS\text{-}S=\{r\text{-}s_1,r\text{-}s_2,r\text{-}s_3,\ldots,r\text{-}s_n\}$,$r\text{-}s_i\in R\text{-}s$.

本文中所提到的环境,如不做特殊说明,则表示的是操作系统环境,不涉及中间件、数据库等其他资源.操作系统从资源的层面由硬件资源和软件资源组成,系统环境状态指的是航空电子系统所在软件环境的资源状态,在特定的操作系统环境中,资源可以通过以下几个方面来量化:操作系统版本、CPU 状态、内存状态、磁盘状态、数据库状态以及各网络接口状态等.

3.1.1 配置文件一致性检测

定义 23(配置文件的一致性). 对于两个配置文件 $conf_1$ 和 $conf_2$,$conf_1=(Name_1,\{(a_i,v_i)\})$,$conf_2=(Name_2,\{(a_j,v_j)\})$,其中$(a_i,v_i)\in A_1\times V_1$,$(a_j,v_j)\in A_2\times V_2$,若同时满足以下条件,则认为软件配置文件 $conf_1$ 和 $conf_2$ 是一致的,表示为 $conf_1==conf_2$;否则就认为两者是不一致的,表示为 $conf_1!=conf_2$:

(1) $Name_1=Name_2$;

(2) $A_1=A_2$;

(3) $V_1=V_2$;

(4) $\forall(a_i,v_i)\in A_1\times V_1,\exists(a_j,v_j)\in A_2\times V_2$ 使得 $a_i=a_j,v_i=v_j$.

需要注意的是,由于对于不同配置文件,关注的属性有所不同,那么实际应用时进行比对的属性集合也就有所不同.

本文中配置文件一致性检测针对的是分系统级的配置文件.通过提取某一分系统的软件配置文件信息,与既定标准进行比对,查看是否符合配置文件一致性的规定,目的是保证当前的软件配置项文件与需求是一致的.图 8 所示的是配置文件一致性检测流程.

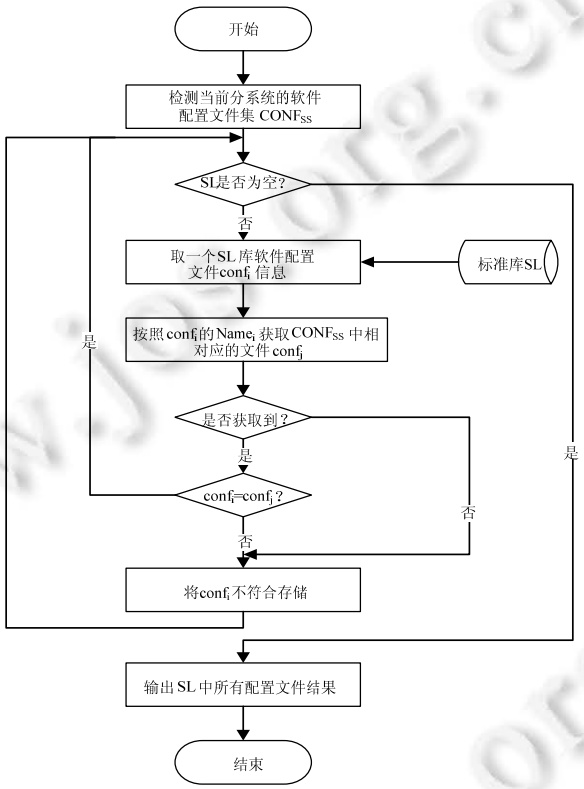


Fig.8 Configuration files conformance testing process

图 8 配置文件一致性检测流程

图中所示的标准库 SL,是需要预先存储的一个用于比对的库,该库包含一个完整的、标准的、指定的软件配置文件信息.对于每一个分系统 ss 的软件配置来说,系统是否具有一致性,取决于实际系统中存在的所有配置文件信息与相对应的标准库 SL 中的信息比对结果.算法 1 是配置文件一致性检测算法.

