文章编号:1674-8190(2023)04-158-10

面向民用飞机的复杂航电系统软件研制与管理方法

尹伟,韩光辉,肖前远,缪万胜,康介祥
(中国航空无线电电子研究所,上海 200241)

摘 要: 现如今,民用飞机航电系统的功能日趋完备、结构日趋复杂、费用日趋昂贵、研制周期日趋紧张,亟需解决软件研制过程中的需求不准确、架构设计过于耦合、过程产物难以追踪等问题。基于对数字化研制、流程研制、持续集成、云开发环境等先进技术与方法的研究,提出面向复杂民用飞机航电系统软件研制问题的解决方法,即构建基于模型的需求开发方法、开放式航电软件架构、民用飞机软件研制过程体系,搭建基于Linked Data的集成开发环境和软件持续集成与验证环境,在航电系统软件研制过程中进行应用。结果表明:本文提出的面向民用飞机的复杂航电系统软件研制与管理方法是可行的,且该方法已在C919民用飞机显示系统研制过程中进行了应用,提升了软件研制效率和质量。

关键词: 航电系统软件;大规模软件;软件架构;软件工程开发环境
中图分类号: V247; TP311.5 **文献标识码:** A
DOI: 10.16615/j.cnki.1674-8190.2023.04.17

Software development and management methods for complex avionics systems for civil aircraft

YIN Wei, HAN Guanghui, XIAO Qianyuan, MIAO Wansheng, KANG Jiexiang
(China Aeronautical Radio Electronics Reasearch Institute, Shanghai 200241, China)

Abstract: The software development process encounters several challenges due to the increasingly functional, complex, expensive, and condensed development cycles of civil aircraft avionics systems, including inaccurate requirements, overly coupled architecture design, and difficult tracing of process outputs. Based on the research of advanced technologies and methods, such as digital development, process development, continuous integration, cloud development environments, a solution is proposed for software development issues in complex civil aircraft avionics systems. In this paper, it is demonstrated how the solution for civil aviation software development is used in the process of developing avionics systems software. The viability of the complex avionics system software development and management method for civil aircraft proposed are also demonstrated. To increase the effectiveness and caliber of software development, the method has been used in the development of the C919 civil aircraft display system.

Key words: avionics system software; large-scale software; software architecture; software engineering development environment

收稿日期: 2022-08-10; **修回日期:** 2023-02-02
基金项目: 工信部民机预研项目(MJZ2-3N21)
通信作者: 尹伟, yinw008@avic.com
引用格式: 尹伟, 韩光辉, 肖前远, 等. 面向民用飞机的复杂航电系统软件研制与管理方法[J]. 航空工程进展, 2023, 14(4): 158-167.
YIN Wei, HAN Guanghui, XIAO Qianyuan, et al. Software development and management methods for complex avionics systems for civil aircraft[J]. Advances in Aeronautical Science and Engineering, 2023, 14(4): 158-167. (in Chinese)

0 引　言

2005年,美国卡内基梅隆大学与美军方提出有关未来的超大规模系统软件的问题,如去中心化、弹性需求、异质平台的大规模互联、庞大的无线和有线设备网络、计算元素、自动化设备和人类在网络物理、社会技术生态系统中共存等关注点,开展了一系列相关领域的技术研究,包括内容感知移动计算、网络物理系统、社会技术生态系统、泛在计算、自适应计算和战术或"边缘"计算等[1]。面向航空领域,航电系统逐渐成为大规模、复杂的系统,其研制及管理方式发生了巨大变化。航电系统是一个由多个系统、多种环境、多项任务、多种资源构成的相互关联、相互支持、相互集成和相互制约的复杂系统[2],具有多目标、多信息、多专业、多任务、多功能、多资源和多过程组成的复杂系统构成与管理特征[3]。随着计算能力和网络等硬件技术的发展,综合到航电系统的功能越来越多,导致航电系统的软件规模和复杂程度持续增加,由此带来了系统的开发、验证、优化及维护的巨大挑战[4]。

软件是现代数字电子设备的"心脏",它使系统的灵活性远远超过模拟系统。软件可以表述算法、逻辑语句、数据和控制流程,其投资是巨大的,A320的航电系统大约有80万行代码,B777的航电系统有超过400万行代码,运行在超过50个硬件平台上[5]。面向如此规模的复杂航电软件,由美国海军航空系统司令部发起并组织联合波音等多家公司提出了未来机载能力环境(Future Airborne Capability Environment,简称FACE)[6]项目,其目的是通过一个通用的软件运行环境,在不同的有人和无人飞行器平台上实现软硬件的互操作,目前其标准已发展到3.1版本。同时,在欧洲也提出了类似的软件架构,如英国和法国共同资助的面向组件的架构(ECOA)[7],目的是促进软件的可移植性和重复使用,以减少复杂的实时飞机软件系统的生产、修改成本和时间。目前FACE已用于民用飞机项目中,如Integrated-178。

在软件研制方面,IBM为了解决大规模的问题,提出Harmony SE/ESW方法[8],并通过Rational系列工具环境进行开发;面向工业环境,西门子公司建立Polarion ALM方法论,串联大量开发工具,并与管理业务相结合;泰雷兹航电公司提出ARCADIA[9]方法,基于模型工程方法,构建满足工程系统、硬件和软件架构设计环境;欧洲航电公司提出ASDE环境,引入具备形式化技术的SCADE等工具,期望成为欧洲嵌入式系统和软件开发的标准;Honeywell公司采用的HiLiTe(Honeywell Integrated Lifecycle Tools & Environment)[10]实现了软件开发到测试验证的自动化,并且实现高安全需求(如DO-178B)下达成目标所需的过程。对于国外航电公司,他们拥有大量相关技术的知识产权,且仅在其公司内部使用,国内一直处于跟随学习的状态。面向航空类软件工程,DO-178B/C为机载设备软件研制提供了指导,使其在符合适航要求安全性水平下,实现预期的功能,但DO-178B/C主要关注产品的目标符合性,而对于软件组织如何开展项目的组织和管理却没有相关描述。

