http://htkz.cn
引用格式：高猛，王晓玲，朱晓程. 航天嵌入式软件安全性验证技术[J]. 航天控制，2025，43(2)：72-78.（GAO Meng，WANG Xiaoling，ZHU Xiaocheng. The verification technology for aerospace embedded software safety [J]. Aerospace Control，2025，43(2)：72-78.）

航天嵌入式软件安全性验证技术

高　猛，王晓玲，朱晓程
北京航天自动控制研究所，北京 100854

摘　要　作为影响安全苛刻系统的重要因素，软件安全性问题日益受到关注。本文结合航天嵌入式软件工程实践，以软件安全性需求为线索，聚焦典型安全性问题，从源代码安全性质的形式化验证、软件安全性需求的自动化测试两个维度，分析总结了包括安全性专项分析、源代码静态分析、源代码模型检测、基于故障模型的安全性测试及关键字驱动的自动化测试等若干关键技术，提出了完整的航天嵌入式软件安全性验证的技术解决方案，研制了自主可控的软件保证支撑平台及工具，以系统提升航天嵌入式软件的可信保证能力。

关键词　航天控制；嵌入式软件；中断驱动；软件安全性；形式化验证；自动化测试
中图分类号：TP392　　**文献标识码**：A　　**文章编号**：1006-3242(2025)02-0072-07

The verification technology for aerospace embedded software safety

GAO Meng, WANG Xiaoling, ZHU Xiaocheng
Beijing Aerospace Automatic Control Institute, Beijing 100854, China

Abstract　As a critical factor affecting safety-critical systems, it has been drawn increasingly attention to software safety issues. Based on engineering practices in aerospace embedded software testing and verification, the software safety requirements are taken as a clue and typical safety problems are focused in this paper. The two dimensions are introduced by the formal verification of source code safety properties and the automated testing of software safety requirements which analyze and summary key technologies, including special safety analysis, source code static analysis, source code model checking, fault-model-based safety testing and keyword-driven automated testing. A comprehensive technical solution for aerospace embedded software safety verification is proposed, and an independent controllable software assurance support platform and tools are developed to systematically enhance the trustworthy assurance capabilities of aerospace embedded software.

Key words　Aerospace control; Embedded software; Interrupt driven; Software safety; Formal verification; Automated test

收稿日期：2025-03-05
作者简介：高　猛（1982-），男，硕士，高级工程师，主要从事软件可靠性及嵌入式软件测试等方面的研究工作。

0　引　言

软件是复杂航天系统的神经中枢,许多关键、复杂的功能均由软件实现,软件的规模和复杂度急剧增加。作为影响安全苛刻系统的重要因素,软件安全性问题日益受到关注。2016年欧空局“夏帕瑞丽”火星着陆器坠毁、日本“瞳”卫星在轨解体以及2020年美国波音“星际客机”载人飞船首飞失败都是近年因为软件安全性问题导致的重大航天事故。

软件安全性(Software safety)一词最早出现在1979年美国发布的MIL-STD-1574A[1]中。美国航天局在NASA-STD-8719. 13C[2]中指出,软件安全性是指软件在生命周期内,应用安全性工程技术,确保软件采取有效措施以提高系统安全性,使系统发生安全性事故的概率降低到一个可以接受的水平内。结合航天工程实践,软件安全性是指软件能够经受有害事件并保持航天系统安全和业务连续的能力,其重点关注系统在发生错误、产生故障或异常时软件的故障诊断和处理能力。

航天嵌入式软件大多属于中断驱动型程序,通常采用中断加主程序或中断加线程的结构,中断(线程)的触发时间和触发顺序不确定且执行交错空间庞大。另外,还具有运行环境恶劣、运行周期较长、数值运算复杂及故障诊断严苛等特点。由于软件运行内在的不确定性,一些对安全性影响较大的深层次问题通常在罕见的执行交错条件下才会发生。上述特点对传统的软件安全性验证技术提出了挑战。具体表现如下:

1)源代码层面:据统计,数据竞争、时序冲突及原子性违反等并发错误,以及数据计算溢出、数组越界、缓冲区溢出、空指针引用、变量使用前未赋初值、计算除零及内存泄漏等运行时错误仍是影响航天嵌入式软件安全性的典型多发问题。采用先进的程序分析技术,实现源代码安全性质的形式化验证是可信软件工程亟待解决的问题。