算法 1. 配置文件一致性检测算法.

```
Input: CONFss: set; SL: set.
Output: CRT: set.
for (i=1; i≤|SL|; i++)
{
    for (j=1; j<|CONF^ss|; j++)
    {
        if (conf^SS(j)==conf^SL(i))
        {
        conf^SL(i).flag=true;
```

```
                }
            }
        }
    Display(SL);
```

3.1.2 系统环境状态适配性检测

定义 24(状态适配性). 对于资源 r 来说,若其当前状态 $r\text{-}s=(r\_Name,\{(p_s,v)\},F),p_s\in P_s,v\in V$,以及资源对应的指标 $p_i=(r\_Name,\{(p_{pi},b)\}),p_{pi}\in P_{pi},b\in B$,若资源状态 $r\text{-}s$ 满足如下条件,则认为资源 r 当前状态 $r\text{-}s$ 满足指标 p_i,记为 $r\text{-}s◎pi$:

(1) $r\_Name=r\_Name$;

(2) $P_{pi}\subseteq P_s$;

(3) $V\Phi B$(这里,$V\Phi B$ 表示 V 中的值介于 B 中每个指标的量值之间).

例如,若某航空电子系统 es 的性能指标集合为

PI^{es}={(*CPU*,{(MHz,[600,∞])}),(*Memory*,{(*All*,[512MB,∞])}),(*Disk*,{(*capacity*,[512MB,∞]),(*iospeed*,[4.5MB/s,∞])})},

而当前内存资源状态为

(*Memory*,{(*All*,4.0G),(*Usability*,3.9G),(*Remain*,22%),(*Used*,78%)},{(*new*,{*size*},{*return*},{*size_t*},{{*void**,*NULL*}}),(*malloc*,{*size*},{*return*},{*size_t*},{{*void**,*NULL*}})},(*relloc*,{*mem_address*,*newsize*},{*return*},{*void**,*unsigned int*},{{*void**,*NULL*}}),(*free*,…),…),

则表示对于内存这个资源来说,当前系统环境满足航空电子系统的系统设计需求,即 $Memory◎pi2$.

状态适配性检测针对的是分系统所在计算机系统环境的资源需求,检测的目的是为了让系统环境达到系统所需要的性能指标.一般将性能指标用一定格式(一般是 XML)的文件进行描述,然后逐个读取指标项对系统检测,直到整个指标集合为空为止,状态适配性检测流程如图 9 所示.

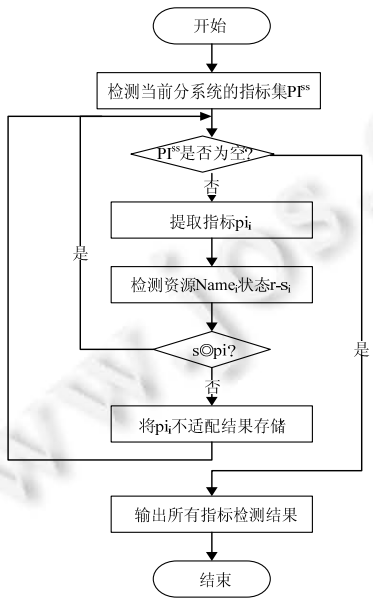


Fig.9 State suitability testing process

图 9 状态适配性检测流程

根据前文定义的系统环境和指标,给出状态适配性检测算法.

算法 2. 系统环境状态适配性检测算法.

Input: PI^{SS}: *set*, *OS-S*:*set*.

Output: *SRT*:*map*. //(*r-Name*,*satus*,*flag*)元组组成集合,*flag* 表示是否达标

```
for (i=1; i<|PI^SS|; i++)
{
    for (k=1; k<|PI^SS|; k++)
    {
        get r-s(j) from OS-S by r-Name;
        if (r-s(j)◎pi^SS(k))
        {
            put (r-s(j),true) into SRT;
        }
        else
        {
            put (r-s(j),false) into SRT;
        }
    }
}
Display(SRT);
```

3.2 系统运行时(动态)检测方法

动态检测是在航空电子系统运行状态下进行的检测.本文先将航空电子系统的运行过程转化成状态机,再通过状态机来进行运行时的状态检测,并将获取系统运行时的状态与构建的航空电子系统执行过程状态图进行对比,以判断每次运行结果的正确性.

3.2.1 窗口树检测用例的生成

在航空电子系统窗口树建模中,窗口与窗口之间的连接为 *api* 调用事件序列,参考基于路径的测试覆盖准则,针对窗口树的检测应偏重对事件序列覆盖的特点,本文提出了窗口树检测的覆盖准则.对于一个窗口树,它的检测用例集 T 应该满足如下准则:

(1) 节点覆盖准则:检测用例集 T 满足节点覆盖准则,当且仅当窗口树中所有的窗口 w,至少存在一个检测用例 $t\in T$,包含窗口.

