软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2022,33(5):1652−1673 [doi: 10.13328/j.cnki.jos.006554]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一个机载软件需求形式化建模与分析实例研究*

胡 军[1,2], 吕佳润[1], 王立松[1,2], 康介祥[3], 王 辉[3], 高忠杰[3]

[1](南京航空航天大学 计算机科学与技术学院, 江苏 南京 211106)

[2](软件新技术与产业化协同创新中心, 江苏 南京 210007)

[3](中国航空无线电电子研究所 软件部, 上海 200233)

通信作者: 胡军, E-mail: hujun@nuaa.edu.cn

摘 要: 现代民机机载软件系统的功能与复杂度在快速增长的同时还必须满足更严格的安全标准, 使得在机载软件需求层级必须进行诸如一致性、完整性等分析与验证成为重要的挑战. 工作基于一个自主设计实现的面向机载软件自然语言需求形式化建模与分析工具平台 (ART) 展开对座舱显控软件子系统 (EICAS) 需求的建模与分析, 包括: ART 工具平台所采用的变量关系 (VRM) 理论模型、平台架构和平台工具链, 基于多范式的需求一致性、完整性形式化分析方法, EICAS 系统的条目化初始自然语言需求的形式化建模和需求模型的自动化分析过程, 如: 需求条目的预处理、规范化处理、需求模型自动生成以及多范式分析等; 给出了工程需求实例研究的经验总结和思考.

关键词: 机载软件形式化建模; 变量关系模型; 自然语言需求建模; 形式化方法

中图法分类号: TP311

中文引用格式: 胡军, 吕佳润, 王立松, 康介祥, 王辉, 高忠杰. 一个机载软件需求形式化建模与分析实例研究. 软件学报, 2022, 33(5): 1652–1673. http://www.jos.org.cn/1000-9825/6554.htm

英文引用格式: Hu J, Lü JR, Wang LS, Kang JX, Wang H, Gao ZJ. Case Study on Formal Modeling and Analysis of Airborne Software Requirements. Ruan Jian Xue Bao/Journal of Software, 2022, 33(5): 1652–1673 (in Chinese). http://www.jos.org.cn/1000-9825/6554.htm

Case Study on Formal Modeling and Analysis of Airborne Software Requirements

HU Jun[1,2], LÜ Jia-Run[1], WANG Li-Song[1,2], KANG Jie-Xiang[3], WANG Hui[3], GAO Zhong-Jie[3]

[1](College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China)

[2](Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 211107, China)

[3](Software Department, Chinese Aeronautical Radio Electronics Research Institute, Shanghai 200233, China)

Abstract: While the function and complexity of modern civil aircraft airborne software are growing rapidly, those safety standards for airborne software (such as DO-178B/C, etc.) must be satisfied at the same time. It raises more challenge to analyze and verify the consistency and integrity of airborne software requirements on the early stage of system development. This study introduces a formal modeling and analysis tool platform (avionics requirement tools, ART) for airborne software natural language requirements, and carries out a case study of the requirements of cockpit display and control software subsystem (EICAS). Firstly, the semantics of a formal variable relationship model (VRM) is given, also the platform architecture and tool chain of ART are descripted. Then, a methodology of formal analysis of requirement consistency and integrity based on multi-paradigm is given. After that, some details of the case study of EICAS are shown including: how to make a pre-modeling process of initial natural language requirements and the automatic analysis process of requirement model, such as the preprocessing and standardization of original requirement items, automatic generation of VRM models and multi-paradigm based formal analysis, etc. Finally, some experiences of this case study are drawn.

Key words: formal modeling for airborne system; variable relation model (VRM); natural language requirement modelling; formal method

* 基金项目: 工信部民机专项项目 (DAB1900501)
本文由“领域软件工程”专题特约编辑汤恩义副教授、江贺教授、陈俊洁副教授、李必信教授以及唐滨副教授推荐.
收稿时间: 2021-08-10; 修改时间: 2021-10-09, 2022-02-07, 2022-02-22; 采用时间: 2022-03-03; jos 在线出版时间: 2022-03-09

目前在世界运输类飞机的机载软件研发长期工程经验积累中已经形成一个共识, 即: 为了提高质量和降低成本, 必须建立系统工程以及软件工程的研发框架, 并在机载软件产品周期的早期阶段, 进行需求分析和验证, 确保需求的完整性和一致性[1−3]. 机载软件需求中存在的错误对机载系统的安全性有重大影响, 历史上曾经导致灾难性的飞行事故[4,5].

在民用飞机机载软件的国际适航安全标准中也给出基于需求的研发方法的强制要求. 如: 1992 年 RTCA 发布的 DO-178B《机载系统和设备合格审定中的软件考虑》标准[6]中, 给出了机载软件研发中的重点, 即: 强调以需求为中心的开发过程和验证目标, 不涉及具体软件技术[7−9]. 在最新版的 DO-178C[10]中还引入了形式化方法标准 (DO-333[11]). 虽然形式化方法[12,13]在计算机科学领域有着很长的发展历史, 但是如何应用不同的形式化方法理论, 并结合实际工程应用领域来形成航空工程中适用且有效的方法和工具, 来对这些不同层级上的机载软件应用需求进行构造、分析与验证, 在国内外仍然都是一个非常大的挑战.

本文工作内容主要介绍了在机载软件工程领域, 采用一个自主设计实现的面向自然语言需求的形式化建模与分析工具平台 ART (avionics requirment tools), 对民机座舱显示控制软件系统的一个典型模块-发动机显示与警告系统 (EICAS) 详细的形式化建模与分析的过程. 在 ART 中所用的形式化理论模型是变量关系模型 VRM (variable relation models)[14−16] (来源于四变量理论模型 FVM (four variable model)[17]. 通过工程实例分析我们还获得了一些形式化方法在工程实践中的经验, 并进行了思考和总结.

本文工作的主要贡献在于: 面向机载软件领域的自然语言需求实施形式化手段, 基于 VRM 表格化模型给出了一系列设计与实现, 形成了从理论模型、平台架构到平台工具链的完整工作. 同时总结了在机载软件领域下实施软件工程需求的研究经验, 为其他领域的自然语言需求形式化建模提供了一个有效范例.

本文第 1 节首先对民用航空领域中机载软件系统的安全关键特征以及相关的适航安全标准 (如: DO-178B/C 等) 进行了概述; 第 2 节给出了本文工作中所设计的一个面向运输类民用机载软件领域的自然语言需求形式化建模与分析工具平台 ART, 包括其变量关系 (VRM) 理论模型、平台架构和平台工具链的总体描述, 并重点说明了其中的基于多范式的需求一致性、完整性形式化分析方法; 第 3 节中给出了对座舱显控系统软件中的一个典型子系统-发动机显示与警告系统 (EICAS) 的案例分析, 详细说明了如何将 EICAS 系统的条目化初始自然语言需求进行形式化建模并自动化分析的过程, 包括: 需求条目的预处理、规范化处理、需求模型自动生成以及多范式分析等; 第 4 节给出了对需求实例研究的工程经验总结和分析; 最后一节是相关研究分析与未来工作.

1 民机航电软件与相关安全标准

DO-178B 标准的目的是指导设计与开发安全可靠的民用机载软件系统. 但是从 DO-178B 最初在 1992 年发布以来, 随着计算机科学与软件工程领域的快速发展, 现代民机的机载系统以及软件的开发过程和技术都发生了非常显著的变化, 如: 机载软件工程人员已经开始使用面向对象编程 (如: C++, Ada 等) 和各种建模、分析及代码生成技术 (如: UML, SCADE, Simulink 等). 其中, 最明显的一个变化就是现代机载软件的代码规模变得越来越庞大, 功能复杂度也在大幅增长[18,19], 如: 现代民机上一套典型的 Honeywell 飞行管理系统 (FMS) 中的软件源代码规模至少在 1 百万行, 并且随着系统功能的变化, 其代码规模每年都在以至少 2% 的速度在增长.

2012 年, DO-178B 升级为 DO-178C[10]. 新版的 DO-178C 综合考虑了计算机系统与软件领域的最新技术发展, 主要针对现代飞机系统中所必需的大规模复杂机载软件的研发和管理. DO-178C 继承了 DO-178B 的核心文件、原则和过程, 同时增加了高层级建模的支持 (DO-331)、面向对象 (DO-332) 和形式化方法 (DO-333). 其中, 形式化方法标准的引入是在 DO-178C 中的一个重要特色, 说明在高安全关键特征的机载软件领域开始认可计算机科学领域中的形式化方法具备在工程领域提升机载软件安全性的能力. 此外, 在 DO-178C 中, 仍然强调各个层级上的需求 (系统需求, 高级需求, 低级需求等) 及相关活动是机载软件研发和认证过程中的核心关注点, 形式化方法的引入也是围绕这一点展开的.

2 航电需求建模与分析平台 (ART)

在 DO-333 形式化方法标准中, 并没有具体的限定必须应用哪一类形式化方法, 而是给出了应用形式化方法需要满足的各类分析和验证目标; 也就是说, 无论采用何种形式的形式化方法, 都需要明确其在适航安全标准中所要求的某些目标[11], 如: 在需求阶段需要满足需求内容的一致性、完整性等目标; 其相应的软件支撑工具也应该结合工程实践来达到这些目标. 本节将介绍一个自主设计实现的面向民机机载软件领域的自然语言需求形式化建模与分析工具平台 (ART). 具体内容包括: ART 工具平台的形式化理论模型-变量关系模型 (VRM), ART 平台的架构和多个工具链的功能说明, 基于多范式分析的需求模型一致性、完整性形式化分析的基本方法等几个方面.

2.1 VRM 形式化模型

在计算机科学领域中, 通常采用多种定义的数学符号及逻辑公式来构建形式化理论和方法. 这些符号的设计没有考虑工程问题的特征和工程人员应用的要求, 使得目前形式化方法难以被机载软件领域工程人员所理解接受. 事实上, 各种形式的表格才是机载软件工程开发人员所熟悉的. 因此, 在 ART 平台中采用同时具备表格化特征和形式化语义的 VRM 需求模型[19].

VRM 模型来源于四变量模型[17], 并根据目前机载软件领域特征进行了定制调整, 如: 现代机载软件通常是通过航空数据总线进行数据包的读取和解析, 很少跟外部各类传感器设备的物理信号数据直接关联, 因此简化了原四变量模型中系统级的监视变量和控制变量集合; 在机载软件的数据输入和输出之间存在多种输入数据的组合或合并处理之后再进行输出或显示的特征, 因此在 VRM 模型中引入中间变量集合来承载此类信息等.

VRM 模型保留了原四变量模型中同时具备表格化和形式化语义特征的核心特征. 直观上来看, VRM 模型是基于一种二维表格来建立的需求结构, 但是这些表格的语义是有着严格的形式化定义. 这类似于关系数据库理论中的二维关系数据, 表面上看是一张二维表格, 但其具备由关系演算逻辑严格定义的形式化语义. 以下仅给出 VRM 模型中其中部分内容的案例展示, 详细内容可参考文献 [14−16].