2)系统级层面:受限于领域需求分析的复杂性和需求规约技术的应用,软件安全性需求分析和设计难免出现遗漏,导致状态转换逻辑冲突、非法数据闭环引入、接口时序不匹配及单粒子翻转等软件可靠性设计不足等问题时常出现,给航天型号安全运行带来极大隐患。软件安全性验证的充分性要求给传统的以测试用例生成为核心的安全性测试提出了挑战。

针对上述问题,本文结合航天嵌入式软件测试验证工程实践,以型号软件安全性需求为线索,聚焦典型问题,从源代码安全性质的形式化验证和系统级安全性需求的自动化测试两个维度,分析总结若干关键技术,形成完整的航天嵌入式软件安全性验证技术解决方案,保证航天嵌入式软件的质量。

1　软件安全性需求分析

航天嵌入式软件开发广泛采用基于过程的“V模型”,如图1所示。软件安全性需求的分析、设计和验证贯穿整个软件开发全过程,是一个不断迭代、逐步求精的过程。

其中,确定软件安全性需求是整个软件安全性工作的关键,目标是通过安全性分析方法获得相对完整、正确的软件安全性需求集合。软件安全性问题多由软件安全性需求获取不全导致,如对硬件失效和异常情况的处理不完善等。航天领域通常采用的软件安全性分析方法包括:

初步危险分析(Preliminary hazard analysis, PHA)是一种表格形式的安全性分析方法,对系统或子系统存在的危险类别、出现条件及可能造成的结果进行分析。PHA提供危险清单的初步框架,并记录通用的危险因素。

软/硬件交互分析(Hardware and software interaction analysis, HSIA)适用于设计的早期阶段,重点关注由软件控制的硬件产品可能出现的故障模式,其输出结果可能影响硬件设计和软件需求。

软件故障树分析(Software fault tree analysis, SFTA)是一种是自顶向下的分析技术,选取显著影响系统的故障事件作为顶事件,通过逻辑门与基本事件表达危害发生的因果链,分析引起系统故障的各种软件原因。

软件失效模式及影响分析(Software failure mode and effect analysis, SFMEA)也是一种表格形式的安全性分析方法,是一种自底向上的分析技术,它以失效模式为基础,以失效影响或后果为中心,根据分析层次和因果关系进行推理、归纳,识别薄弱环节,提出改进措施。

通过上述软件安全性分析工作,得到两个输出:

1)在软件任务书中标识配置项的安全关键等

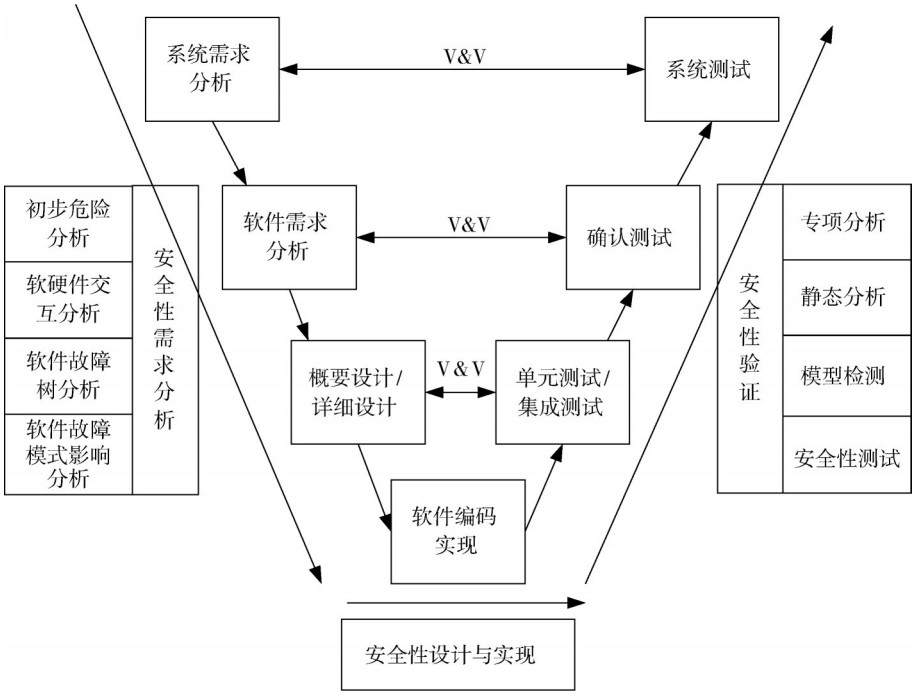

图 1　基于过程的“V 模型”

