

高可信软件工程技术

陈火旺,王 戟,董 威

(国防科技大学计算机学院,湖南长沙 410073)

摘 要: 随着软件在信息社会中发挥日益重要的作用,人们对软件可靠性、可靠安全性和保密安全性等可信性质的要求也愈来愈高.本文讨论了高可信软件工程技术现状和面临的主要挑战,给出了基于形式化方法的高可信软件技术的发展趋势和突破点.

关键词: 软件工程;软件可信性;软件保证

中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2003) 12A-1933-06

High Confidence Software Engineering Technologies

CHEN Huo-wang, WANG Ji, Dong Wei

(School of Computer, National University of Defense Technology, Changsha, Hunan 410073, China)

Abstract: As the software plays more and more important roles in today's information society, the software reliability, safety and security are strongly required. The state of art of its engineering technologies for this high confidence software and the challenges it faced are described in this paper. Based on the formal methods, this confidence software developing trends and its key technical points are also discussed.

Key words: software engineering; high confidence software; software assurance

(上接封二)

科研通信

基于小波压缩域的统计纹理特征提取方法	李晓华, 沈兰荪 (2123)
基于星载电子侦察与成像侦察的数据融合技术	王 壮, 樊 昀, 王 成, 康少单, 孙兆林, 胡卫东, 郁文贤 (2127)
基于指纹特征数字水印算法的身份认证技术研究	张毅刚, 焦玉华, 牛夏牧, 俞龙江 (2131)
运用遗传算法综合稀疏阵列	王玲玲, 方大纲 (2135)
基于免疫算法的 E 面波导分支耦合器优化技术	许 殿, 史小卫 (2139)
并行共形 HDID 算法及其在 PBG 结构仿真中的应用	张 玉, 宋 健, 梁昌洪 (2142)
任意平面阵列的干扰抑制	李 平, 史小卫 (2145)
高动态扩频信号的载波跟踪技术研究	程乃平, 任宇飞, 吕金飞 (2147)
特定领域软件框架的提取方法研究	吴毅坚, 赵文耘 (2151)
一种模型选择优化准则及其在高光谱图像非监督分类中的应用	吴 昊, 郁文贤, 匡纲要, 李智勇 (2154)
基于协方差控制的集中式传感器分配算法研究	周文辉, 胡卫东, 余安喜, 郁文贤 (2158)
基于 Beta-Prime 统计模型和 QGD 分类器的 SAR 图像地物分类方法	付 琨, 孙真真, 吴一戎 (2163)
一类可由基本模型嵌套组成的工作流系统的性能分析方法	田立勤, 林 闯 (2167)
FKAOS: 一种面向 Agent 需求工程方法	刘宗田, 邵 翀, 孙志勇, 刘 炜 (2171)
混合维空间遮蔽关系的表示与推理	王生生, 刘大有 (2175)
一种可变测试集的协议一致性测试方法	吕欣岩, 赵保华, 屈玉贵 (2179)
编队飞行 InSAR 的平地效应与地形高度效应分析	唐 智, 李景文 (2183)
基于 IP 突发信源模型数据包的定量计算研究	刘晏兵, 孙世新, 唐 红 (2187)
可行方向算法与模拟退火结合的 NMF 特征提取方法	陈卫刚, 戚飞虎 (2190)
多级弯曲磁微梁执行器力-磁耦合宏模型	方玉明, 黄庆安, 李伟华 (2194)
自主机器人视觉与行为模型及避障研究	梁 冰, 洪炳, 曙 光 (2197)
4H-SiC npn BJT 特性研究	龚 欣, 张进城, 郝 跃, 张晓菊 (2201)
多天线系统的广义谐振研究	李 龙, 梁昌洪, 史 琰 (2205)
一种对光照变化鲁棒性的彩色目标检测方法	曙 光, 洪炳, 梁 冰 (2210)
X 波段 MEMS 膜开关的阻抗分析模型	郑惟彬, 黄庆安, 廖小平, 李拂晓 (2213)