我国的C919航电显示系统代码已超过了300万行,具备大规模复杂系统软件特征。相对于应用系统、互联网系统的软件生产力,是航电系统软件的15倍[11],可见航电软件研制难度非常大;同时面对机载设备安全适航,航电系统软件的工程化要求更高,质量要求苛刻。在基于DO-178C的软件质量保证要求下,提出了通过准时交付率、合格率和客户满意度的质量目标,降低软件问题且提高变更请求的效率[12]。相比国外而言,国内航空行业研制技术多借鉴互联网行业的方式和方法,缺少相关内容研制的方法。

大飞机显示系统软件是一种具备大规模复杂特征[13]的软件,研制过程中参考了一些方法,用于降低系统集成带来的涌现性,并分解复杂功能目标,降低系统的综合复杂度。本文从复杂航电软件研制若干问题入手,对我国研制复杂系统软件的方式开展讨论,吸取先进的技术与方法,提出适合于复杂民用飞机航电系统软件研制问题的解决途径和方法,并针对构建这样的复杂系统软件若干问题,深入探讨解决方案。

1 复杂航电系统软件研制若干问题

现有系统的复杂性很难用单一解决方案解决,如采用SysML、UML试图解决复杂问题并不完全有效,当建立了愈加复杂的用例图、时序图、活动图等,却依旧解决不了复杂系统交联产生的指数级涌现性。

分解的过程[14]必然会造成系统整体性与内部交互性的流失,即使在后期进行了完整的集成,也无法保证流失的整体性和内部交互性得到彻底还原。

复杂软件系统面对现有软件开发和维护技术,存在构造难以设计、开发和部署的特点,运行状态下其固有内在冲突和各类交互始终存在大量不确定性[15]。

大量的信息交互使系统及软件工程的规模越来越大[16],繁多的工具链体系令项目团队不堪重负。面向大规模复杂航电系统需要解决以下问题:

1) 如何正确分析需求、完成需求编制?

2) 复杂系统的软件架构设计如何开展?

3) 如何追踪中间过程产物?

4) 数据与人员、团队如何一致?

1.1 正确描述需求

开发大规模系统需要满足各种利益相关方广泛的需求,其变化是必然的,在需求不断演化和变更的过程中,容易发生矛盾和不一致。大部分需求由自然语言描述,虽然自然语言并不是表达需求的完美方式,但其依然是目前唯一的能够涵盖各种所需求概念的通用表达方式。想要明确、准确、避免歧义地描述需求并不容易,即便使用图形建模方法(SysML/UML/MARTE等)替代书面表达方式,不同人的理解也不尽相同,最终仍需要文本化的需求。

为完成更复杂的飞机功能,机载软件规模猛增,与之对应的需求条目数也急剧增加。以某个具有100万行代码规模的飞行显示器软件为例,其需求条目数达到了33 000条,按照每页10条需求计算,高层需求将近3 300页[17]。如此大规模的需求数目,要做到每条需求均定义完整、清晰、可行、可验证,难度较高。

造成需求歧义[18]的原因主要有描述不充分、语义二义性、指代二义性、语法二义性、对术语理解不一致等类型。IEEE830—1998是较早提出软件需求规格编写规程的标准之一,分析了好的需求应具备以下要求:正确、明确、完整、一致、重要性/稳定性排序、可验证性、变更分析、可追踪。

面向复杂系统,可以使用形式化方法,如需求的四变量模型、需求状态机语言(RSML)实现具备工程化的需求描述方式,能够从初始的、非正式的系统描述到详细的、正式的需求规约过渡。

1.2 领域统一架构

航空电子系统的高度综合化直接导致软件规模的成倍增加,软件层次、接口、交联关系复杂,控制点分散,同时为满足民用飞机适航要求,增加多等级、多分区软件系统的功能,需要降低软件耦合性,重用软件模块,提升软件的强交互能力,因此面对大规模复杂系统的软件设计需要具有领域驱动架构设计。领域驱动设计[19]是由Eric Evans最早提出的综合软件系统分析和设计的面向对象建模方法,如今已经发展为一种针对大型复杂系统的领域建模与分析方法。区别于4+1视图方法,将要解决的业务概念和业务规则转换为软件系统中的类型以及类型的属性与行为,通过合理运用面向对象的封装、继承、多态等设计要素[20],降低或隐藏整个系统的业务复杂性,并使得系统具有更好的扩展性,应对纷繁多变的现实业务问题[21]。领域驱动设计是一个渐进的设计过程,它需要对知识和设计的所有方面进行不断迭代。

1.3 追踪过程产物

DO-178C对A级软件的系统需求与软件高层需求、软件高层需求与软件低层需求、需求与测试用例、测试用例与测试规程、测试规程与测试结果之间的双向可追踪性提出了很高的要求;实际上还需要包括软件低层需求与代码、代码与二进制目标码之间的可追踪性。然而由于系统之间存在大量交织的复杂逻辑和交联,在实际项目研制过程中很难达成。

为了能在时间节点前完成项目研制,航电软件项目也采用敏捷开发[22]的方法,但适航证据的提交还得依托具体完整的生命周期数据,因此建立追踪性是必要的。

追踪性本身是非常重要的,尤其对于系统安全性而言,研制过程数据的可信性、正确性,变更影响分析和维护等都至关重要。随着民用航电机载设备安全关键特性及航电系统的综合化程度不断提高,软件工程过程数据间追溯关系的复杂性也大幅增加,给适航安全性审定工作带来较大挑

