安全攸关系统的领域建模与形式化验证方法的研究进展与趋势

CCF 抗恶劣环境计算机专业委员会

马殿富[1] 牛文生[2] 程 胜[3] 马 中[4] 宋 富[5] 罗 杰[1] 葛 宁[1]
陆 平[1] 牟 明[2] 王 闯[2] 邱化强[3] 唐忆滨[4] 戴新发[4]

[1]北京航空航天大学，北京
[2]中航机载共性技术工程中心，扬州
[3]北京神舟航天软件技术有限公司，北京
[4]武汉数字工程研究所，武汉
[5]上海科技大学，上海

摘 要

本文从安全攸关软件系统的领域建模方法和形式化验证两个方面出发，对领域建模语言设计，安全攸关软件系统的需求、架构设计和详细设计建模的研究和应用进展进行了分析，对形式化验证方法在操作系统、CPU、编译器和软件模型四个层面上的研究进展和发展趋势进行了分析，并提出了未来可能的研究方向。同时，本文也对领域建模和形式化验证方法在我国航空、航天、航海三个领域的应用现状进行了调研分析，整理了这些领域未来的需求，并提出了发展建议。

关键词：安全攸关系统，领域模型，形式化验证，航空，航天，航海

Abstract

This paper analyzes the research and application progress of domain modeling and formal verification of safety critical software system. The designing of domain modeling language, and requirements, architecture design and detailed design modeling of safety critical software system is analyzed first. Then, the research progress and development trend of formal verification methods in operating system, CPU, compiler and software model are analyzed, and the possible future research directions are proposed. At the same time, this paper also investigated and analyzed the application status of domain modeling and formal verification methods in the fields of aviation, aerospace and navigation in China, summarized the future needs of these fields, and put forward development suggestions.

Keywords: safety critical system, domain model, formal verification, aviation, aerospace, navigation

1 安全攸关软件系统的领域建模方法

1.1 安全攸关软件系统的领域建模研究进展

特定领域语言（Domain-Specific Language，DSL）是指专用于某个特定应用领域的计算机语言，包括特定领域的标记语言 HTML、特定领域的建模语言 MATLAB 和特定领域的编程语言 Make 等。用户可以根据特定的应用领域需求自定义 DSL。与 DSL 相对的是适用于多类应用领域的通用语言（General-Purpose Language，GPL），如标记语言 XML、建模语言 UML 以及编程语言 C 和 Java 等。特定领域建模语言（Domain-Specific Modeling Language，DSML）用于建立更为抽象的领域模型。

本文主要针对安全攸关软件领域的建模语言进行调研，总结其研究和应用进展，并展望未来发展趋势。如图 1 所示，对于软件开发而言，建模方法可以支撑系统分析设计及软件配置项定义、需求分析、架构设计、详细设计等各个开发阶段，其中前两个阶段为问题域，解决“做什么”的问题，后两个阶段为方案域，融合领域设计经验，解决“怎么做”的问题。本文将重点调研其中的领域建模语言定义方法、软件需求建模语言、软件架构设计建模语言以及软件详细设计建模语言的研究进展。

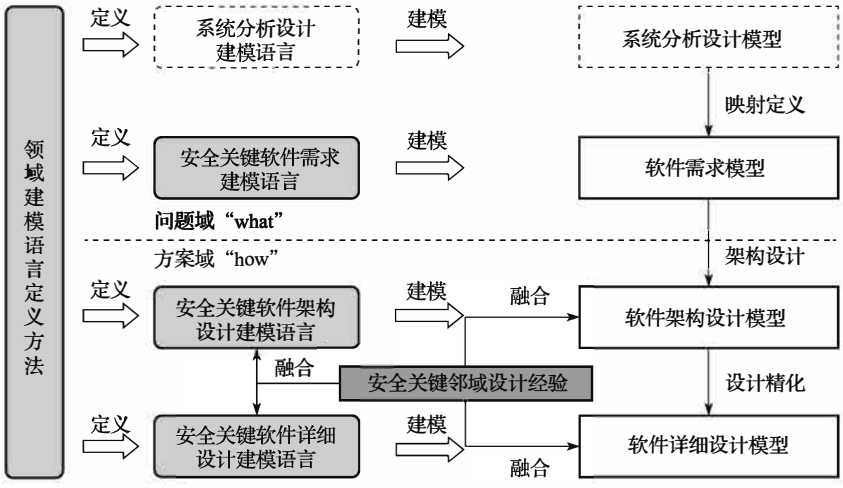

图 1 安全攸关软件的建模方法与建模语言的逻辑关系

1.1.1 领域建模语言的定义方法

定义特定领域语言的难点在于需要深入理解领域知识和领域开发需求，因此往往需要专业领域人员与语言开发人员共同参与，在语言中融合领域知识和经验，经过多轮迭代改进，最终完成满足领域要求的语言定义。2005 年，Mernik 等人总结了什么情况下适

合采用特定领域语言，以及定义特定领域语言的通用方法[1]。他们将 DSL 的开发过程划分为决策、分析、设计、实现、部署等五个阶段。决策阶段考虑开发 DSL 的成本和收益，提出了基于任务自动化程度、产品线、数据结构、GUI 构建等评估指标的 DSL 开发决策模式。分析阶段收集多源的领域数据，如文档、专家经验、存量代码等，应用知识工程和数据挖掘方法捕获领域知识、表征领域知识以及建立领域知识的本体模型，进而采用非形式化或形式化的 FAST(Family-oriented Abstraction, Specification, Translation)[2] 等方法分析领域知识，得到领域范围定义、领域本体、领域概念描述以及领域概念的特征模型。在设计阶段，主要梳理目标 DSL 与现有语言的关系，以及描述内容的形式化本质。DSL 的实现阶段主要关心语言开发环境和编译器的设计与实现。这些阶段并非按照严格的串行顺序执行，而是往往需要反复迭代，逐渐改进 DSL。

针对领域建模语言，Frank 于 2013 年总结了元模型语言的选择方法和特定领域建模语言的开发方法，并提出了语言需求分析和设计原则，用于指导涵盖语言定义的需求分析、建模场景分析、元语言建立等步骤的特定领域建模语言的定义过程[3]。由于 DSML 的开发成本较高，Tolvanen 等人[4]通过实证研究评估了典型领域建模语言的开发成本，发现相比于 DSML 对软件开发的收益，其成本并不算高。北航的张莉教授等人于 2013 年、2016 年提出了基于 GCVL(modeling Goal, domain-specific Conceptual model, architecture Viewpoint, and architecture description Language) 的领域软件架构设计建模方法的元方法，建立了定义领域大型软件密集型系统的架构设计语言的通用框架[5][6]。软件密集型系统是指那些软件对其设计、开发、部署及演化有着重要影响的复杂系统。软件架构设计对大型软件密集型系统整个生命周期有着重要影响，因此，该领域的架构设计建模的研究也一直备受关注。为了满足该领域软件的架构设计建模需求，需要定制领域架构设计建模方法，包括建模语言和建模过程。该工作应用所提出的领域建模元方法和框架，定制了面向船舶指控系统可靠性风险分析的领域软件架构设计建模方法，并验证了所提方法的可扩展性。

1.1.2　安全攸关软件的需求建模

目前需求建模主要聚焦四个方向：一是基于需求的自然语言描述，研究需求的条目化、结构化、需求抽取、需求质量检查和分析[8,9]等问题，严格意义上不能称为需求建模，可以理解为需求描述规范化；二是将自然语言描述的需求条目形式化或半形式化，基于需求模型检查需求中的不一致、冲突、冗余等质量问题[10,11]；三是建立需求的形式化或半形式化模型，定义执行语义，通过仿真等手段帮助用户快速确认需求，主要适用于人机交互等逻辑复杂但功能简单的系统，或是业务流程的逻辑较为简单的信息系统；四是对需求性质进行定性或定量的形式化描述，表达为线性时序逻辑 (LTL)[12] 等逻辑公式或性质观测器[13]，应用于形式化验证系统设计是否满足需求性质[14]。

目前针对特定领域软件的需求建模方法和建模语言的研究还比较少，主要原因在于：领域软件的特性往往体现在设计中，各类软件的功能需求和非功能需求往往在数量和复杂度方面具有差异，但总体来说具有相似的需求描述模式。因此，该方向主要研究如何建立和管理领域软件需求的概念模型，更有效地提供需求信息，支撑领域设计建模。由

于人机交互软件的需求描述往往体现交互行为，并且其功能需求的描述模式比较固定，因此在安全攸关人机交互软件的需求建模方面，目前已开展了一些研究工作。法国的 Lecrubier 等提出了 LIDL 人机交互需求建模语言[15]，并基于 LIDL 语言提出了安全攸关人机交互软件的模型驱动开发过程，包括设计建模与形式验证、代码生成与形式验证等方法[16]。

1.1.3 安全攸关软件的架构设计建模

安全攸关软件的架构设计建模语言主要采用 SysML、AutoSAR 以及 AADL 等。SysML 是由 OMG 组织提出的一种工程领域的描述语言，目前已成为开放国际标准，适用于系统工程和软件工程，支持对嵌入式系统进行建模，包括需求模型、架构的模块定义模型、活动图等详细设计模型[17]。在 SysML 的基础上，可以自定义体现领域特征的 Profile 配置文件，形成面向特定领域的 SysML 建模语言，如：Ge 通过配置 UML 的 Profile，定义了面向实时性质评估的安全攸关软件架构建模语言[14]。AUTOSAR 适用于定义和设计汽车电子控制单元（ECU）软件的建模语言，描述组件、接口、通信、功能分布等架构设计信息[18]。SAE 组织在 MetaH 和 UML 建模语言的基础上，提出了嵌入式实时系统体系结构分析与设计语言 AADL(Architecture Analysis and Design Language)[19]，能够以标准化的方式对嵌入式安全攸关软件进行架构设计建模，其语法简单、可扩展，广泛应用于航空航天领域[19]。基于安全攸关软件的架构模型，能够对架构中的实时性质、组件接口定义正确性、组件交互时序性等进行分析验证；能够生成代码框架，再通过调用代码库实现组件行为逻辑，确保安全攸关软件的架构设计可靠、可控。

1.1.4 安全攸关软件的详细设计建模

工业单位普遍采用 SCADE 和 Simulink 作为安全攸关软件的详细设计建模语言。SCADE(Safety-Critical Application Development Environment) 是法国 Esterel Technologies 公司研制的安全攸关软件集成开发环境，其建模语义基于 Lustre 同步语言实现[20]。SCADE 中的详细设计建模环境融合了航空任务软件的设计模式和特点，其代码生成器 KCG 已通过航空 A 级软件认证，满足适航认证要求，主要用于欧洲的航空领域[21]。相比于 SCADE 而言，Simulink 提供了更丰富的模型资源库，普遍应用于美国的航空领域[22]以及国内外的汽车电子领域[23]。

此外，还有众多工作致力于定制特定安全攸关软件的详细设计建模语言，满足特定的分析和验证需求，如 Wang 等人提出了面向周期化控制领域软件的详细设计建模语言 SPARDL，能够描述控制软件的周期化行为和状态模式之间的转移机制，以及描述周期驱动行为、过程启动、时间条件等领域特征行为，可应用于航天器和汽车的控制软件建模开发[7]。Tariq 等基于 Simulink 提出了面向物理信息系统（Cyber Physical Systems，CPS）的领域设计建模语言，定义物理模块、信息系统模块以及两者间的接口模块等系统架构组成，也能够描述功能模块的详细设计[24]。这些语言更为定制化，并在 Simulink 等语言的子集上附加了领域特征和领域设计经验，对于特定应用群体来说更为实用和简洁。

1.2 国外安全攸关软件的建模应用进展

国外研究机构和工业单位对安全攸关软件的建模基础理论、建模机制、建模方法进行了广泛的应用探索。本文选取航空领域最具代表性的波音公司和空客公司、航天领域最具代表的美国国家航空航天局（NASA）和欧洲航天局（ESA）等机构进行调研。这些机构均在新型号或新产品的软件设计中采用了模型驱动软件工程方法，涉及系统任务和需求定义及管理、系统功能和内外部接口设计、电子设备和软件的生产及测试、系统架构设计和系统综合等各个阶段，提出或应用了特定领域建模语言，结合形式化验证方法，取得了显著成效。

1.2.1 国外航空航天软件开发方法的发展历程概述

国外航空航天软件从传统的软件工程化方法发展到当前普遍被认可的基于模型的软件开发方法经历了近 40 年的时间，期间的发展过程大致可以划分为三个阶段。

（1）起步阶段（解决开发方法和开发环境问题） 该阶段中，以空客为代表的航空企业早在 20 世纪 80 年代开始尝试在飞控和告警系统的开发中采用图形化和形式化的设计模型来替代直接开发软件代码。在产生良好的示范效应后开始将这种开发方法大规模应用于多个系统，并采用集成开发环境 SCADE。与此同时，国外研究机构对基于模型的基础理论、建模机制等进行了深入研究及应用探索，研制了 Simulink、Modelica 等典型集成开发环境。

（2）发展阶段（解决应用问题） 该阶段最具代表性的波音公司、空客公司、NASA、ESA 等单位均在新型号或新产品的软件设计中全面采用基于模型的开发方法，涉及系统任务和需求定义及管理、系统功能和内外部接口设计、电子设备和软件的生产及测试、系统架构设计和系统综合等各个领域，并形成了以 DO-178 为代表的机载软件开发过程标准和以 AUTOSAR 为代表的汽车电子系统软件开发标准。

（3）成熟阶段（解决工具链问题） 2005 年左右，国外航空、航天、汽车等行业的龙头企业开始建立一批基于模型的预研项目，围绕着安全攸关软件开发，联合多家工业单位、高校研究所以及工具开发单位，在先进开发方法和工具链研制方面持续投入，形成了一批有示范效应的科研和应用项目，建立了较为稳定的生态。