收稿日期: 2003-09-30; 修回日期: 2003-12-10

基金项目: 国家自然科学基金项目 (No. 90104007, No. 60233020, No. 60303013); 国家 863 项目 (No. 2001AA113202); 霍英东青年教师基金 (No. 71064)

1 引言

随着计算机应用的不断发展,软件已渗透到国民经济和国防建设的各个领域,在信息社会中发挥着至关重要的作用。一方面,软件为人们提供了新的生活和工作方式,软件成为信息基础设施。例如网上银行、电子商务等,离开了软件,这些应用恐怕不会出现。另一方面,软件不仅在逐步地代替传统硬件和物理设备,而且在系统中承担了集成的作用。高技术产品往往是软件密集型系统,小到日常使用的手机,大到联接全球的 Internet 应用,软件成为人们生活的一部分。相应地,人们也越来越依赖软件。然而,软件的生产现状不能令人满意。著名的 Chaos 报告^[1]对美国 8380 个软件项目的统计,仅有 16% 的项目按时按预算完成,有 53% 的软件项目超预算,其余 31% 的项目被取消。与此同时,软件质量不能令人满意。软件常常发生失效,并对人们的工作生活带来不利的影响,甚至造成巨大的损失。例如,软件失效导致 1996 年 6 月欧洲 Ariane 五型火箭首发失败^[2]。因此,人们对软件的正确性、可靠性、可靠安全性和保密安全性等可信性质给予了十分的关注,如何在软件的开发和运行中保证软件具有高可信性质也成为软件理论和技术的重点研究方向。

软件技术指软件开发、运行和维护的技术,其形态可以是概念、过程、方法、算法、工具或环境。软件技术可分为三种不同类型。第一类是软件系统技术,指在软件系统运行中直接使用的软件技术。这类软件技术与具体运行的软件系统类型密切相关,例如操作系统技术、中间件技术等。第二类是软件开发和维护技术,指在软件系统开发和维护中使用的软件技术。这类技术通常与软件生命周期相关。例如,软件面向对象的需求分析、设计、编码、测试技术。第三类是软件过程技术,指通过软件生产线使软件的开发和维护更有效的软件技术。因此,软件的高可信性质的获得和保证需要这三个层面上的技术支撑。例如,从软件系统运行的层面上,软件保密安全性需要操作系统的访问控制和保密安全策略的支撑;从软件系统的开发层面上,软件可靠安全性需要软件形式化规约和验证技术的支撑;从软件过程技术层面上,软件可靠性可以一种工程化的方法在软件可靠性工程中度量和跟踪。

高可信软件系统要求能充分地证明或认证该软件系统提供服务时满足一些关键性质(称为高可信性质)。典型的例子如核电站控制软件,其可靠安全性十分重要,需要给予充分的验证。近十年来,软件可信性研究受到了国际上广泛的重视。在美国, DARPA、NSF、NASA、NSA、NIST、FAA、FDA 和其他 DoD 机构都积极参与关于高可信软件和系统的研究开发。美国 NSIC 先后形成一系列报告: America in the Age of Information: A Forum^[3]、Research challenges in high confidence systems^[4]、setting an interagency high confidence systems (HCS) research agenda^[5]和 High Confidence Software and Systems Research Needs^[6]。高可信软件工程技术是指获得和保证高可信软件系统关键性质的软件开发技术和过程技术。本文主要探讨高可信软件工程技术现状和发展趋势,给出一些需要深入研究的课题。

2 软件可信性质的挑战

软件系统的可信性质是指该系统需要满足的关键性质;当软件一旦违背这些关键性质会造成不可容忍的损失时,称这些关键性质为高可信性质。软件可信性质常常有以下几种。

可靠性(reliability)

在规定的环境下、规定的时间内软件无失效运行的能力;

可靠安全性(safety)

软件运行不引起危险、灾难的能力;

保密安全性(security)

软件系统对数据和信息提供保密性、完整性、可用性、真实性保障的能力;

生存性(survivability)

软件在受到攻击或失效出现时连续提供服务并在规定时间内恢复所有服务的能力;

容错性(fault tolerance)

软件在故障(硬件、环境异常)出现时保证提供服务的能力;

实时性(real time)

