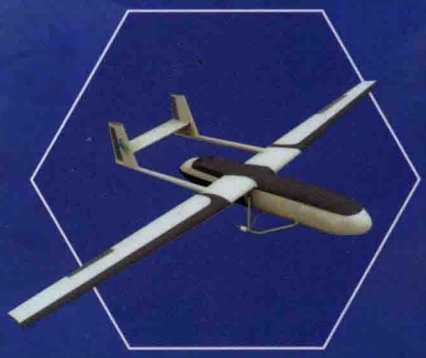
装备科技译著出版基金

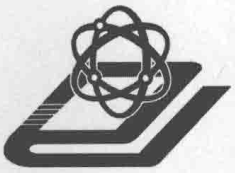
航天任务自主性需求工程

[爱尔兰] Emil Vassev　Mike Hinchey 著
崔晓峰 译

Autonomy Requirements Engineering for Space Missions



国防工业出版社
National Defense Industry Press



装备科技译著出版基金

航天任务自主性需求工程

Autonomy Requirements Engineering for Space Missions

[爱尔兰] Emil Vassev　Mike Hinchey 著
崔晓峰　译

国防工业出版社
·北京·

著作权合同登记　图字:军-2015-250号

图书在版编目(CIP)数据

航天任务自主性需求工程/(爱尔兰)埃米尔·瓦瑟夫(Emil Vassev),(爱尔兰)麦克·辛奇(Mike Hinchey)著;崔晓峰译. —北京:国防工业出版社,2017.12

书名原文:Autonomy Requirements Engineering for Space Missions

ISBN 978-7-118-11406-5

Ⅰ.①航… Ⅱ.①埃… ②麦… ③崔… Ⅲ.①航天系统工程-软件需求 Ⅳ.①V57②TP311.52

中国版本图书馆CIP数据核字(2017)第312855号

※

国防工业出版社出版发行

(北京市海淀区紫竹院南路23号　邮政编码100048)

三河市众誉天成印务有限公司印刷

新华书店经售

*

开本710×1000　1/16　**印张**14½　**字数**268千字

2017年12月第1版第1次印刷　**印数**1—2000册　**定价**79.00元

(本书如有印装错误,我社负责调换)

国防书店:(010)88540777　　发行邮购:(010)88540776

发行传真:(010)88540755　　发行业务:(010)88540717

译者序

2013年12月14日，国人瞩目的“嫦娥”三号月球探测器如期降落在月球虹湾。探测器从15km高度开始，在十几分钟时间里，完全依靠自主控制完成了中国航天器的首次地外天体软着陆，其中两次悬停避障的精彩表现，尤其令人印象深刻。由于38万km地月距离带来的信号延迟，这个紧张复杂的过程容不得探测器把拍摄到的着陆环境传回地球，再等待地面分析判断之后发来指令。一切依靠自我感知与决策的“嫦娥”三号不负众望，成功展示了独立适应环境和完成任务的高超“技艺”。

以“嫦娥”三号为例，我们其实已经越来越多地看到，在航天任务尤其是深空探测中，航天器的自主能力正成为任务的核心要素之一。这是因为相比地面控制，自主控制具有一系列重要优势：可以不受天地大时延的限制，实现实时反应、实时控制；可以不需要把大量的感知信息传回地面，节省宝贵的天地通信带宽；可以简化飞控决策实施程序，更高效地完成更复杂的探测活动；等等。但是进步就意味着挑战，自主控制的设计实现绝非易事，系统化的工程过程更是新的课题：一个航天器应当具备什么自主能力，怎样将这种能力从设想变为现实，怎样验证这种能力与预期完全相符，等等。

与其他任何工程问题一样，自主性工程的首要问题是需求问题。需求工程的研究关注需求的导出、表达、交流、确认等，目的是通过系统化的方法，便捷、高效、准确地建立工程的需求，从而为后续的设计与开发提供根本依据和有效驱动。自主性需求是航天器诸多功能和非功能需求中的一种，同时又是非常特殊的一种，如果能够有一种专门针对该类需求的系统化方法，必然能对自主航天器的研发起到巨大推动作用。