级和安全关键功能；

2)在软件需求规格说明中分解安全关键功能，细化完善安全性需求并进行条目化管理，在跟踪矩阵中追溯。

航天嵌入式软件安全关键需求识别应重点关注：

a) 直接导致危险发生(危险源)的软件功能；

b) 保证系统能源安全的工作模式；

c) 启动或恢复故障安全模式的软件逻辑；

d) 显示硬件状态或安全关键功能状态的遥测设计；

e) 硬件加电/断电、主份/备份切换、活动部件动作的控制信号设计；

f) 软件在线升级维护功能或变量设计。

由于软件安全性分析方法大多源于传统的系统安全性分析方法，在软件特征描述方面存在不足，同时，受限于形式化规约技术的复杂性和适用性，导致软件安全性需求存在分析不完整和定义不准确的问题。因此，系统开展软件安全性验证活动是保证软件安全性的关键。

2　源代码安全性分析技术

针对嵌入式软件源代码的安全性分析，主要采用源代码安全性专项分析、源代码静态分析和模型检测等形式化验证技术，检测典型的软件缺陷模式和运行时错误。

2.1　安全性专项分析

安全性专项分析是采用人工审查并以表格化的方式，针对源代码进行数据分析、中断分析、堆栈分析、临界条件分析和可恢复性分析，验证软件安全性设计与实现的正确性和合理性。

1) 数据分析：确认数据项得到适当的定义和使用，包括数据访问冲突分析、数据数制分析及地址空间使用分析等。

2) 中断分析：确认中断的使用不出现问题，包括中断处理流程分析、中断嵌套分析、中断访问冲突分析及未使用中断分析等。

3) 堆栈分析：确认不会发生堆栈溢出的问题，包括静态和动态情况下的堆栈深度分析及升级维护过程注入代码的堆栈分析等。

4) 临界条件分析：确认软件的相关性能指标满足要求，包括响应时间分析、恢复时间分析、故障检测和处理时间分析及切机或复位进入稳态时间分析等。

5) 可恢复性分析：确认出现相关故障时软件可恢复至安全状态，包括看门狗分析、切机分析、复位分析及重要数据备份与恢复分析等。

2.2　源代码静态分析

源代码静态分析主要用于编程规范检查、缺陷

模式检测、安全漏洞检测、代码质量度量、程序理解和故障定位等[3-4]。作为自动化程序验证技术,静态分析已成为航天嵌入式软件开发过程中必须采用的工程技术。典型的关键技术包括抽象解释、符号执行和约束求解等[5]。

领域典型缺陷模式检测是近年来程序验证技术的研究热点。作者系统梳理了近10年航天嵌入式软件归零问题和第三方评测问题(近4000个),总结抽象出计算算法类、中断时序类和总线通信类等11类缺陷类别,共计缺陷模式170项,建立了首个航天嵌入式软件缺陷模式集(Space software defect patterns set,SSDPS),并基于时序逻辑描述语言对部分缺陷模式进行了形式化建模[6]。面向源代码缺陷模式,研制了具有独立自主知识产权的静态分析工具。基于抽象解释、符号执行及约束求解等程序分析技术,在编程规范检查引擎的基础上,研制了缺陷模式检测引擎,实现了60余项代码级缺陷模式的自动检测,其中包括数组下标越界、缓冲区溢出、空指针引用及计算除零等运行时的典型错误,缺陷检出率大幅提升。

2.3　源代码模型检测

源代码模型检测是保证复杂软件系统安全性质正确性的一种形式化方法,其基于全路径覆盖的穷举式检查,易于发现软件中的算法错误、并发错误等隐藏缺陷,验证过程自动化程度高,错误定位准确[7]。然而,现有的模型检测技术由于存在状态空间爆炸、不能有效支持中断驱动型程序检测等缺点而少有工程应用。

为此,针对型号中常见多发的整数溢出问题,作者围绕中断驱动型程序有界模型检测技术进行了深入研究。在大量真实案例分析基础上,基于整数溢出变量传递性和无后效性的特点,提出一种基于变量依赖关系的程序模型约简技术,将约简后的程序抽象为静态单赋值形式SSA。针对中断驱动型程序,采用抽象解释、摘要机制对中断函数特征进行表示,提出基于干扰变量的中断驱动程序顺序化方法。整数溢出有界模型检测基本框架如图2所示[8],其步骤简要说明如下:

1) 解析待处理程序,得到其中全部的干扰变量信息。

a)干扰变量:存在一个全局变量V,主函数中对全局变量V进行读操作,至少在一个中断中对全局变量V进行写操作或读写操作;

b)干扰变量信息:由干扰变量V、主函数上使用V的语句ST、语句ST对应的位置LOC以及中断程序对应的中断向量号VEC组成。

2) 对中断函数进行抽象处理,得到中断函数摘要。

将中断函数ISR视为被调用函数,其函数体可描述为:while(true){ISR();},对以上函数进行抽象解释迭代分析,计算出中断函数上的各干扰变量值区间不变式后生成中断函数摘要。中断函数摘要由中断向量号、干扰变量及干扰变量值区间不变式组成的三元组的形式进行记录。

3) 在主程序中插入中断函数摘要,得到顺序化后的程序。

对干扰变量进行遍历,根据中断向量号找到该干扰变量对应的中断函数;将中断函数摘要插入到主函数语句ST之前完成顺序化操作。

4)获得顺序化程序的静态单赋值SSA形式,进行程序模型状态空间约简。

按循环边界k将函数展开,对SSA中的语句集合进行遍历,仅选取对整数溢出变量进行写操作的语句,并记录到约束集合C中。

5)使用有界模型检测工具CBMC对约简后的程序进行整数溢出检测。

将约束集合C中的语句进行合取操作得到程序模型CM;将待验证的整数溢出断言语句转换为性质P;将程序模型CM与性质P的CNF公式进行合取

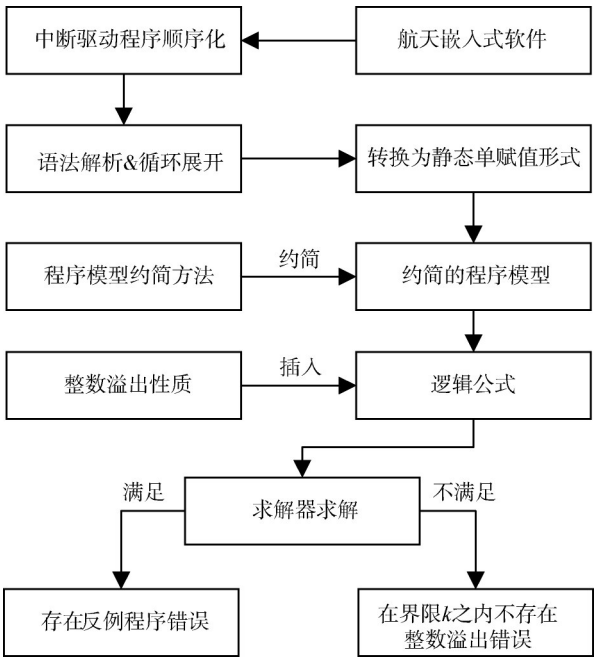

图2　整数溢出有界模型检测基本框架

操作构建逻辑公式，并编码为SMT模型求解。若有解，则表明存在一条错误执行路径并给出相应的反例；反之则表明在界限k之内不存在整数溢出错误。

真实航天嵌入式软件应用实例表明，上述方法在保证整数溢出问题检出率的前提下，不仅能够提高分析效率，还使得已有的模型检测技术能够适用于中断驱动型程序整数溢出检测。相关研究工作初步实现了形式化验证技术在航天嵌入式软件领域的工程应用。

数值计算溢出、数组越界、缓冲区溢出、空指针引用、计算除零、内存泄漏及变量使用前未赋初值等基本的软件安全性质均可通过有界模型检测技术进行检测，如何提高检测效率和准确率是未来研究的方向[9]。

3 软件安全性测试技术

3.1 安全性测试策略

经验表明，需求遗漏和需求错误是引入软件安全性风险的最主要原因。因此，除了测试用户需求中的显式安全性需求外，还要基于领域知识和工程经验识别隐性安全性需求，防止需求遗漏。软件安全性测试应重点验证系统在发生错误、产生故障或异常条件等工况下的软件处理能力。其测试策略主要包括以下几个方面：

1）针对安全关键功能（如发射全流程、保证能源安全的工作模式及重要数据保护功能等）必须单独测试安全性需求。

2）基于故障模式的健壮性测试。一方面，对软件在故障或异常情况下（如注入错误数据、输入错误操作及突然断电等）的反应进行测试，验证软件防止危险状态措施的有效性；另一方面，通过故障注入模拟硬件相关的故障模式（如单粒子效应、通信接口异常及非法中断触发等），验证软件故障诊断与重构设计的有效性。

