软件安全性研究综述

樊晓光[1] 褚文奎[1,2] 张凤鸣[1]
(空军工程大学工程学院 西安 710038)[1] (94795部队 芜湖 241000)[2]

摘 要 软件是安全性关键的软件密集型系统(比如综合航电系统)的一个重要安全因子,软件安全性已逐渐成为软件工程和安全工程交叉领域的研究热点之一。对软件安全性的内涵与外延进行了剖析,给出了软件安全性定义。讨论了软件安全性的度量模型。着重从软件工程的视角对软件安全性的开发过程、设计方案、评估方法与认证技术等现状进行了综述,并探讨了软件安全性的研究方向。

关键词 安全因子,软件安全性,软件工程,安全工程,系统工程,安全性关键系统,综合航电
中图法分类号 TP311.5,N945,V247 **文献标识码** A

Surveys of Software Safety

FAN Xiao-guang[1] CHU Wen-kui[1,2] ZHANG Feng-ming[1]
(Institute of Engineering, Air Force Engineering University, Xi'an 710038, China)[1]
(No. 94795 Unit of PLA, Wuhu 241000, China)[2]

Abstract As software is one of the important safety factors in a software-intensive and safety-critical system, e.g., an integrated modular avionics(IMA) system, software safety is to be a mainstream research direction in the crossing fields between software engineering and safety engineering. The paper analysed firstly the meanings and extensions of software safety, and then gave a definition of it. Measuring models of software safety were then discussed. The paper focused on the state-of-the-art of software safety from a software engineering perspective about development processes, designed alternatives, assessment techniques and certification methods. The potential research directions of software safety were finally pointed out.

Keywords Software factor, Software safety, Software engineering, Safety engineering, Systems engineering, Safety-critical system, Integrated modular avionics(IMA)

安全性关键系统(safety-critical system,SCS)[1]泛指具有潜在破坏力的一类系统。此类系统一旦失效,就能够造成严重的后果,比如人员伤亡、财产损失或环境破坏等。SCS安全性的核心研究内容是采用各种管理、组织、技术等措施,减弱SCS各组成要素导致系统进入不安全状态的概率或减轻SCS失效后果。

近年来,软件在SCS中的应用越来越广泛。从航空航天领域到核工业领域,从电力系统到医疗系统,软件承担着安全性关键的指挥控制功能。同时软件在SCS中的应用规模也与日俱增。比如,在F-22战机的综合航电系统[2]中,软件实现的航电功能高达80%[3],软件代码达到170万余行。而在F-35战机的先进综合航电系统中,软件代码达到500~800万行。这表明,越来越多的SCS日益软件密集化,逐渐形成安全性关键的软件密集型系统。

另一方面,由SCS软件引发的事故或事件却频发不断:发生在20世纪90年代的Ariane 5运载火箭、SOHO太空船等5起航天器事故的罪魁祸首是软件[4];2004年12月20日,一架F-22因飞行控制软件故障而坠毁[5];2007年2月11日,12架F-22在穿越国际日期变更线时又因软件缺陷问题造成导航故障,战机被迫在无导航和通信能力下危险返航[6]。由此,软件已成为决定SCS(包括新型航空航天器)安全与否的重要因素之一。

1 软件安全性内涵与外延

1.1 软件安全性内涵

"软件安全性(software safety)"一词最早出现在1979年发布的Mil-Std-1574A[7]标准中,1986年由美国加州大学的Leveson教授引入计算机科学领域[8]。安全性多用于刻画具有能量、毒性或能够进行质量运动的物理设备或系统,而软件作为一种寄生性逻辑实体,不会由于能量辐射、毒性挥发、质量运动对人造成伤害或对环境造成破坏。因此,"软件安全性"至今备受争议,甚至被认为是一个误用的术语[9]。

到稿日期:2010-04-13 返修日期:2010-07-25 本文受总装备部国防预研基金项目(9140A17020307JB3201),空军工程大学工程学院优秀博士学位论文创新基金(BC07003)资助。

樊晓光(1965—),男,博士,教授,CCF会员,主要研究方向为综合航电技术等;褚文奎(1980—),男,博士,工程师,CCF会员,主要研究方向为综合航电、软件安全性等,E-mail:chuwenkui@126.com(通信作者);张凤鸣(1963—),男,教授,博士生导师,主要研究方向为综合航电、信息系统工程等。

值得指出的是，尽管“安全”本身是一个系统问题，作为系统重要组成要素的软件不会直接危及生命、财产和环境等安全，但借助软件实现的人机交互却可能因软件失效造成人员误操作从而形成危险。对于无人机交互的嵌入式 SCS 而言，也可能因为软件错误地控制系统而造成灾难性后果。以航电软件为例，飞机在飞行高度低于预期值时，航电软件应及时发出近地告警提示飞行员，否则容易造成飞机坠毁。由此，McDermid 认为，软件安全性只是软件在系统上下文中对系统安全性的贡献的速记形式[10]。

目前很多标准[11-13]和学术文章[8,10]都给出了软件安全性的定义。比较具有代表性的是：

(1)NASA 8719.13A[13]：“软件安全性是指在软件生命周期内，应用安全性工程技术，确保软件采取积极的措施提高系统安全性，确保降低系统安全性的错误已经减少到或控制在一个风险可接受的水平内。”

(2)Leveson[8]：“软件安全性是指确保软件在系统上下文中执行不会导致系统发生不可接受的风险。”