(2) 边覆盖准则:检测用例集 T 满足边覆盖准则,当且仅当对于窗口树中所有的边 $r\in R$,则至少存在一个检测用例 $t\in T$ 包含 r,即窗口树中的每两个节点间的树枝至少被检测用例覆盖 1 次.

鉴于此,本文采用深度优先遍历方法来生成检测用例.主要思想是:访问窗口节点 w,将该节点对应的事件序列加入到一个用例中,若该节点是叶节点,则将此时的用例加入到用例集,并设置该节点的状态为“已访问”;若该节点为父节点,则从节点 w 未被访问的子节点中选取一个顶点节点 w_1,从 w_1 出发进行深度优先遍历.算法 2 给出窗口树检测用例的生成算法.

算法 3. 窗口树检测用例生成算法.

Input: *root*:*Window_Node**,*case*:*Case_Seq*.

Output: *Case_Set*:*vector*.

```
Create_Caseset (Window_Node* root,Case_Seq case)
{
    append root into case;
    if (root→child_num==0)
    {
```

```
            add case into Case_Set;
            root→visited=true;
            return;
        }
        i=0;
        while (i<child_num && !root→child[i] →visited)
        {
            Create_Case(root→child[i],case);
            i++;
        }
    }
```

接下来根据航空电子系统的窗口树中的窗口节点来设置检查点.检查点是用来刻画软件行为轨迹的监测点,是由系统任务执行的指标属性的值构成的元组,一般设置于每个窗口与其父窗口之间的边所对应的 *api* 调用序列的功能结束位置.在检查点处输出系统运行到该检查点时候的系统的状态,可以是消息数据、上下文、时间戳、内存占用率、CPU 占用率等信息.当检测用例集执行完成后,会得到一个行为轨迹.

3.2.2 动态检测过程

前文所构建的航空电子系统执行过程状态图是无法被计算机直接接受的,因此需要先将其进行符号化,即转换成可以进行状态验证的.转换过程限于篇幅,在此不再赘述.

动态检测可以看作是在航空电子系统运行过程中,判断系统的行为轨迹是否可以被源自系统设计的有限状态自动机所接受.因此在动态检测之前需要先根据航空电子系统的设计,抽取任务主体,进行分阶段处理,得到阶段分析图,进而抽取关键功能以及相关 *api* 组成窗口和窗口对应的事件序列(*api* 调用序列),构造窗口树,生成以事件流为基础的检测用例集并设置行为轨迹检查点;同时构造以任务为主体的状态流转图,并将其转换为计算机可接受识别的有限状态自动机.如图 10 所示.

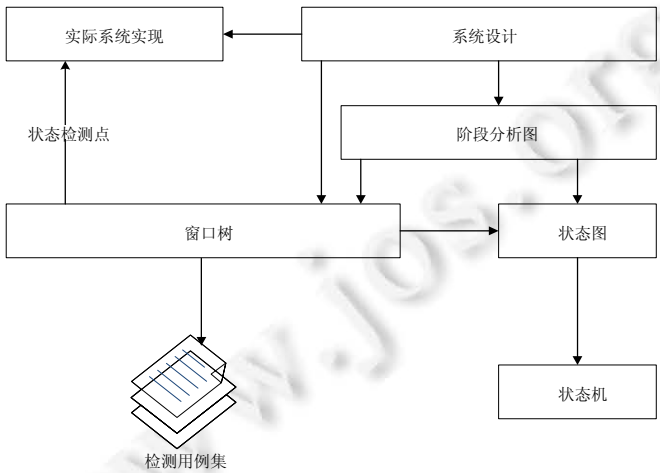


Fig.10 Preparation process of dynamic test

图 10 动态检测准备过程

接下来进入动态检测流程(如图 11 所示),根据窗口树设置检查点,为检测用例设置参数,在系统上运行检测用例,得到系统的行为轨迹,将系统的行为轨迹符号化为有限状态自动机可识别的输入符号序列Σ^+.将符号序列Σ^+输入有限状态自动机中运行,若能得到输出,则认为该用例得到的系统行为轨迹能被源自系统设计的有限状态自动机所接受;否则就认为该用例得到的系统行为轨迹不能被源自系统设计的有限状态自动机所接受.循环执行检测用例集中的每个用例,直到检测用例集中的所有用例都被执行了至少 1 次为止.