3）安全关键功能的测试应满足语句、分支及条件的覆盖性要求，并从用户角度满足覆盖输入，包括输入域覆盖、各种相关功能的覆盖、相关变量可能组合的覆盖、设计输入空间与实际输入空间区域的覆盖等。

3.2 测试需求规则集

安全性测试需求规则集是目前航天嵌入式软件测试验证活动的优秀实践，来源于领域知识和工程经验，涵盖了已知的系统级软件故障模式。规则集中的测试需求项作为通用安全性测试需求，被测软件本身的领域安全性需求作为专用安全性需求，二者结合形成“通用+专用”软件安全性测试验证模式。典型的系统级软件故障模式示例如表1所示。

表1 系统级软件故障模式（示例）

类别	故障模式	
运行时序	任务执行时间超时	数据采集时序错误
	软硬件时序不匹配	中断信号频繁触发
	触发非法外部中断	中断服务程序超时
硬件接口	输入指令及数据错误	输出指令及数据错误
	1553B总线通信异常	CAN总线通信异常
	串行总线通信异常	敏感器/传感器故障
程序安全	程序加载失败	无法正常复位/切机
	程序跑飞	程序陷入死循环
	硬件狗失效	单粒子效应

3.3 测试用例标准集

测试用例标准集是针对某一领域软件开展测试应参照的测试用例组合。测试用例标准集针对该领域软件的典型功能、性能指标，涵盖各种已知的故障模式，确定测试方法和测试数据，依据标准集可以对领域软件进行全面、权威的测试。

基于测试需求规则集中的故障模式，针对性开展安全性测试用例设计，构建测试用例标准集，形成安全性需求、故障模式和测试用例的映射关系，填补软件安全性测试方法的缺失，提高测试用例的揭错能力，满足航天嵌入式软件安全性测试的要求。

3.4 关键字驱动的自动化测试

关键字驱动的自动化测试是当前软件测试领域的主流技术。其核心思想是通过关键字，实现业务逻辑、测试数据和测试脚本的三分离，将复杂的测试设计简化为层次清晰的表格设计，提高了测试用例和测试脚本的复用性，利于形成测试资产；同时降低回归测试成本，缩短测试周期，提高测试自动化水平。

3.4.1 关键字定义

关键字是一种规范化测试描述语言，是对测试指令集的抽象和封装。基于关键字，面向系统的业务逻辑对测试用例进行开发和封装，构建满足测试需求的测试序列，形成公用的测试构件，实现测试自动化和测试用例复用。

1)关键字描述范式

航天嵌入式软件测试关键字描述范式:

Keyword[$ Name]:[$ Time; $ Operation; $ Data1、$ Data2、$ Data n; $ Attributes]

解释:关键字[名称]:[执行时间;操作;数据 1;数据 2;数据 n;含义说明]

以 1553B 总线通信功能测试的相关关键字为例进行说明:

a) 关键字:发送 RT 总线数据

$ {time}	$ {cmdID}	$ {stringOrPath}	$ {cycTime}

含义说明:第 $ {time}秒,通过子地址 $ {cmdID}以 $ {cycTime}秒为周期发送数据 $ {stringOrPath}。

b) 关键字:设置 RT 总线寄存器

$ {time}	$ {address}	$ {data}	$ {mask}	$ {isSet}

含义说明:第 $ {time}秒,设置 $ {isSet}地址为 $ {address}的 RT 总线寄存器值为[$ {mask}& $ {data}]。

2)关键字类别

关键字由基本关键字和组合关键字组成。结合航天嵌入式软件测试特性,抽象出基本关键字(包括:平台控制、总线通信、数据采集、数据查看和故障设置 5 类基本关键字),通过逻辑组合,构建特定场景的组合关键字,然后逐层扩展,形成与功能相关的测试操作序列,最终生成满足各种测试需求的测试用例集并自动生成测试脚本。

3.4.2　自动化测试框架

关键字驱动的航天嵌入式软件自动化测试框架如图 3 所示。其中:

1)应用层:选择测试需求项,基于定义好的关键字,明确相关数据源,采用表格方式编写测试用例,并同步自动生成测试脚本。同时,提供预期结果描述。

2)解析层:基于关键字,自动生成满足定义规则的测试脚本,并对脚本进行解析,作为仿真测试环境的测试输入。

3)执行层:测试脚本提供的指令和数据驱动被测软件运行,自动记录测试输出并与测试预期结果进行自动判读,确认是否满足测试需求。最后,按照模板要求自动生成测试说明和测试报告。