文献[14]指出，软件安全性之所以未能有效付诸实践的两个主要原因是“软件安全性概念的开发人员或用户未能正确理解哪些要素可以确保系统安全”和“未能在更大的或更宽泛的系统中考虑软件安全性”。

根据对软件安全性的理解，给出软件安全性的下述定义：

定义 1(软件安全性)

$$SW\_safety = \int_{SW} Sys\_safety(SW, HW, LW, Env)\mathrm{d}SW$$

软件安全性(SW_safety)是指既定软件对既定系统安全性(Sys_safety)的贡献，衡量的是在特定环境下和规定时间内不会因软件功能失效或需求缺陷等诱发系统危险的能力。其中，系统安全性(Sys_safety)受到操作人员(LW)、软件(SW)、硬件(HW)和外部环境(Env)的共同作用。外部环境(Env)泛指与该系统交互作用的外部系统或其他要素。比如，对于飞行控制软件而言，其安全性研究涉及到飞控软件内部构件之间的交互，飞控软件与飞控系统底层硬件之间的交互，飞控软件与飞行员、综合航电系统等其它系统之间的交互。

1.2 软件安全性外延

在可信计算[15]领域，与安全性(safety)密切相关的术语还包括保密性(security)、完整性(integrity)、可靠性(reliability)、可生存性(survivability)和可信性(dependability)。

鉴于上述术语之间的易混淆性以及国内对部分术语翻译的不一致性(比如将 safety 译为防危性[16]、保险性[17]，将 security 译为安全性[17])，此处着重阐明安全性与这些术语之间的区别和联系，以便更好地理解其内涵(见图 1)。

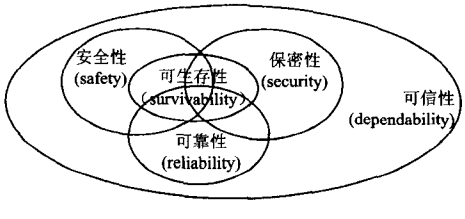

图 1 软件安全性及其联系

软件可靠性是同软件安全性一样与失效密切相关并一直混为一谈的一种软件属性。它衡量的是，在特定条件下和一定时间内，软件提供连续正确服务的能力。二者之间的区别和联系体现在：①软件可靠性关注所有的软件失效，而软件安全性仅关注能够引起危险后果的软件失效；②如果软件任务不能完成将影响系统的安全性，那么消除这些软件失效将同时提高软件的可靠性和安全性；③消除那些能够引起危险后果的软件需求缺陷可能提高了安全性，却对软件可靠性无益；④如果软件存在安全性需求方面的缺陷，即使消除所有的软件失效(尽管提高了软件可靠性)也不一定能提高软件安全性；⑤可靠的软件可能不安全，安全的软件并不一定可靠。有关软件可靠性的内容参见文献[18]。

软件保密性[15]融合了可用性(availability，指某项服务在某个时刻可以获得的概率)、隐私性(confidentiality，指软件已存储信息、代码、数据不被访问的能力)和完整性(integrity，指软件不被意外或未授权修改的能力)，着重关注如何保护软件使其不被恶意攻击。软件安全性和软件保密性之间的联系和区别主要表现在：①二者都与威胁或风险相关。安全性主要处理危及人员生命财产安全的威胁，而保密性主要消除危及数据私有性的危险；②安全性需求和保密性需求可能都会与某些重要的功能需求或任务需求相冲突；③安全性和保密性需求都需要在系统层面谋划布局，且这些需求在系统上下文外是难以处理的；④安全性和保密性都需要一些极其重要的需求，这些需求决定了系统能否被使用；⑤保密性着重关注的是对分级信息进行的未授权恶意访问行为，而安全性多关注的是无意行为。有关软件保密性的综述参见文献[19,20]。

软件可生存性描述的是当软件系统的一部分遭受攻击陷入瘫痪时，关键服务仍能够使用的程度。文献[21]认为可生存性应包含系统、威胁、自适应性、服务可持续性和实时性等 5 个关键部分。软件可生存性与软件安全性之间的联系在于：①确保软件生存的服务可能同时确保软件安全性；②生存性丢失的系统可能依然安全；③不安全的系统可能具有生存性。有关软件可生存性的内容参见文献[21,22]。

软件可信性包括可用性、可靠性、安全性、隐私性、完整性、可维护性(maintenance)6 个属性[15]。它强调的是软件行为在任意操作条件下都是可预测的，能够抵御其它软件、病毒以及一定的物理干扰造成的破坏。有关软件可信性的综述参见文献[15,23]。

2 软件安全性度量模型

SCS 的本质是以风险换利益。从辩证的视角看，SCS 没有风险就没有利益。换言之，“安全”是一个相对的概念。为了追求某些利益，在技术、效能、时间、费用等约束下，只要风险降低到某种程度(称之为安全阈值)，即认为 SCS 是安全可接受的。SCS 安全阈值的界定是定性的，并因各种 SCS 而异，由此给 SCS 软件安全性度量带来了诸多问题[24]。

当前度量软件安全性的工作主要体现在两个方面，即基于风险的和基于缺陷的测度方法。

2.1 风险测度法

风险测度法是度量安全性广为采用的一种方法。该方法以风险总值(Risk)作为安全性指标，以每一个危险(hazard)发生概率 $\rho(hazard)$ 和该危险可能导致的后果 $\varepsilon(hazard)$ 作为衡量风险的要素。一种普遍接受的计算模型[8]是：

$$Risk = \sum_{hazard} \rho(hazard) \times \varepsilon(hazard) \quad (1)$$

DO-178B[25]要求 SCS 软件失效率为 10^{-9}～10^{-7} 次/小时，这就几乎无法借助软件失效的历史统计数据恰当地预测某项软件功能失效的概率。由此，Ericson[26]认为式(1)安全性度量模型不适用于 SCS 软件。替代地，他提出了一种软件危险风险系数(software hazard risk index)测度方法。该方法与控制硬件的软件代码类别以及软件失效的后果严重性有关，通过一系列的查表方法最终可获得软件危险风险系数。