本书正是致力于这种专门针对航天器自主性需求的系统化工程方法，即“空间任务的自主性需求工程方法”。书中定义和描述了一种“自主性需求工程”(ARE)方法。ARE的目的是通过提供一个用于导出和表达自主性需求的机制和方法，在无人的航天任务中集成和提升自主性。ARE依赖于“面向目标的需求工程”(GORE)方法来导出和定义系统目标，并借助建立“一般自主性需求”(GAR)模型来导出和定义辅助性的以及最终可选性的目标，此外还使用一

种知识表示语言(KnowLang)对自主性需求进行规约。书中详细讨论了一个基于欧洲航天局(ESA)的 BepiColomo 任务的概念证明实例,表明了 ARE 的处理自主性需求的能力。此外,本书还讨论了自主性需求工程的形式化验证等问题和方法。

本书作者埃米尔·瓦塞(Emil Vassev)博士,是一位计算机科学家,目前是爱尔兰软件工程研究中心(Lero)的高级研究学者。他长期从事自主性需求工程、自感知和自适应系统的知识表示等研究,领导了一系列与美国国家航空航天局(NASA)、ESA 合作的研究项目,发表了一系列研究成果。本书就是其研究成果自主系统规约语言(ASSL)、自适应系统知识表示与推理框架(KnowLang)等的系统化阐述。

自主航天任务前景美好,但是道路还很漫长。自主计算的研究还处在起步阶段,这既是本书开创性的价值所在,也意味着还有更多的方法有待探索、发现、应用,直至成功推进这个领域更加成熟和进步。

由于译者的知识不足和时间仓促,翻译中难免有不当或谬误之处,敬请读者原谅和指正。

译者

2017 年 4 月

前　言

近年来，欧洲航天局（ESA）和美国国家航空航天局（NASA）的航天任务都逐步在飞行器和地面系统中引入自主性，从而增加任务所能获得的科学数据、开展新的科学实验，以及降低任务成本。在新的空间探测任务中，真人和机器人的探测都得到重视。即使探测活动中可以有人参与，也必须在任务的定义和设计中，针对人工照料空间设施的收益、成本、风险以及可行性进行仔细评估。风险和可行性是驱动使用无人飞行器，以及在可能的地方使用自动化和机器人技术的主要因素。

无人空间探索任务的发展，与机器人飞行器中自主性的集成和普及密切相关。ESA 和 NASA 现在都在采用自主计算作为开发自主性飞行器系统的一个有益范型，但是在解决自主性问题的过程中，采用的还是传统的开发方法。经验已经表明，传统的软件开发方法不适用于这些任务，因为它对自主性本身关注的很少。因此，应当采用的是新的、针对自主性的软件开发方法。

目前，自主系统的需求工程呈现为一个宽阔的开放研究领域，确定的解决方案尚不存在。自主性需求的导出和表达，是自主飞行器工程师当前需要解决的最大挑战之一。本书给出自主性需求工程（ARE）方法，目的是帮助软件工程师正确导出、表达、验证和确认自主性需求。ARE 是爱尔兰软件工程研究中心（Lero）与 ESA 的欧洲空间研究与技术中心（ESTEC）的一个联合项目的成果。

埃米尔·瓦塞（Emil Vassev）
麦克·欣奇（Mike Hinchey）
爱尔兰利默里克（Limerick）

目　录

第1章　航空航天软件工程现状

摘要:本章讨论航空航天领域软件工程的前沿现状。应用于航空航天的软件工程要想取得成功,必须认识到航空航天系统需要满足众多标准和很高安全性需求的特点。因此,航空航天系统的开发强调验证、确认、审定和测试。本章讨论软件开发的复杂性,以及领先的航空航天组织——例如美国国家航空航天局(NASA)、欧洲航天局(ESA)、波音、洛克希德·马丁——当前使用的软件工程过程。这些组织中的软件开发项目使用了基于螺旋的方法,重点在于验证。本章还讨论方法、技术和体系结构。当前有一种新的自主航空航天系统(如无人机和机器人空间探索系统)正在出现,它们融合了诸如集成健康管理、自监视和器上决策等特征。对于自主航空航天系统,合适的、专门的软件工程方法的缺少,是产生许多与需求、建模和实现相关的内在问题的原因。用于自主系统的需求工程呈现为一个宽阔的开放研究领域,目前还只有很少量的方法提出。