1.2.2 空客

空客早在 20 世纪 80 年代开始尝试在飞控和告警系统的开发中采用基于模型的软件开发方法。在产生良好的示范效应后开始将该方法大规模应用于多个系统。在空客的模型驱动开发需求拉动下，法国 Est erelTechnologies 公司研制了集成开发环境 SCADE。目前，空客在需求到概要设计阶段主要采用半模型方式，基于 SysML/AADL 等建模语言定义各系统的功能及系统间的交互接口；而在详细设计阶段已经实现了全模型开发，形成了服务于设计过程的电气/机械/机电/电液全模型和环境模型（流体、动力学），以及服

务于集成过程的 Iron Bird 中的系统模型。

在开发工具方面，空客主要依赖于 SCADE 集成开发环境，它针对采用传统嵌入式软件开发方法在开发安全攸关应用时遇到的软件开发各阶段描述不一致、语义不精确、手工编码引入的潜在错误、需求与设计错误较晚才能发现、软件更新复杂、验证开销大等问题，提出模型驱动的正确构造（Model-driven development，Correct-by-Construction）软件开发方法，支持 DO-178B 软件开发过程从需求到源代码生成的全生命开发周期。SCADE 为整个软件开发过程提供统一的建模机制，包括描述连续控制系统的数据流图和描述离散控制系统的层次化安全状态机，提供满足 DO-178B/C 的 A 级安全标准的代码产生器 KCG，可以替代生成源代码的 DO-178 标准的 A 级认证测试，省去源代码生成时间和认证时间，使得软件开发生命周期从传统的 V 形演变成 Y 形。空客使用 SCADE 已使软件开发时间与成本比传统开发方法减少 25%~30%。

在开发标准方面，空客遵循美国联邦航空管理局（FAA）制定的 DO-178B/C《机载系统和设备合格审定中的软件考虑》标准。大部分航空机载安全攸关大规模复杂软件的安全等级需要达到 10^{-9}。DO-178B/C 标准用以规范机载软件的开发验证过程，从而确保机载软件满足高安全性需求。DO-178B/C 根据软件所具有的不同功能，分析其失效后所带来的后果，来确定软件的安全等级，然后根据不同等级的软件匹配不同程度的过程要求和交付件要求，从工程方法论上实现成本、质量、效率上的有机平衡。DO-178B 标准用于规范机载系统和设备的软件开发过程，确保开发的软件在功能上正确，在安全上可信，并指导软件认证到相应的安全级别，达到适航要求。DO-178B 自 1992 年推出以来，已成为所有新的航空软件认证的事实标准。自 DO-178B 发布以来，航空电子软件开发已有了显著的改变，最明显的改变是软件程序变得越来越庞大和复杂。在 DO-178B 的基础上，航空无线电技术委员会（RTCA）针对管理现代航空电子设备所必需的大量扩增的软件，于 2012 年发布了新的 DO-178C 标准。其中定义了 71 个目标（即针对每个需求的文档和测试）来满足不同的软件信任等级的量化评估。例如，软件等级 A 要满足 71 个目标（含 30 个需要独立的目标）。DO-178C 继承了 DO-178B 的核心文件、原则和过程，消除了一些 DO-178B 中的二义性，同时增加了与当前行业发展和验证实践密切相关的补充文档，包括 DO-330（软件工具鉴定考虑）、DO-331（基于模型的开发和验证）、DO-332（面向对象及相关技术）及 DO-333（形式化方法）。DO-178C 引入前向和后向的追踪，设计人员可以采用先进技术，使软件开发更容易。

空客已实现在航空机载软件方面的全模型驱动开发，在方法和标准方面已经处于成熟阶段，在各个系统的开发中取得了经济效益：在领域建模方面，继续采用 SCADE 集成开发环境，在需求分析中采用 SysML 建模工具，在架构设计中采用 AADL 建模工具；在软件验证方面，在某些系统上实践了 Caveat、Frama-C、Astrée、aiT 等形式化分析和验证工具；在关键技术研究方面，与法国 CEA、LAAS-CNRS、IRIT、ENAC、ONERA、INRIA 等长期合作，联合攻关抽象建模方法和验证技术的瓶颈问题，持续投入基于模型的安全攸关软件开发方法和工具链研制，在长期的合作项目中为安全攸关软件的模型驱动开发模式和工具链建立了稳定的生态。

1.2.3　波音

波音早在2005年开始采用基于模型的方法和工具进行民机设计与开发，早期采用V模型进行基于模型的安全攸关系统开发与验证。第一个标志性的系统始于2016年波音的2nd世纪企业系统（2CES）计划，这也是波音全面数字转型的标志。2020年波音提出了Diamond模型，包含了建模、模拟、设计、交付四个方面的融合。在工具方面，主体采用Cameo Enterprise Archiecture工具和SysML语言来进行系统建模、模拟和验证，同时开发一系列的Cameo插件如Custom/C. O. T. S/JPL OpenMBEE等来扩展Cameo的建模、模拟和验证能力，并将统一的模型存放于云端服务器。波音采用基于模型的系统工程将有关777X系统的所有信息集中在统一的模型中（通常称为“单一事实来源”）。该模型可以支撑系统开发的完整生命周期，覆盖需求分析建模、系统设计和验证。采用统一的模型，使得决策者和供应商等利益相关者以及开发团队可以从不同的视角和粒度级别访问模型，同时确保信息的一致性。基于领域模型的软件开发能够在加快航空系统开发的同时，保证软件的高安全可靠质量。目前波音的生产线只需11天就可以生产一架飞机，所需时间较从前的22天大幅减少，同时777X的ADRF项目证明，基于模型的方法可提高系统质量和沟通效率，大幅减少开发成本和时间与风险。同时可以在程序的早期阶段验证确认，避免歧义，减少错误信息和其他规范缺陷。

波音在方法和标准方面已经处于成熟阶段；在工具方面，继续采用成熟的集成开发环境Cameo Enterprise Architecture，采用SysML进行需求与系统设计建模，在某些系统上实践了IKOS、Coq、Ivory等形式化分析和验证工具。同时开发一系列插件来满足实际开发的需要。在研究方面，波音与美国东北大学、佐治亚理工学院等研究所和高校长期合作，联合攻关方法和技术瓶颈问题，探索更先进的领域建模开发和形式验证方法，研制适用于航空机载软件模型驱动开发的工具系统。

1.2.4　NASA

2011年，NASA指出当前复杂工程系统开发中存在的一些常见问题，如系统的规模和复杂度增速过快，基于文档的系统描述不易于理解系统各部分间的交互，文档存在数量众多、缺乏逻辑性、难以实现知识继承等问题。为此，NASA各中心纷纷使用基于模型的开发方法进行系统开发，如格林研究中心（GRC）从2007年到2016年实现了基于模型的开发方法，喷气推进实验室（JPL）也花了7年时间初步实现该方法。基于模型的开发方法已用于NASA多个主要的项目中，如空间通信网、欧罗巴项目以及火星任务等。

通过使用模型，可以有效地降低项目的总开支。首先，增加系统设计阶段的投资，可以有效地降低项目总成本。NASA成本和经济分析处（Cost and Economic Analysis Branch）主任分析了NASA历史上各个项目的超支情况后，指出大部分超支的原因都在于没有很好地对项目进行定义，导致需求变更，使得开销增加。

美国国防部资助的 Cougaar 子项目应用了模型技术，该项目提供了一个 Agent ㊀开发和运行环境，可以实现任务协同中的关键模型和算法。模型还用于 NASA 的一个概念任务——ANT(Autonomous Nano-Technology Swarm)。具体的开发流程是：首先定义 3 个平台无关模型（M-RAAF、AOM 和 SOM）和一个平台相关模型；然后定义这些模型之间的转换，从 M-RAAF 到 AOM，再到 SOM，再转化为平台现骨干模型；最后，将平台相关模型转化为代码。

目前，NASA 已在火星任务软件中使用基于模型的开发方法，通过模型，能够更好地理解技术基线、改善系统设计过程的交流和理解问题、聚焦于历史设计的偏差、帮助飞行系统工程师实现所需模块。NASA 在新一代火星探测器“毅力号”的设计过程中使用了以下模型技术：①进行模型化的翻译，将设计数据文档转化为基于标准 SysML 语言的设计模型，以进行设计数据的正确性验证、交付物的自动生成和复杂模型自动排版布局（供不同人员查阅）；②采用标准 BPMN 语言描述探测器与地面控制人员的各种行为，也使团队对设计的完整性更有信心；③采用模型集成的方式，针对不同设计团队或领域之间的设计，自动生成文档和报告，并进行一致性检查，确保文档之间数据一致性；④使用基于模型的设计方法，自动生成项目交付所需的文档，节省项目评审的准备时间；⑤选择基于模型的设计方式，自动生成设计文档，确保设计数据的复用，能对设计模型的完整性进行校核；⑥基于 SysML 模型，借助第三方工具的数据过滤和自动排版功能，完成设计信息的选择性展示，促进设计交流。NASA 通过应用基于模型的开发方法，使火星任务的总开支节省了 3.27 亿美元。

在基于模型的开发过程中，NASA 使用 MagicDraw 实现系统建模，使用 IBM Rational Rhapsody 为工程师及开发者提供可视化开发环境。NASA 还用到了如下开源技术：Ecore/Eclipse EMF+OCL 用于建模领域知识，Henshin 或 VIATRA 描述探测或转换规则、Java 实现分析或适应性函数，MOMoT、MOEA 和 VIATRA DSE 利用遗传算法进行优化。

1.2.5 ESA

ESA 在 2019 年 12 月成功发射的系外行星特征探测卫星 CHEOPS 研制中成功应用了基于模型的开发方法，来自十余个国家的二十多个单位参与研制，使用 UML 等模型对领域概念、状态机、流程、时序行为、可重用组件、接口等进行建模并实现基于模型的 C 语言代码自动生成。基于模型的开发模式帮助 CHEOPS 在花费不到 5 000 万欧元、研制周期不超过 4 年的情况下研制成功。

ESA 在 2020 年 9 月公开披露的文档中显示，它在最新的多用途载人航天器 ESM 子系统的研发中成功采用了基于模型的开发方法。该子系统集成了超过 2 万个模块，包含推进、电源、压力、太阳帆板、热能、流体控制等。通过基于模型的开发方法有效地管理多模块之间的信息以及自动生成接口文档，从而解决了大量模块之间的接口控制文档（Interface Control Document，ICD）和信息（Information）之间的不一致问题。ESM 标准

㊀ Agent，一般译为智能体、主体。

化了模块之间的设计修改流程，避免了信息冗余和不一致问题，极大降低了人工维护和验证的成本。

以上两个典型案例体现出安全攸关软件的模型开发方法适用于多方参与研制、大规模系统集成的一致性管理以及协作信息和流程的标准化设计。相比于基于文档的开发模式，基于模型的开发优势体现在降低成本和研制周期、标准化流程和高度一致性、模型的自动维护和验证等方面。

2 安全攸关软件系统的形式化验证方法

安全攸关软件的可靠性不仅依赖于软件本身，还依赖于运行该软件的 CPU 和操作系统的安全性，以及编译该软件的安全性。因此，为提高安全攸关软件的安全性和可靠性，有必要研究 CPU、操作系统以及编译器的形式化验证问题。

计算机系统的信息安全是当前信息技术领域极其重要的研究方向。由于系统软件结构复杂、并发度高，开发与调试都十分困难，一些隐藏的错误很难用通常的软件测试技术发现，而这些错误可能会造成灾难性的后果。形式化方法可以通过数学推理的方法，检测系统描述不一致性或不完整性，以提早发现这些隐藏的错误。因此，形式化方法是提高软件系统，特别是高等级安全攸关系统的安全性和可靠性的重要手段。目前，欧美国家已在相关安全标准和软件认证上规定必须使用形式化方法。RTCA 的 DO-178B[25]《机载系统和设备认证中的软件要求》用于规范机载系统和设备软件开发过程，指导软件安全级别认证。2012 年发布的 DO-178C[26]增加了形式化验证要求，强调采用形式化方法，开展基于模型的开发和验证。国际信息技术安全评估通用标准 Common Criteria(ISO/IEC 15408)[27]定义了计算机系统的安全等级；其中，最高等级 EAL 7 要求系统开发必须采用经形式化验证的设计和测试。英国国防部发布了安全攸关标准 Def Stan 00-55 和 Def Stan 00-56 标准，要求安全攸关软件开发必须采用形式化方法。

2.1 操作系统的形式化验证方法

采用形式化方法进行操作系统（OS）验证，需要对 OS 建立模型，将非形式化的概念转化为模型中形式化的部分，然后再进行设计和验证。目前对 OS 的验证主要进行系统模型、代码实现的一致性验证（即功能正确性）和系统安全性验证。

CertiKOS 是一款经过严格验证的系统操作。2016 年耶鲁大学的 Flint 团队采用 Coq 定理证明器，验证 CertiKOS 的功能正确性[28][29][30]和安全性相关属性[31]。Leroy 等人验证了 CompCert 编译器[32]，从而奠定了 CertiKOS 项目的基础，因为 CertiKOS 的底层机器模型是 CompCert X86 汇编模型。2019 年 FLINT小组以 CertiKOS 为基础，构建了一个经过验证的实时 OS 内核 RT-CertiKOS[33]。

安全嵌入式 I*A*(secure embedded I*A*，seL4）是一款嵌入式系统微内核[34][35]。2004

年澳大利亚国家 CT 实验室对 seL4 进行了功能正确性验证；用于形式化规范定义以及证明 seL4 内核的 C 代码（共 8 700 多行）的 Isabelle/HOL 代码共有 20 多万行[36]；该设计与验证工作花费了共 20 人·年的时间。为了提升操作系统形式化验证的程度，近几年 Klein 带领的项目组又在多核系统验证[37]、缓存验证[38]、实时性验证[39]等方向进行了探索。

华盛顿大学 UNSAT 小组的宗旨是提高计算机系统的正确性和性能。Hyperkernel[40]是由该小组设计、实现并形式化验证的 OS 内核。此外，他们还设计了一种 push-button 方法，用于构建可证明正确的 OS 内核以及文件系统[41]。

PikeOS[42]是一种用于航空电子和汽车环境的工业操作系统，包含一个微内核和一个 PikeOS 系统组件。文献［43］给出 PikeOS 的 Isabelle/HOL 形式化规范，涵盖了分区间通信、内存、文件提供程序、端口和事件等功能，共包含超过 4000 行的 Isabelle/HOL 代码。