Fenton 认为，度量软件安全性应当考虑其它控制事件、触发事件对危险发生的影响以及后果减轻措施对安全性的影响[27]。如果不考虑上述两点，那么在式(1)中分配危险发生概率和危险后果就毫无意义。这种观点是十分合理的。对于失效的飞行控制软件而言，如果不加以及时控制或处理，就可能会影响飞行安全，比如 F-22 坠毁事故[4]。如果允许飞行员动态重构飞行控制软件或采取其它控制措施，那么就有可能消除危险。基于这种思想，Fenton 等人开发了 AgenaRisk 安全性评估系统。不过，该模型同样也需要输入某种软件失效概率以及控制事件、触发事件等的成功概率，这就不可避免地引入主观判断。另外，该模型没有给出风险排序或总风险值，不便于优先决策。

借鉴可靠性的定义模式，文献[16]尝试用平均危险失效时间度量 SCS 软件安全性，并给出了安全性评估指标和可靠性评估指标之间的定量对应关系。由于软件失效率和失效检测率的不可确定性，实际上该模型并不能发挥效用。此外，鉴于本文有关软件安全性和可靠性之间关系的讨论，该文献建立的模型也只有在软件功能失效同时影响安全性和可靠性的前提下才存在一定的合理性。

2.2 残留致险缺陷数测度法

文献[24]试图提出一种多元、多模型、多阶段(3M)的软件安全性度量框架。该模型希望在开发、使用、变更等阶段分别采用 Holstead 模型、模糊模型、经验模型等评估软件功能复杂性、技术支持、开发水平、测试结果等对软件安全性的影响，最终分别得到测试前、后的软件残留致险缺陷数。虽然并未给出相关模型的具体细化方法，也没有说明该指标的具体用意，但该文献提出了度量软件安全性的一种重要思路，即残留致险缺陷测度法。

文献[28]分析了缺陷导致危险产生的过程。尽管软件设计型缺陷是导致危险生成的主要根源，但值得指出的是软件安全性并不一定与残留致险缺陷数成反比。这是因为取决于软件运行剖面，只要这些缺陷没有被激活就不会对系统安全构成威胁。但对于某一款软件而言，我们总可以认为，残留致险缺陷数越少，软件安全性就越高。

文献[14]基于 McCall 软件质量模型提出了一种 SCS 软件安全性的度量框架。在该框架中，软件安全性与系统危险分析、需求完整性、基于安全性约束条件的设计、运行时管理、安全性关键的测试等有关。这与文献[24]的工作存在异曲同工之妙。不过，该模型并没有给出具体的单一或多个度量指标。实际上该模型关注的是如何提高 SCS 软件安全性。

3 以安全性为中心的软件开发过程

Heimdahl[9]认为，开发一个安全的系统无外乎 4 个步骤：①识别系统危险，定义安全性需求；②确定系统各组成要素如何触发上述系统危险；③为系统各组成要素派生相应的安全性需求；④开发上述组成要素，并使其满足对应的安全性需求。尽管上述过程看起来颇为简单，但开发满足安全性需求的软件的过程却要复杂得多，往往需要伴随软件开发过程开展相应的安全工程活动。目前，以安全性为中心的软件开发过程可以概括为两类，即传统的基于过程的开发方法和日益兴起的基于模型的开发(model-based development，MBD)方法。

3.1 基于过程的开发

当前基于过程的 SCS 软件开发多采用“V”模型[29]，其目的是在软件开发的每一阶段生成相应的验证测试计划，使其免受低层软件开发细节的影响。文献[30]认为，采用该模型开发专业领域的软件，还需要一门融合了系统工程与软件工程的交叉学科——软件系统工程——提供目标系统的领域知识，以便软件工程师充分理解系统需求，从而在软件设计中减少需求和设计缺陷，增强系统的安全性。图 2 是文献[31]给出的基于过程的 SCS 软件开发模型，该模型在航电工业中应用较广。

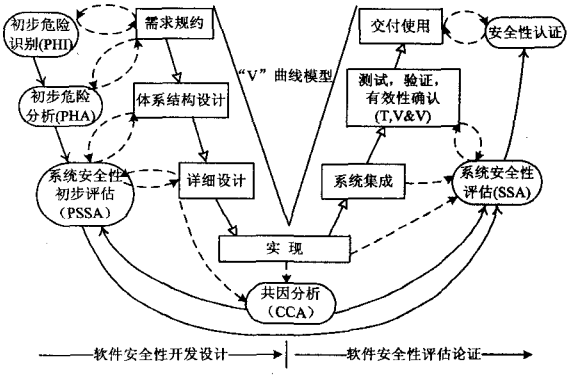

图 2 “V”型 SCS 软件开发过程

在“V”模型左侧，通过在需求规约、体系结构设计、详细设计、软件实现等阶段分别开展初步危险识别(preliminary hazard identification，PHI)、初步危险分析(preliminary hazard analysis，PHA)、系统安全性初步评估(preliminary system safety assessment，PSSA)、共因分析(common cause analysis，CCA)等安全工程活动，分析、识别、派生软件承担的安全性需求，确保其正确性、完整性、一致性，并使其在软件设计时充分考虑这些需求。