对领域工程而言, 具体需求内容是在 VRM 模型表格中的内容来描述. VRM 模型中包括 3 种表格: 条件表, 事件表和模式转换表. VRM 模型中都对它们定义了详细的形式化语义. 以下用文献 [20] 中灯光控制系统的简化版本作为示例来简要说明表格模型的直观描述和形式化语义.

表 1 是一个条件表的例子, 表格中的 4 个条件对应于 4 个需求条目; 其直观语义是: 中间变量 tRemLL 在不同工作模式 (如: unoccupied, occupied 或 temp_empty) 下, 当满足不同的条件时, 可以取值的各种情况.

表 1 条件表示例

Mode (模式)(mcStatus)	Condition (条件)	
unoccupied	true	false
occupied	mIndoorLL>tCurrentLSVal	mIndoorLL≤tCurrentLSVal
temp_empty	mIndoorLL>tCurrentLSVal OR tOverride	mIndoorLL≤tCurrentLSVal AND NOT tOverride
tRemLL(输出量)	0	tCurrentLSVal-mIndoorLL

表 1 所对应的形式语义可以表达为一个逻辑公式如下:

$$tRemLL = F_1(mcStatus, mIndoorLL, tCurrentLSVal, tOverride) =$$
$$\begin{cases} 0 \text{ if } mcStatus = unoccupied \vee (mcStatus = occupied \wedge mIndoorLL > tCurrentLSVal) \vee \\ (mcStatus = temp\_empty \wedge (mIndoorLL > tCurrentLSVal \vee tOverride = true)) \\ tCurrentLSVal - mIndoorLL \text{ if } (mcStatus = occupied \wedge mIndoorLL \leqslant tCurrentLSVal) \vee \\ (mcStatus = temp\_empty \wedge mIndoorLL > tCurrentLSVal \wedge tOverride = false) \end{cases}$$

表 2 则给出一个事件表的例子, 基于新旧状态依赖关系集合: {mFMOverride, mcStatus}和{mFMOverride′, mcStatus′}, 定义了中间变量 tOverride 的值 (见表 2).

表 2 事件表示例

Mode (模式)	Event (事件)	
—	@T(mMFOverride) WHEN mcStatus!=occupied	@T(mcStatus=occupied)
tOverride′(输出量)	true	false

表 2 所对应的形式语义可以表达为一个逻辑公式如下:

$$tOverride' = F_2(mFMOverride, mcStatus, mFMOverride', mcStatus')$$
$$= \begin{cases} true \text{ if } mMFOverride = false \wedge mMFOverride' = true \wedge mcStatus \neq occupied \\ false \text{ if } mcStatus \neq occupied \wedge mcStatus' = occupied \end{cases}$$

表 3 是一个同样来自灯光控制系统的模式转换表的例子, 决定了模式集 mcStatus 所处模式随 mOccupied 和 mT3 变量值变化的特性.

表 3 模式转换表示例

Old Mode (源模式)	Event (事件)	New Mode (目标模式)
unoccupied	@T(mOccupied)	occupied
occupied	@F(mOccupied)	temp_empty
temp_empty	@T(DUR(NOT mOccupied)>mT3)	unoccupied
temp_empty	@T(mOccupied)	occupied

表 3 所对应的形式语义可以表达为一个逻辑公式如下:

$$mcStatus' = F_3(mcStatus, mOccupied, mT3, mOccupied', mT3')$$
$$= \begin{cases} temp\_empty \text{ if } mcStatus = occupied \wedge mOccupied = true \wedge mOccupied' = false \\ unoccupied \text{ if } mcStatus = temp\_empty \wedge mOccupied = false for more then mT3 seconds \\ occupied \text{ if } (mcStatus = unoccupied \wedge mOccupied = false \wedge mOccupied' = true) \vee \\ (mcStatus = temp\_empty \wedge mOccupied = false \wedge mOccupied' = true) \end{cases}$$

更多的其他有关事件表和模式转换表所对应的直观表格示例和对应的形式化逻辑语义定义可以见参考文献 [14,20].

在本节的最后, 再给出在实际工程中的一个条件表实例 (表 4). 实例表示的是在 EICAS 系统中, 不同的发动机状态参数的输入量组合情况下, 显示推力模式应该正确显示哪些信息. 显控软件具有一个典型的特征, 即: 输入/输出量取值组合所形成的条件可能会非常多, 因此, 表 4 中的显示格式做了调整, 但仍然保持了 VRM 模型语义.

表 4 EICAS 条件表实例

	状态参数	变量取值 (每一列之间是“或”, 每一行之间是“与”)							
输入	NormalTOSelected	T	F	F	F	F	F	F	F
	FlexTOSelected	F	T	F	F	F	F	F	F
	TO1DerateSelected	F	F	T	F	F	F	F	F
	TO2DerateSelected	F	F	F	T	F	F	F	F
	NormalCLBSelected	F	F	F	F	T	F	F	F
	MCTSelected	F	F	F	F	F	T	F	F
	CruiseSelected	F	F	F	F	F	F	T	F
	GASelected	F	F	F	F	F	F	F	T
条件	Reduced thrust takeoff	T							
输出	Text (文字)	TO	FLEX-TO	D-TO1	D-TO2	CLB	CON	CRZ	G/A
	Font (字体)	Small							
	Color (颜色)	Green 100							

该条件表的语义表格函数如下所示 (其中, 将输入变量分别以 $ip_1, p_2, \ldots, p_8$ 表示, 用 $ip_{m,1}, ip_{m,2}$ 表示 ip_m 取值情况, 条件 Reduced Thrust Takeoff 以 RTT 表示, op_1, op_2, op_3 表示输出变量, 用 $op_{1,k}, op_{2,k}, op_{3,k}$ 表示输出变量取值情况):

$$(op_i, cv_i) = f_i(ip_1, \ldots, ip_8, c_i) =$$
$$\begin{cases} 'TO' & \text{if}(ip_1 = T \wedge ip2 = F \wedge ip3 = F \wedge ip4 = F \wedge ip5 = F \wedge ip6 = F \wedge ip7 = F \wedge ip8 = F) \wedge RTT = T \\ 'FLEX-TO' & \text{if}(ip_1 = F \wedge ip2 = T \wedge ip3 = F \wedge ip4 = F \wedge ip5 = F \wedge ip6 = F \wedge ip7 = F \wedge ip8 = F) \wedge RTT = T \\ 'D-TO1' & \text{if}(ip_1 = F \wedge ip2 = F \wedge ip3 = T \wedge ip4 = F \wedge ip5 = F \wedge ip6 = F \wedge ip7 = F \wedge ip8 = F) \wedge RTT = T \\ 'D-TO2' & \text{if}(ip_1 = F \wedge ip2 = F \wedge ip3 = F \wedge ip4 = T \wedge ip5 = F \wedge ip6 = F \wedge ip7 = F \wedge ip8 = F) \wedge RTT = T \\ 'CLB' & \text{if}(ip_1 = F \wedge ip2 = F \wedge ip3 = F \wedge ip4 = F \wedge ip5 = T \wedge ip6 = F \wedge ip7 = F \wedge ip8 = F) \wedge RTT = T \\ 'CON' & \text{if}(ip_1 = F \wedge ip2 = F \wedge ip3 = F \wedge ip4 = F \wedge ip5 = F \wedge ip6 = T \wedge ip7 = F \wedge ip8 = F) \wedge RTT = T \\ 'CRZ' & \text{if}(ip_1 = F \wedge ip2 = F \wedge ip3 = F \wedge ip4 = F \wedge ip5 = F \wedge ip6 = F \wedge ip7 = T \wedge ip8 = F) \wedge RTT = T \\ 'G/A' & \text{if}(ip_1 = F \wedge ip2 = F \wedge ip3 = F \wedge ip4 = F \wedge ip5 = F \wedge ip6 = F \wedge ip7 = F \wedge ip8 = T) \wedge RTT = T \end{cases}$$

2.2 基于多范式的需求模型一致性与完整性分析

建立形式化 VRM 模型的目的是在需求阶段就可以对自然语言需求中常见的一致性、完整性问题进行形式化分析, 在本小节中以下给出 ART 中的多范式定义说明[14], 用于分析需求模型的一致性和完整性.

2.2.1 VRM 基本范式

VRM 基本范式的符合性, 是模型分析的基本要求. 即需求模型应符合如下要求.

1) 常量集合: 集合中所定义的常量必须有效, 即具备常量名、常量值, 及值类型符合要求.

2) 变量集合

- 变量集合中所有变量必须有定义.
- 变量集合中状态变量必须初始化, 初始值满足类型定义.
- 变量集合中的模式集变量, 还需要定义模式集合和初始模式.

3) 类型集合: 自定义类型必须存在父类型.

4) 表格函数

- 不存在空表.
- 必须使用变量集合中定义的变量.
- 同一变量的类型在跨表中保持一致.
- 源模式与目标模式必须属于同一模式集.
- 表格函数中的表达式定义无语法错误.

2.2.2 第一范式

第一范式约束单个表的输入完整性 (输入完整性范式). 表格中的输入变量取值必须包括了其值域范围, 如: 条件表/事件表中必须覆盖输入状态变量 r 的值域范围 $VR(r)$. 为此定义输入完整性约束为:

$$input_j \in VR\left(SV_{\mu(i)}\right) \wedge \left(\bigcup_{j=1}^{n} input_j = VR\left(SV_{\mu(i)}\right)\right).$$

对于条件/事件表, 其语义为: 对于指定的关联模式集 $SV_{\mu(i)}$, $VR\left(SV_{\mu(i)}\right)$ 为该模式集的值域, 表格函数的“模式”列中所有模式构成集合 $\bigcup_{j=1}^{n} input_j$, 若其与 $VR\left(SV_{\mu(i)}\right)$ 为等价集合, 则指定的表格满足第一范式约束. 如表 5 是以 *SI* (safety injection, 安全注入开关) 作为输出量的条件表, 其输入量为模式集 *Pre* (第 1 列), 表示水压模式, $\bigcup_{j=1}^{4} Pre_j = \{High, Permit\}$, 由前面的定义可知 $VR(Pre) = \{High, Permit, Low\}$. 因此 $\bigcup_{j=1}^{4} Pre_j \neq VR(Pre)$. 可以判断条件表不符合第一范式约束.

表 5 *SI* 条件表

模式集*Pre*	条件	输出变量*SI*
High	true	off
Permit	true	off
High	false	on
Permit	false	on

输入完整性对于模式转换表, 其语义有所不同, 定义为: 对于被该表指定模式间转换行为的模式集 $SV_{\mu(i)}$, $VR\left(SV_{\mu(i)}\right)$ 为该模式集的值域, 表格的"源模式"列中所有模式构成集合 $\bigcup_{j=1}^{n} input_j$, 若其与 $VR\left(SV_{\mu(i)}\right)$ 为等价集合, 则指定的表格满足第一范式约束.