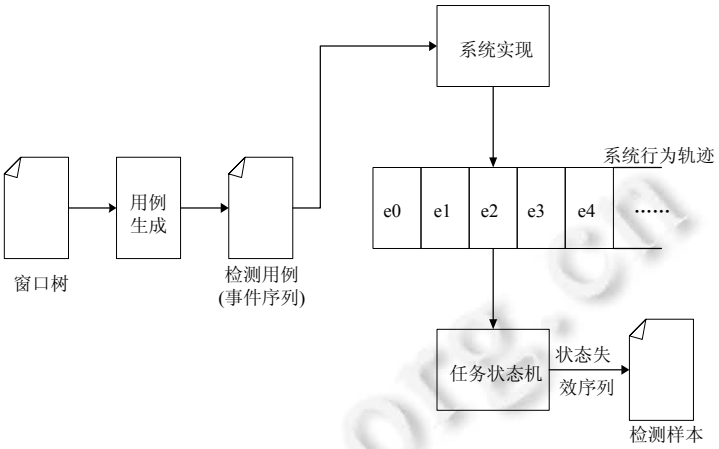


Fig.11 Dynamic testing process

图 11 动态检测流程

4 系统的实现与验证

4.1 实 现

为了验证本文所提出的系统级检测方法,本文设计并实现了一个综合检测系统.综合检测系统由两部分组成:综合检测控制系统和检测中间件.如图 12 所示.

综合检测控制系统分布在独立的计算机系统上,完成主要的检测功能,包含的功能模块有静态检测模块、动态检测模块、消息监听模块和显示模块.静态检测模块负责实现配置文件一致性检测和状态适配性检测;动态检测模块负责判定由检测中间件传过来的输入是否能被接收以及对最后结果的统计分析;消息监听模块则负责消息的接收和转发.

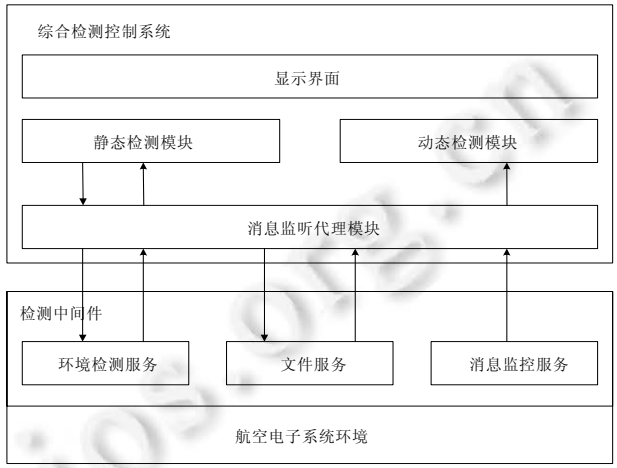
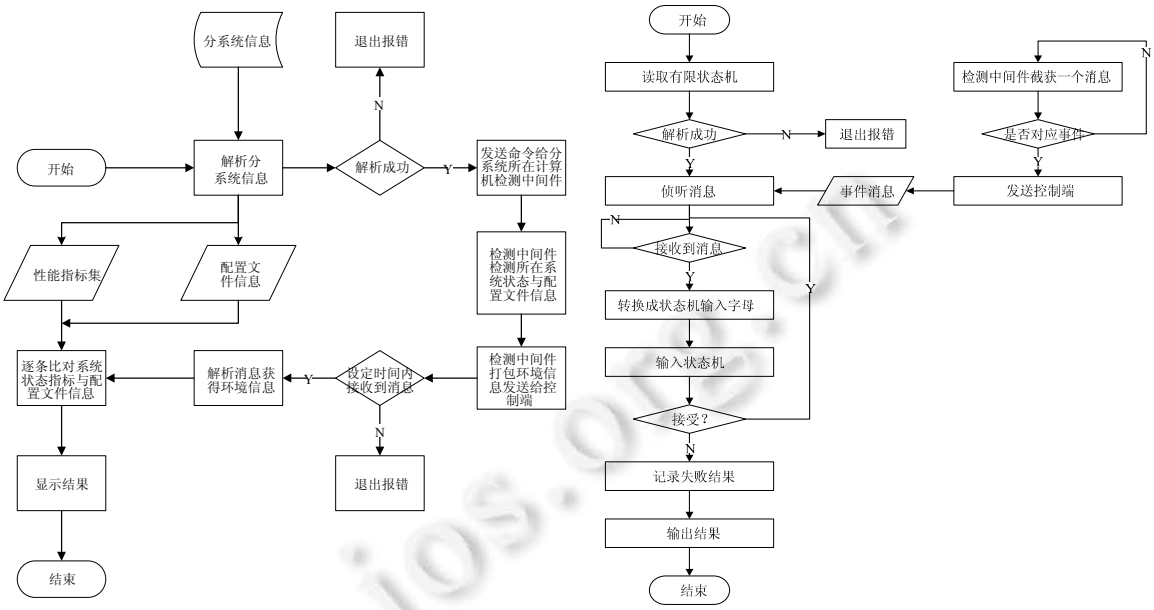