实时操作系统 INTEGRITY-178B[44]主要用在 B-2 和 F-35 等军用飞机以及空客 A380 等民用飞机。美国国家信息安全保障合作组织联合国家标准和技术研究院于 2005 年对 INTEGRITY-178B 进行认证。Green Hills 公司宣布 INTEGRITY-178B 是第一个通过通用标准评估分级 6+ 级（EAL 6+）的操作系统。

LynxOS[45][46]是一款完全符合 POSIX 标准的强实时操作系统，被广泛用于航空电子、航天系统、电信等领域。LynxOS 的其中一个版本 LynxOS-178 是最早获得 DO-178 认证的实时操作系统。

ARINC 653 是航空电子应用程序接口标准。Zhao 等使用 EventB 对该标准进行了形式化验证[47]，发现 ARINC 653 第 1 部分中的三个错误，并检测到了三种规范不完整的情况。

Zephyr[48]是 Linux 基金会的一个开源物联网操作系统。Zhao 等在 Isabelle/HOL 中开发了并发 C 代码的建模语言及验证方法，并对 Zephyr 的内存管理模块进行建模[49]，最终从 Zephyr 的内存管理中发现 3 个严重 Bug，包括系统调用死循环等。

μC/OS-II[50]是一个商用抢占式实时多任务内核。冯新宇等提出了一种内核验证框架[51]，用 5.5 人·年的时间验证了 μC/OS-II 的关键模块，包括定时器中断处理程序和四种同步机制等。

用于汽车应用的 ORIENTAIS 是一个基于 OSEK/VDX 标准的实时操作系统。华东师范大学的何积丰院士团队[52]采用面向目标代码的验证方法，对 ORIENTAIS 进行建模与验证，证明了 ORIENTAIS 的接口实现遵循 OSEK/VDX 规范。

2.2 CPU 的形式化验证方法

CPU 是计算机系统的核心，其正确性和安全性直接关系到应用系统是否安全，尤其是在航空航天领域，安全攸关的机载软件系统对于 CPU 的安全性要求更是高于普通系统。随着计算机技术的发展，CPU 的性能得到了不断提升，但与此同时其复杂程度也越来越高。一旦 CPU 设计流程中出现错误，可能会引起严重的后果。因此，针对 CPU 设计的正确性验证变得十分重要。

首先，介绍 CPU 验证方面的工作。IBM 在 POWER7 处理器的设计过程中，使用断言工具 SixthSense 和 RuleBase[53][54]对处理器进行验证；仅对浮点运算部件的验证，便发现了数百个错误。美国犹他（Utah）大学 MPV 研究小组与 Intel 合作[55]，对 Murphi 语言进行扩充，并对层次化存储系统的一致性协议进行验证。Lamport 提出了一种基于 Clock 时间戳的证明机制，证明了 Origin2000 的存储一致性协议设计的正确性，即该协议符合存储顺序一致性的要求[55]。AAMP[56]是美国 Rockwell Collins 公司开发的航电微处理器系列，其指令集属于 CISC 类型，有超过 200 条指令；AAMP 的验证使用了 ACL2 定理证明器[56]。

其次，为了验证处理器，需要对处理器的指令集和架构等进行建模。2010 年，Fox 等针对 ARMv7 指令集，提出了一个基于 HOL4 逻辑的指令集抽象模型[57]。2017 年，Goel 等人利用 ACL2 建模工具，对 32 位的 X86 指令集架构进行了建模，并对 X86 汇编程序进行形式证明[58]。澳大利亚新南威尔士大学的 Syeda 和 Klein 对 ARMv7 架构的内存地址映射管理单元做了抽象化的建模[59]。Zhe Hou 等人采用 Isabelle/HOL 对 SPARCv8 架构进行了建模，包含 SPARCv8 架构的寄存器模型等[60]；该模型还包含可执行指令的抽象状态机，用以执行二进制的 SPARCv8 汇编代码，并对二进制汇编代码进行验证。

再次，指令流水线是处理器的一个重要功能，其正确性对处理器的安全性有重要的影响。流水线技术首次出现在 Intel 的 486 芯片中[61]。Makiko 等人在 2000 年提出了一个流水线自动生成系统——PEASS-III[62-64]，实现了针对 MIPS 架构的 5 级流水线的自动生成。Daniel 等人[65-67]在 2001 年提出了一种流水线微处理器的自动设计和验证的方法，并通过使用定理证明对流水线的转换过程进行了形式验证。Burch 则提出了一个流水线自动验证方法[68]；该方法基于模型检测和等价性检验，以非流水线 CPU 结构为参考，证明流水线结构在功能上实现了非流水线的参考模型。

最后，介绍国内有关 CPU 验证的研究工作。龙芯处理器是中国科学院计算技术研究所自主研发的通用 CPU。在龙芯 1 号处理器的验证中，首先采用了模拟验证的方法，随后采用遗传算法对测试用例的生成过程进行改进，提高了验证的效率[69]。龙芯 2 号则采用了等价性检验和模型检测两种方法。为此，开发了用于浮点加法部件的模型检验器 ArithSMV[70]；该系统采用了基于 * PHDD 的模型检验方法，实现了 * PHDD 的加法、减法和乘法等操作以及 * PHDD 的大小比较指令；增加了对条件模型检验方法的支持，并集成了基于可满足性判定的验证方法。

2.3 编译器的形式化验证方法

编译作为整个计算机系统中的重要环节之一，所有高级语言都必须通过编译才能成为计算平台可执行的程序。编译的正确性和安全性对于整个计算机系统具有非常关键的意义——如果编译不正确，整个安全攸关系统的安全性无从谈及。在安全攸关领域，特别是航天航空机载软件系统，符合安全规范的软件编译架构是一项具有挑战的项目。一方面，现代的编译体系构成复杂，开源的编译器如 GCC 和 LLVM，很难被机载软件系统

所使用；另一方面，现在的软件编译系统没有按照航天航空机载软件 DO-178C 的标准进行架构，无法保证高安全性。

由于安全攸关系统功能和非功能性需求不断提高，软件编译的复杂度也随之急剧增加。一个 A 级安全关键软件必须是由 A 级编译产生代码。因此，安全攸关编译是安全攸关系统的重要保证，如何设计满足 DO-178C 规范标准的高质量安全攸关编译，是工业界共同面临的新需求，也是学术界面临的新难题。

针对编译设计正确性验证技术，主要分为两大类：一类是集成安全规则的静态校验技术，形成安全校验编译；另一类是形式化验证技术，形成验证编译[71]。

1967 年，John McCarthy 和 James Painter 通过研究编译器的正确性问题，证明了从一些简单数学逻辑表达式映射到机器语言的编译算法的正确性[72]。1972 年，Milner 和 Weyhrauch 提出了一种用于证明编译正确性的机器实现方法[73]。Hoare 和何积丰用语义制导的方式，研究正确性编译器的生成[74]。Stepney 等人在文献［75］中描述了如何从语言的形式化定义出发构造编译器，并使用 Prolog 语言定义源语言和目标语言的指称语义，通过遍历语言的树结构来构造正确性证明。Gaul 等人[76]通过把编译器的证明过程和开发流程结合起来，减少证明工作量的同时，也使开发过程符合软件工程规范。

目前最具有里程碑意义的工作是 Leroy 带领 CompCert 项目组首次完成了对一个完整且实际的编译过程的正确性验证；整个证明过程完全形式化，并且都是机器自动生成的[77-79]。为了支撑形式化证明的自动化，CompCert 采用辅助定理证明工具 Coq 对编译过程进行重构；在编译过程中，完成了从函数式语言 Clight 到汇编代码 Power PC 的编译（整个过程由八种中间语言间的编译构成），然后使用 Coq 对整个编译过程的正确性进行证明。

在国内，中国科学技术大学苏州高等研究院安全实验室的葛琳、刘诚、华保健等人设计并实现了一个校验编译器的原型系统——PLCC，并设计断言语言 PAL 来描述安全规范[80,81]。关于复杂嵌入式系统的架构语言 AADL[82,83]的编译，北京航空航天大学计算机学院赵永望老师团队对 AADL 进行了分析，并基于模板技术研究了从 AADL 到 C 代码的自动生成技术[84,85]，还在神舟公司的 GNC 应用模型项目上进行验证工作。另外，赵老师团队完成了将 AADL 模型转换到 CSP 模型的形式化验证工作[84]。

目前支持编译器测试的工具有 PV-Suite[86]和 CV-Suite[87]。PV-Suite 是 Perennial 公司开发的商用编译器和操作系统测试套件；它利用断言对 C、C++和 Java 等语言进行一致性测试。CV-Suite 是 Plum Hall 公司开发的商用编译器测试程序包。

在安全攸关领域中，结合模型检测技术对编译流程进行验证，通过遍历所有的状态空间，自动检测具有有穷状态的系统，并构造不满足性质定理的反例；这种技术的优势在于能够实现全自动化，需要人工操作的工作比较少，所以工业界已经有了广泛的应用。在对 C 语言编译器的模型检测技术中，支持对 C 语言代码验证的典型工具有 SLAM[88]、BLAST[89]、MAGIC[90]等。其中，MAGIC 工具是基于“语义弱模拟”理论来验证程序与规范是否一致；简单来说，其验证过程是程序标号变迁系统与程序规范标号变迁系统是否为弱模拟的判定过程。BLAST 是一个对谓词抽象、模型检验等计算进行优化以提高验

证效率的工具，它使用了惰性抽象技术，包括 On-the-Fly 的抽象方法和 On-Demand 的精细化方法[89]；BLAST 成功验证了 13 万行源代码的 C 程序[89]。SLAM 被应用于验证微软开发包中的驱动程序的安全性，由 C2BP、BEBOP、NEWTON 三种工具组成：C2BP 是一个将 C 代码抽象成布尔程序的谓词抽象工具，BEBOP 是一个检验布尔程序的工具，NEWTON 用于精化布尔程序到其他谓词的工具。另外，对于 Java 程序编译的模型检验工具也有很多，例如 ESC/Java[91]、Java Path Finder[92]和 Bandera[93]等，前两个是基于谓词抽象技术，最后一个则基于程序切片方法。

近年来国内也开展了安全攸关软件研究的立项和研究工作。中国科学院软件研究所林惠民院士等组织立项和形式化技术方面有关的研究[94]，其成果主要在模型检测的形式化验证方向上。国防科大陈火旺院士等讨论了高安全可信软件工程技术的现状和面临的主要挑战[95]，并指出基于形式化方法的高安全可信软件技术的发展趋势和突破点。

2.4 安全攸关任务软件模型的形式化验证方法

与面向领域的建模方法一致，机载任务软件模型的形式化验证需要面向任务软件的三个模型层次：需求模型、架构设计模型和软件模型。

在需求模型的验证方面，大部分工作仍然是验证需求与设计或软件模型的一致性问题，仅针对需求层模型存在的问题进行验证的工作还比较少，主要用于检查需求模型中的冲突问题，如陈小红等针对铁路互锁软件的安全性需求定义了领域需求建模语言，采用模型检测技术检查安全需求模型间的一致性，发现需求间的冲突等问题[96]。

在架构设计模型的验证方面，针对嵌入式软件架构的构件建模、组合性质、构件间组合验证的问题，Kopetz 等人提出了实时嵌入式软件组合构建理论，明确了构件化设计、组合构造的思想和原则[97]。白晓颖和贺飞等将组合构件模型的验证方法总结为基于契约和基于不变量、基于模型检查等三类[98]。IA 模型、BIP 模型、PA 模型、HRC 模型等都是基于契约的模型，均可使用基于契约的方法进行验证。Verimag 实验室在 BIP 模型的可组合性验证中引入了不变量验证技术，用以验证软件架构中的构件不变量、交互不变量、系统不变量以及时间约束[99]。基于模型检查的软件架构设计验证技术主要用于评估架构设计模型的实时形式、进行软件安全性评估和缺陷检测等[100]。通常将 AADL、SysML、UML 等描述的架构模型通过模型转化生成 BIP 等验证层模型，完成时序逻辑等表达的架构性质验证[101,102]。

在软件模型的验证方面，由于国内外工业单位普遍采用 SCADE 和 Simulink 对软件建模，因此一直以来都有大量工作研究这两类建模语言的验证问题。SCADE 集成环境早在 2004 年整合了基于可满足性求解的 Prover 验证插件[103]，Walde 等人在真实通航飞行器上验证了两个版本的飞控软件实现是否一致[104]，Shi 等人将 SCADE 模型转化为 NuSMV，用于验证安全性需求[105]，Basold 等人在 SCADE 和 SMT 之间引入了一种中间语言，用于证明语言转换过程中的正确性问题[106]。针对 Simulink 模型的验证问题，Simulink 集成环境中已整合形式化验证插件 Simulink Design Verifier，用于自动验证和检测软件模型中的

整数溢出、死逻辑、数组访问越界、被零除等问题，其验证技术与 SCADE Prover 类似，均基于可满足性求解技术[107]；Reicherdt 等人将 Simulink 的模型转化为 Boogie 模型，用于验证汽车控制器的功能[108]；中国科学院的詹乃军老师等提出了基于混成霍尔逻辑（Hybrid Hoare Logic）的 Simulink/Stateflow 模型验证方法，并成功验证了高铁列控系统模型[109]。

2.5 未来研究方向

针对操作系统、CPU 以及编译器验证的研究，有以下几个方面的机遇和挑战：

（1）多核 OS 以及分布式 OS 的高效测试验证　目前，多核并发底层软件的形式化验证技术研究刚起步，仍然面临极大的挑战。此外，近些年随着大数据时代的到来，分布式计算成了从海量数据中挖掘数据价值的必然选择。而支持分布式计算的分布式 OS 的安全性问题也成为目前研究的一个热点。

（2）操作系统自动化验证方法　虽然形式化方法在保证软件安全性、正确性及可靠性方面得到认可，但是受限于其验证效率，至今未能广泛应用于工业界。操作系统的形式化验证仍然是困难和费时的。自动化验证解决方案通过最大限度地减少对人力资源的需求，缩短验证软件系统所需的时间，提高生产率。如何运用自动化验证技术形式化验证操作系统也是值得探索的课题。