1.1　引言:航空航天工业特点

航空航天工业可以说具有极其复杂的特点,其中诸如波音和空客这样的制造商,以及ESA和NASA,它们的航空航天项目产生出生命周期很长的产品(长达34年)。当前,航空航天工业的最终产品或服务来自于极大量企业的合作,飞行器的不同部件的设计和生产都是通过外包或合作的方式完成。例如,波音787"梦幻客机"的70%的设计和制造都被外包。因此,航空航天工业的特点要求不同大小的公司之间的紧密协作,导致了不同层次的依赖性。同时,产品的多样性经常带来计划之外的研发合作、变化的共识/冲突,以及设计和开发过程的大量迭代。因此,诸如波音或空客这样的制造商对其战略供应商的能力和效能具有很大的依赖性。

1.1.1　注重安全性

航空航天工业(如NASA和ESA)遵循"为风险的最小化而进行设计"的策

略,即强调安全性(safety)原则。这对于确保充分级别的安全性得到正确的规约①、设计和实现,是很有必要的。以下是在许多工业中常见的基本安全性原则和方法[49],它们被用于使事故的可能性最小化,以及在一旦发生一个事故时减小其不良后果:

(1) 危险排除和限制;

(2) 屏障和互锁;

(3) 失效安全(fail-safe)设计;

(4) 失效风险最小化;

(5) 监视、恢复和逃逸。

这些原则和方法是一个综合的安全性设计中的基本要素,经常要将它们组合使用才能达到预期目的。

1.1.2 标准化

标准化对于航空航天业务的各个方面都是基础性的,它们是"互操作和互联的工具,保证可靠性、可重复性以及质量的公共需求,安全与审定的基础,以及传递变更的最有力机制之一"[76]。标准形成了在全球设计、建造以及支持航空产品时所使用的技术资料的一个最大来源。过于复杂、经常重复的航天标准体系,则又导致为了协调这些基于竞争的标准的规章性需求而增加的代价、由于多个标准而进行的多个符合性评估和质量管理而增加的代价、以及由于冗余和交叠的标准和标准设施而增加的成本和低效性。

航空航天工业协会(AIA)理事会给出了:

(1) 用于支持全球航空航天工业的标准体系的关键需求;

(2) 按照定义的需求,当前使用的主要标准开发模型和组织;

(3) 对于航空航天工业要求的最佳标准的一组建议。

NASA 和 ESA 都提供了自己的软件工程专门标准。例如,ESA 的软件工程部门(软件工程和标准化部[70])负责软件工程化,既包括 ESA 内部标准——通过共同主持软件标准化和控制委员会(BSSC),也包括外部软件标准——主要是与欧洲空间标准化合作组织(ECSS),以及有时与国际标准化组织(ISO)一起。

1.1.3 复杂性

为了满足标准和安全性的规章,航空航天系统的软件开发需要实施严格的

---

① 规约,也称"规格说明",是指严格而详细的描述说明,兼有名词和动词的含义。在软件工程中,"规约(规格说明)"通常就是"需求规约(需求规格说明)"的简称。——译者

质量控制。例如,ESA 已经启动了许多举措来研究解决项目中的软件问题,主要是与时间进度和安全性相关的问题。其中的主要发现之一是:高复杂性使得航空电子系统及其软件十分脆弱[71]。由于持续增长的系统复杂性:

(1) 系统定义(需求和设计)的完成在项目中越来越靠后;