Fig.12 Structure of integrated testing system

图 12 综合检测系统结构

检测中间件部署在航空电子系统所在的计算机环境上,负责收集系统的环境信息、软件配置文件以及系统流转过程中的系统消息.图 12 所示的环境检测服务是为了响应从综合检测控制系统发来的收集环境信息的命令,将收集到的航空电子系统所在环境的操作系统版本、CPU、内存、磁盘等性能指标需求信息传递给综合检测控制系统;文件服务则负责收集综合检测控制系统需要的软件配置文件信息,通常来说,每个分系统都会有支撑运行的系统软件配置文件,在状态适配性检测之前将获取的配置文件与标准配置文件版本进行比对,完成配置文件一致性检测.消息监控服务则用来监控航空电子系统中各个分系统、软件配置项之间的消息流转,并将过滤后的消息发送给综合检测控制系统.

在系统静态检测时,综合检测控制系统向检测中间件发送提取系统环境所需检测的信息(包括操作系统版本、内存、磁盘、CPU 等)命令,检测中间件收到命令后,将收集的环境信息打包通过 UDP 协议发送给综合检测控制系统,由消息监听模块接收并转发,静态检测模块提取性能指标信息文件、配置文件信息等,与综合检测控制系统预置的指标和标准配置文件信息进行比对,得到静态检测结果,如图 13(a)所示.



(a) 静态检测模块工作流程 (b) 动态检测模块工作流程

Fig.13 Testing process of integrated testing system

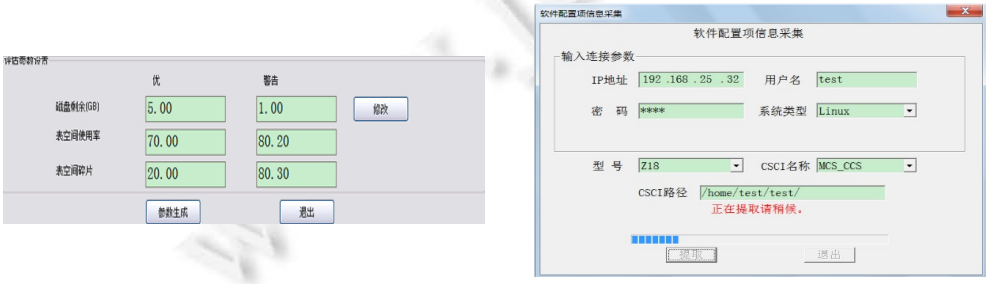
图 13 综合检测系统检测流程

静态检测完成后,若检测结果无误,则进入系统的动态检测.动态检测模块的核心是系统有限状态自动机.在检测开始阶段,将航空电子系统抽象出的状态图转换成有限状态自动机,在航空电子系统中运行根据窗口树生成的用例,检测中间件会检测用例运行过程中所有分系统以及软件配置项之间的消息过程,根据预先设置好的消息过滤规则,选取与系统行为轨迹相关的消息发送给动态检测模块,动态检测模块将消息转换为机器可识别的输入,判断其是否能被有限状态自动机所接受,如图 13(b)所示.

4.2 验 证

(1) 检测中间件

图 14 为原型系统中检测中间件的部分运行效果截图.图 14(a)为参数配置图,配置环境信息评估参数,当环境信息值越界时,不需要检测就发出警告信息.图 14(b)为软件配置项信息采集界面,采集的信息包括连接信息、型号、名称、路径等.