（3）多发射机制的 CPU 数据通路的自动综合方法　CPU 的形式验证与其设计方法密切相关，而自动综合方法则是后续形式验证的基础。现代 CPU 采用多发射机制来提高 CPU 的指令执行效率。然而，目前针对 CPU 模型自动综合的研究，都是针对流水线结构的 CPU 数据通路。因此，需要重点关注多发射机制 CPU 数据通路的自动综合方法，以及多发射机制的正确性和安全性验证方法，以进一步提高 CPU 的设计效率。

（4）保持代码可追踪性的编译器优化技术　优化技术是提高编译效率的常用手段，但使用优化会造成细节缺失，给源代码和目标码之间的可追踪性验证带来困难。如何既保证安全攸关领域的可追踪性需求，又能把优化技术加入安全攸关软件编译中，是未来编译器验证研究中值得关注的问题。

（5）机器学习技术与形式验证方法的结合　机器学习方法在很多领域都取得了令人瞩目的成果。然而，因其缺乏可解释性，在安全攸关软件领域较少得到应用。因此，如何利用机器学习提高软件的形式验证的性能，是未来的一个研究方向。有研究表明，即使是通过形式化验证的软件，仍然可以在其中找到漏洞（bug）[110]。这是因为很多系统软件因其复杂性，几乎不可能形式化验证所有的可能性。因此，可以利用机器学习的方法，处理一些例外事件，提高系统的鲁棒性[111]。

3 领域模型和形式化验证方法在航空领域的应用和需求

3.1 应用现状与成果

3.1.1 领域模型应用现状

领域模型是领域知识各组成部分的抽象形式，同时也表示了领域内各系统的一些共同特征。领域模型可用于软件复用活动，能为领域内新系统的开发提供可复用的软件需求规约，以及指导领域设计阶段和实现阶段可复用软件资产的生产。面向领域的特点使得一般的需求建模方法不再适用于领域模型的建模过程。因此，采用何种形式的领域模型，如何实施对领域的建模活动，成为领域工程研究和实践中的重要问题。

领域模型在机载系统，尤其是安全攸关系统的实际应用中，可以使用模型来描述部分或全部软件需求、部分或全部设计，或者使用模型同时表达软件需求和设计。根据文字描述的软件需求进行建模，以模型的方式表达软件功能，再通过专用的工具直接生成源代码，这是基于模型最常见的应用场景之一。在航空领域，目前国内外航空工业场所针对基于模型的开发和验证大多采用 SCADE 工具来完成[112,113]。在国外，空客、欧洲直升机（Eurocopter）、泰雷兹（Thales）、达索（Dassault）等公司都将 SCADE 应用在领域的安全关键软件开发中[114,115]。如空客已经成功使用 SCADE/KCG 工具开发出了 A340/500-600 机型飞行控制系统软件，取得了令人满意的效果[116]。SCADE 工具在国内航空领域应用相对较少，其中成都飞机设计研究所最早将 SCADE 成功应用在某型号飞机的飞行控制系统软件研制过程中，加快了项目研制进度。另外由 615 所研制的 C919 大型客机的显示系统软件、平显系统软件中均使用了 SCADE 进行软件设计建模[117]。

3.1.2 形式化验证方法应用现状

形式化方法通过严谨的语义对系统进行建模，借助计算机以及严格的数学方法对系统及其软硬件相关特性进行验证，削减人因因素影响，提高了验证结果的可靠性，是解决复杂系统以及软硬件验证的关键，因此逐渐成为国际上的研究热点。

形式化验证方法中最常用的是定理证明和模型检验。定理证明方法可以有效地避免空间爆炸问题，虽然尚存在自动化程度较低等不足，但在工业界也得到了初步的应用，如罗克韦尔柯林斯公司用 PVS 工具验证了导航模块的可靠性[118]。空客和达索公司尝试使用定理证明来替代原本的机载代码单元测试[119]等。目前定理证明方法的研究方向主要集中在如何提高证明的自动化程度，以及如何较好地实现从软件代码到待证明定理的抽象转化[120,121]。

模型检验方法基于状态搜索的原理，它针对较小的有穷状态系统进行状态空间的遍

历，检查有无缺陷从而完成对系统的验证。模型检查技术因为自动化程度较高，因此在航空航天、轨道交通、核能等工业领域有着相对广泛的应用。如 Havelund 等人使用 SPIN 工具对某航天器的计划执行模块进行了形式化验证[122]。Bochot 等人对 A380 客机的地面扰流功能进行了形式化验证[123]。在国内，中国民航大学的金志威等人提出了一种基于模型检验的机载电子硬件验证方法[124]。上海航天控制技术研究所构建了一种基于 SCADE 状态机的形式化验证框架，提高系统的安全性和可靠性[125]。然而在验证一些状态量较为复杂的系统时，模型检验方法会出现空间爆炸的问题，因此目前该方法的主要研究方向就在于如何减弱空间爆炸带来的影响[126-128]。

3.2 领域未来需求及建议

现今，国产自主可控的航空机载设计软件，尤其是安全攸关系统的设计软件由于国际技术壁垒，始终是遏制以民机为代表的国产航空器研制和发展的重要卡脖子技术。随着科技的进步与国家航空产业的不断升级，现代航空器的性能愈加先进、功能日益复杂。作为承担和实现功能的核心载体，机载软件的规模、类型、交互性和复杂度也不可避免地持续增长，这导致了机载软件在不同程度上出现由于人为、硬件、协作等因素引起安全问题的可能性变大，软件设计和行为更难控制，导致系统脆弱性难以定位和预测，使安全性和可靠性显著降低。

由于航空器的特殊性，机载软件的安全性问题可能导致严重的事故和后果。典型的案例如波音 787，在使用现有的机载软件设计及安全性分析方法进行研制后，还是因为电源着火导致停飞，波音 737 Max 飞机更是因为 MCAS 系统设计和安全性分析问题导致印度尼西亚狮航的空难事件。这些机毁人亡的空难事故对民众生命财产和科技快速换代造成了严重的伤害，提出了严峻的挑战。

上述安全攸关系统软件设计的宝贵经验和惨痛教训均预示着在航空器领域，尤其是民机研制中，为攻克高安全、高可靠的航空机载设计软件瓶颈问题，满足国产大飞机适航符合性重大需求，领域模型和形式化验证在未来需要进行深刻的变革和持续的改进，尤其在先进的高可靠、高安全复杂系统形式化验证方法推进，和 SCADE、Simulink、Modelica 等先进支持工具优化方面，未来需解决好下述技术问题。

3.2.1 复杂系统形式化验证方法推进

（1）高可靠、高安全复杂系统形式化验证方法建设　对于复杂系统的形式化验证方法的研究未来需着重突破自然语言处理技术和基于模型的形式化验证方法。在理论层面，这是对于大规模状态空间运用数学推演完成穷举测试形式化方法引发空间爆炸问题，改善复杂逻辑系统高可靠、高安全验证自动化的有效途径。

（2）形式化验证的智能工具链建设　未来形式化验证中定理证明和模型检验方法的研究方向应主要集中在如何提高证明的自动化程度，以及如何较好地实现从软件代码到待证明定理的抽象转化，研究、部署和调度围绕形式化验证的工具链建设，支持面向机

载安全攸关系统安全性分析相关的形式化验证落地更为智能化、完备化、平台化。

3.2.2　支持工具优化

（1）程序多核化　近年来，为解决单核处理器的局限性，多核处理器以其更高能效逐渐被更多应用于安全攸关系统的研制中。为满足多核相关需求，未来与 SCADE、Simulink、Modelica 等工具紧耦合的领域建模和形式化验证研究应聚焦于多核目标上可执行代码的生成的优化研究，在追求高效性的基础上需仍能保留原有的精确、无歧义的、确定性语义和平台无关性，提升多核下程序处理的能力。

（2）多语言融合化　在形式化验证方面，未来考虑将软件研制领域市场中的诸如 HLL 等多种语言融入 SCADE 等模型验证及代码自动生成工具的新型形式化流程中，以提升安全攸关系统领域使用模型检查技术进行形式化验证的便利程度和有效性。

（3）基于模型的开发验证流程智能化　未来主要的研究目标可在人工智能领域，将诸如 Simulink 和 SCADE 的工具中传统的基于模型的开发验证流程与新型 AI 研发流程相结合，运用前沿的深度学习、自然语言处理和人因脑机接口等先进科技，提升建模、验证和代码生成的自动化、智能化和效率，并在无人自主航空器的机载系统设计和实现中落地验证，为其在载人航空器的机载系统中的应用奠定基础。

3.3　充分利用现有成果的建议

（1）推动技术创新和成果落地转化　在复杂航空系统中，软件系统越来越多地采用多供应商集成方式，安全攸关软件系统的设计、开发与验证成为航空领域面临的巨大挑战，而基于形式化模型的解决方案已在业界逐渐应用。目前国内外在形式化理论研究方面取得了许多令人满意的成果，如针对系统特性开发了 PVS、HOL4、KeYmaera 等众多的定理证明器。然而，面对航空工程项目及系统的日渐复杂，以及系统安全性分析难度的指数增长，未来应进一步探索形式化方法与航空系统全生命周期过程的深度融合。同时，应在现有成果基础上瞄准领域前沿发展，大力促进形式化理论在实际工业领域的落地，为高安全高可靠的系统研制提供保障。

（2）提升工具自主可控水平　目前国内的航空软件开发验证环境和工具链的研究处于起步阶段，各航空单位使用的环境工具大多来自国外，面临着被国外垄断的局面，存在断供的风险，无法满足国内日益增长的航空安全攸关软件验证需求，也无法满足国家对核心技术自主可控的要求。例如美国政府曾将 58 家中国航空航天及其他领域的公司列入“军事终端用户”（Military End User，MEU）的实体清单，限制其购买一系列美国商品和技术，对其产业链供应链安全造成严重威胁。因此，可将现有成果整合成技术成熟度高、实用的工具系统，提供国产化的软件开发验证环境，解决由 MEU 产生的自主可控风险。

（3）促进军民融合推广应用　基于模型的研制流程在民机领域中已经有较为广泛的应用，但在装备型号研制中尚处于初期阶段。将成熟的领域模型和形式化方法及相关产

品向装备领域推广，系统性地从研制整体框架出发，从根本上将建模技术与过程管理关联起来，对装备研制全周期内所有存在及潜在过程进行归纳与分析，完成从项目管理、技术实现低耦合的松散架构，到统一规范、模型与方法的集成解决方案的转变，能为装备研制提供技术层面的可行革新，进而提升装备质量和性能指标。

（4）提升领域模型跨阶段的管理能力　现有研究的领域模型大多分布于系统不同研发阶段，而它所包含的不同阶段建模要素存在着内在的关联关系。因此，在现有阶段模型的基础上，必须针对领域内不同类型的产品，研究切实有效的方法，建立完整的逻辑信息架构，与不同阶段的信息相对应，并对不同阶段的模型和信息架构中的各个元素进行追踪，以对不同阶段的模型进行有效管理。

（5）增强领域模型复用性　现阶段，系统建模工具中的大部分图形均由人工完成，尚未能自动生成。事实上，需求模型、结构模型、行为模型可重用元素较多。因此如何通过重用已有的系统设计模型来提高系统建模与设计效率，将是模型驱动复杂产品系统建模与设计下一步值得研究与探索的重要问题。针对此问题，可在软件的研制流程中，开展柔性建模工作，建立准确柔性的领域模型，更好地响应需求变更，极大地提高软件的复用率，使软件具有较好的可扩展性。

（6）完善形式化验证工作的充分性　目前航空领域系统形式化验证的工作通常是对已有代码的“设计后验证”，所验证的模型仅仅是实际系统的一个抽象，只能体现系统的可能行为，而不是全部行为。这种方法最大的问题在于必须保证抽象出来的形式化模型与原来的源代码的行为是一致的，而事实上，无论这个抽象过程是自动的还是人工的，都无法避免不一致性。因此，这些验证工作是不充分的，需进一步完善。

（7）改进建模与验证方法的单一性　目前已有的系统建模及验证方法主要侧重于建模或者验证系统的某个方面的属性，很少能够完整地覆盖多个方面的属性建模和验证需求。因此需研究可覆盖多方面属性的建模和验证方法，改进其单一性，提升形式化验证的完备性。

4　领域模型和形式化验证方法在航天领域的应用和需求

在航天装备系列化、信息化、体系化的发展趋势下，控制系统已从早期的任务单一、以硬件控制为主转变为多任务、复杂应用场景、以软件控制为主、采用大量新技术、软硬件深度耦合的大规模复杂系统。复杂系统带来的涌现性问题，给控制系统研发带来如下困难：

（1）系统需求分析不充分，系统方案缺少系统原型验证手段，导致后续各环节变更频繁。智能化与信息化发展提升了控制系统的复杂性，系统需求识别越来越困难，系统需求分析往往不够充分，甚至在控制系统研制早期还存在想当然的情况，加之系统原型验证的能力不足，使得控制系统方案设计正确性更多依赖于设计者本身的经验以及评审专家的经验，方案合理性、正确性的确认严重滞后，通常要等到软件、单机的验证环节

才能确定系统设计及软件、单机功能分解正确与否；同时也会导致部分方案设计的问题只有系统试验过程中才能发现，且技术状态在后期变更频繁；研制过程头轻尾重，在分析设计环节投入较少，过于依赖在实现环节不断试错、在系统试验中以量保质。

（2）各专业间缺少协同设计的有效手段，存在部分工作重复的情况，导致项目研发时间难以缩减。当前的研制模式各专业设计相对独立，没有形成合力，资产分散在各个专业，没有形成重用资产，缺少协同工具平台及组织资产基础；研制过程中各专业通常串行工作，且相互传递的设计依据基本为文档形式，在流程末端的专业不能最大限度地使用前序专业的工作成果；同时各专业工作间有一定的交叉，致使各专业人员存在一定的重复工作且不能相互借鉴使用；在项目工作不断增多、研制周期不断压缩的情况下，留给开发及验证的时间越来越紧张，产品质量保证的时间不断被压缩，进度和质量越发难以保证。