2.2.3 第二范式

第二范式 (条件范式) 约束包括两部分: 条件一致性和条件完整性. (1) 条件一致性: 保证不出现同时产生矛盾输出值; (2) 条件完整性: 保证任一时刻都能映射到某一输出值. 后者在多条件组合的情况下会显得特别重要. 下面分别说明一下.

(1) 条件一致性约束为: $\forall m_i \in mc\left(\forall j,k \in N_i\left(c_{i,j} \wedge c_{i,k} = false\right)\right)$, 它要求: 在模式 m_i 确定的情况下, 对于产生不同输出的条件 $c_{i,j}$、$c_{i,k}$, 需满足 $c_{i,j} \wedge c_{i,k} = false$. 对于整个条件表, 当且仅当对于所有确定的模式 m_i 都满足上述公式, 才能通过条件一致性检查;

(2) 条件完整性约束为: $\forall m_i \in mc\left(\bigcup_{j\in N_i} c_{i,j} = true\right)$, 它要求: 在模式 m_i 确定的情况下, 对应的所有条件 $c_{i,1} \vee c_{i,2} \vee \ldots \vee c_{i,n} = true$. 对于整个条件表, 当且仅当对于所有确定的模式 m_i 都满足上述公式, 才能通过条件完整性检查.

为了清楚起见, 以下给出两个示例. 示例表 6 是一个违反第二范式的条件表. 最后两行的语义为当模式集 *Pre* 在模式 Low 下, 满足不同的条件时, *SI* 获取的输出值. 由于 $Overriden \wedge Overriden \neq false$, 即当处于模式 Low, 满足条件 Overriden 时, 输出将不确定是 off 还是 on, 存在二义性问题, 不满足条件一致性约束. 我们将表 6 最后一行第 2 列的 Overridden 修改为!Overridden, 则模式 Low 下满足条件一致性约束. 对 High 和 Permit 模式下进行同样的分析检查, 以确定该条件表是否完全满足条件一致性.

表 7 是一个违反条件完整性的示例, 在模式 Low 下, 其条件只考虑了 $WaterPres < 900$ 的情况, 而忽略了 $WaterPres \geqslant 900$ 时的任何可能性, 因此违反了条件完整性; 而在模式 Permit 下, 由于 Block 的值域仅有 off 和 on 两个值, 因此满足条件完整性. 但由于 Low 模式下的问题, 表 7 不完全满足条件完整性约束.

表 6 *SI* 条件表

模式集*Pre*	条件	输出变量*SI*
High	true	off
Permit	true	off
High	false	on
Permit	false	on
Low	Overridden	off
Low	Overridden	on

表 7 Overriden 条件表

模式集Pre	条件	输出Overriden
Low	*WaterPres*<900	false
Permit	Block=on	false
Permit	Block=off	false

2.2.4 第三范式

第三范式约束单个事件表的一致性, 即事件范式. 它的形式化定义为: 在模式 m_i 确定的情况下, 对于产生不同输出的事件 $e_{i,j}$、$e_{i,k}$, 需满足 $e_{i,j} \wedge e_{i,k} = never$. 整个事件表满足第三范式约束的条件为前述的事件一致性对所有模式 m_i 都成立, 即:

$$\forall m_i \in mc\left(\forall j,k \in N_i\left(e_{i,j} \wedge e_{i,k} = never\right)\right).$$

其语义为: 对于某事件表, *mc* 为其关联模式集. 在模式 m_i 确定的情况下, N_i 为所有模式列为 m_i 的行号集合, 由于事件可以视为状态的切换, 每个事件都等价于由事件发生前的条件和事件发生后的条件构成的条件对. 两事件的合取为 *never*, 相当于满足这两个事件发生前和发生后的两对条件分别合取, 其中任意一个合取式为永假式, 即两个事件不会同时发生. 对于处于该模式下的不同表格行的任意两行, 都需满足这两行的事件不会同时发生. 对于整个事件表, 当且仅当所有模式 m_i 满足事件不会同时发生, 才符合事件一致性约束.

对示例表 8 检查其是否符合事件一致性约束: 在模式 Low 下, 合取所有事件.

表 8 Overriden′事件表

模式Pre	事件event	输出Overriden′
Low	@T(Block=on)WHEN(Reset=off)	true
Permit	@T(Block=on)WHEN(Reset=on)	true
Low	(@T(Pre=High))‖(@T(Reset=on))	false
Permit	@T(Pre=High)WHEN(Reset=on)	false

$(@T(Block=on)WHEN(Reset=on))\wedge(@T(Pre=High)\vee@T(Reset=on))=never$, 不违反事件一致性; 但是在模式 Permit 下, 求其不同事件的合取 (第 3、5 行, 第 2 列), 得到:

$$(@T(Block=on)WHEN(Reset=on))\wedge(@T(Pre=High)WHEN(Reset=on))\neq never.$$

即: 当处于模式 Permit 下, 若此时 Block 由 off 转变为 on, Pre 由!High 转变为 High, 且 $Reset=on$, 则输出不确定是 true 还是 false. 由此可知, 表 8 违反了第三范式.

2.2.5 第四范式

第四范式 (输出完整性) 要求表格对输出量产生的输出值必须完全覆盖其值域范围. 满足此范式可以检查出是否有输出情况的遗漏.

输出完整性约束的形式化定义为:

$$output_j\in VR(V_i)\wedge\left(\bigcup_{j=1}^{n}output_j=VR(V_i)\right).$$

其语义为: 所有表格关联的输出量的可能取值情况 $VR(V_i)$ 都包含在表格中的输出值构成的集合 $\bigcup_{j=1}^{n}output_j$ 中. 这里的 V_i 是由表格函数产生输出值的输出量, 只能为受控变量 CV 或中间变量 IV. 表 9 对 SI 条件表进行第四范式的检查. 对于输出变量 SI, 表格中只存在其值为 off 的情况 (第 3 列), 而 $VR(SI)=\{on,off\}$. 故其违反输出完整性.

表 9 SI 条件表

模式Pre	条件	输出变量SI
High	true	off
Permit	true	off
Low	Overridden	off

上面所给出的 VRM 需求模型的多范式定义与模型分析算法, 目前已经在 ART 平台的 VRM 模型自动分析工具中实现了.

2.3 ART 平台架构和工具链

ART 是一个面向 DO-178C/DO-333 机载软件适航标准的软件需求分析与验证工具平台 (如图 1 所示). ART 项目的总体目标是考虑现代民机机载软件需求中的领域特征, 面向自然语言描述的需求条目, 建立包含条件、事件、模式等核心要素的工程实用的形式化需求模型的建模方法 (即本文中所提到的 VRM 模型), 然后基于 VRM 的形式化语义, 对需求模型展开一致性、完整性等形式化分析与验证; 最后, 经过形式化分析验证的 VRM 需求模型用来自动生成面向需求的测试用例集, 包括航空 A 级软件所需满足的 MC/DC 测试用例集.

在图 1 中给出了 ART 工具平台设计并应用的场景中的 5 个阶段.

第 1 阶段是 VRM 模型的构造; 将机载软件领域中常见的自然语言条目化需求 (通常用 DOORS 需求管理工具来进行管理) 导入到 ART 工具中, 由需求建模人员通过规范化建模方法和领域概念库将这些条目化需求按照规范化模版要求进行处理, 然后由模型生成引擎自动将这些规范化的条目需求构造出 VRM 表格模型, 即模型分析所需的条件表、事件表和模式表等[15];

第 2 阶段是 VRM 模型的形式化分析: 针对需求模型中可能存在的一致性、完整性问题, 使用模型自动分析引擎展开基于多范式的模型形式化分析[14]. 这些范式包括了前述的输入完整性、输出完整性、条件一致性、事件一致性等各类要求.

第 3 阶段是 VRM 模型的形式化验证: 对已经满足各类范式要求的 VRM 模型, 采用模型检验技术继续进行面向系统安全性质的验证. 在目前 ART 工具中是通过一个模型转换框架将 VRM 模型自动转换为 NuSMV 模型[21], 并提供了系统安全性质的时序逻辑公式的规范化模版输入方式, 最后调用 NuSMV 进行验证.

第 4 阶段是面向需求的测试用例自动生成: 对前面已经通过形式化分析与验证的 VRM 模型, 测试自动生成发动机会对其进行解析, 划分为多个逻辑子句, 构造相应的语义控制树, 然后根据用户定义的不同安全关键等级标准 (如: A 级、B 级等) 来生成相应的面向需求的测试用例集[16].

第 5 阶段是在实际的航空机载软件中进行实例应用, 在本文中第 3 节中所描述的 EICAS 系统需求实例分析就是这个阶段的部分工作展示.

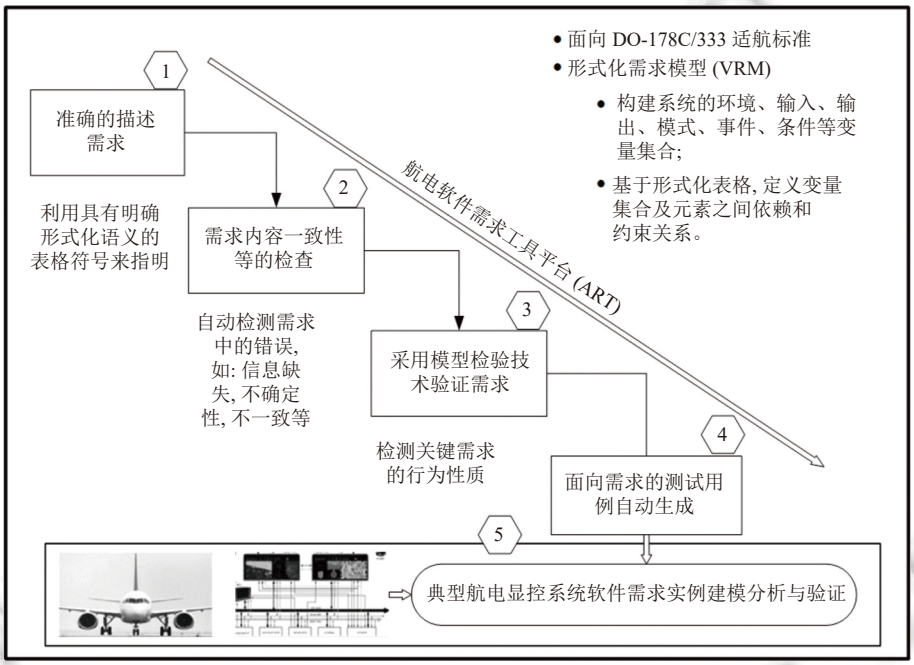


图 1 机载软件需求平台 ART

图 2 中给出了 ART 工具平台的主要框架. ART 基于 Eclipse 的插件框架, 集成了需求建模、分析、验证、测试用例生成等过程中多个不同工具, 形成一套工具链平台. ART 平台架构主要分为 3 层: 用户接口层、内核层和数据存储层. 用户层是面向需求建模不同阶段的用户, 提供不同的用户界面接口; 内核层则是多个不同的工具引擎, 包括: 领域概念库构造、VRM 模型生成、VRM 模型分析、模型验证以及测试用例自动生成、工程报告生成等; 数据存储层是 ART 平台各个工具的输入/输出所依赖的数据模型文件和后台数据库. 图 3 中给出了 ART 工具链中需求建模和测试用例生成的用户界面示例.