软件在指定的时间内完成反应或提交输出的能力。

高可信软件系统中会涉及上述性质的一个或多个。这些可信性质与软件的功能性混合在一起,使得高可信性质的获得和保证变得复杂。究其原因,软件作为人类连续的高度复杂的智力产品,其科学原理和工程规律远未得到充分的认识,从而缺乏有效地生产满足高可信软件的软件技术。事实也表明了这一点,1999 年的 PITAC 报告指出开发可靠和安全软件的技术不足^[7]。

高可信软件工程对目前的软件理论和技术提出了严峻的挑战,涉及到一系列科学问题。

第一、软件系统的行为特征。如何定性/定量地描述软件的行为?如何建立各类复杂的软件结构和系统对应的系统行为?这是软件理论的基本问题。软件的静态语法与其动态语义的分离是造成软件行为难于描述和推理的原因。随着软件规模的增大,软件中并发、实时、分布、移动等特性的出现,这些问题的认识亟待深入。

第二、软件可信性质与软件行为的关系。如何描述软件可信性质及其与软件行为的关系?我们看到,上述关于软件可信性质的描述是非形式化的抽象陈述,必须建立起性质和软件行为之间的内在联系及其严格的描述,才能在软件开发中,设计并验证所需的可信性质。

第三、面向软件可信性质的设计和推理。如何将软件可信性质(通常非操作性的)融入软件设计(操作性)?可信性质通常是软件系统的全局约束,伴随着软件开发过程逐步地“设计”出来,最终获得这些可信性质。如何针对可信性质发现一种分而治之的策略从而控制复杂性,是进行面向可信性质的软件设计和验证的关键。

第四、软件系统的可信性质的确认。如何发现和评估软件系统是否具有可信性质?量化是一种工程科学成熟的重要标志,需要对软件可信性质有合适的度量方法,并能在软件过程

中进行跟踪。

3 高可信软件工程技术现状

高可信软件系统在很大程度上不会因为系统中存在的错误、环境的异常或恶意的攻击而导致系统的失效。这要求系统的行为在软件开发时就能得到准确的把握,即系统的行为是可预计的。这就与软件理论和形式化方法发生了自然和本质的联系。

3.1 高可信软件理论

人们对程序理论开展了长期的研究。早期的成果有 Hoare 的顺序程序的公理化理论^[8],它通过前后置断言给出了顺序程序的部分正确性和完全正确性的形式推理系统。顺序计算语义理论的主要思想是把程序模型看作是输入输出映射。当并发程序计算出现后,程序模型的重点转向系统间的交互及其随时间变化的激励/响应模式,形成了 CSP^[9]、CCS^[10]、I/O 自动机^[11]、标记变迁系统及程序的时序语义^[12]等模型。相应地,在实时计算出现后,一个重要的方向是对上述并发模型进行实时扩充,例如时间自动机^[13]。在这些程序理论中,可以通过定理证明或模型检验的方式验证系统的一些并发性质(例如,安全性和活性),也可以验证系统间的行为关系(例如,互模拟)。近年来,嵌入式应用的发展,推动了关于混合了离散计算和连续计算的混合系统程序理论的研究,并发模型中加入连续计算成分,例如混合自动机^[14]。移动应用推动了程序理论关于移动演算的进展。然而复杂应用的高可信软件的程序理论远未获得共识。

传统工程技术往往能在进行产品生产之前,通过建立模型进行演算、测试和验证,从而保证产品生产出来后具有期望的性质。软件工程与传统工程技术相比,目前尚不具备类似的基础。综观已有的程序理论,多数集中在面向程序正确性,或者若干理论拼接,不足以作为软件可信性质的统一基础。这在技术层面上体现在两个方面。一方面,对于单个的可信性质,分析、设计和保证该性质的技术常常是分离的,存在语义的沟壑。例如,对反应式系统的可靠安全性,有失效模式和影响分析技术、容错计算、调度、形式化验证等技术,但这些技术仍是各自分离的,没有形成一个集成技术框架。另一方面,对多个可信性质,这些性质之间对设计决策会造成冲突,例如,容错性质往往会影响到实时性。目前的程序理论未能作为指导性性质的融合、折衷的基础。