（3）新任务形势下系统的复杂性带来了团队规模的增大，各专业间、不同人员间工作的一致性与完备性保证难度加大，导致实现环节存在理解上的偏差。为适应目前国内外越来越严峻的形势，型号研制任务的难度越来越大，这也导致了控制系统的复杂性急剧增加，以及型号研制团队规模的增大。在目前的工作模式下，协调会越开越多，协调文档越写越厚，协调问题时有发生，尤其是软件项目，系统或架构设计后过渡到软件设计时前期成果很少能重用，且基于文档任务要求传递方式往往导致在实现环节存在理解上的偏差。

4.1 应用现状与成果

基于模型的系统工程、模型驱动软件开发方法在航天领域正在开展研究，并形成了部分成果。

基于模型的系统工程方面，空间站系统初步形成了基于模型的需求管理、系统设计、仿真验证及制造测试能力。

通过条目化需求管理，空间站建立了需求管理系统，包含总体-舱段-分系统需求体系，形成了条目化的型号技术要求管控模式；多学科集成仿真，面向空间站研制阶段设计仿真方案验证需求，空间站系统总体、电总体、数管、GNC、机械臂等分系统联合建立了空间站能源、环热控、信息、姿轨控、推进等专业的系统级模型，开展了空间站三舱组合体飞行过程仿真、机械臂转位仿真等；全三维协同设计，已实现了全三维设计、三维下厂，并以三维设计模型为主线，将结构设计、热设计、总装设计以及所有单机模型纳入统一模型架构，保障机械接口设计的正确性。相关情况如图 2~图 5 所示。

初步建立了 MBSE 相关工具链，模型驱动系统研发平台由 Rhapsody 或 MagicDraw（模型体制相同）构成，模型驱动软件研发平台由 RTCASE 软件需求建模工具、SCADE 软件设计工具集（主要包括 SCADE Architect、SCADE Suite、SCADE Test）以及 LDRA 测试验证工具集构成，形成从架构到设计到实现的双向追踪与联合仿真。

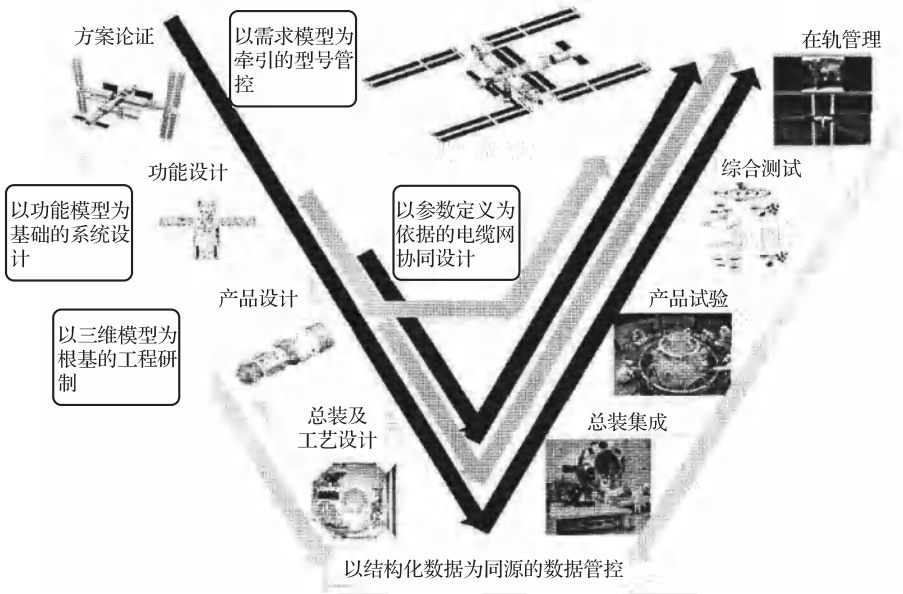

图 2　空间站基于模型的系统工程研制模式

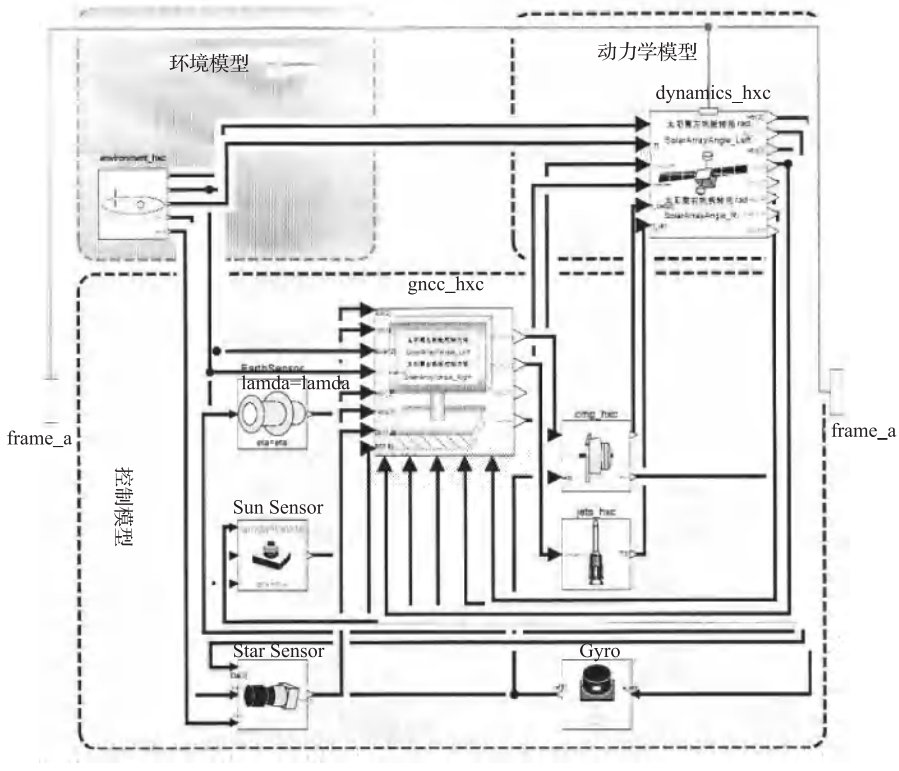

图 3　姿轨控专业仿真模型

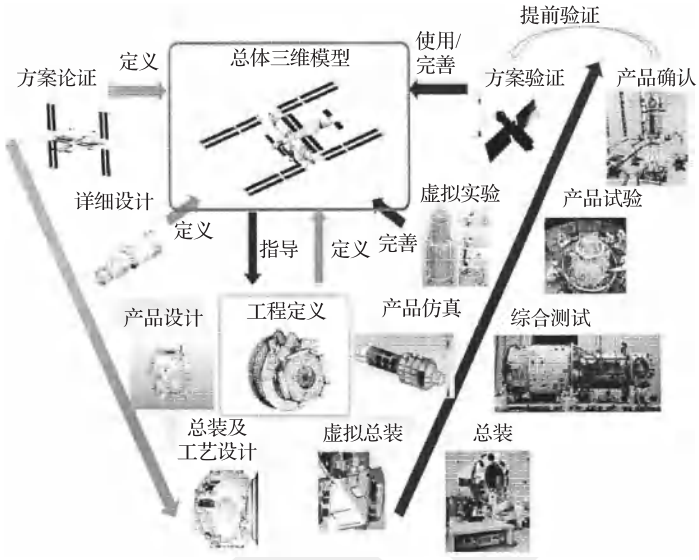

图 4　空间站全三维协同设计

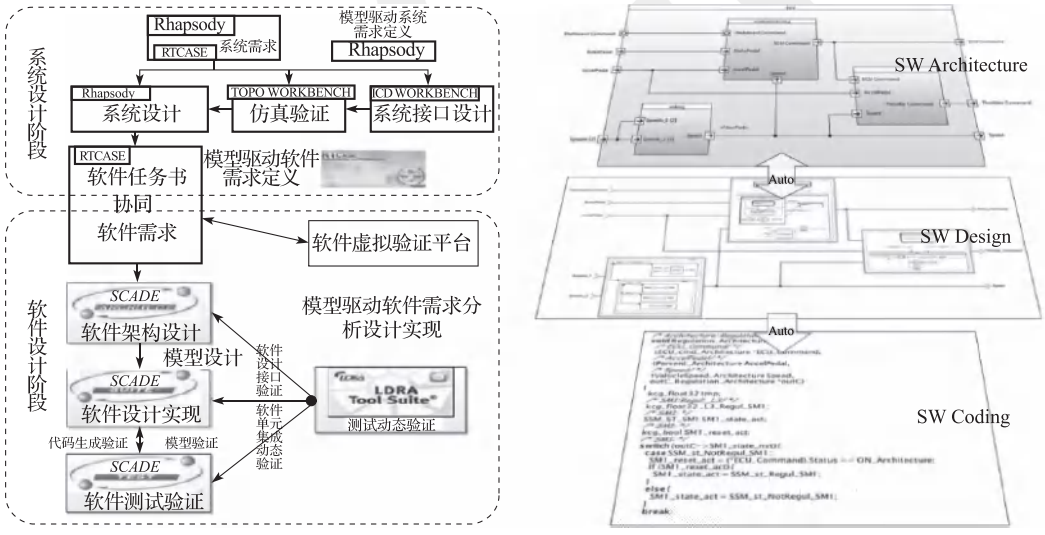

图 5　基于模型的商用工具整合

模型驱动软件开发方法方面，航天目前的研究不如 MBSE 较为成熟，当前基于 SCADE 工具和 MATLAB/Simulink，初步形成了基于模型的软件研制方法，并已经在部分型号论证和模样阶段初步应用，利用 Simulink 建立的模型，进行软件设计、开发、验证、人机交互设计与仿真等工作。基于模型的软件系统工程实践如图 6 所示。

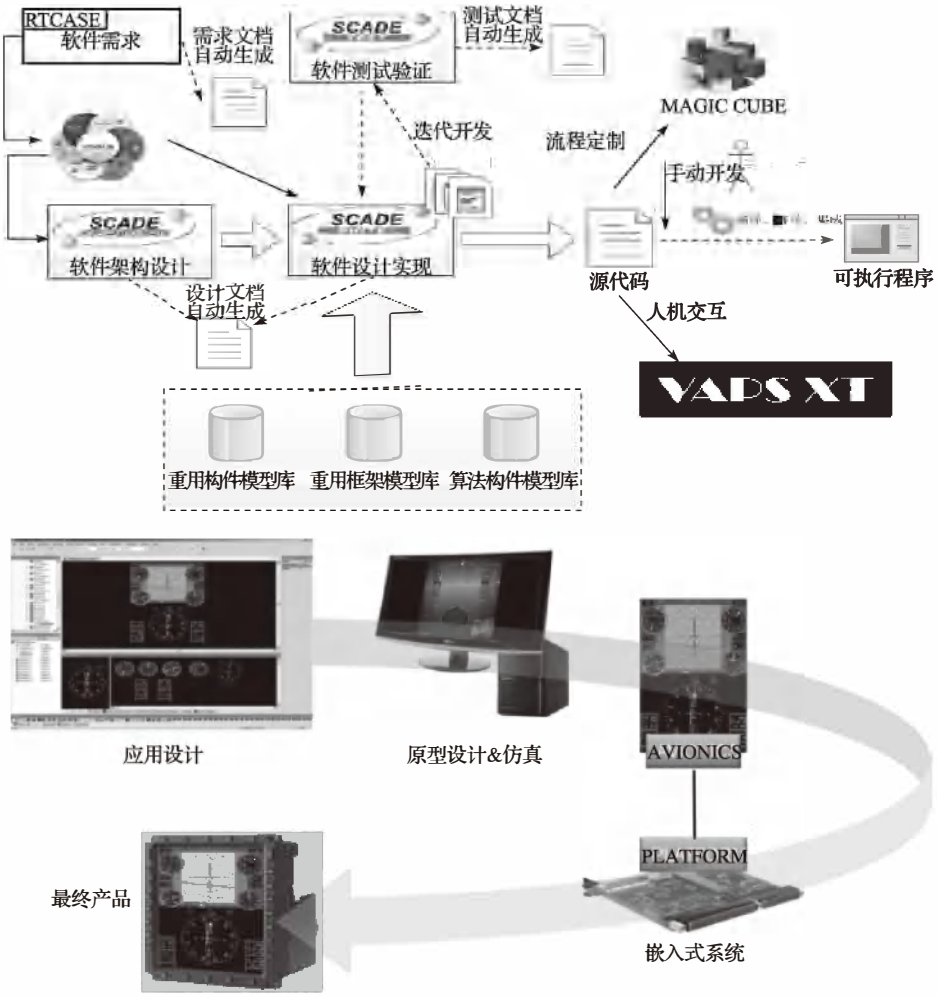

图 6 基于模型的软件系统工程实践

4.2 领域未来需求及建议

从航天飞行器和运载火箭等型号的未来发展趋势来看，智能化和网络化是大势所趋，软件定义装备将是主要技术途径，软件的地位和作用将更加突出，软件将更加复杂、难度更高，软件研制难度的快速增长亟需与时俱进的模型驱动软件开发及形式化验证等新技术、方法和工具的研究和应用，切实保障航天工程未来发展需要。

针对航天后续任务，研究和推广基于模型的系统工程和模型驱动软件开发模式，制定模型驱动开发流程、领域模型、基础构件、软件架构等方面的规范和标准，研发领域建模语言、领域软件参考架构、领域基础模型、领域软件构件库、模型驱动软件工具集，建立自主可控的载人航天模型驱动软件产品线，实现软件的工业化生产，大幅提升软件研制效率和质量保证水平，整体软件研制能力达到同行业国际先进水平。模型驱动的软