在“V”模型右侧，通过在软件实现、系统集成、测试/验证/有效性确认、交付使用等阶段开展共因分析、系统安全性评估(system safety assessment，SSA)等安全工程活动，评估、论证软件是否满足安全性需求或是否达到期望的安全性级别。

值得指出的是，软件的安全性“素质”主要取决于前期开发设计(V 模型左侧)，而不是在软件实现后(V 模型右侧)通过其它保护措施辅助实现。有关上述安全工程活动的目的和支撑技术参见 ARP-4754[32]和 ARP-4761[33]标准。

3.2 基于模型的开发(MBD)

在基于过程的软件开发方法中，SCS 软件的有效性确认和验证活动在“V”模型右侧进行，此时如果发现软件无法满足安全性需求或存在安全性方面的重大设计缺陷，再返工弥补，势必代价巨大。文献[9]表明在该开发方法中，软件有效性确认和验证所消耗的资源占到整个软件开发的 50%～

70%。这些弊端促使了 MBD 方法的出现。在 MBD 中，软件开发的重心在于构建能够准确反映系统需求的形式化或半形式化模型。为了实现该模型，MBD 要求在需求规约和体系结构设计阶段就开展有效性确认和验证活动。在特定的集成开发环境（比如 SpecTRM 工具集[34]）中，反映系统功能行为的需求模型能够自动转换为代码，甚至还能实现自动测试并产生可信的测试案例。如此，MBD 将“V”模型转变成了“Y”模型[35]，如图 3 所示。

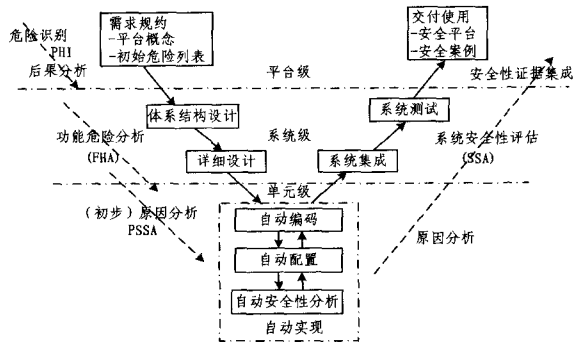

图 3 “Y”型 SCS 软件开发过程

MBD 缩减了“设计-集成-测试”的工作量，降低了缺陷、错误引入的可能性，节约了开发代价。但由于自动化技术的引入，也出现了许多新问题。比如，如果模型集成开发环境曲解了建模语言的语义，那么在该领域模型内执行的所有测试都是无效的。如果自动生成的代码不正确，那么最终软件实现可想而知。如果该领域的分析工具不能捕获错误的模型，那么该模型就可能被接受。目前，尚无法为模型集成开发环境、代码生成器、分析工具等提供一定的可信级别。

4 软件安全性设计

一个不容争辩的事实是，不管遵循多么严格的开发过程，软件总是或多或少地存在残留致险缺陷。比如，文献[36]在对一段遵循 DO-178B 开发过程的软件代码进行静态代码分析时，发现软件缺陷密度高达 23 个/kLoC，其中 6%是致险缺陷。而 MBD 广泛应用形式化技术存在的一个严重问题是泰坦尼克效应[8]（titanic effect），即过度信任系统假设和模型，而未能考虑软件失效情况的应对措施。因此，为了实现或提高软件安全性，不仅要遵循某些开发过程，还应该在软件设计中采取积极的应对策略。

Leveson[8]认为 3 个可选方案是（按优先级高低）：①阻止或降低危险发生概率的设计，比如采用 lockout，lockin，interlock 等技术；②运行时危险探测与处理方面的设计，比如采用看门狗、例外处理、前向恢复、后向恢复等技术；③提供对危险的告警设备、程序及培训。

目前技术方面的研究基本上围绕前两个方案进行。比如安全内核（safety kernel）技术[37]试图将整个系统分为功能和非功能两个组成部分，然后再分级设计安全性。在操作系统方面，主要通过提供可预知的调度算法确保系统的安全性，比如文献[38]提出的最短紧急距离优先的多级关键任务调度算法以及集成式多级防危机制。

为了提高综合航电系统的安全性，ASAAC 提出利用各种故障探测机制分别监控应用软件层、操作系统层、模块支持层和硬件层等健康状态的思想[53]。对于发生的软硬件故障，可以通过运行时蓝图提供的信息进行故障屏蔽、故障定位、故障隔离或系统重构等。此项技术目前还处于理论研究阶段，尚未工程实现。

5 软件安全性分析与评估

针对软件开发和软件产品（即“V”或“Y”模型的左侧和右侧）进行安全性分析与评估的目的不同。后者主要是通过一定的分析和评估手段，确认软件是否实现了期望的安全性或是否满足了安全性需求，这通常由第三方开展，实际上就是软件安全性认证活动。有关软件安全性认证内容参见第 6 节。此处着重论述针对软件开发的安全性分析与评估方法。

需求是“纲”，设计是“目”，纲举目张是软件实现安全性的根本前提。Leveson[8]认为，软件设计缺乏远见以及规约错误是影响软件安全性的最主要原因。文献[35]通过单元测试搜集的证据以及文献[38]对 1980－1996 年 8 大灾难性事故原因的分析结果共同证实了这一论断。由此，目前针对软件开发的安全性分析与评估工作主要集中在软件需求规约和体系结构设计两个阶段。