(a) 参数配置图 (b) 软件配置项信息采集

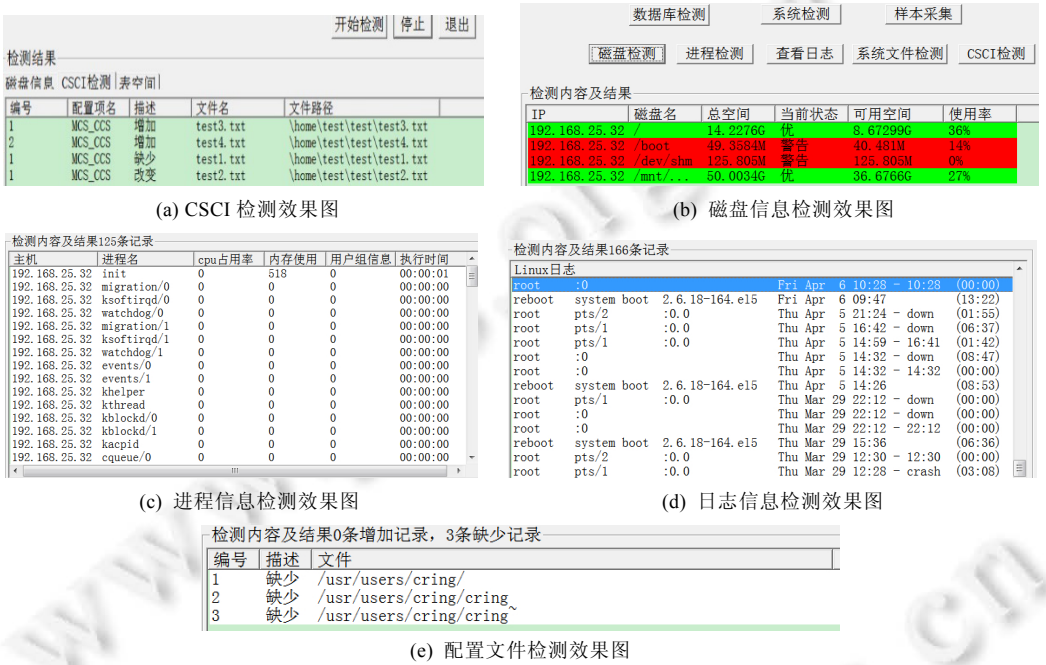
Fig.14 Part of the testing middleware operating results screenshot

图 14 检测中间件的部分运行效果截图

(2) 静态检测

图 15 为原型系统中综合检测控制系统中静态检测模块的部分运行效果截图.其中,图 15(a)显示的是将所

采集的软件配置项特征信息与标准软件配置项特征信息进行比对后的结果,结果清晰地显示了 3 个配置文件与标准配置文件相比对,其不同之处的详细信息;图 15(b)清晰地显示了磁盘检测的内容和结果;图 15(c)为进程检测效果图,用来显示被检测系统某次开启的所有进程的相关信息;图 15(d)为日志文件信息检测功能,是用来获取被测系统上所有日志和系统文件的信息,主要包括记录警告和错误的日志信息;图 15(e)为配置文件检测,用来分析是否缺少必要的文件内容.



(a) CSCI 检测效果图 (b) 磁盘信息检测效果图

(c) 进程信息检测效果图 (d) 日志信息检测效果图

(e) 配置文件检测效果图

Fig.15 Part operating results screenshot of the static testing module in integrated testing system

图 15 综合检测系统中静态检测模块的部分运行效果截图

(3) 动态检测

动态检测是针对不同的任务进行的,对任务处理过程进行状态性的检测,在检测前通过文件导入状态机并进行相关的配置,在状态匹配过程中选用相应的任务进行检测,图 16 是航空电子系统中气象探测任务的动态检测效果图.

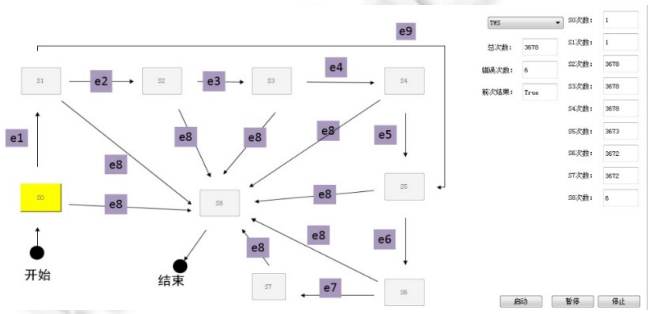


Fig.16 Dynamic testing module operating results screenshot of meteorological observation mission in avionics systems

图 16 航空电子系统中气象探测任务的动态检测效果截图

(4) 系统运行测试

我们在航空电子系统模拟仿真平台上,通过综合检测系统分别运行了 5 组不同检测用例,运行结果见表 1.