件研制流程如图 7 所示。

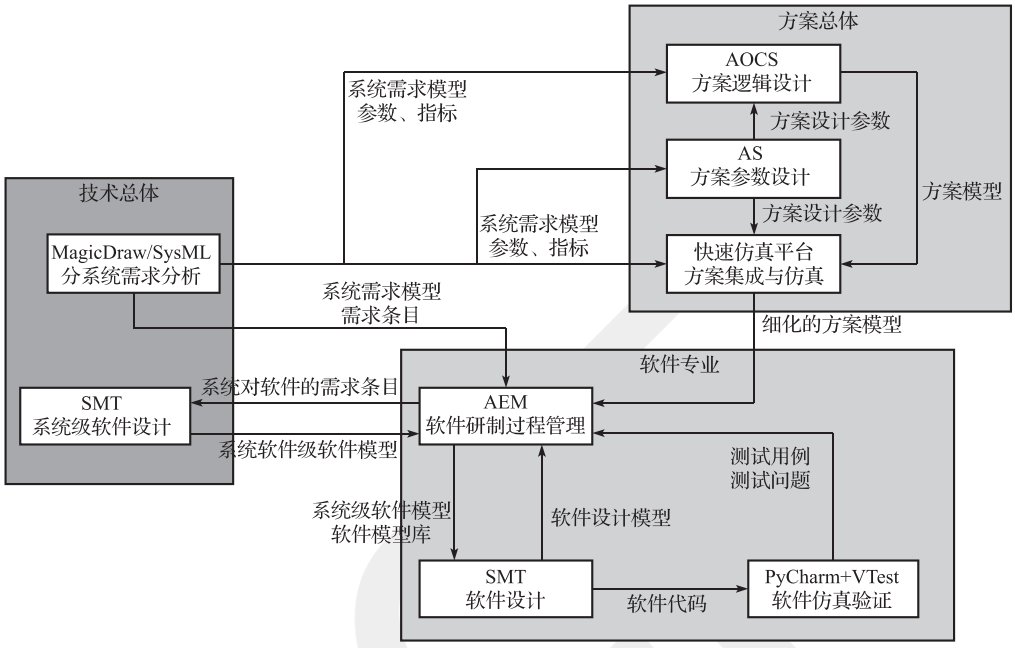

图 7　模型驱动的软件研制流程

建议从以下几方面开展工作：

（1）建立基于模型的研制体系，优化系统及软件研制流程。形成多专业并行的工作模式，提供可协同设计的载体与平台；提前进行方案验证，避免把验证工作都放在集成测试或系统试验阶段，提前发现问题，从而降低缺陷发现和修复的成本。依托基于模型的系统工程可以建立覆盖从系统需求分析到单机设计、软件编码阶段全流程协同设计体系，同时模型可以在生命周期较早阶段提供分析、设计、验证的手段，将研制环节和质量保证环节前移。

（2）建立基于模型的研发平台，提升各专业间协同工作的水平。基于目前国内外通用的支撑平台，针对控制系统型号研制的特点，研究其在控制系统研制全生命周期各阶段中，不同专业间在平台辅助下协同工作的方法，提升各专业间协同工作的水平。目前成熟的商用模型驱动平台可以解决多人协作进行分析和设计产品传递面临的一系列问题。

（3）建立模型的组织资产库，提供可视化、可配置的重用模式。依托统一的平台，将目前各专业可重用的资产进行建模，形成模型组织资产库，并研究可视化、可配置的重用模式，提高重用资产的利用率。基于模型可以进行可视化的组织资产复用，同时文档自动生成可以确保与原有质量评审体系依旧可以实施。

综上所述，开展基于模型的系统工程能够改进现有控制系统研发流程，有效提升系统研制能力；从开发人员来看，能够提升研制效率和质量；从型号全生命周期来看，能够明晰需求、控制变更、促进协调、提升效率；从一个组织来看，能够积累组织资产、规范流程。

5 领域模型和形式化验证方法在航海领域的应用和需求

随着我国海洋装备从以近海为主逐步发展为远近海协同发展，错综复杂的海洋环境促使军民各型号舰船研制需求逐渐体现出智能化、信息化的发展特性，当前舰船设计领域，特别是舰载综合处理计算机的设计呈现出了与传统舰载计算机设计不同的特点和方法。

在设计过程中，舰载综合处理计算机作为安全攸关系统所面临的挑战正在不断增加。首先，随着我国远洋舰船的研制能力不断增强，舰艇及其核心的舰载计算机系统及其他电子装备受到远海条件下高湿度、高震动型和极端恶劣天气条件考验的频率和强度都在不断增加，影响舰船正常航行和作业的各类软硬件故障的出现概率和严重程度也在不断攀升。其次，舰船信息化、智能化程度的不断提升使一些舰船航行、作业相关控制系统的设计和使用理念也发生着变迁。现代舰船的控制非常复杂，物理结构、舰载货物、环境天气和航行状态都有可能影响舰船的姿态和作业动作。在需要解析的信息指数级增长的同时，舰船姿态控制的方式逐渐变得更加精细和繁杂，对面向相关操作的舰载综合处理计算机的实时性要求进一步提升。最后，与复杂海洋航行、作战环境一起发展的，还有舰载综合处理计算机所面临的安全挑战。舰载综合处理计算机不仅需要为不断增加的各类电子设备获取的信息的及时处理和分析提供保障，还需要为不断增加的各类端侧计算平台提供灵活合理的计算服务。端侧设备和用户的大量加入对维护舰载综合处理计算机的信息安全而言是新的考验。与此同时，远近海复杂多变的国际局势和错综复杂的环境也对舰载综合处理计算机的安全防护能力提出了高要求，安全可信操作系统保护技术、虚拟化技术、访问控制技术逐渐成为舰载综合处理计算机设计中的关键技术之一。

这些新设计思路和新技术的引入，使得舰载综合处理计算机的设计任务变得更加复杂和艰巨。在新形势下，为了解决复杂的抗恶劣环境安全关键计算机系统的设计和验证，形式化方法成为舰载综合处理计算机设计的核心方法，而形式化验证也为舰载综合处理计算机中面向上述可靠性、实时性和安全性的设计方案论证提供了重要保障。

5.1 应用现状与成果

5.1.1 面向可靠性的舰载综合处理计算机的形式化验证

对舰载综合处理计算机这一安全攸关系统而言，首先被关注的特性是可靠性。一方面，舰载综合处理计算机在其应用场景下很容易受到各类软硬件故障的影响和干扰。这是因为远海、深海中高湿度、高温差、强酸性和极端天气不利于电子设备的正常工作运转；而舰船运行过程中往往会产生和遭遇较高频率和较高强度的震动，这一因素也会对计算机系统的运行和维护构成威胁。尽管可以采用容错性能较高的工业级器件构建舰载

综合处理计算机系统，但是上述因素还是不可避免会对其造成一定的影响，导致舰载综合处理计算机系统不得不进行一系列可靠性设计。

针对这一情况，舰载综合处理计算机在所采用的基于模型的系统工程方法设计中引入了多模冗余等方法对每个组件进行容错处理，从而确保整体系统的可靠性。对于多模冗余设计中单个组件和整体系统硬件设计的正确性，则是依赖形式化验证手段来实现的。

目前，舰载综合处理计算机设计中主要是引入商业化的成套硬件辅助与设计形式化验证工具链实现上述功能。该工具链在组件设计方案确定、组件硬件结构设计完成后的组件验证和虚拟仿真阶段引入。工具链中包含的主要功能模块分为抽象问题表示模块、问题编码模块和问题推理引擎三个部分。与系统硬件设计紧密相关的数据结构首先由抽象问题表示模块抽象为形式化的模型；然后，问题编码模块负责将抽象出的问题表示转换为满足特定推理引擎要求的数据结构；最后，问题推理引擎针对问题编码模块移交的问题进行具体的推理和验证，从而确定舰载综合处理计算机中各个组件可靠性容错设计的覆盖情况。根据基于模型的系统工程方法的设计流程，经过验证后的组件经过必要的设计修正和重验证后，可以被视为具有符合要求可靠性的系统组件进行系统集成验证。通过虚拟集成技术和上述形式化验证工具链，舰载综合处理计算机设计者能够对其系统硬件可靠性进行完备的整体验证。

在软件设计方面，在当前的舰载综合处理计算机的设计中，也开始考虑通过引入STPA，实现对软件需求缺陷、系统交互等潜在的软件安全性问题归因的量化分析。在舰载综合处理计算机的软件系统研制过程中，首先根据整个系统的功能组成，构建系统的分层控制结构模型；通过对历史系统级事故的归因和危险分析，确定潜在的非安全行为，从而确定软件的安全性需求。然后，将安全性需求进一步形式化。接着，使用 Simulink Stateflow[129] 等工具针对系统中软件的安全控制过程进行抽象，构建出软件安全控制行为逻辑的状态图模型及其对应的形式化表示。最后，按照形式化方法依托模型检查工具完成对软件安全属性的验证。

5.1.2　面向实时性的舰载综合处理计算机的形式化验证

智能化是现代舰船的发展趋势。近年来，自动化技术和计算机技术的发展为舰船的自动化提供了强大的助力。舰船的自动化对提高舰船运行效率和作业效率、改善舰船工作人员和乘员的工作生活条件、提高舰船的可靠性有重要的意义。

当前，我国智能舰船的设计和制造正处在迅猛发展之中。在智能化舰船这一复杂的集成系统中，既存在动力系统、电力系统和导航系统这样早已有之，而今则融合了新技术、新方法的主要航行作业系统，还增加了偏航系统、电站配电系统等伴随其他门类新技术产生、能有效助力舰船航行和作业的新生辅助机械控制系统。这些系统结构复杂，运行环境恶劣，对精确控制的要求非常高，传统的人类经验与简单机械结构辅助的方式几乎不可能满足当前智能舰船航行作业过程中的实时性控制要求。

因此，将 PLC 等专为面向工业控制而设计的特殊计算机系统引入舰船机电控制及相关辅助系统中的设计思路，已经成为近来智能舰船设计的潮流。PLC 具有可接驳多种类

型信号数据接口的特性，采用了接近控制电路的编程语言进行描述，典型的编程语言包括了功能块图（FBD）、梯形图（LD）、指令表（IL）、结构化文本（ST）和功能顺序图（SFC）五种国际标准指定的语言[130]。其基本结构与功能包括了存储器、各类逻辑运算、控制指令等。随着 PLC 技术研究的深入，PLC 的性能，尤其是冗余技术和可靠性等方面的性能，已经获得了长足的发展。因此，美军在实施开放架构的机电控制系统（OAMCS）计划中就大量采用了 PLC 及其相关编程软件。我国在当前的智能舰船设计中，也在逐步进行将 PLC 等相关技术融入舰载综合处理计算机的尝试，以利于舰船的模块化、系统化设计。

舰船控制系统对实时性的要求很高，除了要求系统对指令做出正确反馈之外，还需要系统满足对应的时间约束。在这种情况下，即使对普通计算机系统影响有限的延迟攻击等攻击方式，也会对舰载综合处理计算机中面向舰船控制的软硬件功能部件造成巨大影响，从而影响舰船正常的航行和作业，严重的甚至会对舰船工作人员和乘员、舰船和舰船负载造成人员伤亡和经济损失。考虑到实时性因素，需要依托形式化验证工具链建立时间自动机模型，模拟系统中各个组件传递信息、处理数据、判决输出等全流程机制，从而对系统进行可达性判定，分析系统的安全性，并针对该形式化验证分析设计优化策略，然后对多种优化策略再次进行时间自动机建模分析。通过反复的优化-迭代过程，实现对舰船控制系统的实时性设计与分析。

除此之外，如 Petri 网、贝叶斯网络、自动机等其他模型检测方法也能将给定舰船控制系统程序转化为形式化的模型，实现基于状态迁移系统的自动验证技术，检查给定的系统在功能和时效上是否满足规范，从而确保舰船控制系统的安全可信。

5.1.3 面向安全性的舰载综合处理计算机的形式化验证

目前，面向信息处理的舰载综合处理计算机的算力需求正处于爆炸式增长之中。早期的舰载综合处理计算机只需要为核心显控台提供服务，仅需要处理少数传感器获取和通信设备收发的信息，用户少、进程可控、数据安全可信，计算机系统的安全保障较为简单。然而，面对信息化时代的海洋，舰船装备亟待处理的信息量和需要服务交流的用户数量指数式增加。特别是进入智能化时代以来，舰载综合处理计算机广泛面临着提供更加复杂的智能化服务需求，舰艇内、舰艇间以及舰艇和无人舰船之间多个信息源和信息处理终端之间需要频繁密集地通过舰载综合处理计算机进行数据协同处理和交互，不同层级、拥有不同权限、对计算资源需求各不相同的大量用户接入舰载综合处理计算机，要求提供数据访问/共享、协同处理等服务。此外，当前也出现了将 PLC 等舰船控制系统融入舰载综合处理计算机的设计潮流。

这种现状推动了当前舰载综合处理计算机以“将网络作为高性能计算机”作为主要实现架构，通过网络连接对庞大的计算资源进行统一管理和调度。然而，舰载综合处理计算机与舰船控制系统密切结合，具有对安全性高度敏感的特点。将舰船控制系统融入舰载综合处理计算机的设计思路，虽然有利于提高控制系统的互操作性、互联性、开放性，有利于利用标准化的通信协议，能够降低开发成本、提高生产效率，但同时也带来

了一定的安全隐患，使得系统和设备更易受到外部网络的攻击。在民用云计算领域中暴露的数据安全管理问题促使舰载综合处理计算机设计者必须提出对应的解决方案。

为了保证舰载综合处理计算机的数据存取和传输的安全性，当前舰载综合处理计算机的设计中引入了多种访问控制模型。结合身份验证、授权和访问控制模型，舰载综合处理计算机能够依据不同用户的身份对特定系统、资源和应用程序的访问进行允许、限制或拒绝的判定和控制。由于舰艇、乘员和负载之间通常需要灵活组合，舰载综合处理计算机中的用户和数据拥有者往往也不在同一个安全域中，系统通常是通过用户的属性或特征而不是通过预定义的身份来表示用户。因此，舰载综合处理计算机中通常采用了针对云计算复杂环境的存取访问控制模型保证数据安全性。典型的存取访问控制模型包括基于角色的存取访问控制模型（RBAC）、基于使用控制的存取访问模型（UCONAC）、基于属性的访问控制模型（ABAC）、高时效安全访问控制（TESAC）等。当舰艇信息处理终端需要从舰载综合处理计算机中获取数据或其他云资源时，需要向舰载综合处理计算机提出请求。舰载综合处理计算机根据采用的访问控制模型对用户进行一系列访问控制，通过验证用户从数据拥有者处得到的相关密钥和证书确定用户的访问权限。