在软件安全性需求分析方面，Leveson[8]认为可以采用实时逻辑、时间 Petri 网、故障树分析以及软件故障树分析等技术。不过实践证明，上述方法都与软件的复杂度有关，普遍存在工作量较大的问题。很多学者也探讨了需求规约和分析的改进方法，比如 p-time Petri 网方法[39]、支持向量机和故障树相结合的方法[40]。由 Leveson 团队开发的 SpecTRM 工具集是目前解决需求分析的有效工具之一。

在软件体系结构的安全性评估方面，文献[41]提出的轻量级 PSSA 软件体系结构安全性分析方法能够节省工作量和费用。文献[42]给出了一种基于体系结构权衡分析法的软件体系结构安全性评估方法。文献[43]提出了一种利用灾难性事件覆盖度评估软件安全性的思想，不过未能给出软件危险分析过程。

从确保软件安全性分析的效果看，文献[44]认为软件安全性分析方法至少应：①提供基本的形式化模型——避免需求规约的模糊性；②具有合成性能——通过软件构件组合决定系统的安全性；③采用系统建模方法——软件、硬件甚至人的行为都以统一的或综合的模型进行建模；④具有充分的表达能力——能够以较为直接的方式捕获公共失效场景，比如顺序失效、单点失效；⑤提供自动化支持——能够以最小的代价重复安全性分析。

目前开展的很多工作都可归结到这 5 个方面。比如文献[45]给出了基于进程代数模型和基于信任逻辑模型的安全性分析方法，文献[35,46]也通过构建基于模糊集、贝叶斯网络等的形式化模型开展软件安全性评估项目。英国 York 大学开发的失效传播和变换符号（failure propagation and transformation notation，FPTN）[47,48]是一项颇具代表性的形式化安全性分析技术。在系统建模和自动化支持方面，SpecTRM 工具集也是典型工具之一。

6 软件安全性认证

认证工作一般由第三方展开，主要通过一系列的验证活动确保产品遵循了相应的设计标准、能够满足需求并且不违反相关法律法规。SCS 软件的安全性认证一般要求确认 SCS

软件是否满足了安全性需求。

6.1 基于过程的认证

在航空航天领域，认证 SCS 软件是否满足安全性需求的一种常用方法是，论证该软件开发过程是否遵循了某些面向过程的标准，比如 DO-178B，IEC61508[49] 等。这些标准一般根据目标软件的开发确信级别（development assurement level，DAL）或安全性完整级别（safety integrity level，SIL）定义相应的开发、测试过程。只要软件遵循了相应级别的开发、测试过程，即认为软件达到了相应的 DAL 或 SIL，并不给出安全性需求得到满足的直接证据。这种认证方法一般称之为基于过程的认证。

尽管遵循严格的开发过程在某种程度上确实能够减少软件中的残留致险缺陷数，但这并不能说明软件的 DAL 或 SIL 级别越高，失效率就越低，因为其间未必存在必然联系。此外，基于过程的认证方法既未能给出安全性级别分配指南，也未能说么推荐各级别所采用工具的依据。

6.2 基于产品证据的认证

与面向过程的认证方法不同的是，英国 York 大学 McDermid 领导的团队致力于开发软件产品安全性案例，挖掘软件安全性需求得到满足的直接证据，从而论证软件的安全可接受性。这种基于产品证据的认证思路是：①识别软件在系统上下文能够触发危险的所有失效模式；②提供证据表明这些失效模式要么不能发生，要么发生概率极小、是可接受的，要么能被探测并可以控制其影响。

为了形式化描述基于产品证据的认证方法，McDermid 团队开发了目标结构化构造符号（goal structuring notation，GSN）[31,50]，通过由上而下的安全性目标结构化分解过程，将系统、软件的安全性目标和需求派生到各个软件构件上，由其提供满足安全性需求的证据，进而论证软件或系统的安全性。

值得说明的是，在基于产品证据的认证方法中，不断细化的安全性目标最终依赖的可直接参考的证据可能是基于某种过程标准开发实现的解决方案。换言之，除非这些可直接参考的证据都是客观事实，否则与基于过程的认证方法相比，基于证据的方法只是将假设安全的对象进行了细粒度划分，并未从本质上解决安全性认证问题。

尽管基于产品证据的认证方法富有吸引力，但如果用来替代基于过程的认证，还需要解决下述问题：(1)如何确保安全性案例中的论证的正确性。Greenwell 的工作表明很多安全性案例中存在推理逻辑方面的谬误[51]；(2)当前安全性案例中可接受的证据类型形态各异，缺乏安全可接受的证据类型的界定。比如有些安全性案例[52]认为代码达到 MC/DC 覆盖测试即认为满足了规约，或者认为自动编码器生成的源代码能够满足需求规约；(3)目前安全性案例的接受规则尚不明确，同时存在主观过程[9]。

6.3 累计认证

随着模块化 COTS 产品在安全性关键的软件密集型系统中的应用越来越多，针对因系统变更（比如软硬件升级）而产生的安全性二次认证问题，传统的认证方法面临着巨大的经济压力。以综合航电系统为例示之：①传统的认证方法主要基于系统静态配置生成安全性案例。而综合航电系统具有配置灵活的特点，动态生成的有效配置多达数百万种。穷举所有的航电配置并开发相应的安全性案例从经济、时间等代价上而言是不明智的。②在系统局部变更时，传统的安全性认证方法要求重新认证整个系统的安全性。但对于综合航电系统而言，往往只是数量有限的 COTS 产品存在安全性问题，而绝大数 COTS 产品都是安全的，无须再次认证。鉴于综合航电系统软件规模巨大，软件局部变更后重新认证整个系统的安全性势必代价不菲。