(2) 软件需求永远是不稳定的,软件工程师经常在软件已经处于组装、集成和测试(AIT)时还不得不实现仍然在变化中的需求;

(3) 软件开发的速度要求越来越快。

1.1.4 平台多样性

航空航天产品使用各种操作平台和运行环境。例如,ESA 的产品使用了不同的操作系统,包括 IRIX、Solaris、Linux、Windows 等。平台的多样性给开发过程带来了另一层复杂性。这里引用 ESA 建模与仿真组的一位仿真工程师 Peter van der Plas 的话:"当 ESA 的科学家要求一个新的功能时,我们就进行改造,发布一个新的版本,并在每一个平台上进行测试。我们拥有的平台越多,需要花的时间就越长。"

1.2 航空航天软件工程过程

通常,对于将要开发的任何软件系统,为该项目选择适当的开发生命周期过程(软件过程)是非常重要的事情,因为所有其他活动都是从该过程导出的。软件过程的概念意味着一组活动,其目标是软件的开发或演化。在所有的软件过程中,最通常的活动是:

(1) 规约——确定系统的功能和它的开发约束;

(2) 开发——生成软件系统;

(3) 确认——检查软件是否是客户想要的;

(4) 演化——改变软件以响应变化的要求。

一个航空航天软件的开发过程使我们必须谨记:航空航天系统需要满足各种标准,并且还要有高的安全性需求。为此,航空航天系统的开发强调验证、确认、审定以及测试。软件开发过程在技术上必须是充分的和高效费比的,能够控制航空航天系统设计的复杂性和满足安全性需求,并对其中包含的软件进行审定。将项目的生命周期拆分成若干个阶段,可以使得整个开发过程被组织成更加易于管理的片段。"软件开发过程应当在适合管理和预算环境的时间点上,

提供给管理者对于取得的进展的递增的可视性。”①在 NASA、ESA、波音和洛克希德·马丁开展的大多数最新的航空航天软件开发项目中，使用了某种基于螺旋的方法取代瀑布过程模型，其重点在于验证[56]。瀑布模型[61]是一个经典的软件生命周期，其中软件经历从一个阶段到下一个阶段的有序的转换过程。该模型类似于软件演化的一个有限状态机描述。在复杂的组织设置（如 NASA 和 ESA）中，瀑布模型及其衍生版本对于帮助结构化、分配人力以及管理大型软件开发项目可能是最有帮助的，这也是它的主要目的之一[8, 61]。

如图 1.1 所示，一个通常的航空航天软件开发过程包含密集的验证、确认和审定步骤，从而能够生成足够安全和可靠的控制系统[68]。

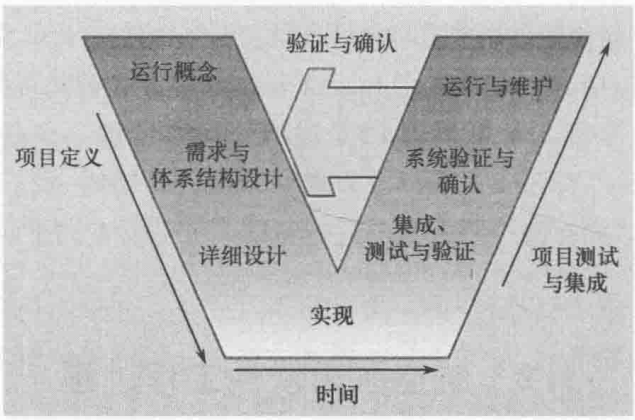


图 1.1　NASA/ESA 软件开发过程的一个通用视图[56]

（1）运行概念（Concept of Operatons）阶段（在 ESA，该阶段被称为系统与软件协同工程[68]），目的是将软件开发与系统需求关联起来。在这里，系统是指任何航天器/航空器子系统，由其产生航天器/航空器上的软件需求（如数据处理、姿态控制、温度控制、能源管理）。