3 EICAS 需求的形式化建模与分析实例研究

ART 工具平台目前正在国产民机系统的机载软件中进行工程实例应用. 本节给出一个显控软件系统中 EICAS 模块需求实例, 包括了从自然语言到 VRM 建模以及分析的过程, 但本文中暂未包括后续形式化验证和测试用例自动生成. 以下内容主要有: EICAS 模块的概述, 条目化自然语言的预处理, 需求的规范化和模型生成, 模型的形式化分析及结果等.

3.1 机载显示控制软件 EICAS 的概述

EICAS (发动机指示和机组警告系统) 是现代飞机中综合显示系统的一部分, 为驾驶员提供发动机系统和其他

相关系统状态的显示 (如: 发动机转速、滑油温度、燃油消耗量以及液压系统等). EICAS 属于座舱显示控制系统的一个软件子系统, 其指示的发动机等系统的关键信息给机组人员提供了飞行中至关重要的参考数据, 如果显示出错将会导致非常严重的安全隐患. 常见的 EICAS 界面如后文的图 4 所示.

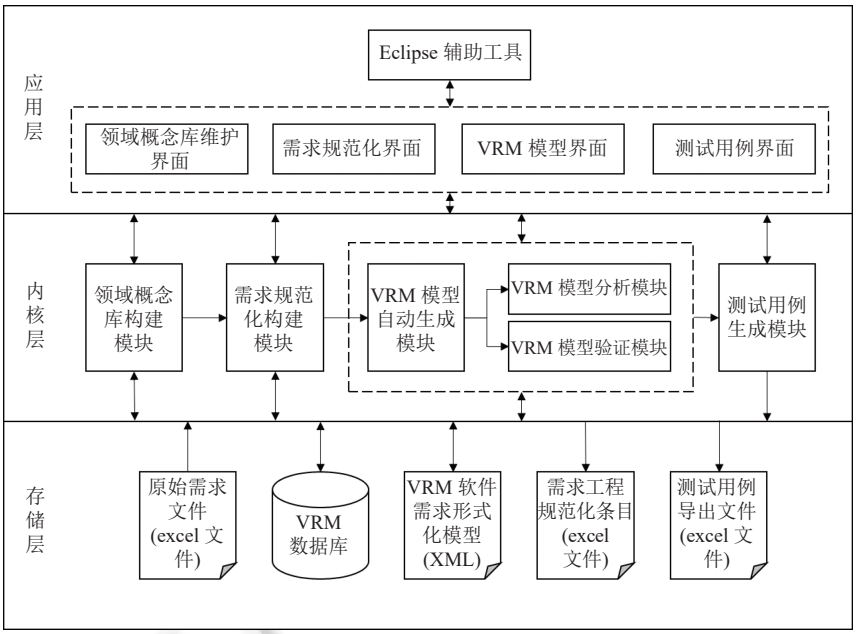
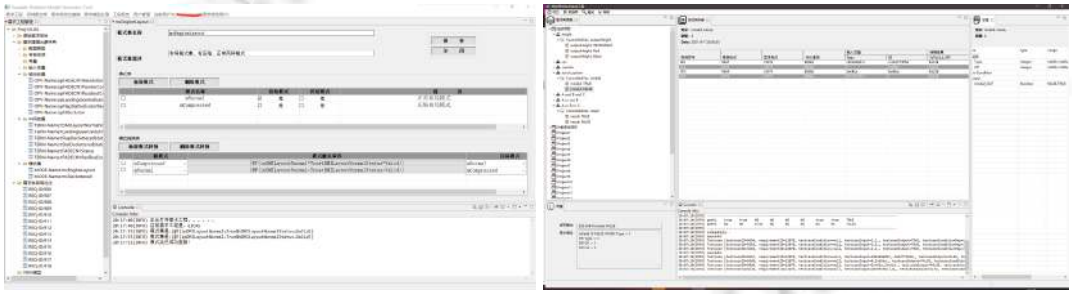


图 2 ART 的总体框架



(a) ART 需求建模界面 (b) ART 测试用例自动生成界面

图 3 界面图展示

3.2 条目化需求的预处理

在 ART 工具平台中, 需求预处理的目的是建立一个 EICAS 系统需求的领域概念库, 包括 EICAS 系统中常见的各种模式、输入/输出量、数据类型、中间变量等的定义. EICAS 领域概念库一旦建立起来, 在以后其他机型的 EICAS 系统需求建模过程中就可以复用这个库, 然后进行本地定制调整. 表 10 中给出了一部分经过预处理之后目前可公开的相对完整的 EICAS 需求条目示例, 用于本文后续的 VRM 模型构造和分析说明.

表 10 中的需求条目 1、2、3、4 的语义是描述 EICAS 系统的两个工作模式集, 以及如何进行模式转换的逻辑. 这四条需求可用于领域概念库定义中的模式转换建模; 其他的 13 条需求则通过 ART 进行基于模版的规范化建模, 最后自动生成 VRM 模型中的条件表或事件表. 具体部分工作描述如下. 图 5 所示为在 VRM 模型框架指导下所建立的条目化需求建模预处理的流程图.

预处理流程开始时, 首先对原始需求文档进行逐条需求条目的预分类. 建模人员需要与工程人员进行讨论, 并

确定各条需求具体要表达的语义, 防止对需求错误的理解. 经过逐条讨论, 建模人员可将需求分类为 3 类: 建模为模式转换的需求、建模为条件表的需求和建模为事件表的需求.

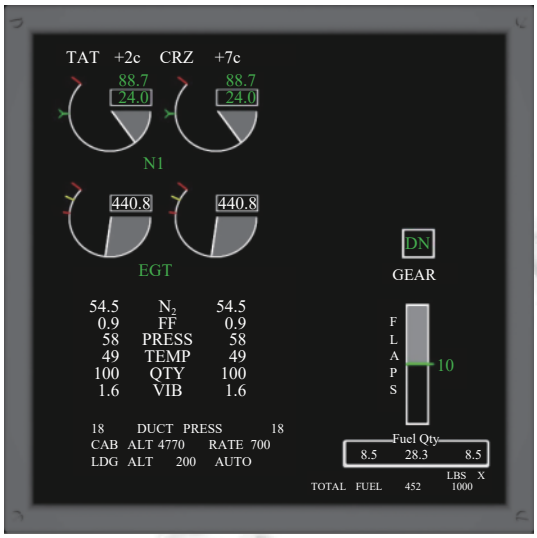


图 4 EICAS 显示界面示意图

表 10 EICAS 原始需求条目

需求编号	需求内容
1	正常模式的EICAS应进入防拥模式, 当1.起落架指示处于防拥; 且2.缝襟翼处于防拥.
2	压缩模式的EICAS应进入防拥模式, 当1.起落架指示处于防拥; 且2.缝襟翼处于防拥.
3	EICAS的界面布局应进入正常模式, 当ipDMILayoutNormal有效且为真.
4	EICAS的界面布局应进入压缩模式, 当ipDMILayoutNormal有效或为假.
5	当EICAS接收到的数据位于显示范围之外, 但处于ICD定义的信息之内时, EICAS应该将该数据显示为无效.
6	当EICAS收到的数据合法, 且数据的读数范围在有效的显示范围内时, EICAS应认为该数据为有效.
7	N1指示应该采用下示的分辨率显示NA读数: 1.0.1%对于(0%–99.9); 2.1%对于100%及以上.
8	N1指示应显示N1指针为白色, 当1.ipFDAECN1为合法, 且2.a.ipFADECN1RedlineExceedance为不合法, 或b.ipFADECN1RedlineExceedance为合法且为假.
9	N1指示应显示N1指针为红色, 当1.ipFADECN1为合法, 且2.ipFADECN1RedlineExceedance为合法且为真.
10	N1指示应显示N1读数为白色, 当1.ipFADECN1为合法, 且2.a.ipFADECN1RedlineExceedance为不合法, 或b.ipFADECN1RedlineExceedance为合法且为假.
11	N1指示应显示N1读数为红色, 当1.ipFADECN1为合法, 且2.ipFADECN1RedlineExceedance为合法且为真.
12	当ipFADECN1无效时, 应移除N1指针显示.
13	当ipFADECN1无效时, N1指示应显示N1读数为”--.-”样式.
14	当正常EICAS进入防拥模式时, 应移除起落架和襟缝翼指示.
15	当压缩EICAS进入防拥模式时, 应移除起落架和襟缝翼指示.
16	当EICAS进入防拥模式时, 应设置op_EI_Declutter信号为真(适用于压缩和非压缩模式).
17	当EICAS退出防拥模式时, 应设置op_EI_Declutter信号为假(适用于压缩和非压缩模式).

随后对这 3 类需求进行相关信息的提取 (如: 变量定义、条件、事件、模式等): 先考虑从模式转换型需求中提取构建模式集的信息, 此时由于领域概念库还未完全构建出来, 模式集信息中模式转换的部分可以是暂时空缺. 接着从 3 类需求中提取出其中所有条件或事件引用的输入变量、由各条需求决定取值的输出变量, 以及其中隐含的可能需要新设计的中间变量等, 然后根据需求中的语义, 来设置所有该变量的类型、精度等属性, 与工程人员进

行确认, 所构造的这些变量的信息是完全符合原始需求语义信息的.

最后, 将上述模式集信息、变量与数据类型信息汇总为一个领域概念库, 并将领域概念库中的各种变量信息作为反馈数据, 将前序步骤中尚未构造完整的模式转换表补充完整, 作为对领域概念库的逆向完善.