解决上述问题的一种可能途径是采用北约标准 STANAG 4626[53] 提出的累计认证（incremental certification）方法。其初衷是对于局部变更的综合航电系统，寻求只对变更部分进行二次安全性认证就能确保整个系统安全的方法。如此，二次认证的代价将只与系统变更部分的规模和复杂性有关，而与系统整体无关。较之传统的认证方法，累计认证将能节约大量二次认证费用。

安全性累计认证从 2005 年提出至今，尽管其技术和过程尚不成熟，但其思想却展现了巨大的活力，多个组织都在积极开拓研究。比如，英国国防技术战略部（defence technology strategy）目前将安全性累计认证作为一项关键的技术优势项目正在加紧推进研究。欧洲工业航电工作组（industrial avionics work group，IAWG）、欧洲民用航电设备局（EUROCAE）和美国航空无线电技术委员会（RTCA）也就安全性累积认证的方法、过程、应用进行了探讨。

实现安全性累积认证的一种可能方法是模块化方法。2002 年 NASA 发布的模块化认证[54]报告就飞机中使用的软件构件的模块化认证工作进行了研究，提出了基于假设-保证推理（assume-guarantee reasoning）的认证方法。该方法尽管比较合理，但并不完善，因为其定义的规则并不能用于正确的系统。IAWG 认为需要 4 步才能实现模块化累计认证[55]：①识别变更场景；②定义安全性案例结构，并进行优化；③识别出所有的依赖-保证关系（dependency-guarantee relation）；④生成安全性论据。依托 IAWG 开发模块化安全性案例的经验和教训，文献[56]就安全性模块化累积认证的原因、方法、时机、人员等基本问题进行了概述。英国国防部标准 DS 00-56 issue 3[57] 定义了模块化安全性案例的开发环境与管理过程。基于模块化认证研究，EUROCAE 和 RTCA 联合制订了 ED-124/DO-297 标准“综合模块化航空电子开发指南与认证考虑”[58]，提供了一种将综合航电系统集成到飞机上的认证方法，其中包含了软件安全性模块化认证方面的考虑。

结束语 本文试图从原因、定义、度量模型、开发过程、设计方案、评估方法和认证技术 6 个方面总结软件安全性 20 余年的发展状态和成果。尽管时隔多年，Leveson[8] 有关软件安全性的论述至今仍然成立：“原因很好理解，定义仍然备受争议，如何实现悬而未决”。

结合本文有关软件安全性的论述，我们认为进一步值得研究的方向包括：

(1)建立统一的软件安全性度量模型，用作软件安全性评估的依据。比如，建立一种与提供满足安全性需求的证据数量和质量、是否遵循标准化开发和测试过程、软件开发环境的可信性以及软件是否包含容错、纠错机制等有关的软件安全性定量测度模型。

(2)制定刚性的安全性认证标准，加强安全性累计认证的科学理论研究，包括正交的证据类别、证据的可信性、证据的获取方法等。

(3)鉴于软件安全性还涉及社会问题[8]，技术手段只能提供有限的解决方案[13,59]，建立一套集技术、管理和组织于一体的软件开发方法将是十分必要的。

参 考 文 献

[1] Storey N R. Safety critical computer systems[M]. Boston: Addison-Wesley Longman publishing Co., Inc., 1996

[2] 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[J]. 航空学报，2009，30(10)：1912-1917

[3] 沈玉龙，崔西宁，马建峰，等. 综合化航空电子系统可信软件技术[J]. 航空学报，2009，30(5)：938-945

[4] Leveson N G. The role of software in spacecraft accidents[J]. Journal of Spacecraft and Rockets，2004，41(4)：564-575

[5] 422nd Test and Evaluation Squadron. Executive summary: aircraft accident investigation，F/A-22 S/N 00-4014[EB/OL]. http://www.f-22raptor.com/pdf/af.exsum_f22crash.pdf. 2004-12-20

[6] Defense Industry Daily. F-22 squadron shot down by the International Date Line[EB/OL]. http://www.defenseindustrydaily.com/f22-squadron-shot-down-by-the-international-date-line-03087/，2007-3-1

[7] USAF. MIL-STD-1574A—1979 System safety program for space and missile system[S]. Arlington: Department of Defence，1979

[8] Leveson N G. Software safety: why，what，and how[J]. Computing Surveys，1986，18(2)：125-163

[9] Heimdahl M. Safety and software intensive systems: challenges old and new[C]// 2007 Future of Software Engineering(FOSE'07). Washington DC: IEEE Computer Society，2007：137-152

[10] McDermid J A. Software safety: where's the evidence? [C]// Proceedings of the 6th Australian Workshop on Safety Systems and Software. Brisbane: Australian Computer Society，2001，3：1-6

[11] 总装备部. GJB/Z 102—2004 军用软件安全性分析指南[S]. 北京：总装备部，2004

[12] Department of Defence. MIL-STD-882B—1984 Software safety program requirements[S]. Arlington: Department of Defence，1984

[13] National Aeronautics and Space Administration. NASA-STD-8719.13A—2003 Software safety NASA technical standard[S]. Washington DC: National Aeronautics and Space Administration，2003

[14] Swarup M B，Ramaiah P S. An approach to modeling software safety in safety-critical systems[J]. Journal of Computer Science，2009，5(4)：311-322

[15] Avizienis A，Laprie J C，Randell B. Fundamental concepts of dependability[R]. UCLA CSD Report No. 010028. Log Angeles: University of California，2000