为了确保舰载综合处理计算机访问控制模型的安全性，计算机系统设计者引入了形式化方法验证该访问控制系统的可信性。具体地，诸如模型检测器 PAT（Process Analysis Toolkit）[131]在内的成熟模型检测工具已经被介绍和使用到了舰载综合处理计算机访问控制模型的形式化验证阶段。该工具此前已经被广泛应用于并发实时系统、概率模型和传感器网络等多个领域的仿真推理和验证，它可通过不同的功能模块将多种模型检测方法嵌入工具中，能有效检验系统的死锁、可到达性和 LTL 的各种属性，进而顺利发现系统的各类错误和漏洞。针对访问控制的形式化验证就是利用了该工具的特点，通过对访问控制模型的无死锁性，验证证明访问控制模型能够正常运转；通过形式化入侵者可能造成的影响检测系统的反应，验证访问控制模型的顺序性、完备性和安全性；通过与其他典型应用模块的联合仿真，验证访问控制模型的性能开销对实时性指控系统的影响在可控范围之内。

5.2　领域未来需求及建议

尽管形式化验证技术已经在舰载综合处理计算机的设计中有了初步的应用和尝试，然而目前仍然有大量问题和挑战，亟须解决。

首先，现代舰船设计是一个典型的复杂系统设计问题，其基本特点是非线性、多目标、多约束、多耦合和并行。因此，单一设计模式很难解决对象、约束和规程之间的关系。这一问题对于舰载综合处理计算机而言也同样成立，与其设计紧密相关的物理特性、数据获取方式、通信协议、决策博弈理论等问题，很多并不是舰载综合处理计算机设计单位之所长，需要结合各学科、各领域的专业理论进行协同设计，才能确保舰载综合处理计算机的安全性。然而，由于历史原因，各学科各自有独立的形式化方法，目前尚未形成能够统一综合集成进行形式化验证的平台。因此，当前的舰载综合处理计算机中，

涉及跨学科、跨领域的问题只能依赖测试仿真和经验范式等传统设计方法。这一问题已经成为我国舰载综合处理计算机领域亟须突破的问题之一。

其次，出于安全性方面的考虑，当前舰载综合处理计算机的形式化验证主要依赖经过行业实践检验的工具链，该工具链主要由国外工业软件构成。这一形势对我国发展自主可控的海洋装备构成了严重的威胁。而随着国际局势的波动，舰载综合处理计算机所需的形式化验证工具后续使用的安全性成为一个问题。因此，需要构建自主可控的计算机系统形式化验证工具，为可持续的智能化舰船设计和发展奠定基础。

最后，我国海洋装备的发展史见证了舰载综合处理计算机不断前进的历程。随着舰载综合处理计算机复杂程度的逐渐提高，形式化验证技术逐渐成为舰载计算机领域甚至是舰船设计领域的关键技术之一。智能化时代的到来和舰船控制系统的融入更加凸显了形式化验证技术的重要性和复杂性，成为保障舰船航行、作业必不可少的关键设计环节。

6 结束语

总之，当前面向安全攸关软件系统的领域建模和形式化验证方法已经取得了较大的进步，在国际上一些大型的航空航天企业和研究机构也已经进行了一些成功的应用尝试，然而国内在这方面还处于起步阶段，将领域建模和形式化验证真正用于安全攸关软件系统的开发还存在一些有待解决的问题，但在多个领域中现有的软件开发模式已经开始不能适应当前领域的发展要求，采用领域建模和形式化验证等增强软件开发质量的新手段已经成为未来发展的一种趋势。因此，探索如何将领域建模和形式化验证等方法用于我国航空、航天和航海领域的安全攸关软件系统开发，是保障我国安全攸关软件生产安全的重要途径。

参考文献

[1] MERNIK M, HEERING J, SLOANE A M. When and how to develop domain-specific languages[J]. ACM computing surveys (CSUR), 2005, 37(4):316-344.

[2] WEISS D M, LAI C T R. Software product-line engineering: a family-based software development process[M]. Upper Saddle River: Addison-Wesley Longman Publishing Co., Inc., 1999.

[3] FRANK U. Domain-specific modeling languages: requirements analysis and design guidelines[M]. Berlin: Springer, 2013.

[4] TOLVANEN J-P, KELLY S. Effort used to create domain-specific modeling languages[C]// Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems. 2018.

[5] FAN ZQ, YUE T, ZHANG L. A generic framework for deriving architecture modeling methods for large-scale software-intensive systems[C]// Proceedings of the 28th Annual ACM Symposium on Applied

Computing. 2013.

[6] FAN ZQ, YUE T, ZHANG L. SAMM: an architecture modeling methodology for ship command and control systems[J]. Software & systems modeling, 2016, 15(1): 71-118.

[7] WANG Z, et al. Spardl: a requirement modeling language for periodic control system[C]// International Symposium On Leveraging Applications of Formal Methods, Verification and Validation. Berlin: Springer, 2010.

[8] GUO WZ, ZHANG L, LIAN X L. Automatically detecting the conflicts between software requirements based on finer semantic analysis[Z]. arXiv 2021:2103.02255.

[9] EZZINI S, et al. Using domain-specific corpora for improved handling of ambiguity in requirements[C]// 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). New York: IEEE, 2021.

[10] WESTON N, CHITCHYAN R, RASHID A. Formal semantic conflict detection in aspect-oriented requirements[J]. Requirements engineering, 2009, 14(4): 247.

[11] DEGIOVANNI R, et al. Goal-conflict detection based on temporal satisfiability checking[C]// 2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE). New York: IEEE, 2016.

[12] PNUELI A. The temporal logic of programs[C]// 18th Annual Symposium on Foundations of Computer Science (SFCS 1977). New York: IEEE, 1977.

[13] GE N, et al. RT-MOBS: a compositional observer semantics of time Petri net for real-time property specification language based on μ-calculus[J]. Science of computer programming, 2021,206: 102624.

[14] GE N. Property driven verification framework: application to real time property for UML MARTE software design[D]. Toulouse: Université Paul Sabatier, 2014.

[15] LECRUBIER V, D'AUSBOURG B, AIT-AMEUR Y. The LIDL Interaction Description Language [C]// EICS. 2015.

[16] GE N, et al. Formal development process of safety-critical embedded human machine interface systems[C]// 2017 International Symposium on Theoretical Aspects of Software Engineering (TASE). New York: IEEE, 2017.

[17] HAUSE M. The SysML modelling language[C]// Fifteenth European Systems Engineering Conference. 2006.

[18] FÜRST S, et al. AUTOSAR-A Worldwide Standard is on the Road[C]// 14th International VDI Congress Electronic Systems for Vehicles, Baden-Baden. 2009.

[19] FEILER P H, GLUCH D P, HUDAK J J. The architecture analysis & design language (AADL): An introduction[R]. Pittsburgh: Carnegie-Mellon University, 2006.

[20] HALBWACHS N, et al. The synchronous data flow programming language LUSTRE[C]// Proceedings of the IEEE. 1991.

[21] DORMOY F-X. SCADE 6: a model based solution for safety critical software development [C]// Proceedings of the 4th European Congress on Embedded Real Time Software (ERTS'08). 2008.

[22] DAJANI-BROWN S, et al. Formal modeling and analysis of an avionics triplex sensor voter[C]// International SPIN Workshop on Model Checking of Software. Berlin: Springer, 2003.

[23] MARKEL T, et al. ADVISOR: a systems analysis tool for advanced vehicle modeling[J]. Journal of power sources, 2002,10(2): 255-266.

[24] TARIQ M U, FLORENCE J, WOLF M. Design Specification of Cyber-Physical Systems: towards a Domain-Specific Modeling Language based on Simulink, Eclipse Modeling Framework, and Giotto[C]//

ACESMB@ MoDELS. 2014.

[25] RTCA. Software Considerations in Airborne Systems and Equipment Certification: DO-178B [S]. Washington: RTCA Inc., 1992.

[26] RTCA. Software Considerations in Airborne Systems and Equipment Certification: DO-178B [S]. Washington: RTCA Inc., 2011.

[27] Common Criteria for Information Technology Security Evaluation (v3.1, Release 5)[EB/OL]. https://www.commoncriteriaportal.org.

[28] GU T, KOENIG J, RAMANANANDRO T, et al. Deep specifications and certified abstraction layers[J]. ACM SIGPLAN Notices, 2015, 50(1): 595-608.

[29] GU T, SHAO Z, CHEN H, et al. CertiKOS: An extensible architecture for building certified Concurrent OS Kernels [C]// Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah. 2016.

[30] CHEN HN, WU X, SHAO Z, et al. Toward compositional verification of interruptible OS kernels and device drivers[C]// Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation, Santa Barbara. 2016.

[31] COSTANZO D, SHAO Z, GU R. End-to-end verification of information-flow security for C and assembly programs[J]. ACM SIGPLAN notices. 2016, 51(6): 648-664.

[32] KREBBERS R, LEROY X, WIEDIJK F. Formal C semantics: CompCert and the C standard[C]// International Conference on Interactive Theorem Proving. 2014.

[33] LIU M, et al. Compositional verification of preemptive OS kernels with temporal and spatial isolation: technical report, YALEU/DCS/TR-1549[R]. New Haven: Department of Computer Science, Yale University, 2019.

[34] LIEDTKE J. On micro-kernel construction[J]. ACM SIGOPS operating systems review, 1995, 29(5): 237-250.

[35] GIEN M. Micro-kernel architecture key to modern operating systems design[J]. UNIX review, 1990, 8(11): 58-60.

[36] KLEIN G, ELPHINSTONE K, HEISER G, et al. seL4: formal verification of an OS kernel[C]// Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, New York. 2009.

[37] ANDRONICK, MORGAN C, KLEIN G. Formal Model of a Multi Core Kernel-based System[R]. Sydney: National ICT Australia Limited, 2018.

[38] SYEDA H T, KLEIN G. Formal reasoning under cached address translation[J]. Journal of automated reasoning, 2020: 1-35.

[39] HEISER G, KLEIN G, MURRAY T. Can we prove time protection? [C]// Proceedings of the Workshop on Hot Topics in Operating Systems, New York. 2019.

[40] NELSON L, SIGURBJARNARSON H, ZHANG K, et al. Hyperkernel: push-button verification of an OS kernel[C]// The 26th Symposium on Operating Systems Principles, Shanghai. 2017.

[41] SIGURBJARNARSON H, BORNHOLT J, TORLAK E, et al. Push-button verification of file systems via crash refinement[C]//12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). 2016.

[42] KAISER R, WAGNER S. Evolution of the PikeOS microkernel[C]//First International Workshop on Microkernels for Embedded Systems. 2007.

[43] VERBEEK F, HAVLE O, SCHMALTZ J, et al. Formal API specification of the PikeOS separation kernel[C]// NASA Formal Methods Symposium. Cham: Springer, 2015.

[44] Green Hills Software, Inc. INTEGRITY real-time operating system[EB/OL]. http://www.ghs.com/products/rtos/integrity.html.

[45] Lynx. LynxOS-178 RTOS [EB/OL]. [2015-07-21]. http://www.lynx.com/products/real-time-operating systems/lynxos-178rtos-for-do-178bsoftware-certification.

[46] Lynx. LynxSecure Separation Kernel Hypervisor[EB/OL]. http://www.lynx.com/products/hypervisors/lynxsecure-separationkernel-hypervisor/.

[47] ZHAO Y, YANG Z, SANÁN D, et al. Event-based formalization of safety-critical operating system standards: an experience report on ARINC 653 using Event-B[C]//2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE). New York: IEEE, 2015: 281-292.

[48] The Zephyr Project[EB/OL]. [2018-12-31]. https://www.zephyrproject.org/.

[49] FENG Z, YONGWANG Z, DIANFU M, et al. Fine-grained Formal Specification and analysis of buddy memory allocation in Zephyr RTOS[C]//2019 IEEE 22nd International Symposium on Real-time Distributed Computing (ISORC). New York: IEEE, 2019: 10-17.

[50] The real-time kernel: μC/OS-II[EB/OL]. http://micrium.com/rtos/ucosii/overview.

[51] XU F, FU M, FENG X, et al. A practical verification framework for preemptive OS kernels[C]// International Conference on Computer Aided Verification. Cham: Springer, 2016: 59-79.

[52] SHI J, HE J, ZHU H, et al. ORIENTAIS:formal verified OSEK/VDX real-time operating system[C]// IEEE 17th International Conference on Engineering of Complex Computer Systems, Paris. 2012.

[53] LEROY X. The CompCert C verified compiler: documentation and user's manual[R/OL]. Rocquencourt: INRIA Paris-Rocquencourt, 2012. http://compcert.inria.fr/man/manual.pdf/.

[54] AHARON A, GOODMAN D, LEVINGER M, et al. Test program generation for functional verification of PowerPC processors in IBM[C]// Design Automation Conference. San Francisco, CA: IEEE, 2005: 279-285.

[55] INGO S. System modeling and design refinement in ForSyDe[J]. Handbook of hardware/software codesign, 2017, 23(1): 99-140.

[56] WILDING M M, GREVE D A, RICHARDS R J, et al. Formal verification of partition management for the AAMP7G Microprocessor[M]. New York: Springer US, 2010: 30-40.

[57] FOX A C J, M O MYREEN. A trustworthy monadic formalization of the ARMv7 instruction set architecture[C]// International Conference on Interactive Theorem Proving. Edinburgh, UK: Springer, 2010: 243-258.

[58] GOEL S, HUNT W A, KAUFMANN M. Engineering a formal, executable x86 ISA simulator for software verification[J]. Provably correct systems, 2017, 4(8): 173-209.