战。目前大多数工具仅实现了单一过程之间的追踪,其他的都需要人工来完成,需要采用具有关联的信息系统[23]加以实现,如知识图谱、自然语言处理等。

1.4　软件适航过程

虽然在航空设备软件的研制方面国内具有一定的基础和经验,也建立了一些基本研制流程和技术规范,大量航空企业还通过了GJB 5000A/B的三级审查,但是这些企业对民用飞机适航目标、流程和提供的数据仍不是很明确,未健全满足适航的整个软件开发周期,也未覆盖软件开发全生命周期,特别是未能达到适航要求软件过程的机载A级软件研制能力的需求。从软件本体数据考虑,团队、人员与工程产品之间,包含了任务、需求、模型、代码以及文档等相互关联关系,这些关系依赖于过程管理和控制能力。随着软件规模提升,复杂度和重要性也相应增加,对飞机的安全性影响也来越高,并且还不断引入先进的、前沿的技术,使软件研制的适航审定通过的难度越来越高。软件工程本体示意图如图1所示。

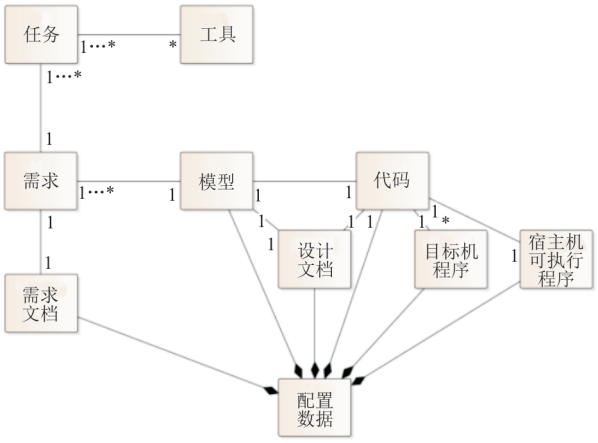

图1　软件工程本体示意图

Fig. 1　Software engineering ontology diagram

2　复杂航电系统软件问题解决途径

2.1　模型开发方法

2.1.1　基于模型的需求开发方法

在一个大型项目中,通常先通过对需求的建模来了解系统运行的边界和工作模式;再通过对架构的建模来分解系统的功能,并对功能进行解耦,同时找出各个功能之间的接口与交互关系;最后用设计模型对架构中分解好的功能进行实现,从而达到基于全生命周期过程的软件建模开发。

基于以往需求特征和项目经验,形成需求规约,开展需求建模分析,通过动态运行验证需求的合理性、正确性,证明方法合理、可行。对每个抽象级别、层级需要关注特定的系统部分、细节,从而降低需求规约的复杂度。

航电系统的需求建模是在IBM Rhapsody工具中进行的,实现基于SysML的用例图和块图,并满足需求规约;需求管理基于IBM DOORS工具,统一数据源,同时辅以在DOORS中实施版本管理,改进了以往同样的需求文档存储于众多单机且存在众多版本,导致无法确定最新状态的情况。

2.1.2　基于模型的航电显示软件开发

根据基于模型的方法,结合需求建模,基于模型的航电显示软件开发流程包括四项活动:

1) 设计建模:根据航电软件的需求,在软件架构的指导下,对每个构件进行详细设计,包括模块划分、模块间的协同、数据流与控制流、数据结构等。显示系统采用逻辑建模和控制建模结合的方式进行设计,项目中使用SCADE工具完成。

2) 设计验证:对设计模型或设计文档进行分析、检查、评审、模拟仿真、模型覆盖分析等,根据软件需求验证设计模型或文档的准确性、一致性、可行性。

3) 代码生成:在设计建模工具的支持下,根据设计模型自动生成目标代码,用于软件测试和实现。

4) 设计管理:使用配置管理工具,对软件设计模型、软件设计文档、软件设计数据等进行版本管理与基线管理;使用DOORS等工具,与软件需求、软件架构、白盒测试用例建立跟踪关系。

2.2　开放式软件架构技术

随着系统越来越大,越来越复杂,对于软件架构的新要求[24]也随之出现,依托多种软件元素特征实现系统功能[25]。在航空电子领域,通过对目前主流的ARINC653、ASAAC、GOA及FACE等开放式航电软件架构标准进行研究,结合我国民用飞机大规模多团队复杂系统软件开发的实际需要,融入面向服务的思想,通过关注点分离技术,形成面向民用飞机航电系统的大规模复杂系统软件架构,如图2所示。