3.2 软件形式化方法

形式化方法的重要目标是以一种严格的工程方法进行软件的开发,它作为一种思想、方法、技术渗透到软件开发的各个活动中。其理论基础是数学化的程序理论。因此程序理论的发展和不足直接影响了形式化方法在工程中的应用。

形式化方法是关于在计算系统的开发中进行严格推理的理论、技术和工具,它主要包括形式化规约技术和形式化验证技术^[15]。统计表明,传统的非形式化的软件工程技术对软件质量的保证具有一个难以逾越的顶点,而形式化方法的实践证明形式化方法是提高软件质量的重要途径。在从高层规范至最终实现的过程中,选用适当的、以形式化方法为基础的工具

具进行辅助设计和验证,对提高安全攸关系统的可信度有很大帮助。

形式化规约技术使用具有严格数学定义语法和语义的语言刻画软件系统及其性质,可以尽早发现需求和设计中的错误、不一致、歧义和不完全。顺序系统的形式化规约技术侧重于对状态空间的描述,其主要思想利用集合、关系和函数等离散结构表达系统的状态,用前后置断言表达状态的迁移,例如 Z^[16]、VDM^[17]。并发系统的形式化规约技术侧重于对系统并发特性的描述,其主要思想是用序列、树、偏序等来表达系统的行为,例如 CSP、CCS、Statecharts^[18]、时序逻辑^[12]。RAISE 语言和方法^[19]则综合了这两种思路。数学上的抽象和组合概念是形式化规约技术的重要基础。

形式化验证是在形式化规约的基础上建立软件系统及其性质的关系,即分析系统是否具有所期望性质的过程,主要分有两种途径:模型检验和定理证明。模型检验技术是通过搜索待验证软件系统模型的有穷状态空间来检验系统的行为是否具备预期性质的一种有穷状态系统自动验证技术^[20,21]。在模型检验中,系统用有穷状态模型建模;其性质规约通常是时序逻辑或模态逻辑公式,也可以用自动机语言描述;通过有效的搜索来检验有穷状态模型是否满足规约,如果不满足,它还能给出使性质公式为假的系统行为轨迹。符号化模型检验技术大大提高了可有效应用模型检验技术的系统规模,使得模型检验在工业界逐步得到应用。模型检验已经成为形式化方法中系统验证的重要途径。模型检验应用面临的主要问题是状态爆炸问题。目前的解决思路有两类,一类途径是通过发现模型的状态空间的结构特点来缓解状态空间爆炸问题,主要的方法包括符号化模型检验、对称模型检验、偏序模型检验、On-the-fly 模型检验等;另一类途径是通过抽象和分解把复杂系统的验证转化成模型检验可以处理的问题,主要的方法包括抽象方法、组合方法等。

模型检验在硬件设计和通信协议的形式化验证上获得了巨大的成功。例如,利用模型检验技术,有效地发现了多处理器 cache 一致性协议中的一些错误,而这些错误在传统的模拟中往往未能发现。著名的基于模型检验的并发系统自动验证工具有 SMV^[22]、SPIN^[23]等。

定理证明技术是将软件系统和性质都用逻辑方法来规约,通过基于公理和推理规则组成的形式系统,以如同数学中定理证明的方法来证明软件系统是否具备所期望的关键性质。基于定理证明的形式化验证技术可以看作是以软件系统为公理获得其性质的证明过程。主要的计算机辅助定理证明工具有 PVS^[24]、HOL^[25]等。近年来,定理证明的自动化程度得到明显的提高。

模型检验技术和定理证明技术优势互补。模型检验技术的重要优势是它的自动化程度高,并且当系统不能具备所期望的性质时,能给出相应的反例路径;而它的缺点是软件系统的状态爆炸问题使得它的扩展性受到较大制约。定理证明技术则在它能够基于无穷域上的归纳法处理无穷状态空间,然而,它在自动化程度上远远不如模型检验,而且难于在证明失败后如同模型检验技术那样提供易于理解的可读反例。形式