[59] SYEDA H T, KLEIN G. Program verification in the presence of cached address translation[C]// Interactive Theorem Proving-9th International Conference. Oxford, UK: Springer, 2018: 542-559.

[60] HOU Z, SANAN D, TIU A, et al. An executable formalisation of the SPARCv8 instruction set architecture: a Case Study for the LEON3 Processor[C]// International Symposium on Formal Methods. Limassol, Cyprus:Springer International Publishing, 2016: 388-405.

[61] BEERS R. Pre-RTL formal verification: an Intel experience[C]// ACM/IEEE Design Automation Conference. Anaheim, CA, USA: IEEE, 2008: 806-811.

[62] ITOH M et al. PEAS-III: an ASIP Design Environment[C]// International Conference on Computer Design, Austin, TX, USA:IEEE, 2000: 430-436.

[63] KITAJIMA A, ITOH M, SATO J, et al. Effectiveness of the ASIP design system PEAS-III in design of pipelined processors[C]// Asia and South Pacific Design Automation Conference. Yokohama, Japan: IEEE, 2001: 649-654

[64] ALOMARY A, NAKATA T, HONMA Y, et al. PEAS-I: A hardware/software co-design system for ASIPs[C]// European Design Automation Conference. Hamburg, Germany:IEEE, 1993: 2-7.

[65] KROCNING D, PAUL J W. Automated pipeline design[C]// Design Automation Conference(DAC). Las Vegas, NV, USA:IEEE, 2001: 810-815.

[66] KROCNING D, PAUL J W. Proving the correctness of pipelined micro-architectures[C]// Formal Methods in Computer-Aided Design. Austin, TX, USA:IEEE, 2000: 161-178.

[67] VELEV M N, Formal verification of pipelined microprocessors with delayed branches[C]// International Symposium on Quality Electronic Design (ISQED'06). San Jose, CA:IEEE, 2006: 4-299.

[68] BRYANT R E, O'HALLARON D R. Computer systems: a programmer's perspective[M]. New York: Pearson, 2015: 391-446.

[69] ALGLAVE J, MARANGET L, TAUTSCHNIG M. Herding cats: modelling, simulation, testing, and data mining for weak memory[J]. ACM transactions on programming languages and systems, 2014, 36(2):7.

[70] 吴瑞阳, 汪文祥, 王焕东. 龙芯 GS464E 处理器核架构设计[J]. 中国科学, 2015, 45(4): 480-500.

[71] HOARE T. The verifying compiler: a grand challenge for computing research[C]//Joint Modular Languages Conference. Berlin: Springer, 2003: 25-35.

[72] MCCARTHY J, PAINTER J. Correctness of a compiler for arithmetic expressions[J]. Proc. sympos. appl. math, 1967, XIX:33-41.

[73] MILNER R, WEYHRAUCH R. Proving compiler correctness in a mechanized logic[C]. 1972.

[74] HOARE C A R, HE J F. Refinement algebra proves correctness of compilation[M]//Programming and Mathematical Method. Berlin: Springer, 1992: 245-269.

[75] STEPNEY S, WHITLEY D, COOPER D, et al. A demonstrably correct compiler[J]. Formal aspects of computing, 1991, 3(1): 58-101.

[76] GAUL T, GOOS G, HEBERLE A, et al. An architecture for verified compiler construction[C]//Joint Modular Languages Conference. 1996.

[77] BLAZY S, DARGAYE Z, LEROY X. Formal verification of a C compiler front-end[M]// FM 2006: Formal Methods. Berlin: Springer, 2006:460-475.

[78] LEROY X. Formal certification of a compiler back-end[J]. Principles of programming languages, 2005, 41(1):42-54.

[79] LEROY X. Formal Verification of an Optimizing Compiler[M]// Term Rewriting and Applications. 2007:1-1.

[80] 刘诚,陈意云,葛琳,等. 一个出具证明的编译器原型系统的实现[J]. 计算机工程与应用,2007,43(21):99-102.

[81] 葛林,陈意云,华保健等. 汇编代码验证中的形式规范自动生成[J]. 小型微型计算机系统,2008(7):1219-1224.

[82] SAE. Architecture Analysis & Design Language: standard SAE AS5506[S]. Warrendale: SAE, 2004.

[83] SAE. Architecture Analysis & Design Language: standard SAE AS5506A[S]. Warrendale: SAE, 2009.

[84] WANG Y, MA D, ZHAO Y W, et al. An AADL-based modeling method for ARINC653-based avionics software[C]// IEEE, Computer Software and Applications Conference. New York: IEEE Computer Society, 2011:224-229.

[85] YANG Z, HU K, ZHAO Y W, Ma D, et al. From AADL to timed abstract state machines: a verified model transformation[J]. Journal of systems & software, 2014, 93(2):42-68.

[86] 何群. C编译器自动测试工具的剖析与移植[J]. 计算机工程, 2004(20):95-97.

[87] Plum Hall Inc.. The Plum Hall Validation Suite For CTM[Z/OL]. [2021-08-21]. http://www.plumhall.com/stecl.html.

[88] Ball T, Rajamani S K. The S LAM project: debugging system software via static analysis[C]//ACM SIGPLAN Notices. ACM, 2002, 37(1): 1-3.

[89] Henzinger T A, Jhala R, Majumdar R, et al. Lazy abstraction[J]. ACM SIGPLAN Notices, 2002, 37(1): 58-70.

[90] Chaki S, Clarke E M, Groce A, et al. Modular verification of software components in C[J]. IEEE Transactions on Software Engineering, 2004, 30(6): 388-402.

[91] Nelson G. Extended static checking for java[C]//International Conference on Mathematics of Program Construction. Springer Berlin Heidelberg, 2004: 1-1.

[92] Havelund K, Pressburger T. Model checking java programs using java pathfinder[J]. International Journal on Software Tools for Technology Transfer (STTT), 2000, 2(4): 366-381.

[93] Corbett J C, Dwyer M B, Hatcliff J, et al. Bandera: Extracting finite-state models from Java source code[C]//Software Engineering, 2000. Proceedings of the 2000 International Conference on. IEEE, 2000: 439-448.

[94] Huimin L. Research on High Reliability Software: Investing into the Future of Information Technology [J]. Bulletin of the Chinese Academy of Sciences, 2002, 6: 001.

[95] Chen H W, Wang J, Dong W. High confidence software engineering technologies[J]. Acta Electronica Sinica, 2003, 31(12A): 1933-1938.

[96] CHEN X H, et al. Automating consistency verification of safety requirements for railway interlocking systems[C]// 2019 IEEE 27th International Requirements Engineering Conference (RE). New York: IEEE, 2019.

[97] KOPETZ H. Software engineering for real-time: a roadmap[C]// Proceedings of the Conference on the Future of Software Engineering. 2000.

[98] 王博, 白晓颖, 贺飞, 等. 可组合嵌入式软件建模与验证技术研究综述[J]. 软件学报, 2014, (2): 234-253.

[99] MOUELHI S, CHOUALI S, MOUNTASSIR H. Invariant preservation by component composition using semantical interface automata[C]// Proceedings of the Sixth International Conference on Software Engineering Advances ICSEA. 2011.

[100] GRUNSKE L, COLVIN R, WINTER K. Probabilistic model-checking support for FMEA[C]// Fourth International Conference on the Quantitative Evaluation of Systems (QEST 2007). New York: IEEE, 2007.

[101] CHKOURI M Y, et al. Translating AADL into BIP-application to the verification of real-time systems[C]// International Conference on Model Driven Engineering Languages and Systems. Berlin: Springer, 2008.

[102] BERTHOMIEU B, et al. Formal verification of AADL specifications in the topcased environment[C]//

International Conference on Reliable Software Technologies. Berlin: Springer, Heidelberg, 2009.

[103] ABDULLA P A, et al. Designing safe, reliable systems using scade[C]// International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Berlin: Springer, 2004.

[104] WALDE G, LUCKNER R. Bridging the tool gap for model-based design from flight control function design in Simulink to software design in SCADE[C]// 2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC). New York: IEEE, 2016.

[105] SHI J, et al. SCADE2Nu: a tool for verifying safety requirements of SCADE models with temporal specifications[C]// REFSQ Workshops. 2019.

[106] BASOLD H, et al. An open alternative for SMT-based verification of SCADE models[C]// International Workshop on Formal Methods for Industrial Critical Systems. Cham: Springer, 2014.

[107] HAMON G, et al. Simulink design verifier-applying automated formal methods to Simulink and Stateflow[C]// Third Workshop on Automated Formal Methods. 2008.

[108] REICHERDT R, GLESNER S. Formal verification of discrete-time MATLAB/Simulink models using Boogie [C]// International Conference on Software Engineering and Formal Methods. Cham: Springer, 2014.

[109] ZHAN N J, WANG S L, ZHAO H J. Formal verification of Simulink/Stateflow diagrams[M]. Cham: Springer, 2017.

[110] FONSECA P, ZHANG K, XI W, et al. An empirical study on the correctness of formally verified distributed systems[C]// the Twelfth European Conference. New York: ACM, 2017.

[111] FULTON N, PLATZER A . Safe reinforcement learning via formal methods: toward safe control through proof and learning[C]// AAAI Conference on Artificial Intelligence. 2018.

[112] COLACO J-L, PAGANO B, et al. SCADE 6: from a Kahn Semantics to a Kahn Implementation for multicore[C]// 2018 Forum on specification & Design Languages (FDL). 2018.

[113] LOPES M K, FRANÇA R B, et al. Enhancing range analysis in software design models by detecting floating-point absorption and cancellation[M]// Information Technology New Generations. Berlin: Springer, 2018.

[114] 方伟, 周彰毅. SCADE 在航空发动机 FADEC 软件开发中的应用[J]. 航空发动机, 2016, 42(5): 43-47.

[115] 王亮亮, 薛芳芳, 曹琳,等. 机载 FMS 航路管理磁航向角计算方法研究及 SCADE 建模实现[J]. 航空计算技术, 2018, 48(5): 14-17.

[116] 程黎. SCADE 在无人机飞行控制软件设计中的应用[D]. 西安:西安电子科技大学, 2011.

[117] 吴康. 机载安全关键软件模型验证技术研究[D]. 天津:中国民航大学, 2020.

[118] COFER D, MILLER S. DO-333 certification case studies[C]// Internation Symposium on NASA Formal Methods. Berlin: Springer, 2014, 1-15.

[119] MOY Y, LEDINOT E, et al. Testing or formal verifcation: DO-178C alternatives and industrial experience[J]. IEEE software, 2013, 13(6): 50-57.

[120] PETOT G, BOTELLA B, et al. Instrumentation of annotated C programs for test generation[C]// 14th IEEE International Working Conference on Source Code Analysis & Manipulation. 2014.

[121] KOSMATOV N, PETIOT G, et al. How testing helps to diagnose proof failures[J]. Formal aspects of computing, 2018, 30: 629-657.

[122] HAVELUND K, LOWRY M, PENIX J. Formal analysis of a space-craft controller using SPIN[J].

IEEE transactions on software engineering, 2001, 27(8): 749-765.
[123] BOCHOT T, VIRELIZIER P, et al. Model checking flight control systems: the Airbus experience[C]// International Conference on Software Engineering-companion Volume. IEEE. 2009.
[124] 金志威, 田毅, 芦浩,等. 基于模型检测的机载电子硬件验证方法研究[J]. 现代电子技术, 2019, 42(16): 6-9.
[125] 林荣峰, 施健, 朱晏庆,等. 基于 STP 方法的 SCADE 模型形式化验证框架[J]. 计算机工程, 2019, 45(10): 70-77.
[126] 侯刚, 周宽久, 勇嘉伟,等. 模型检测中状态爆炸问题研究综述[J]. 计算机科学, 2013(6A): 77-86.
[127] 王瑞. 基于 SAT 的符号化模型检验技术研究[D]. 长沙:国防科技大学, 2014.
[128] 张程灏. 机载软件建模及其形式化验证方法研究[D]. 成都:电子科技大学, 2017.
[129] ZOU L, ZHAN N, WANG S, et al. Formal verification of Simulink/Stateflow diagrams [C]// Proceedings of the International Symposium on Automated Technology for Verification and Analysis. Berlin: Springer, 2015.
[130] JOHN K-H, TIEGELKAMP M. IEC 61131-3: programming industrial automation systems[M]. Berlin: Springer, 2010.
[131] SUN J, LIU Y, DONG J S. Model checking CSP revisited: introducing a process analysis toolkit[C]// Proceedings of the International symposium on leveraging applications of formal methods, verification and validation[C]. Beriln: Springer, 2008.

作者简介

马殿富 北京航空航天大学，教授，安全关键软件、服务计算、非结构化数据组织与处理，CCF 会士、监事、CCF 抗恶劣环境计算机专业委员会常务委员。

牛文生 中航机载共性技术工程中心，研究员，计算机技术，软件技术，CCF 抗恶劣环境专业委员会副主任。

程 胜 北京神舟航天软件技术有限公司，研究员，实时操作系统、数据库管理系统、装备计算机系统。

马　中　武汉数字工程研究所，研究员，舰载嵌入式计算机技术，CCF 抗恶劣环境专业委员会委员。

宋　富　上海科技大学，长聘副教授，形式化验证，系统安全和人工智能安全，CCF 形式化方法专业委员会委员，CCF 系统软件专业委员会委员，CCF 高级会员。

罗　杰　北京航空航天大学，副教授，形式化验证，知识推理和群体智能。

葛　宁　北京航空航天大学，副教授，形式化方法、模型驱动工程，CCF 形式化方法专业委员会、CCF 软件工程专业委员会委员。

陆　平　北京航空航天大学，副教授，形式化方法、大数据、并行计算。

牟　明　中航机载共性技术工程中心，研究员，计算机技术，软件技术。

王　闯　中航机载共性技术工程中心，高级工程师，计算机技术，虚拟集成与仿真验证技术。

邱化强　北京神舟航天软件技术有限公司，工程师，嵌入式操作系统、装备嵌入式系统。

唐忆滨　武汉数字工程研究所，工程师，计算机系统结构、舰载嵌入式计算机技术。

戴新发　武汉数字工程研究所，研究员，舰载嵌入式计算机技术。