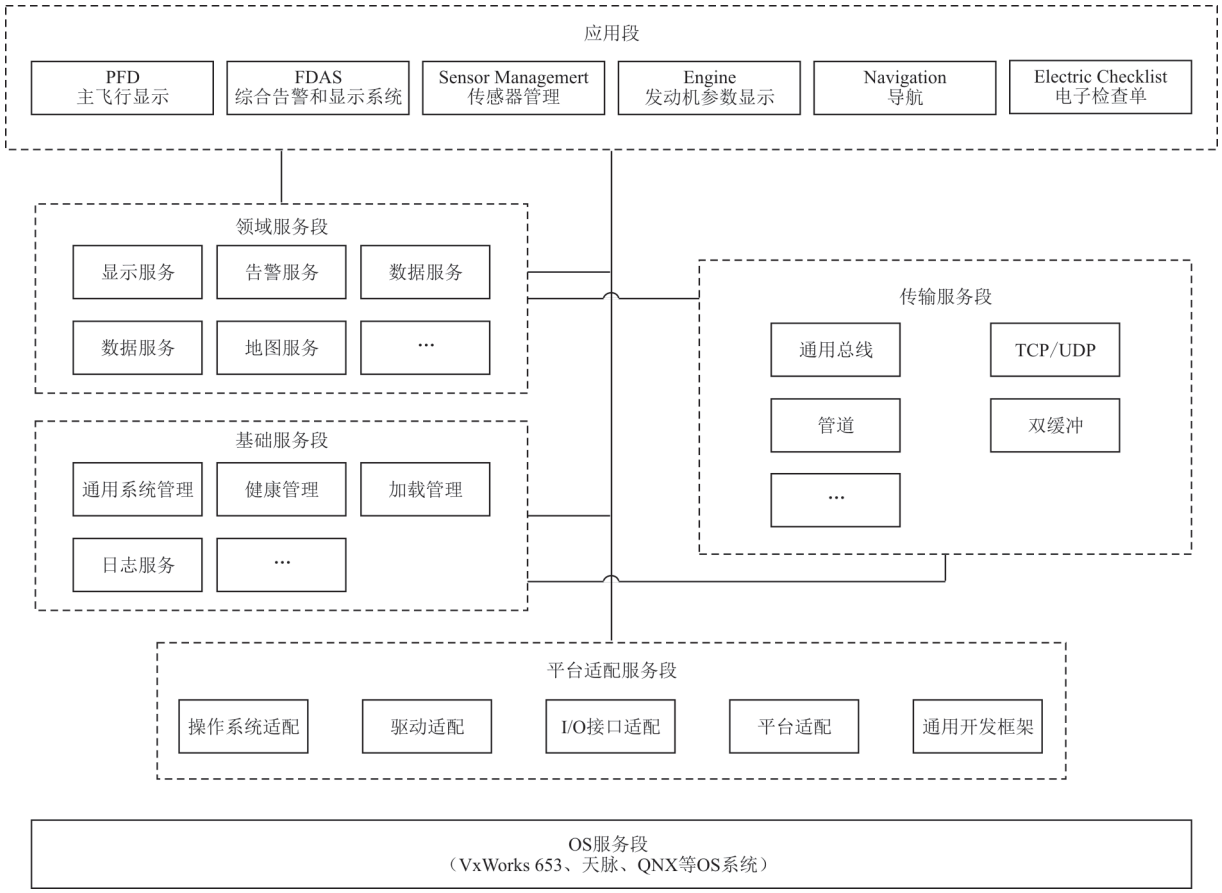

图 2 民用飞机显示系统架构

Fig. 2 Civil aircraft display system architecture

基于开放式的软件架构分段方式明确软件架构中各核心元素的概念、范围和功能需求；给出软件架构中各核心元素间的接口形式、数据类型和交互方式。这样的软件架构为软件开发的组件、相互交联，提供了开发指导，提高软件开发效率，并能为适航审查过程提供支持。

2.3 持续集成与验证技术

2.3.1 航电软件的持续集成

在早期型号研制过程中，软件的调试与功能验证都需要等待真实硬件生产出来以后才可以进行，同时软件开发人员对不同机型的系统要重新编码、调试和测试。特别是在系统集成阶段，每个机型都有不同的集成验证环境，需要在不同的地点，有时由于项目进度紧急，甚至直接去真实飞机上集成和验证，这样既无法保证软件质量，又会在后期造成系统综合的困难，时间并未得到节省。

软件集成人员在项目初期架构制定完成后就开始着手持续集成平台的搭建，自动构建脚本，自动测试脚本在开始集成前均已准备完成，代码改动均可以通过持续集成平台看到集成结果，并进行验证，定期频繁地进行集成，每次集成结果均向全部项目成员发布。统一的平台和各型号软件之间的延续性极大地缩减了开发周期和成本。