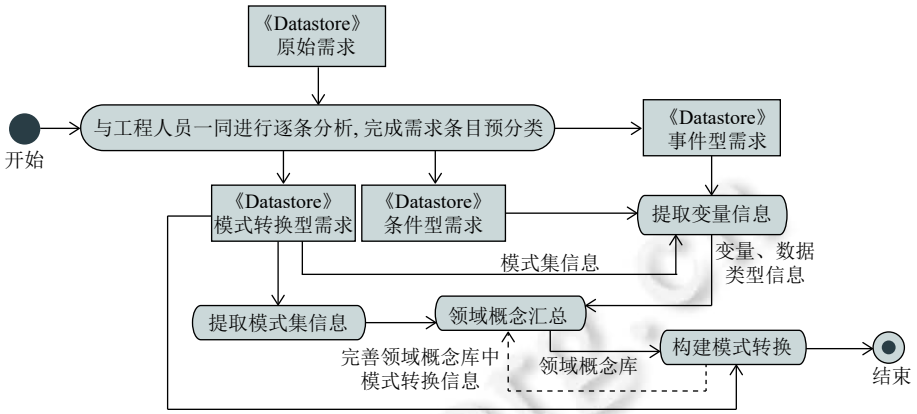


图 5 原始需求预处理流程图

下面是应用上述流程对 EICAS 条目化需求进行预处理的过程详细说明. 首先构建领域概念库中的模式集: 从需求 1 和 2 中给出的“进入防拥模式”可判断, 系统应具有防拥模式, 且有两种状态“处于”和“不处于”; 因此建立“防拥模式集”mcDecluttered, 其包含的模式有 mOff 和 mOn, 各自代表的语义为“处于防拥模式”和“不处于防拥模式”. 那么“进入防拥模式”的需求条目在后面就可以表达为在模式转换表中的 mcDecluttered 模式集从 mOff 转换为 mOn 模式的一条模式转换. 从需求 3 和 4 中给出的“进入正常模式”和“进入压缩模式”可判断, 系统还具有另外两种不同的布局模式“正常”和“压缩”; 因此需要建立另一个模式集“布局模式集”mcEngineLayout, 其包含的模式有 mNormal 和 mCompressed. 当系统处于两种模式时分别代表的语义为: “处于正常布局模式”和“处于压缩布局模式”. 那么“进入正常模式”和“进入压缩模式”分别可以表示为在模式转换表中 mcEngineLayout 从 mCompressed 转换为 mNormal 和从 mNormal 转换为 mCompressed 的模式转换. 上述两个模式集的模式转换具体定义还需后续步骤中其他的变量都设计完成后再进行完整的定义.

接下来, 从原始需求条目中提取输入/输出变量, 有些条目中还需要通过设计新的中间变量来承载原始需求中的某些主语, 或者某些条件、事件中需要进行判断的对象的语义. 以下给出其中的两个例子说明:

(1) 经与工程人员讨论, 并由需求 5 和 6 内容了解到, EICAS 系统从外部获得各种输入数据时, 这些输入数据都会附加一个信号量, 其取值范围为:

{GLOB_DATA_NO_DATA, GLOB_DATA_NO_COMPUTED_DATA, GLOB_DATA_FUNCTIONAL_TEST, GLOB_DATA_EMPTY, GLOB_DATA_NORMAL_OPERATION, GLOB_DATA_UNFRESH, GLOB_DATA_OUT_OF_RANGE}.

为此, 需要对所有输入变量建立对应的输入信号量, 名称为: 输入变量名加 Status 后缀; 其相应的数据类型设为 Status, 父类型为枚举型, 其值域为{NoData, NoCompData, FuncTest, Empty, Normal, Unfresh, OutOfRange}.

(2) 在需求 8、9、12 中都有“显示 N1 指针为白色”“显示 N1 指针为红色”“移除 N1 指针”等自然语言的描述性内容, 都需要设计变量来承载其语义信息. 为此设计变量 opFADECN1PointerColor, 和相关的数据类型 PointerColor, 父类型为枚举型, 值域为{NotDisplayed, Red, White}.

通过预处理整理得到的这些信息, 在 ART 的领域概念库中建立的变量字典和数据类型字典, 部分内容如表 11 和表 12 所示.

接下来再根据所建立的变量字典和数据类型字典, 将需求 1、2、3 和 4 中关于模式转换的内容进行精确的定义, 建立模式转换表. 如: 需求 1、2 中, 由于前置条件为 EICAS 布局模式分别为压缩和正常时, 对结果产生没有影

响, 因此将两条需求合并后移除对布局模式的判断条件, 最后得到的 mcDecluttered 包含如表 13 所示的模式转换. 在这里很容易看出, 在这个可公开的 EICAS 需求模型例子中, mcDecluttered 模式转换其实是不完整的, 因为在模式转换表中只有从 mOff 到 mOn 的转换, 缺少从 mOn 到 mOff 的模式转换, 不满足输入完整性范式 (第一范式). 这一点在后续模型分析的过程中也得到了确认. 需求 3 和 4 所建立的有关显示布局模式 mcEngineLayout 的转换表如表 14 所示.

表 11 EICAS 需求模型的变量字典 (部分)

变量名	值域	类别	初始值	关联模式集
ipLandingGearDecluttered	False, True	输入	False	无
ipFlapDecluttered	False, True	输入	False	无
ipSlatDecluttered	False, True	输入	False	无
ipDMILayoutNormal	False, True	输入	False	无
ipFADECN1	0.0–110.0	输入	0.0	无
ipFADECN1RedlineExceedance	False, True	输入	False	无
ipLandingGearDeclutteredStatus	NoData, NoCompData, FuncTest, Empty, Normal, Unfresh, OutOfRange	输入	Normal	无
ipFlapDeclutteredStatus	NoData, NoCompData, FuncTest, Empty, Normal, Unfresh, OutOfRange	输入	Normal	无
ipSlatDeclutteredStatus	NoData, NoCompData, FuncTest, Empty, Normal, Unfresh, OutOfRange	输入	Normal	无
ipDMILayoutNormalStatus	NoData, NoCompData, FuncTest, Empty, Normal, Unfresh, OutOfRange	输入	Normal	无
ipFADECN1Status	NoData, NoCompData, FuncTest, Empty, Normal, Unfresh, OutOfRange	输入	Normal	无
ipFADECN1RedlineExceedanceStatus	NoData, NoCompData, FuncTest, Empty, Normal, Unfresh, OutOfRange	输入	Normal	无
tLandingGearDeclutteredStatus	Invalid, Valid	中间	Invalid	无
tFlapDeclutteredStatus	Invalid, Valid	中间	Invalid	无
tSlatDeclutteredStatus	Invalid, Valid	中间	Invalid	无
tDMILayoutNormalStatus	Invalid, Valid	中间	Invalid	无
tFADECN1Status	Invalid, Valid	中间	Invalid	无
tFADECN1RedlineExceedanceStatus	Invalid, Valid	中间	Invalid	无
opFADECN1Resolution	0.1%, 1%	输出	0.1%	无
opFADECN1PointerColor	NotDisplayed, Red, White	输出	NotDisplayed	无
opFADECN1ReadoutColor	YellowDashes, Red, White	输出	YellowDashes	无
opLandingGearIndicatorRemoved	False, True	输出	False	mcEngineLayout
opFlapSlatIndicatorRemoved	False, True	输出	False	mcEngineLayout
opEIDeclutter	False, True	输出	False	mcEngineLayout

在需求模型预处理阶段, 还需要工程开发人员一起明确哪些需求条目是属于条件型的, 哪些是属于事件型需求, 如: 需求 5–13 都是属于条件型, 需求 14–17 是则是属于事件型. 在 VRM 模型中的条件型需求和事件型需求的语义是不同的, 简要来说: 条件型需求只需考虑当前系统状态, 而事件型需求则需要同时考虑当前系统状态和前一个系统状态. 具体的定义可参考文献 [14,15]. 区分条件型和事件型, 会直接关系到后续的模型语义分析.

3.3 需求的规范化与形式化模型生成

上述 EICAS 系统的需求模型预处理工作结束之后, 可以使用 ART 工具中需求规范化建模的功能来对表 10 中需求进行基于模版的规范化处理, 形成规范化的条目化需求集合, 然后将这些规范化的需求条目通过 VRM 模型构造引擎来自动生成各类条件表、事件表和模式转换表. 具体的模版定义、规范化处理和 VRM 表格模型自动生成方法可参见文献 [15]. 以下主要是从 ART 工具的操作层面来简要说明完成 EICAS 的需求规范化过程中的主

要工作和形式化模型生成.

首先, 在 ART 通过新建需求工程 Proj-EICAS, 将 DOORS 中 EICAS 条目化需求自动导入进来 (如图 6(a) 左侧所示). 然后根据每一个需求条目本身的语义信息的不同, 选择“通用条件”“显示条件”“通用事件”和“显示事件”等 4 类模版中的一种来进行规范化处理 (如图 6(a)). 以需求条目编号为 8 的需求为例, 本条需求为条件型需求, 并不是关于 EICAS 界面显示格式的需求, 因此可以选择使用“通用条件”规范化模板 (图 6(b)) 进行规范化处理.

表 12 EICAS 需求模型的数据类型字典 (部分)

类型名称	父类型	值域
Boolean	boolean	False, True
Status	enumerated	NoData, NoCompData, FuncTest, Empty, Normal, Unfresh, OutOfRange
VariableStatus	enumerated	Invalid, Valid
Thrust	float	0.0..110.0
Resolution	enumerated	0.1%, 1%
PointerColor	enumerated	NotDisplayed, Red, White
ReadoutColor	enumerated	YellowDashes, Red, White

表 13 mcDecluttered 模式转换

源模式	转换事件	目标模式
mOff	@T(ipLandingGearDecluttered=True&ipLandingGearDecluttered=True)	mOn

表 14 mcEngineLayout 模式转换

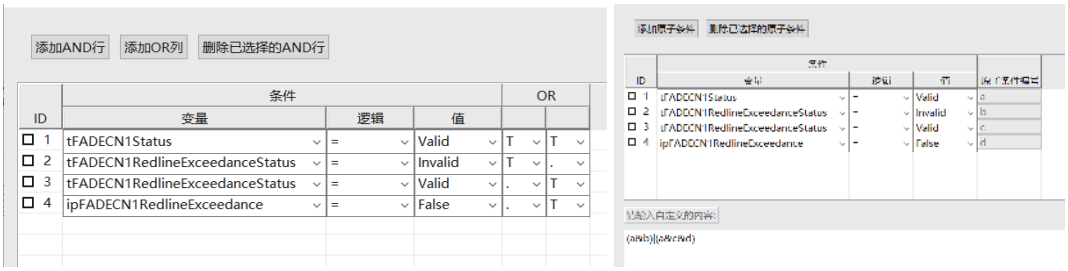
源模式	转换事件	目标模式
mCompressed	@T(ipDMILayoutNormal=True&tDMILayoutNormal=Valid)	mNormal
mNormal	@F(ipDMILayoutNormal=True&tDMILayoutNormal=Valid)	mCompressed

(a) 需求规范化建模主界面 (b) 通用条件模板界面

图 6 界面展示图

定义本条需求规范化的过程需要明确定义: 条件、主体、功能和客体. 其中主体可在下拉框中选择领域概念库中的所有中间变量和输出变量; 客体可在下拉框选择该变量的值域中的值; 功能可在下拉框选择领域概念库中的所有专有名词, 一般是一个谓语动词. 条件的输入有两种输入方式, 其一为图 6(b) 中“条件编辑”所提供 AND-OR 表输入方式[22]. AND-OR 表的输入方式需要工程人员提前将较为复杂的条件转换为析取范式, 然后再按照 AND-OR 表的形式进行输入.