[16] 杨仕平，桑楠，熊光泽. 安全关键软件的防危性测评技术研究[J]. 计算机学报，2004，27(4)：442-450

[17] 方滨兴，陆天波，李超. 软件确保研究进展[J]. 通信学报，2009，30(2)：106-117

[18] Lyu M R. Software reliability engineering: a roadmap[C]// 2007 Future of Software Engineering(FOSE'07). Washington DC: IEEE Computer Society，2007：153-170

[19] Devanbu P T，Stubblebine S. Software engineering for security: a roadmap[C]// Proceedings of the Conference on the Future of Software Engineering. New York: ACM，2000：227-239

[20] Chess B，Mcgraw G. Software security[J]. IEEE Security and Privacy，2004，2(2)：53-84

[21] Westmark V R. A definition for information system survivability[C]// Proceedings of the 37th Annual Hawaii International Conference on System Sciences(HICSS'04). Washington DC: IEEE Computer Society，2004，9：303-312

[22] Knight J C，Strunk E A，Sullivan K J. Towards a rigorous definition of information system survivability[C]// Proceedings of the DARPA Information Survivability Conference and Exposition (DISCEX'03). New York: IEEE Computer Society，2003，1：78-89

[23] Littlewood B，Strigini L. Software reliability and dependability: a roadmap[C]// Proceedings of the Conference on the Future of Software Engineering. New York: ACM，2000：175-188

[24] 丽萌. 安全性苛求系统中关于软件安全性评价的研究[J]. 计算机工程与科学，2002，24(2)：59-61

[25] Radio Technical Commission for Aeronautics，Inc. DO-178B—1992 Software Considerations in Airborne Systems and Equipment Certifications[S]. Washington DC: Radio Technical Commission for Aeronautics，Inc，1992

[26] Ericson C A II. Hazard analysis techniques for system safety [M]. Hoboken: John Wiley & Sons，Inc.，2005

[27] Fenton N，Neil M. Measuring your risks: numbers that would make sense to Bruce Willis and his crew[EB/OL]. http://www.agenarisk.com/resources/white_papers/Masuring_Riks.pdf，2009-11-19

[28] 覃志东，刘晓强，王洪亚. 基于关联风险剖面的软件防危性增长测试[J]. 系统工程与电子技术，2009，31(3)：686-690

[29] Pumfrey D J. The principled design of computer system safety analyses[D]. Heslington: University of York，2000

[30] Biglari H. Past，present and future of safety-critical real-time embedded software development[M]. Frederick: Fairchild Control Corporation，2008

[31] Conmy P M. Safety analysis of computer resource management of software[D]. Heslington: University of York，2005

[32] Society of Automotive Engineers. ARP4754—1996 Certification Considerations for Highly-integrated or Complex Aircraft Systems[S]. Warrendale: Society of Automotive Engineers，Inc，1996

[33] Society of Automotive Engineers. ARP4761—1996 Guidelines and methods for conducting the safety assessment process on civil airborne systems and equipment[S]. Warrendale: Society of Automotive Engineers，Inc，1996

[34] Leveson N G. An approach to designing safe embedded software [M]. London: Springer Verlag，2002，LNCS 2491：15-29

[35] Tribble A C，Lempia D L，Miller S P. Software safety analysis of a flight guidance system [R]. NASA/CR-2004-213004. Washington DC: National Aeronautics and Space Administration，2004

[36] Harrison K J. Static code analysis on the C-130J Hercules safety critical software[C]// Proceedings of the 17th International System Safety Conference(ISSC'99). Orlando(USA): System Safety Society，1999

[37] Anderson T. Safe and secure computting systems[M]. Hoboken: Blackwell Scientific Publications，1989

[38] 杨仕平. 分布式任务关键实时系统的防危(safety)技术研究[D]. 成都：电子科技大学，2004

[39] Dutilleul S C，Defossez F，Bon P. Safety requirements and p-time Petri nets: a level crossing case study[C]// IMACS Multiconference on Computational Engineering in Systems Applications. Beijing: IEEE Computer Society，2006：1118-1123

[40] Koh K Y，Seong P H. SMV model-based safety analysis of software requirements[J]. Reliability Engineering and System Safety，2009，94(2)：320-331

[41] Lisagor O，McDermid J A，Pumfrey D J. Safety analysis of software architectures-"Lightweight PSSA"[C]// Proceedings of the 22nd International System Safety Conference. Providence: System Safety Society，2004：1-10

(下转第 27 页)

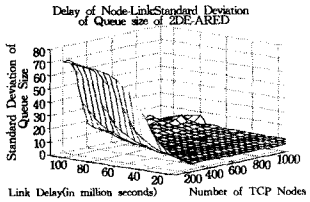

图 12 2DE-ARED 算法的队长标准差

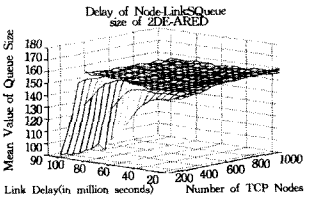

图 13 2DE-ARED 算法的队长平均值

图 14 和图 15 分别给出 PID 算法的瞬时队列长度的标准差和队长平均值。对比图 12 和图 14，相比之下，2DE-ARED 能够较为有效地降低瞬时队列长度的波动性，并且在瞬时队列长度的均值方面也有比较好的表现。