持续集成在机载航电领域的应用难度在于，需要配置不同的编译场景，针对不同的测试环境还需要进一步配置测试目标。

2.3.2 软件持续集成与验证环境

持续集成的形式是频繁进行构建，确保代码正常。通过自动化的脚本对目标环境的软件测试[26]，并且持续开展，其本质是尽快地暴露问题。因为问题暴露得越早，越容易修复。

因此在显示系统研制过程中，应用持续集成与验证技术中的技术要点主要包括：自动构建、多目标环境的自动部署、自动测试。持续集成环境及过程如图 3 所示。

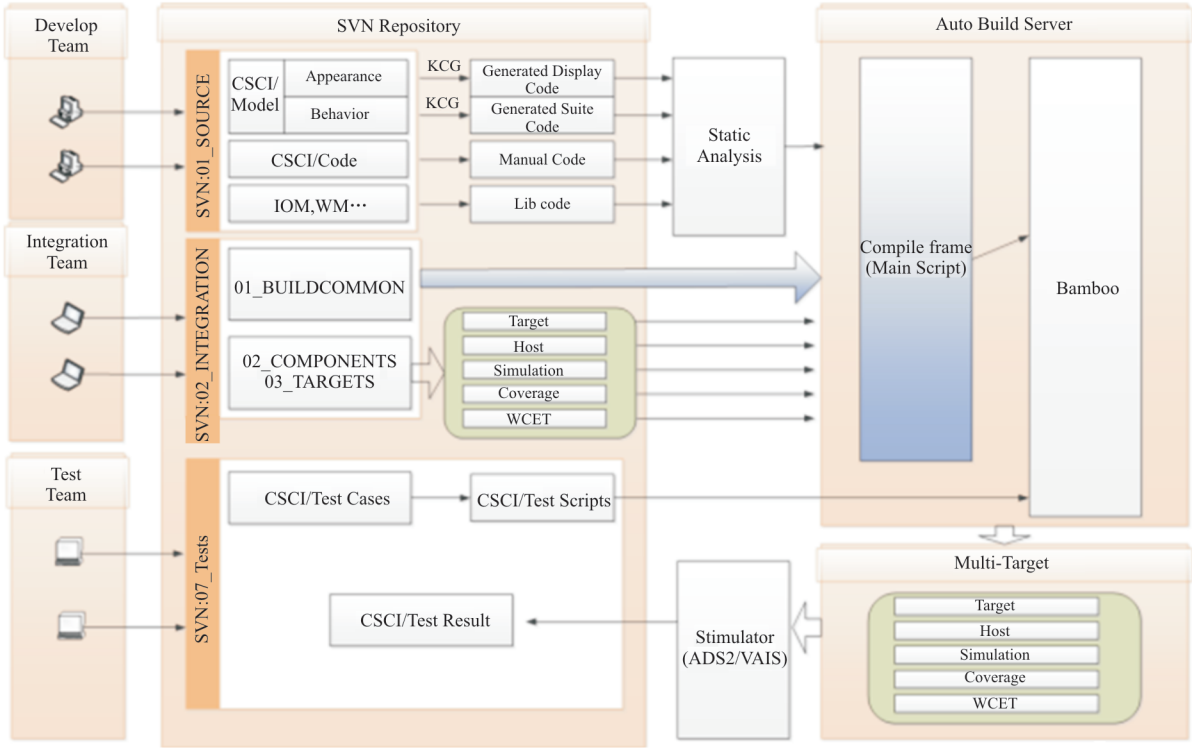

图 3　持续集成环境及过程

Fig. 3　Continuous integration environment and processes

通过整合各环节工具链,以触发器、钩子函数、脚本等方式将各个工具间的接口打通,将以下各步骤串联:

1) 规则检查:代码提交到SVN库时根据项目编码规则进行静态分析、通过静态分析可提交代码,未通过,需要修改代码后再进行提交;

2) 自动构建:持续集成服务器可定时检测到代码是否变更,当发现变更时,执行代码编译过程;

3) 自动部署:通过自动构建的目标码可部署到目标机、宿主机、包含最坏运行时间测试逻辑的目标机、包含覆盖率分析测试逻辑的目标机;

4) 自动化测试:持续集成服务器可根据配置好的测试用例和带有插桩的测试代码执行自动化测试。

通过上述方法在民用飞机航电显示系统软件开发与验证过程的应用,提升了项目研制进度,降低了研制复杂度,为系统可重复构建提供了技术基础和环境基础。

2.4　软件工程过程及研发环境

2.4.1　面向民用航电显示软件的研制过程

民用飞机软件的研制过程需满足适航要求,过程控制的根本目的是保证项目研制过程符合适航要求DO-178B/C及其附件。从研制单位的角度,DO-178B/C只说明了做什么,但没有说明怎么做,无法直接指导机载软件研制工作。DO-178B/C只给出了各生命过程应满足的目标,但并没有给出具体的实施方法(工作方法、输入、输出等)。DO-178B/C给出了生命周期过程,但没有给出各生命周期过程之间的相互关系、数据流向等信息。因此,机载软件项目满足DO-178B/C要求的前提是建立一套研制过程控制方法(机载软件研制流程)。该方法以DO-178B/C目标要求为基础,结合实际项目的具体情况,明确定义生命周期过程,给出各过程的关系,包括输入、输出、工作方法、转换准则、人员角色等关键信息。研制单位按照该流程开展机载软件研制工作,能够针对DO-178B/C的具体目标要求,产生相应证据,从而有效支持其满足DO-178C目标要求。

在实际项目中，可以使用模型来描述部分或全部软件需求，也可使用模型来描述部分或全部设计，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设计（包括软件架构和低级别需求），再通过专用的工具（如SCADE）直接生成源代码，这是基于模型开发常见的应用场景之一。

以基于模型的开发为背景，完成计划过程、需求过程、设计过程、编码过程、测试准备过程、测试与总结过程、构型管理过程、质量保证过程、合格审定准备及联络过程，如图4所示。

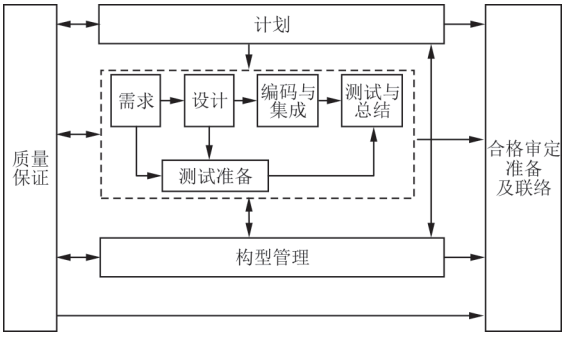

图4 流程相关性

Fig. 4 Process relevance

2.4.2 基于Linked Data的集成开发环境

大规模复杂系统软件开发工具链中存在大量不同种的工具，如建模、编码、调试、测试等，这些工具对于工程人员来说是沉重的负担，因为这些工具软件也是软件系统，将产生和收集大量的数据[27]。然而对于航空电子系统软件，其嵌入式软件的开发受资源限制，需要具备协同的开发体系结构[28]，这样的复杂系统软件分布式协同开发，使得研发过程能够高效、协同、有序的开展[29]。能够对系统开发过程中数据统一管理，实现设计、分析、实验数据的谱系关联，基于知识、流程的统一开发，完整地展现出整个研发历程。

基于Linked Data[30]建立的软件开发环境，支持航空电子软件的多个开发阶段，集成需求工具、设计工具、实现工具、仿真工具、测试工具、版本控制工具等，使开发过程标准化。通过集成开发平台的链接数据，团队可以共享信息和知识，并进行有效的协作。未来，在积累了一定数量的数据之后，可以对关联的数据进行数据分析和挖掘。数据的追踪与关联工具示意图如图5所示。