化验证技术的一个趋势是将模型检验技术和定理证明技术有机的结合起来,这在软件验证上体现出明显的生命力,例如模型证明技术^[26]。

形式化方法的一个重要优点是能以一种严格的方式保证软件可信性,并且其中的模型检验技术为自动化软件工程提供了新的途径。20世纪90年代以来,在国际上,形式化方法已成为软件开发中重要的可信软件技术之一。

3.3 软件需求分析、设计和测试技术

软件的可信性问题自软件开发以来久已存在。人们在软件工程的实践中从需求分析方法、设计和测试多个方面提出了一些方法来试图从开发的角度获得和评估软件的这些性质。例如,在需求分析中软件安全性分析技术^[27],在软件设计中的软件容错技术^[28],在软件测试中的软件可靠性测试技术^[29]。

软件体系结构是指一组具有共同应用背景的软件系统在结构上的共性抽象。近年来,软件体系结构与构件技术成为开发可信软件的重要途径。例如,在安全攸关软件领域中,多版本冗余结构、控制/监视结构、计算/自测试结构是典型的容错软件体系结构。针对常见应用问题的典型设计,并加以分类总结,人们提出了设计模式的概念,与软件构件技术一起形成了面向对象的软件重用技术。采用典型、成熟的设计模式和经过检验的软件构件有利于提高软件的可信性。软件构件开发技术也为高可信软件的开发提供了有益的实践。

软件测试是通过执行软件来判断软件是否具备所期望的性质,是可信软件开发中一个行之有效的、必不可少的、客观地评估软件可信性的方法。在高可信软件开发中,软件测试的开销往往大于50%。现有的软件动态测试方法可分为随机测试和选择性测试两大类。以数理统计理论为基础的软件随机测试方法,根据软件在使用中输入数据空间的概率分布,随机选取测试数据。选择性测试则根据程序的内在结构和软件功能规约,有目的地选择测试数据,主要方法有控制流测试,数据流测试,功能测试,以及针对软件错误的测试等等。目前,对于可信软件,软件测试技术面临重大挑战。例如,安全攸关软件不仅要求在其环境处于正常状态时保证系统的安全性,而且要求环境处于非正常状态时也能使系统安全地进入安全状态,因而往往难以获得充分的数据来测试软件应付危险情况的能力,不能满足可靠安全性测试的需求。

软件工程中分析、设计和测试技术目前仍是在工程上保障软件高可信性的主要手段,但不能满足高可信软件开发的需要。从基础上看,这些技术需要与高可信软件的程序理论结合,提高技术的形式化程度。从系统性、有效性和适用性上看,它们远未形成高可信软件开发的系统化技术体系。

3.4 软件过程技术

软件开发是一个复杂的过程,通过对软件过程的管理和控制是提高软件质量的重要手段。因此,软件过程对高可信软件开发有着重要的影响。

软件过程模型的研究是90年代软件工程的热点。它是软件生命周期理论研究的继续与发展,旨在发现与总结软件开发过程的客观规律,为软件生产过程的标准化和规范化打下

基础。卡内基梅隆大学软件工程研究所提出的CMM^[30]根据一个软件企业的软件开发过程与生产组织结构的特点来定性刻画其软件生产的成熟程度,并将成熟程度与所能够生产的软件规模、所能够达到的软件质量以及软件生产的开销与质量的稳定性联系起来。

国际上已经开始将软件过程模型与可信软件开发联系起来。例如,面向可靠性,净室软件工程法(Cleanroom software engineering)^[31]将软件开发过程置于统计质量控制之下。它可以用于开发有可靠性认证的高质量软件。其基本点是在增量式软件周期中,将形式化方法的规约、设计和验证与可靠性认证的统计测试有效地结合起来。该方法在软件过程中将可靠性作为工程化的跟踪目标。又如,面向安全性,有软件开发过程中错误发生机制与软件质量定量预测模型(FASGEP model)^[32]。但是,无论是对于软件可信性质的度量,还是在软件过程中进行工程化的跟踪,过程控制策略和技术手段现在都很缺乏。