经工程人员使用反馈之后, 在目前 ART 工具中还可以提供第 2 种方式, 如图 6(b) 中“自定义内容”方式, 给不熟悉 AND-OR 条件表的工程人员提供另外一种自由输入条件逻辑公式的方法. 其界面如图 7 所示.



(a) 使用 AND-OR 表的条件构造界面 (b) 用户自定义条件的界面

图 7 界面展示图

具体而言: 先将条件中的原子条件在 AND-OR 表中进行编号, 然后用编号和逻辑运算符在图 7(b) 下面的输入框根据需求条目中的逻辑条件进行自由输入, 此时不需要考虑是否符合析取范式的格式, 只要是符合基本的命题逻辑公式的格式就可以. 当接受了需求中的逻辑条件公式输入之后, ART 平台工具会自动解析转换成析取范式, 放到前面所选择的条件型模版中. 下面仍以编号为 8 的原始需求条目进行规范化过程中的条件公式如何构造为例说明.

经前面的需求预处理和分析, 可以得到需求 4 中的逻辑条件为:

① ipFADECN1 为合法, 且

② ipFADECN1RedlineExceedance 为不合法, 或

③ ipFADECN1RedlineExceedance 为合法且为假.

这个条件显然是“与-或-与”的形式, 需要转化为析取范式才能保存在规范化需求条目中. 可以将原子条件 tFADECN1Status=Valid 编号为 a, 原子条件 tFADECN1RedlineExceedanceStatus=Invalid 编号为 b, 原子条件 ipFADECN1RedlineExceedance=False 编号为 c, 原子条件 tFADECN1RedlineExceedanceStatus=Valid 编号为 d. 然后, 这样就只需要将该需求中条件的语义编辑为一个非常简单明了的逻辑公式: a&(b||(c&d)).

不同的需求建模人员可能会构造出另一个等价的逻辑公式, 如图 7(b) 中表示的 (a&b)||(a&c&d), 但最后都会自动转换为同一个范式. 即系统会在后台自动将其转换为标准的析取范式, 并存放在前面选择的条件型规范化需求条目中 (如图 6(a) 中的右侧). 当然, 这个析取范式通常会很长, 但这个是为了后续模型分析所用, 并不需要读懂. 如果在后续的需求更新中需要修改条件, 只需要重新进入图 7(b) 中的自定义界面进行修改编辑即可. 至此, 就完成了条件型需求 4 的规范化处理.

其他原始需求的处理类似. 如果某一条原始需求的语义比较复杂, 可能需要分解为多条规范化需求来分别处理, ART 工具平台会自动记录和维护原始需求与规范化之后的需求条目之间的一对多的对应关系. 当需求工程中的所有原始需求完成规范化后, ART 工具平台中的“VRM 模型自动生成”功能就可以在系统后台自动生成符合第 2.1 节中 VRM 理论模型的多个条件表、事件表和模式表的形式化模型 (具体的生成方法参见文献 [15,23]). 在表 15 中给出了表 10 中 EICAS 需求条目的规范化处理与模型生成的结果统计.

VRM 模型生成之后, 可以在 ART 中查看所生成表格模型的信息 (如图 8 所示). 在模型的展示界面上主要呈现 3 类信息: 与条件表相关联的变量、与事件表相关联的变量、与模式转换表关联的模式集等. 这里所说的与某变量或模式集相关联就是指这些变量或模式集的取值由这些表格函数来决定的. 如果还想查看更多的细节, 就可以通过该界面上的变量名、模式集名, 进入到一个新的界面, 显示其关联的条件表、事件表或模式转换表的详细内容. 与此同时, 系统后台还会生成一个 EICASmodel.xml 模型文件, 保存在 ART 平台设定好的固定目录下, 用于后续的模型分析、验证和测试用例自动生成.

经过前面的需求模型预处理、自然语言需求规范化处理和 VRM 模型自动生成, 接下来 ART 工具平台就可以基于 VRM 模型的形式化语义, 展开基于多范式的需求一致性、完整性自动化分析[14]. 图 9 中给出了表 10 中的 EICAS 需求进行多范式的模型分析截图, 表 16 中给出了详细范式分析结果的统计信息.

表 15 对表 10 中 EICAS 需求条目的规范化处理与模型生成的结果统计

统计项目		数量
可建模的EICAS原始需求	表10中的EICAS原始需求条目总数	17
	表10中可建模为模式转换的原始需求	2
	表10中可建模为条件表的原始需求	11
	表10中可建模为事件表的原始需求	4
建立领域概念库和原始需求条目的规范化处理	输入变量集合	12
	中间变量集合	6
	输出变量集合	6
	模式集集合	2
	需求规范化之后的条目	30
自动生成 的VRM模型	条件表	9
	事件表	3
	模式转换表	2

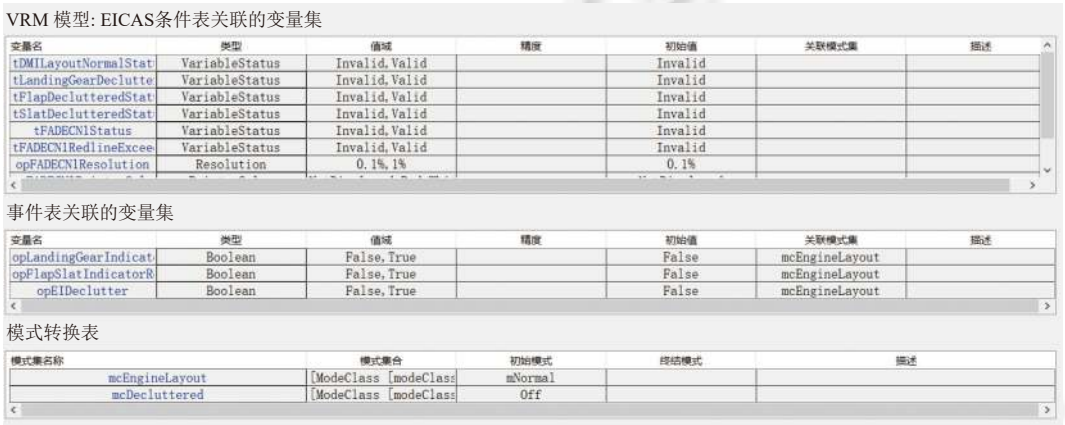


图 8 VRM 模型展示界面

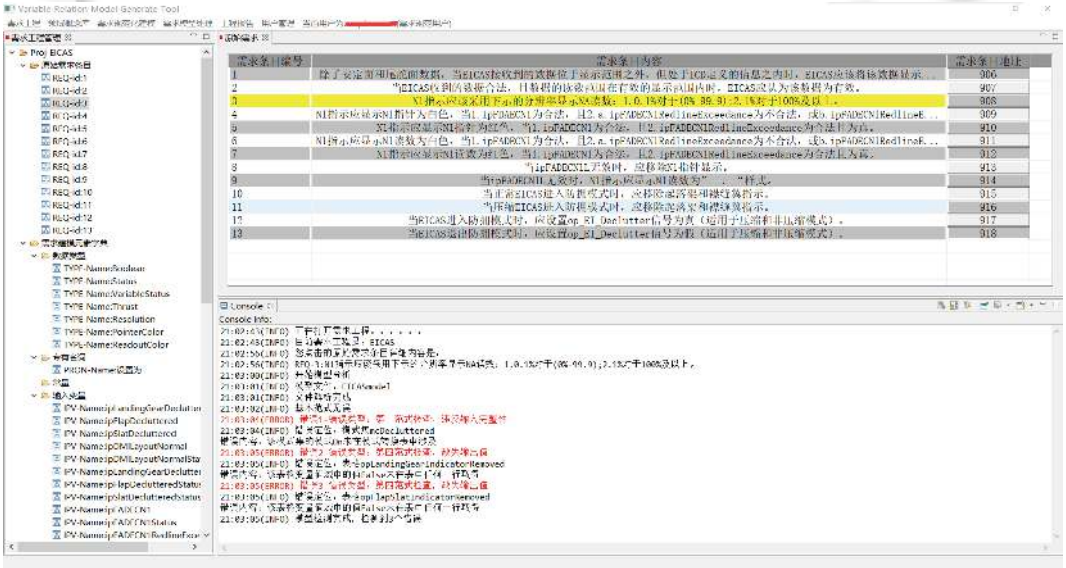


图 9 EICAS 模型分析报告

表 16 EICAS 需求模型分析结果统计

统计项目		数量
多范式的VRM模型形式化分析结果	基本范式错误	0
	输入完整性错误(第一范式)	***1***
	条件一致性错误(第二范式)	0
	条件完整性错误(第二范式)	0
	事件一致性错误(第三范式)	0
	输出完整性错误(第四范式)	***2***

从分析结果可以看出, 经过 ART 平台的模型自动分析引擎的多范式分析, 表 10 中的 EICAS 需求条目确实满足基本范式、条件一致性/完整性和事件一致性等, 但是在输入完整性和输出完整性方面存在范式错误. 在图 9 中可以看出, 当出现范式检查错误的时候, ART 平台能给出错误定位信息, 方便工程人员进行模型中的错误查找.

对上述范式错误的信息进行查找定位后, 发现:

(1) 不满足输入完整性 (第一范式) 的错误点正好是本文第 3.2 节中表 13 中所提及的 mcDecluttered 模式转换表, 该表中只有从 mOff 到 mOn 的转换, 缺少从 mOn 到 mOff 的模式转换. 经过与工程人员的交流确认, 在这部分公开的 EICAS 需求中确实缺少这部分内容.

(2) 不满足输出完整性 (第四范式) 的错误信息表明: 在 VRM 模型的输出变量 opLandingGearIndicator-Removed 和 opFlapSlatIndicatorRemoved 关联的事件表中, 未取到其值域中的取值 False, 造成的结果是违反了第四范式的输出完整性. 根据这样的报错信息来检查原始需求, 可以发现表 10 中的需求 14、15, 分别说明了 mcEngineLayout 处于 mNormal 和 mCompressed 模式下, 发生什么事件时, 要移除起落架和襟缝翼指示; 但是表 10 中没有任何一条其他的需求给出: 当发生什么事件时, 起落架和襟缝翼指示应该被重新显示出来, 可能导致的后果就是一旦这些指示被移除后, 将永远不再显示. 经过与工程人员的交流确认, 这部分公开的 EICAS 需求是一个早期的版本, 确实缺少这部分内容, 在后期的版本中已经进行了调整修改, 但是这个漏洞是在具体代码实现阶段由编程人员发现, 后来才补充完整的.

由此可见, 经过 ART 平台的形式化建模与模型自动分析, 对表 10 中 EICAS 系统需求检查出的这 3 个错误, 在工程上是非常有意义的; 这也表明 ART 平台及其相关的模型方法, 是可以有效发现需求中一致性、完整性的错误. 为了验证 ART 平台对其他范式的有效性, 我们还采用了主动修改 EICAS 需求模型中相关条件或事件的方式 (如: 删除或增加条件/事件表中的某一行条件/事件等), 进行错误注入, 分别对其他范式的错误检查能力进行了验证. 验证结果表明, 得益于形式化模型的精确语义模型, 所做的错误注入信息都可以被多范式的模型分析引擎检查出来.