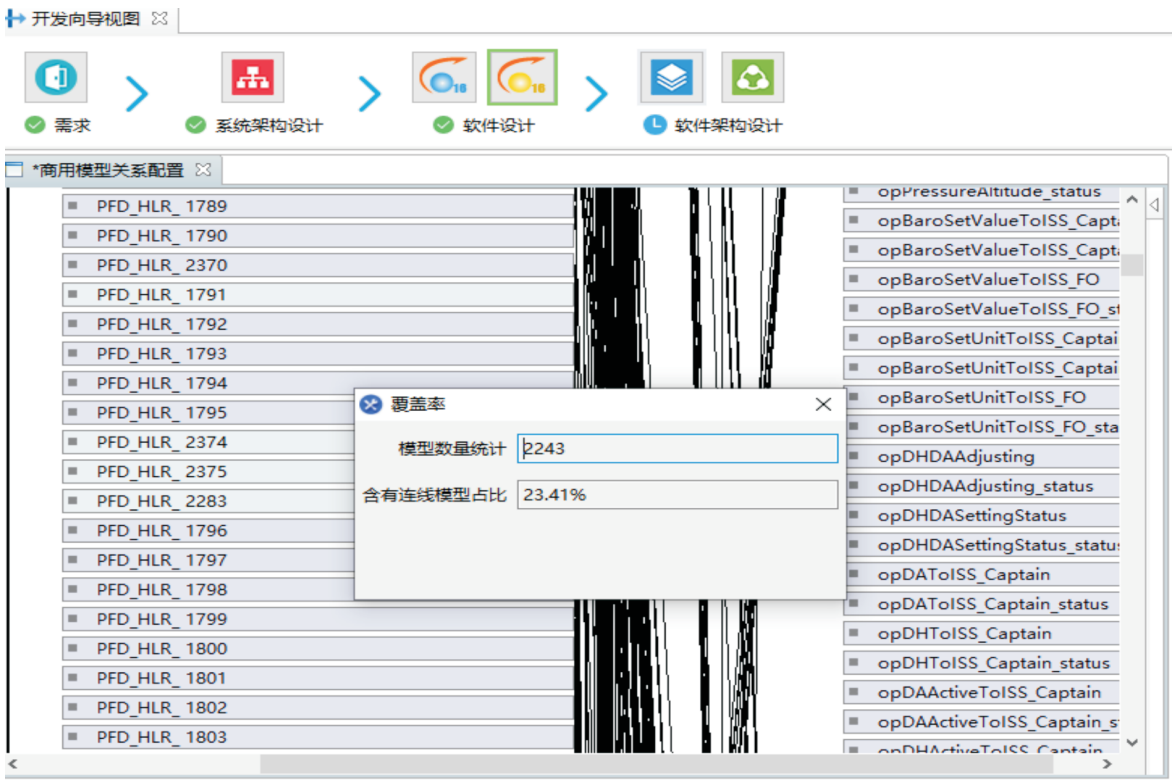

图5 数据的追踪与关联工具示意图

Fig. 5 The diagram of data tracking and correlation tool

HLR(高级需求),LLR(低级需求)、软件架构、源代码建立关联关系,如图 6 所示,用以满足 DO-178C 中表 A-3(6)、表 A-4(6)和表 A-5(5)对可追溯性的要求。

根据民用飞机航电显示软件的研制过程,定制项目的生命周期、工具环境、配置管理环境以及项目目录等,将所使用的工具包括 DOORS、Rhapsody、SCADE 等集成到一起,配置好后按照研制过程让软件设计师在统一平台上开展软件开发。软件集成开发环境如图 7 所示。