3.5 小结

David Parnas在1985年文章中,指出了国防战略系统对软件技术提出的要求以及当时软件技术水平的限制^[33]。经过近20年的研究,一方面,软件工程技术取得重要的进展;另一方面,在系统的高可信方面,开销依然巨大,效果不令人满意。高可信软件工程虽然在一些局部上有突破,但未形成系统的、科学的工程方法和技术。

4 形式化工程方法

应该说,数十年的软件工程研究和实践围绕着高可信软件的开发有了初步的成果。本节探讨以形式化方法为基础的高可信软件工程技术的发展趋势和突破点。

形式化方法将对软件可信性的获得和保证有着不可替代的作用。但是,至今,形式化方法在实际的高可信软件的开发中仍不多见,基本处于实验室的试验阶段,并且,其使用者多是专家型用户。软件开发,包括高可信软件的开发,仍以非形式化软件开发方法为主流。因此,如何以一种工程化的方法研究和应用形式化方法和技术是高可信软件工程的发展方向。

需要充分地研究高可信软件的程序理论,以其为基础,寻求可集成化地处理多种软件可信性质需求的方法和技术。同时,这些方法和技术应当是可扩展的,抽象和分解这两种控制软件复杂性的重要手段需要得到有力的支持。因此,要为高可信软件工程技术提供一个统一的框架,同时支持对可信性质的分而治之。在此框架下,形式化方法流派间的有机集成和互操作是一个发展方向。

“轻量级”的形式化方法将是形式化方法工程化的趋势。需要为形式化提供较强的自动化工具支持,即工具具有形式化的基础,但其使用者无须具备很强的形式化知识。工具的人机交互应是自然的、可读的、易于理解的。这关系到,把软件工程中的主流需求分析、设计和测试技术与形式化技术充分的融合。例如,近年的模型检验技术、静态分析技术、形式化测试技术的发展证明了这一点。

将形式化方法和非形式化实用软件开发方法在模型驱动

框架下有机地集成起来. 这要求形式化方法从面向模型向面向模型和程序方向发展. 例如, 模型检验技术已从单纯的验证软件设计模型, 发展到开始验证程序的研究. 对软件可信性的度量进行研究, 在软件过程的每个软件活动中针对软件可信性集成形式化方法和非形式化方法是高可信软件工程的趋势. 基于这些工作, 出现了基于统计学的原理和方法的统计软件工程 (Statistical software engineering)^[34].

从软件开发范型上看, 形式化方法落后于非形式化方法的发展. 例如, 基于中间件的构件化软件开发在一定程度上提高了软件的可信性, 但是这些技术的形式化基础仍有待研究. 对构件化和增量式的有效支持是形式化工程方法的重要基础. 随着分布计算技术的发展, 面向中间件的形式化开发方法成为一个急迫的研究课题^[35]. 与该方向密切相关的, 将软件体系结构、构件化开发方法形式化和科学化, 建立利用构件组装并开发出高可信软件的方法, 将是未来软件工程, 尤其是 Internet 软件开发的重要方向^[36]. 从软件开发的发展看, Agent 软件开发方法将是下一代软件开发范型. 近来, 分布对象技术向面向 Agent 软件工程的方向发展, 已经应用于许多高可信软件的应用领域 (例如自主式航天探测器、飞行控制软件等), 然而 Agent 软件开发方法目前仍未形成. 因此, 开展面向 Agent 的形式化方法是一个重要挑战.

程序设计语言与软件工程技术有着密切的联系. 程序设计语言的变革历来是计算机科学中里程碑式的进步. 从程序设计语言的角度支持高可信软件开发是一个具有挑战性的课题. 例如, Proof Carrying Code^[37]对移动代码的保密安全性、Control-C 对实时控制系统的可靠安全性^[38]进行了有益的尝试. 将程序设计语言、编译技术和形式化验证结合建立 Verifying Compiler 被认为是一个巨大的挑战^[39]. 对于网络嵌入式软件, 支持高可信软件的面向 Agent 的程序设计语言将是一个趋势, 同时程序设计方法学基础层面上的成果 (例如形式验证、运行时验证) 将更多地呈现在语言设施的设计和实现上, 在新的软件开发平台和运行平台上显现出来.