4 对实例研究的分析

除了本文中 EICAS 系统需求的形式化建模与模型分析之外, ART 平台还被应用在其他一些机载软件的需求建模分析中, 并且从领域工程人员那边都得到了非常积极的结果反馈, 也积累了一些面向航空软件工程的需求形式化建模与分析的经验, 本节以下给出这方面的一些思考和总结, 并阐明了本研究工作的技术特色.

(1) 应用 VRM 模型和 ART 平台能够指导工程人员对需求条目进行良好的分类: 航空软件工程中目前已经能够完成自然语言需求条目化整理, 但是多数缺少对需求语义的分类检查. 按照 DO-178B/C 的软件工程管理要求, 在实际工程项目中工程人员通常都会将高级需求和低级需求整理成自然语言条目化的形式, 并存放在 DOORS 之类的需求管理工具中; 但是 DOORS 这类需求管理工具只能进行需求条目变化的版本管理和追踪, 无法对需求本身进行语法、语义检查. 在 ART 平台的工程应用过程中, 我们发现工程人员在 DOORS 中所存放的条目化需求, 常常会把不同种类的需求条目混杂在一起; 如: 有些需求条目是属于系统级需要满足的性质, 有些需求条目则是针

对某个具体输入量产生的输出的说明, 但是在条目化需求库中都是放在一起管理, 没有进行分类. 从模型分析和验证的角度来看, 前者其实是应该作为模型验证的目标, 而后者才是建立需求模型真正需要的需求条目信息.

(2) 首次建模中对需求预处理并建立相应的领域概念库能够对后续需求变更的管理和形式化建模提供更好的技术支撑: 在 ART 平台中, 条目化需求进行模型预处理的第一阶段工作非常重要, 会对后续的形式化建模与分析阶段产生重大影响. 在每次具体的工程实践中, 进行 ART 的工程应用的第一步都是需要对 DOORS 中的条目化自然语言需求进行分类和预处理, 因为并非每条实际的工程需求都是需要进行形式化建模与分析的. 预处理的目的包括: a) 将原始需求条目中系统级需求性质、可建模的动态行为需求与其他需求区分开来; b) 建立领域概念库, 包括: 提取系统工作模式、对隐含信息建立中间变量、明确输入/输出变量及类型字典等; c) 明确对哪些需求条目进行条件或事件语义的建模决定; d) 还有哪些需求目前还无法处理, 如: 跟时间直接相关的需求等. 表 17 给出了本文研究的可公开部分的 EICAS 系统需求的建模设计与分析的统计数据.

事实上, 最初的 EICAS 原始需求并不止表 10 中的 17 条需求, 而是 40 条 (如表 17 的第 1 行), 但是经过分类和预处理, 只保留了 17 条需求进行 VRM 模型的形式化建模与分析. 从表 17 中可以看出, 其他类别的需求中, 有 6 条需求其实是系统级属性类的需求, 如: “正常布局模式下的 EICAS 应该提供以下信息: (1) 发动机指示, (2) 油量信息, (3)CAS 信息, (4) 起落架信息, (5) Flap/Slat 信息, (6) 配平 (Trim) 信息, (7) 通信指示”; 这条需求实际上表达的是: 不论各种指示量该如何显示, 只要是正常布局模式, EICAS 界面上就应该包括这些所有的指示量; 而这种性质其实是一个系统级的属性, 可以用 CTL 逻辑公式来描述, 进而作为后续模型检验工具进行形式化验证所用.

表 17 EICAS 系统需求统计

	统计项目	数量 (条)
EICAS原始需求	***原始需求条目(包括表10中的需求)***	***40***
	可建模为模式转换的原始需求	4
	可建模为条件表的原始需求	9
	可建模为事件表的原始需求	4
	暂无法进行建模的需求	***6***
	系统级的属性类需求	***6***
	静态信息描述需求	***7***
	注释类需求	***2***
	语义冗余类需求	***2***
规范化需求	基于模版的需求规范化条目	30

此外, 其他还有几类需求, 如: 注释类需求 (2 条)、语义冗余类需求 (2 条)、静态信息描述需求 (7 条, 如: “N1 指示应显示在 EICAS 窗口的左上角”“N1 指示应该显示 NA 的指针尺寸如下: (1) 外圈直径为 27 mm; (2) 内圈直径为 26 mm, 与外圈保持 0.5 mm 的间隙; (3) 从 0 度开始; (4) 指示 210 度”等)、暂无法建模的需求 (6 条, 如: “当 ipFADECN1L 包含迟滞增加时, N1 指针应当顺时针扫过表盘”里包含的“迟滞”与“顺时针”等行为语义在目前的 VRM 模型中还不能有效表达).

因此, 我们认为: 从工程条目化需求出发来对一个软件系统需求进行第一次建模, 一定会经历这个预处理过程, 这个过程非常重要, 并且这个过程并不会节省成本, 但是形式化方法的更多价值是在后续系统更新、修改维护的过程中体现出来. 事实上, 前述的模型预处理过程也是一个需求模型的初步设计过程. 对于 ART 平台的应用而言, 这一步需求预处理的结果就是建立一个 EICAS 系统需求的领域概念库, 包括了 EICAS 系统中通用常见的各种模式、输入/输出量、数据类型、中间变量等的定义. 这个 EICAS 领域概念库一旦建立起来, 在以后其他机型的 EICAS 系统需求建模过程中就可以复用这个库, 然后进行本地定制调整.

(3) 建模中引入的新中间变量能够很好的支持形式化建模与分析: 在 VRM 模型规范化和建模的过程, 通常还需要引入新的变量来承载在原始需求中的某些必要的语义信息. 虽然在一些需求条目中会比较明确的给出输入/输出变量名, 但不少需求条目中并没有, 而且还需要通过设计新的变量来承载原始需求中的某些主语, 以及条件、

事件中会被判断的对象的语义等. 这样的处理是必要的, 因为 VRM 形式化模型的本质是建立在变量关系基础上的; 而且这样处理的好处是: 只要把这些重新设计的变量与概念之间建立的映射关系, 就能够保持原有需求条目的语义, 同时也能进行形式化建模与分析.

(4) VRM 模型中对事件和条件的不同语义定义能帮助工程人员深入理解条件型需求和事件型需求的不同: 工程人员对条件型的需求和事件型的需求常常并不清楚, 需要在需求预处理阶段和建模分析人员一起进行明确化判断. 事实上, 如前所述在 VRM 模型中, 条件型需求和事件型需求的语义是完全不同的: 条件型需求只需考虑当前系统状态, 而事件型需求则需要同时考虑当前系统状态和前一个系统状态之间特定变量集的变化情况, 而且还可以有较为复杂的带条件的事件语义描述. 严格区分事件型需求和条件型需求是 ART 平台中进行 VRM 需求建模与分析的一个主要特征, 相比较而言, 在很多其他的需求模型中, 并不区分条件和事件的语义.

(5) ART 平台的建模过程能有效发现需求中对有效性标识数据的前后引用不一致问题: 在机载软件工程领域, 出于安全性考虑, 几乎所有的输入数据都有一个对应的有效性标识数据, 即表示每个输入数据是否有效. 我们发现, 这种重要特征在大多数需求条目中都会明确提及并使用, 但是在某些需求条目中, 并没有明确说明每一个输入变量都有一个有效性伴随变量, 只是在某个章节分段部分通过所谓注释性的需求条目来说明这个要求, 然后在具体的需求条目中省略了有效性数据的说明. 这可能是在版本修订过程中, 不同的需求撰写人员采用了不同的写作风格所带来的问题; 但是这种前后不一致的做法非常有可能在实际工程中带来数据有效性检查遗漏的安全隐患. 对于这种问题, 我们发现在 ART 平台的需求预处理阶段能够很好的识别出这种情况, 然后进行统一的建模处理.

(6) ART 平台的建模过程可能很好的区分动态行为需求类和静态的需求类: 在机载软件的显控领域, 有相当部分的需求条目内容是关于显示格式、位置和颜色的静态规定, 不涉及到与输入数据变化相关的动态行为信息, 这类需求通过工程人员的人工检查就可以很容易完成, 无需通过 ART 平台建立 VRM 模型来进行严格分析.

(7) ART 平台能够支持需求模型的迭代设计构造过程: 需求模型的设计构造过程其实也是一个反复迭代设计精化的过程. 如: 在模型预处理过程中, 我们与工程人员需要首先逐条需求进行归类分析, 找出哪些是模式集及其转换相关的需求, 并确定模式集及模式取值, 但是在此时还没有完整的变量字典定义, 因此还需要等到变量字典以及类型字典都建立好了之后, 才能回过头来再将模式表中的模式转换完整的建立起来, 这就是一个迭代的过程. 因此, 基于 ART 平台的需求建模与分析过程其实本身就是一个相对闭环的工程设计过程, 也是需要进行良好的过程管理.

5 相关研究

目前在航空应用领域, 从机载软件需求的角度来看国外的相关理论、技术和工具的发展情况, 大致可以分为如下几类: (1) 从实际安全关键系统的开发工程经验中形成的理论与技术, 如: 四变量理论模型[17]、SCR (software cost reduction) 方法[24,25]、RSML (requirements state machine language) 方法[26], CoRE (consortium requirements engineering) 方法[27,28], SpecTRM (specification tools and requirements)[29−31], T-VEC 工具[32]等; (2) 从通用的软件工程领域产生的软件需求规约方法, 如: 统一建模语言 (UML)[33,34]中的用例 (UseCase) 模型的需求捕获与描述方法以及 StateCharts 状态机模型[35]、从 UML 扩展而来的系统建模语言 (SysML)[36]中用于描述系统需求的参数模型, 其典型的工具包括了: Raphsody[37], Statmate[38]等; (3) 从电子硬件系统设计的同步数据流语言发展而来的需求建模与代码生成技术, 如: Matlab 公司的 Simulink 工具[39,40]、基于 Esterel 技术的 SCADE 工具[41]等; (4) 以自然语言或者采用某些特殊语义限定词的结构化自然语言的形式来描述系统和软件需求, 通常用这类方法来描述的需求都是采用条目化的管理方法, 使用诸如 DOORS 之类的需求工具进行管理.

在上述的相关工作中, 四变量理论模型 FVM 是由 Parnas 和 Madey 等人共同提出的机载作战程序软件的

需求规范方法. 其模型中定义了 4 类变量集: 监视量、控制量、输入量和输出量等, 设计了二维表格来描述 4 类量之间需求关系, 并给出表格的形式化定义语义. 本文中 ART 工具平台的理论模型 VRM 就是来源于四变量理论模型.