测试平台对测试过程中各类数据进行结构化统一管理,实现数据间的追踪关系维护,提供完全的执行过程自动记录,实现测试过程可控。同时提供了针对关键字的自然语言用例映射表,自动生成满足用户要求的测试报告。

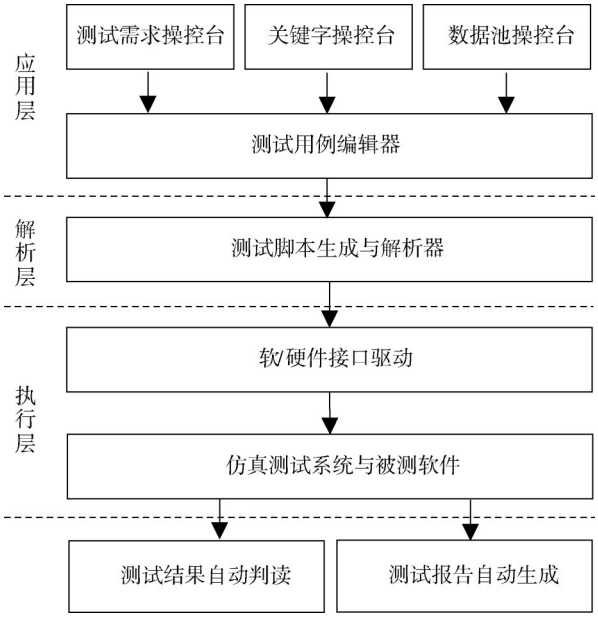

图 3　嵌入式软件自动化测试框架示意图

4　结　论

本文以航天嵌入式软件测试验证的工程实践为基础,以软件安全性需求为线索,聚焦典型安全性问题,从源代码安全性质的形式化验证和软件安全性需求的自动化测试两个维度,分析了包括安全性专项分析、源代码静态分析、源代码模型检测、基于故障模型的安全性测试及关键字驱动的自动化测试等若干关键技术,提出了完整的航天嵌入式软件安全性验证技术解决方案,研制了自主可控的可信软件保证支撑平台及工具,以提升航天嵌入式软件的质量保证能力。相关工作已在以新型运载火箭为代表的航天型号软件研制中得到广泛应用,促进了我国航天型号软件质量的提升。

参　考　文　献

[1] MIL-STD-1574A, System safety program for space and missile systems[S].

[2] NASA-STD-8719.13C, NASA software safety standard[S].

[3] WU X, CHEN L, MINE A, et al. Static analysis of runtime errors in interrupt-driven programs via sequential-

ization[J]. ACM Transactions on Embedded Computing Systems, 2016,15(4):1-26.

[4] BINKLEY D. Source code analysis: a road map[C]// Proceedings of the Conference on the Future of Software Engineering, Minnerpolis,2007:104-119.

[5] 张健,张超,玄跻峰,等. 程序分析研究进展[J]. 软件学报, 2019, 30(1): 80-109. (ZHANG Jian, ZHANG Chao, XUAN Jifeng, et al . Progress in program analysis research[J]. Journal of Software, 2019, 30(1): 80-109.)

[6] 高猛,滕俊元,陈睿,等. 航天器软件典型缺陷模式的自动检测技术[J]. 空间控制技术与应用,2019,45(5):72-78. (GAO Meng, TENG Junyuan, Chen Rui, et al. Automatic detection of typical defect patterns in spacecraft software [J]. Space Control Technology and Applications,2019,45(5):72-78.)

[7] WU X G, WEN Y J, CHEN L Q, et al. Data race detection for interrupt-driven programs via bounded model checkinge[C]// Proc. of the 7th International Conference on Software Security and Reliability Companion . Los Alamitos: IEEE, 2013:204-210.

[8] 高猛,滕俊元,王政. 航天嵌入式软件整数溢出的形式化验证方法[J]. 软件学报, 2021,32(10):2977-2992. (GAO Meng, TENG Junyuan, WANG Zheng. Formal verification method for integer overflow in aerospace embedded software [J]. Journal of Software, 2021,32(10):2977-2992.)

[9] SUNG C H, KUSANO M, WANG C. Modular verification of interrupt-driven software[C]// Proc. of the 32nd IEEE/ACM International Conference on Automated Software Engineering. Los Alamitos: IEEE, 2017:206-216.