5 结束语

高可信软件技术是当前软件技术面临的重大挑战. 从科学意义上, 它要求人们对软件系统开发和运行等规律有更深一步的认识; 从应用价值上, 它关系到人们在信息社会对信息基础设施的依赖和可信程度的提高.

软件工程以提高软件生产率和软件质量为目标, 高可信软件工程是其面向可信性的重要技术组成. 高可信软件工程是软件工程技术中基础性和前沿性方向, 以形式化方法为基础的软件技术将成为突破点和发展趋势.

参考文献:

- [1] Standish Group. The CHAOS Report[R]. Found at <http://www.standishgroup.com>. 1995.
- [2] The Inquiry Board. Ariane 5 Flight 105 Inquiry Board Report [R]. Paris : European Space Agency Press July 1996.
- [3] National Science, Technology Council (NSTC). America in the Age of Information : A Forum on Federal Information and Communications R&D [R]. Bethesda, Maryland July 6 - 7, 1995.
- [4] NSTC. Research challenges in high confidence systems [A]. Proceedings of the Committee on Computing, Information, and Communications Workshop [C]. USA : <http://www.hpcc.gov/pubs/hcs-Aug97/intro.html>, August 6 - 7, 1997.
- [5] High Confidence Systems Working Group, NSTC. Setting an interagency high confidence systems (HCS) research agenda [A]. Proceedings of the Interagency High Confidence Systems Workshop [C]. Arlington, Virginia, 25 March 1998.
- [6] High Confidence Software and Systems Coordinating Group. High Confidence Software and Systems Research Needs [R]. USA : <http://www.ccic.gov/pubs/hcss-research.pdf>, January 10, 2001.
- [7] President 's Information Technology Advisory Committee. Information Technology Research : Investing in Our Future [R]. Report to the President, USA : <http://www.cs.rice.edu/~ken/presentations/PITAC.pdf>, February 24, 1999.
- [8] C A R Hoare. An axiomatic basis for computer programming [J]. Communications of the ACM, 1969, 12(10) : 576 - 580.
- [9] C A R Hoare. Communicating Sequential Processes [M]. Prentice-Hall International Series in Computing Science, Prentice-Hall International, Englewood Cliffs, N J London, 1985.
- [10] Robin Milner. A Calculus of Communicating Systems [M]. USA : Springer, 1980.
- [11] N Lynch, M Tuttle. An introduction to input/output automata [J]. CWI Quarterly, 1989, 2(3) : 219 - 246.
- [12] Zohar Manna, Amir Pnueli. The Temporal Logic of Reactive and Concurrent Systems : Specification [M]. Berlin : Springer-Verlag, 1992.
- [13] R Alur. Timed automata [A]. 11th International Conference on Computer-Aided Verification [C]. LNCS 1633, Berlin : Springer-Verlag, 1999. 8 - 22.
- [14] R Alur, C Courcoubetis, T A Henzinger, P H Ho. Hybrid automata : An algorithmic approach to the specification and verification of hybrid systems [A]. Hybrid Systems [C]. LNCS 736, Berlin : Springer-Verlag, 1993. 209 - 229.
- [15] Edmund M Clarke Jeannette M Wing. Formal methods : State of the art and future directions [J]. ACM Computing Surveys, 1996, 28(4) : 626 - 643.
- [16] J M Spivey. The Z Notation : A Reference Manual [M]. New Jersey : International Series in Computer Science, Prentice Hall, 1989.
- [17] C B Jones. Systematic Software Development Using VDM [M]. New Jersey : Prentice Hall International Series in Computer Science, Prentice Hall, 1986.
- [18] David Harel. Statecharts : A visual formalism for complex systems [J]. Science of Computer Programming, 1987, 8(3) : 231 - 274.
- [19] Mogens Nielsen, Klaus Havelund, Kim Ritter Wagner, Chris George. The RAISE language, method and tools [J]. Formal Asp. Comput, 1989, 1(1) : 85 - 114.
- [20] Edmund Clarke, Bernd Hölger Schlingloff. Model checking [A]. Handbook of Automated Reasoning [M]. Edited by Alan Robinson and Andrei Voronkov, Boston : MIT Press, 2001. 1635 - 1790.
- [21] 林惠民、张文辉. 模型检测 : 理论、方法与应用 [J]. 电子学报, 2002, 30(12A) : 1907 - 1912.