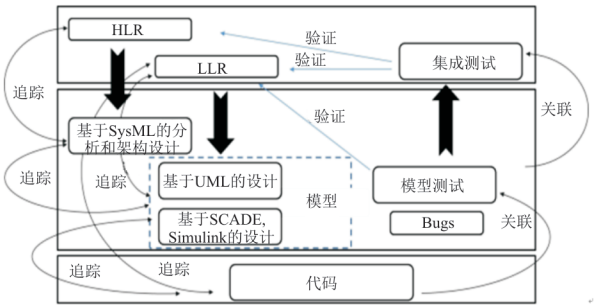

图 6　数据的追踪与过程之间的关联

Fig. 6　The link between data tracking and process

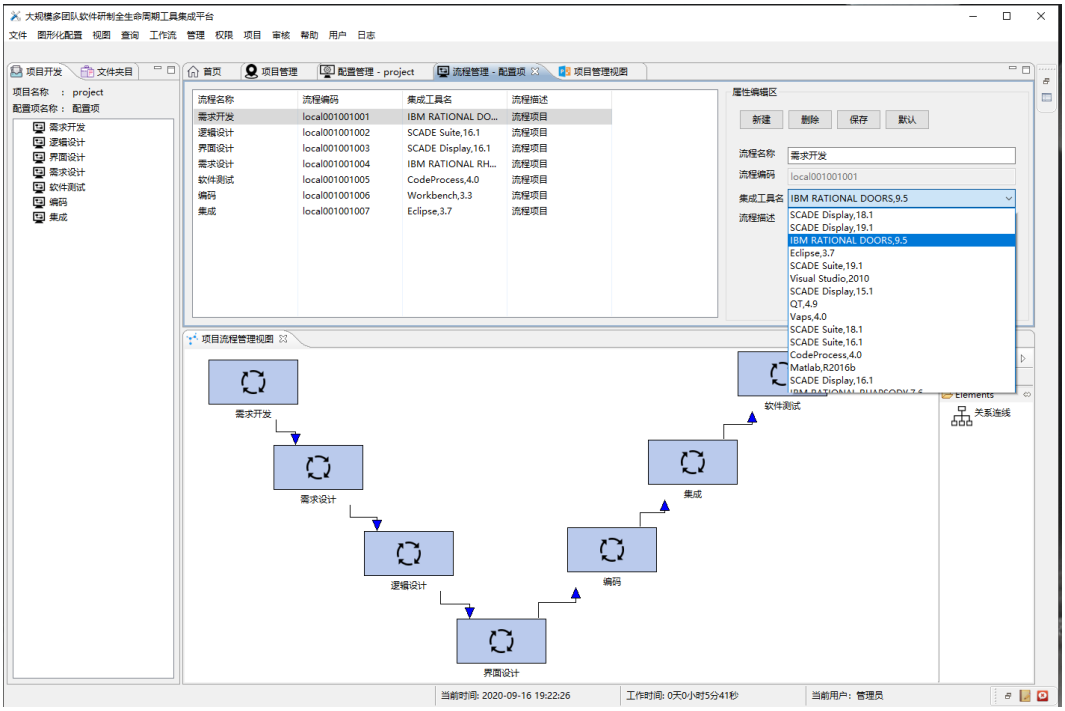

图 7　民用飞机软件集成开发环境

Fig. 7　Civil-plane software integrated development environment

3　应用结果

本文提出的面向复杂民用飞机航电系统软件研制的方法已在 C919 飞机航电显示软件开发和验证过程中开展应用,提高了软件研制效率,降低了软件出错率,并且支持适航审定过程,应用效果如表 1 所示。

表 1　显示系统软件开发的方法应用效果

Table 1　The results of the methods applying for DS system software development

使用方法或者工具	应用方式	获得的效益
SCADE with Code Generator	60% 软件代码自动生成	开发周期缩短 50%,软件错误率降低 5 倍
开放式软件架构	通过 IO 接口隔离软硬件数据传输	开发效率提升 2 倍
持续集成框架和工具链	集成、发布和自动部署 100% 采用持续集成平台	开发效率提升 3 倍,集成、发布和部署无人为失误

4 结 论

1）通过基于模型的需求开发方法和基于模型的软件开发方法，采用DOORS进行需求管理，SCADE Display开发显示软件、SCADE Suite开发逻辑控制软件，改进传统文本性的需求二义性问题，能够较完整地传递功能需求和设计思考。

2）开放式软件架构的构建可为设计思想统一提供参考，该思想已在航电、飞控系统的软件研制中大量应用，并对显示系统交联数据的逻辑设计起指导作用。

3）面向大型客机机载嵌入式软件研制的生命周期和过程，将软件研制的开发、集成、验证等过程纳入持续集成框架下综合考虑，目前研发环境基于持续集成环境，构建了一系列工具链，采用软件持续集成与验证提高了研制效率，加快软件集成速度，并可在交付前提供测试结果。

4）过程体系建设是积累和收集以往开发经验的一个具体方式，结合机载软件开发工具，建立基于Linked Data的集成开发环境，形成软件配置项目中需求—模型—代码的追溯关系，有利于过程数据的积累和适航证据的准备。

本文针对软件研制过程提出的方法适用于大规模复杂民用飞机航电系统软件的研制，采用该方法可使项目研制过程更规范、更高效。然而，本文研究内容范围广泛，涉及到软件工程较多方面，却又是对于航电系统软件开发较为重要的方面，为了解决大规模复杂航电系统软件研制的问题，还需要进一步在实践中检验。如采用形式化的语言实现正确、一致的需求，基于定理证明器、模型检测器等方法及工具减少安全关键软件构建过程中的问题。还可以通过自动化脚本，降低集成与验证过程中人为因素的影响，使复杂系统数据有迹可循，提升软件质量。

参 考 文 献

[1] SEIA. Solar market insight report 2017 year in review[R]. USA: SEIA, 2017.

[2] COLLINSON R P G. Introduction to avionic s systems [M]. 3rd ed. Berlin: Springer, 2011.

[3] HALLE M, THIELECKE F. Tool chain for avionics design, development, integration and test[C]// 1st Workshop on Avionics Systems and Software Engineering. Stuttgart, Germany: [s.n.], 2019: 1-10.

[4] 高翔，李辰. 复杂航电架构的开放式系统标准研究[J]. 航空电子技术，2015，46(2)：26-31，41.
GAO Xiang, LI Chen. Research on the open system standard of complex avionics architecture[J]. Avionics Technology, 2015, 46(2): 26-31, 41. (in Chinese)

[5] ATKEARNE Y. Software: the brains behind U.S. defense systems[R]. USA: Computer Science, 2013.

[6] FACE Consortium. FACE™ technical standard: edition 3.1 [S]. USA: FACE Consortium, 2020.

[7] BAE Systems (Operations) Limited, Dassault Aviation. European component oriented architecture (ECOA) collaboration programme: architecture specification: IAWG-ECOA-TR-007[S]. UK: BAE, 2018.

[8] MCGEE T, MCGEE B. Agile systems engineering[M/OL]. [2022-08-10]. https://www.researchgate.net/publication/300459131_Agile_Systems_Engineering.

[9] VOIRIN J L. Model-based system and architecture engineering with the arcadia method[M/OL]. [2022-08-10]. https://www.researchgate.net/publication/328132426_Model-based_system_and_architecture_engineering_with_the_arcadia_method.

[10] BHATT D, MADL G, OGLESBY D, et al. Towards scalable verification of commercial avionics software[C]// AIAA Infotech@Aerospace Conference and Exhibit. Atlanta, Georgia: AIAA, 2010: 1-8.

[11] REIFER D J. Industry software cost, quality and productivity benchmarks[R/OL]. [2022-08-10]. https://www.researchgate.net/publication/249815530_Industry_Software_Cost_Quality_and_Productivity_Benchmarks.