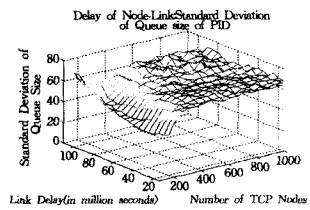

图 14 PID 控制器的队长标准差

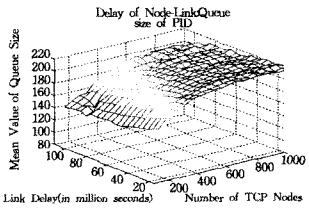

图 15 PID 控制器的队长平均值

结束语 本文首先介绍二阶差分方程的特点，选择连续域-离散化的方法来设计二阶控制器，给出二阶传递函数形式。然后根据控制器的设计目标来确定主要的参数，并确定系统的二阶传递函数，将传递函数转换为离散域的脉冲传递函数，求得相应的差分方程。最后在 NS2 上通过仿真实验验证了算法的正确性和有效性。

参 考 文 献

[1] Jacobson V. Congestion avoidance and control [J]. ACM SIGCOMM Computer Communication Review, 1995, 25(1): 157-187

[2] Floyd S, Jacobson V. Random early detection gateways for congestion avoidance[J]. IEEE/ACM Transactions on Networking, 1993, 1(4): 397-413

[3] 戴航，慕德俊，王林，等. Internet 拥塞控制系统在不同源控制算法作用下的资源竞争分析[J]. 计算机科学，2009，36(11)：40-42

[4] Floyd S, Gummadi R, Shenker S. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management[OL]. http://www.icir.org/floyd/papers/adaptiveRed.pdf, August 2001

[5] Hollot C, Misra V, Towsley D, et al. A control theoretic analysis of RED[C]// Proceedings of IEEE INFOCOM. 2001, 3: 1510-1519

[6] Zheng B, Atiquzzaman M. A framework to determine bounds of maximum loss rate parameter of RED queue for next generation routers[J]. Journal of Network and Computer Applications, 2008, 31(4): 429-445

[7] Zamani M, Karimi-Ghartemain M, Sadati N, et al. Design of a fractional order PID controller for an AVR using particle swarm optimization[J]. Control Engineering Practice, 2009, 17(12): 1380-1387

[8] 陈晓龙，章云，刘治. 多 TCP 协议多链路端拥塞控制特性研究[J]. 计算机科学，2009，36(5)：88-91

(上接第 13 页)

[42] Bate I, Hawkins R, McDermid J. A contract-based approach to designing safe systems[C]// Proceedings of the 8th Australian Workshop on Safety Critical Systems and Software (SCS'03). Darlinghurst: Australian Computer Society, 2003, 33: 25-36

[43] Kornecki A J. Assessment of software safety via catastrophic events coverage[C]// Proceedings of the 21st IASTED International Multi-conference Applied Informatics (AI 2003). Innsbruck(Austria): ACTA Press, 2003: 1139-1144

[44] Wu W, Kelly T. Failure modeling in software architecture design for safety[C]// Proceedings of the 2005 Workshop on Architecturing Dependable Systems. New York: ACM, 2005: 1-7

[45] 杨捷. 安全软件体系结构的形式化方法研究[D]. 武汉：武汉大学，2004

[46] Bieber P, Bougnol C, Castel C, et al. Safety assessment with AltaRica-lessons learnt based on two aircraft system studies[C]// 18th IFIP World Computer Congress. Toulouse: Springer Verlag, 2004, 156: 505-510

[47] Fenelon P, McDermid J. An integrated toolset for software safety analysis[J]. Journal of Systems and Software, 1993, 21(3): 279-290

[48] Fenelon P, McDermid J A, Nicholson M, et al. Towards integrated safety analysis and design[J]. ACM SIGAPP Applied Computing Review, 1994, 2(1): 21-32

[49] International Electrotechnical Commission. IEC 61508—1999 Fundamental safety of electrical/electronic/programmable electronic safety related systems[S]. Geneva: International Electrotechnical Commission, 1999

[50] Kelly T P. Arguing safety-a systematic approach to managing safety cases[D]. Heslington: University of York, 1999

[51] Greenwell W S, Knight J C, Holloway C M, et al. A taxonomy of fallacies in system safety arguments[C]// Proceedings of the 24th International System Safety Conference. Albequerque: System Safety Society, 2006

[52] Nelson S. Certification processes for safety-critical and mission-critical aerospace software[R]. NASA/CR-2003-212806. Washington DC: National Aeronautics and Space Administration, 2003

[53] NATO. STANAG 4626—2005 Modular and open avionics architecture[S]. Brussels: Military Agency for Standardization, 2005

[54] Rushby J. Modular certification [R]. NASA/CR-2002-212130. Washington DC: National Aeronautics and Space Administration, 2002

[55] Industrial Avionics Working Group. Modular Software Safety Case Process[R]. IAWG-AJT-301. Leatherhead: Industrial Avionics Working Group, 2007

[56] Fenn J L, Hawkins R D, Kelly T P, et al. The who, where, how, why and when of modular and incremental certification[C]// 2nd IET International Conference on System Safety. London: IET Software, 2007, CP532: 135-140

[57] Ministry of Defence. DEF-STAN 00-56 draft issue 3—2004 Safety Management Requirements for Defence Systems [S]. London: Ministry of Defence, 2004

[58] Radio Technical Commission for Aeronautics, Inc. RTCA DO-297—2005 Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations[S]. Washington DC: Radio Technical Commission for Aeronautics, Inc., 2005

[59] Alberico D, Bozarth J, Brown M, et al. Software system safety handbook[M]. Patuxent River (USA): Joint Services Software Safety Committee, 1999