- [22] KL McMillan. Symbolic model checking—an approach to the state explosion problem[D]. SCS, Carnegie Mellon University, 1992.
- [23] Gerard J Holzmann. The Spin Model Checker Primer and Reference Manual[M]. Boston: Addison-Wesley, 2003.
- [24] S Owre, J M Rushby, N Shankar. PVS: A prototype verification system [A]. D Kapur, editor. Automated Deduction (CADE-11) [C]. volume 607 of Lecture Notes in Computer Science, Berlin: Springer-Verlag, 1992. 748 - 752.
- [25] Michael J C Gordon. Introduction to the HOL system[A]. TPHOLs [C]. New York, USA: IEEE Computer Society, 1991. 2 - 3.
- [26] Sergey Berezin. Model Checking and Theorem Proving: a Unified Framework[D]. Also a TR number CMU-CS-02-100. Carnegie Mellon University, 2002.
- [27] NASA. Software Safety NASA Technical Standard. NASA-STD-8719. 13A[S]. September 15, 1997.
- [28] Michael R Lyu. Software Fault-Tolerance[M]. New York: John Wiley, 1995.
- [29] John Musa. Software Reliability Engineering[M]. New York: McGraw-Hill, 1999.
- [30] Mark C Paulk, Charles V Weber, Bill Curtis, Mary Beth Chrissis. The Capability Maturity Model Guidelines for Improving the Software Process[M]. Boston: Addison-Wesley Publishing Company, 1998.
- [31] Stacy J Prowell, Carmen J Trammell, Richard C Linger, Jesse H Poore. Cleanroom Software Engineering: Technology and Process[M]. Boston: Addison Wesley Professional, 1999.
- [32] Mike Falla. Advances in Safety Critical Systems—Results and Achievements from the DTI/ EPSRC R&D Programme in Safety Critical Systems [R]. Lancaster University Computing Department, <http://www.comp.lancs.ac.uk/computing/resources/scs/index.html>, 1997.
- [33] David L Parnas. Software aspects of strategic defense systems[J]. Communications of the ACM, 1985, 28(12): 1326 - 1335.
- [34] Siddhartha R Dalal, Jesse H Poore, Michael L Cohen. Innovations in Software Engineering for Defense Systems[M]. Washington: the National Academies Press, D. C., 2003.
- [35] John Hatcliff, William Deng, Matthew Dwyer, Georg Jung, Venkatesh Prasad. Cadena: An integrated development, analysis, and verification environment for component-based systems[A]. Proceedings of the International Conference on Software Engineering (ICSE 2003) [C]. New York: IEEE Press, 2003. 160 - 173.
- [36] 杨芙清、梅宏、吕建、金芝. 浅谈软件技术发展[J]. 电子学报, 2002, 30(12A): 1901 - 1906.
- [37] George C Necula, Peter Lee. Safe kernel extensions without run-time checking[A]. 2nd Symposium on Operating Systems Design and Implementation (OSDI '96) [M]. Seattle, WA, October, 1996. 229 - 243.
- [38] Sumant Kowshik, Dinakar Dhurjati, Vikram Adve. Ensuring code safety without runtime checks for real-time control systems[A]. Proc Int Conf on Compilers, Architecture and Synthesis for Embedded Systems (CASES02) [C]. Grenoble, France, Oct. 2002. 288 - 297.
- [39] Tony Hoare. The verifying compiler: A grand challenge for computing research[J]. JACM, 2003, 50(1): 25 - 35.

作者简介:



陈火旺 男, 1936 年生于福建安溪, 中国科学院院士, 教授, 博士生导师, 主要从事软件工程、系统软件和人工智能基础等方面的研究。



王 戟 男, 1969 年生于上海, 教授, 博士生导师, 主要从事高可信软件技术、软件工程、语义 Web 等方面的研究。



董 威 男, 1976 年生于陕西咸阳, 讲师, 博士, 主要从事高可信软件技术、软件工程等方面的研究。