[12] 周培. 基于DO-178C的机载软件质量保证与管理[J]. 航空工程进展，2021，12(6)：161-166.
ZHOU Pei. Airborne software quality assurance and management based on DO-178C[J]. Advances in Aeronautical Science and Engineering, 2021, 12(6): 161-166. (in Chinese)

[13] NORTHROP L, FEILER P H, GABRIEL R P, et al. Ultra-large-scale systems-the software challenge of the future [M]. USA: Carnegie Mellon University, 2006.

[14] HISSAM S, KLEIN M H, MORENO G, et al. Ultra-large-scale systems: socio-adaptive [R/OL]. [2022-08-10]. https://resources.sei.cmu.edu/asset_files/WhitePaper/2016_019_001_493871.pdf.

[15] 王怀民，吴文峻，毛新军，等. 复杂软件系统的成长性构造与适应性演化[J]. 中国科学：信息科学，2014，44(6)：743-761.
WANG Huaimin, WU Wenjun, MAO Xinjun, et al. Growing construction and adaptive evloution of complex software system[J]. Scientia Senica Informationis, 2014, 44(6):

743-761. (in Chinese)

[16] ISHIDA Y, ADACHI N, TOKUMARU H. A topological approach to failure diagnosis of large-scale systems [J]. IEEE Transactions on Systems Man & Cybernetics, 1985, 15(3): 327-333.

[17] 陈福, 牟明, 戴小氐, 等. 机载大规模复杂软件开发及验证技术[J]. 电子科技, 2016, 29(3): 190-193.
CHEN Fu, MU Ming, DAI Xiaodi, et al. Development and verification of large-scale and complex airborne software [J]. Electronic Sci. & Tech., 2016, 29(3): 190-193. (in Chinese)

[18] LIPCAK J. ROSSI B. A large-scale study on source code reviewer recommendation[EB/OL]. (2018-06-20)[2022-08-10]. https://arxiv.org/abs/1806.07619.

[19] ERIC E. 领域驱动架构设计: 软件核心复杂性应对之道[M]. 北京: 人民邮电出版社, 2016.
ERIC E. Domain-driven design: tackling complexity in the heart of software [M]. Beijing: Posts & Telecom Press, 2016. (in Chinese)

[20] 宁小庚, 黄晓芳. 一种基于领域驱动设计划分微服务的方法[J]. 西南科技大学学报(自然科学版), 2019, 34(1): 80-85.
NING Xiaogeng, HUANG Xiaofang. Approach to divide microservices based on domain driven design[J]. Journal of Southwest University of Science and Technology, 2019, 34(1): 80-85. (in Chinese)

[21] VAUGHM V. 实现领域驱动设计[M]. 北京: 电子工业出版社, 2014.
VAUGHN V. Implementing domain-driven design [M]. Beijing: Publishing House of Electronics Industry, 2014. (in Chinese)

[22] LIN J. Human factors in agile software development [J]. Computer Science, 2015, 101: 115-123.

[23] SOMMERVILLE I, CLIFF D, CALINESCU R, et al. Large-scale complex IT systems [J]. Communications of the ACM, 2012, 55(7): 501-507.

[24] OSTADZADEH S. SHAMS F. Towards a software architecture maturity model for improving ultra-large-scale systems interoperability[EB/OL]. [2022-08-10]. https://arxiv.org/abs/1401.5752.

[25] ITKONEN J, FGRI T E, DINGSYR T. What is large in large-scale? a taxonomy of scale for agile software development [EB/OL]. [2022-08-10]. https://xueshu.baidu.com/usercenter/paper/show? paperid=1g320m10q83r00p0fq120ec0qr753127&site=xueshu_se.

[26] 李英玲, 王青. 持续集成测试用例集优化综述研究[J]. 软件学报, 2018, 29(10): 3021-3050.
LI Yingling, WANG Qing. Test set optimization in continuous integration: a systematic literature review[J]. Journal of Software, 2018, 29(10): 3021-3050. (in Chinese)

[27] MARCUS A. MENZIES T. Software is data too [C]// FSE/SDP 2010: Proceedings of the 2010 FSE/SDP Workshop on Future of Software Engineering Research. New York: ACM, 2010: 229-232.

[28] 陈小平, 谢彬, 李斌. 嵌入式软件协同开发支撑技术[J]. 计算机工程, 2007, 33(18): 90-92.
CHEN Xiaoping, XIE Bin, LI Bin. Supporting technology for embedded software co-development[J]. Computer Engineering, 2007, 33(18): 90-92. (in Chinese)

[29] 伍恒, 张卫民, 王靖. 软件的分布式协同开发环境[J]. 吉首大学学报(自然科学版), 2003, 24(1): 74-76.
WU Heng, ZHANG Weimin, WANG Jing. Distribution and cooperation environment of software development [J]. Journal of Jishou University (Natural Science Edition), 2003, 24(1): 74-76. (in Chinese)

[30] LABRA-GAYO J E, PRUD'HOMMEAUX E, SOLBRIG H, et al. Validating and describing linked data portals using shapes [EB/OL]. [2022-08-10]. https://arxiv.org/abs/1701.0892. 2017.

作者简介:

尹　伟(1979—),男,博士,高级工程师。主要研究方向:航空电子,控制科学与工程,软件工程,适航安全等。

韩光辉(1985—),男,硕士,高级工程师。主要研究方向:航空电子,计算机科学,软件工程等。

肖前远(1985—),男,硕士,高级工程师。主要研究方向:航空电子,软件测试等。

缪万胜(1972—),男,学士,研究员。主要研究方向:航空电子,软件工程等。

康介祥(1969—),男,学士,研究员。主要研究方向:航空电子,软件工程等。

(编辑:马文静)