Table 1 Test results of integrated testing system

表 1 综合检测系统运行测试结果

检测用例组	检测用例数	执行用例数	检测通过数	用例通过率(%)	用时(ms)	静态存储余量平均值(%)	CPU 平均占用率(%)
组 1	252	252	238	94.44	34 234	87.67	66.33
组 2	169	169	154	91.12	23 017	93.46	70.28
组 3	156	156	144	92.31	22 268	90.11	67.31
组 4	147	147	138	93.88	21 399	88.93	63.62
组 5	143	143	132	92.30	21 751	91.54	68.29

从系统运行的测试结果中可以看出,测试用例的执行没有出现遗漏的现象,5 组测试用例的通过率均在90%上,静态存储余量与 CPU 平均占用率在整个检测过程中时刻在变化,但是均值稳定,并且保持在一个较为可观的范围内,测试用例的运行时间受用例的多少影响,综合检测系统在 5 组检测用例运行中没有出现宕机的现象,测试结果很好地说明了本文所提出的系统级综合检测方法是有效和可行的.

5 结 论

本文通过分析航空电子系统的软件体系结构特点,对航空电子系统进行形式化建模,针对系统运行时的信息交互,提出了基于任务分阶段的窗口树建模方法,支持检测用例的自动生成;并以任务为主体,抽象定义并简化了任务主体的状态,构建了系统状态图模型.在此基础上,提出了基于模型检测的系统级综合检测方法,从静态和动态两方面对系统进行综合性检测,最后设计并实现了综合检测系统,验证了所提出方法的有效性.后续的研究工作还包括形式化地讨论被测系统的形式化模型模拟被测系统运行行为的完备性、模型的进一步细化、采样数据处理方法的改进和完善以及检测用例生成算法的优化等.

致谢 感谢审稿人的仔细评阅和中肯的意见.

References:

[1] Zheng ZM, Ma SL, Li W, Wei W, Jiang X, Zhang ZL, Guo BH. Dynamical characteristics of software trustworthiness and their evolutionary complexity. Science in China (Series F: Information Sciences), 2009,52(8):1328−1334. [doi: 10.1007/s11432-009-0137-2]

[2] Zheng ZM, Ma SL, Li W, Jiang X, Wei W, Ma L, Tang ST. Complexity of software trustworthiness and its dynamical statistical analysis methods. Science in China (Series F: Information Sciences), 2009,52(9):1651−1657. [doi: 10.1007/s11432-009-0143-4]

[3] Li YM, Wang JQ, Zheng JG, Wang CQ. About standard avionics ATE of overseas. Computer Measurement & Control, 2004, 12(1):1−5 (in Chinese with English abstract). [doi: 10.3321/j.issn:1671-4598.2004.01.001]

[4] Liu JF, Wang H. Study on architecture of overseas avionics ATS. Measurement & Control Technology, 2002,21(1):5−8 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-8829.2002.01.002]

[5] Yu JS, Li XS. Future trends of the U.S. military ATS. Measurement & Control Technology, 2001,20(12):1−3 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-8829.2001.12.001]

[6] Cai XB, Wang H, Wang HW. Development of French universal automatic test equipment. Measurement & Control Technology, 2000,19(6):1−4 (in Chinese with English abstract).

[7] Wen XS, Xu YW, Yi XS, Chen X. Intelligent BIT Theory and Application. Beijing: National Defense Industry Press, 2002 (in Chinese).

[8] Tong J, Cai YW, Bo W, Ren JT. Analysis of development and application of BIT technology. Ordnance Industry Automation, 2008, 27(4):5−7 (in Chinese with English abstract). [doi: 10.3969/j.issn.1006-1576.2008.04.002]

[9] Guo YM, Cai XB, Zhang BZ, Zhai ZJ. Review of prognostics and health management technology. Computer Measurement & Control, 2008,16(9):1213−1216 (in Chinese with English abstract).

[10] Byington CS, Roemer MJ, Galie T. Prognostic enhancements to diagnostic systems for improved condition-based maintenance. In: Proc. of the 2002 IEEE Aerospace Conf. 2000. [doi: 10.1109/AERO.2002.1036120]

[11] Scheuren W. Safety & the military aircraft joint strike fighter prognostics & health management. In: Proc. of the AIAA. 1998.