（2）需求工程和体系结构设计阶段，强调高层建模，作为强化需求的完整性和一致性的手段。许多飞行软件的硬实时特性要求一个全面的可调度性分析以及使用特定的调度策略（如 Ravenscar[18]）。

（3）详细设计阶段进行的是低层系统建模。在该阶段，除了使用设计方法，还会用到建模语言，不过后者通常更加强调硬实时系统。

（4）实现可以是手工的或自动的。建模使得生命周期可以实现自动化，方式是通过自动生成实际飞行的代码以及一些确认测试。对于特定的适用于航天

① 见《NASA 系统工程手册》：“项目生命周期应当在适合管理和预算环境的时间点上，提供给管理者对于取得的进展的递增的可视性。”——译者

的处理器，对代码进行交叉编译。语言的选择受到航天中使用的硬件的约束影响。

（5）集成、测试和维护阶段由一系列用于对实现进行确认，以及在运行中对系统进行维护的活动组成。测试和验证都能发现设计和实现缺陷，而它们会导致实现之前的和实现步骤中的重复。

ECSS-E40[21]是一个用于在航天系统项目中开发软件的标准。ECSS是ESA、各国空间机构以及欧洲工业协会合作的结果，目的是建立和维护共同的标准。ECSS-E40强调一个以文档为中心的过程模型，用于开发可靠的器上实时软件（DOBERT）。该模型与图1.1所示的模型相似，其中每个文档都是用CASE-ML进行描述，CASEML是一个基于可扩展标记语言（XML）的语言。

ECSS-E40已经被用于ESA的各种项目，包括可靠器上嵌入式实时软件工程环境（DOBERTSEE）/低成本器上软件开发工具包项目[75]，用于为航空电子软件建立一个可以获得的和集成的软件工程环境。

1.2.1 需求工程和建模

需求工程[5, 69]是导出、编档和交流需求的过程。它管理需求的变化，在需求文档中维护需求的可追踪性。需求被编写为软件需求规约。软件需求反映了用户和购买者的需要。通常，需求与未来系统必须提供的服务相关，不过它们也可以是关于产品质量或约束的需求。而且，用户对系统的不同感受和理解，可以导致要么是极其抽象的服务描述（如“生成报告”），要么是详细的特征（如应当实现的数学公式）。因此，为了正确处理需求，需求工程师力图对需求进行分类。有两大类需求：功能（functional）需求和非功能（non-functional）需求。功能需求确定未来系统的运行，非功能需求关乎系统的质量或约束。根据电气与电子工程师协会（IEEE）软件需求实践建议（IEEE 830—1998标准），这两类需求应当关注这样一些问题[42]：

（1）功能——强调软件功能，就是将要提供给用户的服务；

（2）外部接口——包含软件将要如何与人或者其他软件和硬件系统交互的问题；

（3）性能——强调性能问题（对于系统整体以及系统的特别功能），如速度、可用性、响应时间、恢复时间等；

（4）质量属性——未来系统的质量的度量，如可移植性、正确性、易维护性、保密性等；

（5）约束——强调可能影响实现的设计、硬件、文化以及其他问题，如要求的标准、实现语言、数据库完整性原则、资源限制、操作环境等。

1.2.1.1 规约与建模

实践已经表明,在安全关键系统的开发中,形式化语言可以在需求规约和系统建模方面非常有用,如航空电子软件,其中软件的失效很容易引起安全性危害。例如,在 C130J Hercules Ⅱ的控制软件开发中,洛克希德·马丁应用了一个"通过构建保证正确(correctness-by-construction)"的方法,基于形式化的(SPARK)和半形式化的"联合需求工程(Consortium Requirements Engineering)"语言[3]。结果表明,这种组合足以消除大量的错误,在高质量和低成本方面为洛克希德·马丁带来了巨大的收益。在 ESA,需求规约和系统建模与以下方面相关[34]:

(1) 定义数据类型(XML 或 ASN.1)。

(2) 数据组织,如以类和对象的方式。

(3) 行为——行为建模语言可以对状态序列和系统参与的事件进行形式化表示。这通常是基于转换事件序列的状态机,基础是同步或异步模型。

在航天领域中使用的规约和建模语言中,最典型的一些包括规约与描述语言(SDL)、Esterel、Lustre、MatLab/Simulink 中实现的各种语言,等等。特别是,SDL 作为一种用于开发有限状态机的形式化建模的标准化语言,已经被主要应用于电信协议。它已经被证明在一些航天应用中非常有用[69]:

(1) 数据处理系统工作台(数据管理系统的设计确认(DDV));

(2) Meteosat 第二代航天器的失效检测、隔离和恢复(FDIR)——对航空电子可重配置性的器上软件需求进行了建模;

(3) SpaceWire 协议,提供了通信语言的一个标准。

Lustre 是一个异步数据流编程语言[34],用于响应式系统的编程。Lustre 已经被成功应用于关键应用的自动控制软件的开发,如空中客车的软件以及阵风战斗机的作战控制软件[7]。其他的形式化语言(如 B、VDM、PVS)可以被用于软件需求的特别功能。形式化语言的优势来自于严格的数学语义和高度的抽象,使得可以开发软件工具用于自动化的验证与确认。

需求经常是使用自然语言表达的。为了正确处理这样的需求(例如,实现可追踪性,以及避免由于使用自然语言而带来的模糊性),需要对自然语言进行分析。LEXIOR(LEXical analysis for ImProvement of Requirements)[16]是一个提供自然语言分析的工具。该工具包含一个用于编写需求规约的最佳实践规则数据库,以及一个词法分析和解析引擎,用于根据一组预先定义的最佳实践,进行预处理、内容验证以及交互式的编写和编辑。

1.2.1.2 需求度量

需求度量是对所开发的软件进行测量的一个重要部分。这包含需求波动性度量(在开始编码之后,需求发生的变化有多大)、需求可追踪性度量、需求版本度量、需求完整性度量。对需求进行手工测量是一项烦琐的工作,应当使用自动化的需求工具。这样的工具有助于高效地管理需求。例如,IBM Rational Requisition、Dynamic Object Oriented Requirements Systems,以及 Requirements Use Case 工具,是一些可用于对软件需求进行度量的主流自动化需求工具。许多商业化工具提供了需求的可追踪性和版本管理,如 DOORS 或 IRQA。需求度量在 DOBERTSEE[75]软件工程环境中也得到了实现。

1.2.1.3 需求的特性

为了测量软件的质量,需求工程定义了良好需求的特性。以下是 NASA 对良好需求的属性分类的一个简单描述[5]:

(1) 必要性——如果没有它,产品/系统就不能满足用户的实际需要。

(2) 清晰性——如果一个需求有多个解释,建立的系统就可能无法符合用户的需要。清晰至关重要。

(3) 完整性——要了解一个系统的全部未来需求是不可能的,但是所有已知的需求都应当得到规定。

(4) 一致性——需求之间不能相互冲突。

(5) 可追踪性——每个需求的来源应当得到标识。

(6) 可验证性——每个需求应当能够通过验证、分析、审查或演示而得到验证。在可能的情况下要避免否定式需求,如"部件不能过热"。

1.2.2 管理安全性和风险

复杂电子的需求来自于系统和子系统需求、安全性,以及所选择的体系结构、使用的技术和工具、设备将要运行的环境、实现环境带来的约束。对于 NASA 的许多系统,安全性是一个特别重要的需求来源。安全性需求可能进入系统需求,并继而进入复杂电子,或者在需求的每一级分解中被直接要求。

在开发过程中,软件工程师应当指定要求的安全级别,以确保设计和实现安全的系统。NASA 工程师将软件安全性表达为一组确保在正常和异常条件下系统具有可预测行为的特征和规程。而且,如果开发者正确地指定了软件安全性,则"一个计划外事件发生的可能性被最小化,并且其后果得到控制和限制"[37]。NASA 使用了两个软件安全性标准[51],标准定义了:①四个定性的危险严重级