[12] Zhang T. Research on formal verification methods of model of complicated information system [Ph.D. Thesis]. Harbin: Harbin Engineering University, 2011 (in Chinese with English abstract).

[13] Praxis AH. Using formal methods to develop an ATC information system. IEEE Software, 1996,13(2):66−76. [doi: 10.1109/52.506463]

[14] Introduction to TCAS II, version 7.1 [Z]. Version 1.2, U.S. Department of Transportation, Federal Aviation Administration, 2014.

[15] Vassev E, Hinchey M. Developing experimental models for NASA missions with ASSL. In: Proc. of the Workshop on Formal Methods for Aerospace (FMA) EPTCS 20. 2010. 88−94.

[16] Zhu Y, Geng XT, Gao XG. The modelling and analysis of integrated avionics system based on stochastic Petri net. Fire Control and Command Control, 2006,31(1):41−44 (in Chinese with English abstract). [doi: 10.3969/j.issn.1002-0640.2006.01.013]

[17] Miller SP, Greve DA, Wilding MM, Collins R, Rapids C, Srivas M. Formal Verification of the AAMP-FV Microcode. NASA/CR-1999-208992, Menlo Park: SRI Int'l, 1999.

[18] Bieber P, Castel C, Seguin C. Combination of fault tree analysis and model checking for safety assessment of complex system. In: Proc. of the 4th European Dependable Computing Conf. LNCS 2485, Berlin, Heidelberg: Springer-Verlag, 2002. 19−31. [doi: 10.1007/3-540-36080-8_3]

[19] Wiels V, Delmas R, Doose D, Garoche PL, Cazin J, Durrieu G. Formal verification of critical aerospace software. Journal of Aerospace Lab, 2012,4.

[20] Chu WK, Zhang FM, Fan XG. Overview on software architecture of integrated modular avionic systems. Acta Aeronauticaet Astronautica Sinica, 2009,30(10):1912−1917 (in Chinese with English abstract). [doi: 10.3321/j.issn:1000-6893.2009.10.020]

附中文参考文献:

[3] 李永明,王俭勤,郑晋光,王成青.国外标准化通用航空电子自动测试设备现状和发展.计算机测量和控制,2004,12(1):1−5. [doi: 10.3321/j.issn:1671-4598.2004.01.001]

[4] 刘金甫,王红.国外航空电子 ATS 体系结构研究.测控技术,2002,21(1):5−8. [doi: 10.3969/j.issn.1000-8829.2002.01.002]

[5] 于劲松,李行善.美国军用自动测试系统的发展趋势.测控技术,2001,20(12):1−3. [doi: 10.3969/j.issn.1000-8829.2001.12.001]

[6] 蔡小斌,王红,王宏伟.法国通用自动测试平台 ATE 发展综述.测控技术,2000,19(6):1−4.

[7] 温熙森,徐永威,易晓山,陈循.智能机内测试理论与应用.北京:国防工业出版社,2002.

[8] 同江,蔡远文,伯伟,任江涛.BIT 技术的发展现状与应用分析.武器装备自动化,2008,27(4):5−7. [doi: 10.3969/j.issn.1006-1576.2008.04.002]

[9] 郭阳明,蔡小斌,张宝珍,翟正军.故障预测与健康状态管理技术综述.计算机测量和控制,2008,16(9):1213−1216.

[12] 张涛.复杂信息系统模型的形式化验证方法研究[博士学位论文].哈尔滨:哈尔滨工程大学,2011.

[16] 朱岩,耿修堂,高晓光.基于随机 Petri 网的综合航电系统建模及分析.火力与指挥控制,2006,31(1):41−44. [doi: 10.3969/j.issn.1002-0640.2006.01.013]

[20] 褚文奎,张凤鸣,樊晓光.综合模块化航空电子系统软件体系结构综述.航空学报,2009,30(10):1912−1917. [doi: 10.3321/j.issn:1000-6893.2009.10.020]

李睿(1983−),女,江西高安人,博士生,CCF 学生会员,主要研究领域为软件自动化方法,基础软件.

连航(1988−),男,硕士,主要研究领域为软件自动化方法.

马世龙(1953−),男,博士,教授,博士生导师,主要研究领域为计算机软件与理论,网络环境下的计算模型,海量信息处理计算模型,网络计算技术及其应用.

黎涛(1990−),男,硕士生,主要研究领域为软件自动化方法.