相比较而言, 统一建模语言 UML/SysML 所提供的图形化符号在完整性、一致性和数学的严谨性上强调的不够. 仅使用 UML/SysML 提供的需求描述方法不足以满足现代安全关键机载系统和软件的需求设计、分析和验证的要求. 而基于同步数据流的图形化建模语言 (如: Simulink 和 SCADE) 设计初衷为系统和软件的设计, 近些年被应用于建模需求规范. 但从 DO-178C 中要求的多层级需求来看, Simulink 和 SCADE 描述的需求模型最多属于 DO-178C 中的低级需求. 其原因是: 在这些模型中通常都包含了很明显的设计信息和细节. 最后一类限定自然语言的需求方法 (如: RSL[42]采用限定的结构化的自然语言描述需求), 通常使用 DOORS 对条目化需求进行管理. 但 DOORS 本身只是一个需求管理工具, 并不能对需求的语义进行检查. 因此, 当系统的规模增大, 所构造的大量需求条目必不可少存在二义性、完整性等问题, 显然 DOORS 工具无法支撑 DO-178C 任何一个层级的需求分析和验证的任务要求.

除了上述相关工作之外, 还有其他一部分面向自然语言需求的工作包括文献 [43,44], 此类工作的主要思想也是通过模版化的手段将自然语言描述的详细需求 (主要是面向航天领域) 转换为 AADL 架构模型 [45], 然后通过 AADL 模型的仿真运行或者形式化验证技术来对需求进行验证分析. 此类工作并不是在需求层级上进行一致性、完整性分析, 而是转换到系统架构层级, 用系统架构来验证需求是否符合用户的功能要求.

与本文工作中的 ART 平台功能比较类似的商业工具是 T-VEC[31]. T-VEC 也是一个基于四变量理论模型, 面向软件需求的建模与测试用例自动生成的软件工具. 经过对比分析, 在形式化建模方面, T-VEC 工具仅提供了一个文本化的模型编辑界面, 所有的变量定义、表格函数定义以及条件逻辑公式的编辑都完全由用户自行输入, 而 ART 工具平台提供了一个较为完整的模型预处理过程支持, 通过建立领域概念库、自然语言的规范化处理、条件的逻辑表达式简便输入构造法等来支持工程人员来尽可能的做好需求预处理这个重要的阶段. 此外, 在模型分析方面, T-VEC 仅提供类似基本范式和第一范式的检查, 无法像 ART 平台一样还能完成其他的几个范式所对应的需求一致性和完整性分析功能.

6 总结与未来工作

综上所述: 本文工作首先对民用航空领域中机载软件系统的安全关键特征以及相关的适航安全标准 (如: DO-178B/C 等) 进行了概述; 然后给出了所设计的自然语言需求形式化建模与分析工具平台 ART, 包括其变量关系 (VRM) 理论模型、平台架构和平台工具链的总体描述, 并重点说明了基于多范式的需求一致性、完整性形式化分析方法. 然后选择了在显控系统软件中的一个典型模块-发动机显示与警告系统 (EICAS) 进行案例分析, 详细说明了如何将 EICAS 系统的条目化初始自然语言需求进行形式化建模并自动化分析的过程, 包括: 需求条目的预处理、规范化处理、需求模型自动生成以及多范式分析等. 最后还给出了对 EICAS 需求实例研究的工程经验总结和思考.

因此, 本文工作的主要贡献在于: 面向机载软件领域的自然语言需求实施形式化手段, 基于 VRM 表格化模型给出了一系列设计与实现, 形成了从理论模型、平台架构到平台工具链的完整工作. 同时总结了在机载软件领域下实施软件工程需求的研究经验, 为其他领域的自然语言需求形式化建模提供了一个有效范例.

未来的工作包括: 继续结合实际的工程应用, 坚持“从工程中来, 到工程中去”的基本原则, 着重从工程应用中发现真实的问题, 来持续的修正、更新 ART 工具链平台的形式化理论模型、分析与验证算法群以及工具用户体验; 如: 部分用户对复杂机载软件的工程需求可能是以模块化结合条目化的形式来呈现, ART 工具后续还需要考虑在理论、算法和工具实现等各个层级上如何有效支持模块化建模、分析与验证; 基于 NuSMV 的自动化验证的结果中反例行为数据如何能直观简要的展示给用户, 并准确的反馈到模型层级进行错误定位, 且能形成相应的认证数据报告等工作. 因此, 我们需要在未来进行更大规模的实例应用研究, 通过工程实践和反馈, 不断提升 ART 工具平台在国家重大工程中的有效性.

References:

[1] Heimdahl M, Leveson N, Redler J, Felton M, Lee G. Software assurance approaches, considerations, and limitations: Final report. DOT/FAA/TC-15/57, Federal Aviation Administration, U. S. Department of Transportation, 2016.

[2] Leveson NG. Role of software in spacecraft accidents. Journal of Spacecraft and Rockets, 2004, 41(4): 564–575. [doi: 10.2514/1.11950]

[3] Rierson L. Developing Safety-Critical Software: A Practical Guide for Aviation Software and DO-178C Compliance. Boca Raton: CRC Press, 2013.

[4] Conradi K. Report on the serious incident to Boeing B787-8, ET-AOP London Heathrow Airport 12 July 2013. UK: AAIB Press, 2015.

[5] Japan Transport Safety Board. Aircraft serious incident investigation report All Nippon Airways Co. Ltd. JA804A. Report No. AI2014-4, Japan Transport Safety Board, 2014.

[6] RTCA DO-178B. Software Considerations in Airborne Systems and Equipment Certification. 2nd ed., RTCA Press, 1992.

[7] Lempia DL, Miller SP. Requirements engineering management findings report. DOT/FAA/AR-08/34, Federal Aviation Administration, U. S. Department of Transportation, 2008.

[8] Lempia DL, Miller SP. Requirements engineering management handbook. DOT/FAA/AR-08/32, Federal Aviation Administration, U. S. Department of Transportation, 2009.

[9] Federal Aviation Administration. Advanced Avionics Handbook. New York: Skyhorse Publishing, 2011.

[10] Rierson, Leanna. Developing Safety-critical Software: A Practical Guide for Aviation Software and DO-178c Complia. Taylor & Francis, 2013.

[11] Cofer D, Miller SP. Formal methods case studies for DO-333. Guidance Systems, 2014.

[12] Wang J, Zhan NJ, Feng XY, Liu ZM. Overview of formal methods. Ruan Jian Xue Bao/Journal of Software, 2019, 30(1): 33–61 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5652.htm [doi: 10.13328/j.cnki.jos.005652]

[13] Chen G, Yu LY, Qiu ZY, Wang Y. Logic based formal verification methods: Progress and applications. Acta Scientiarum Naturalium Universitatis Pekinensis, 2016, 52(2): 363–373 (in Chinese with English abstract). [doi: 10.13209/j.0479-8023.2015.131]

[14] Shi MY. Research on system requirements and design safety analysis method based on formal model [MS. Thesis]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2020 (in Chinese with English abstract). [doi: 10.27239/d.cnki.gnhhu.2020.000101]

[15] Hu JC, Hu J, Wang WX, Kang JX, Wang H, Gao ZJ. Constructing formal specification models from domain specific natural language require-ments. Journal of Chinese Computer Systems, 2021, 42(8): 1639–1648 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-1220.2021.08.012]

[16] Wang WX, Hu J, Hu JC, Kang JX, Wang H, Gao ZJ. Automatic test case generation from formal requirement model for avionics software. In: Proc. of the 6th Int'l Symp. on System and Software Reliability (ISSSR). Chengdu: IEEE, 2020. 12–20. [doi: 10.1109/ISSSR51244.2020.00011]

[17] Parnas DL. Software aspects of strategic defense systems. Communications of the ACM, 1985, 28(12): 1326–1335. [doi: 10.1145/214956.214961]

[18] Huo M, Deng ZW. Development trend of foreign military avionics. Avionics Technology, 2004, 35(4): 5–10 (in Chinese with English abstract). [doi: 10.3969/j.issn.1006-141X.2004.04.002]

[19] Shi JJ. Research on system safety modeling and analysis based on four-variable model [MS. Thesis]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2016 (in Chinese with English abstract).

[20] Heitmeyer C, Bharadwaj R. Applying the SCR requirements method to the light control case study. Journal of Universal Computer Science, 2000, 6(7): 88–92.

[21] Zhang Y, Hu J, Wang LS, Kang JX, Wang H, Gao ZJ. Formal verification method for SCR requirement model. Journal of Chinese Computer Systems, 2022, 43(1): 193–202 (in Chinese with English abstract).

[22] Heimdahl MPE, Leveson NG, Reese JD. Experiences from specifying the TCAS II requirements using RSML. In: Proc. of the 17th DASC. AIAA/IEEE/SAE. Digital Avionics Systems Conf. Proc. Bellevue: IEEE, 1998. C43/1–C43/8. [doi: 10.1109/DASC.1998.741499]

[23] Hu J, Hu JC, Wang WX, Kang JX, Wang H, Gao ZJ. Constructing formal specification models from domain specific natural language requirements. In: Proc. of the 6th Int'l Symp. on System and Software Reliability (ISSSR). Chengdu: IEEE, 2020. 52–60. [doi: 10.1109/ISSSR51244.2020.00017]

[24] Heitmeyer CL. Software cost reduction. In: Marciniak JJ, ed. Encyclopedia of Software Engineering. 2nd ed., New York: John Wiley & Sons, Inc., 2002. [doi: 10.1002/0471028959.sof307]

[25] Hager JA. Software cost reduction methods in practice: A post-mortem analysis. Journal of Systems and Software, 1991, 14(2): 67–77. [doi: 10.1016/0164-1212(91)90091-J]

[26] Leveson NG, Heimdahl MPE, Hildreth H, Reese JD. Requirements specification for process-control systems. IEEE Trans. on Software

Engineering, 1994, 20(9): 684–707. [doi: 10.1109/32.317428]

[27] Faulk S, Brackett J, Ward P, Kirby J Jr. The core method for real-time requirements. IEEE Software, 1992, 9(5): 22–33. [doi: 10.1109/52.156894]

[28] Faulk SR, Kirby J Jr, Finneran L, Moini A. Consortium requirements engineering guidebook. Technical Report, SPC-92060-CMS, Herndon: Software Productivity Consortium, 1993.

[29] Howard JR, Anderson PW. The safety risk of requirements incompleteness. In: Proc. of the 20th Int'l System Safety Conf. Denver, 2002. 225–234.

[30] Lee G, Howard J, Anderson P. Safety-critical requirements specification and analysis using spectrm. In: Proc. of the 2nd Meeting of the US Software System Safety Working Group. 2002.

[31] Leveson NG, Reese JD, Heimdahl MPE. SpecTRM: A CAD system for digital automation. In: Proc. of 17th DASC. AIAA/IEEE/SAE. Digital Avionics Systems Conf. Proc. Bellevue: IEEE, 1998. B52/1–B52/8. [doi: 10.1109/DASC.1998.741474]

[32] Blackburn MR, Busser RD. T-VEC: A tool for developing critical systems. In: Proc. of 11th Annual Conf. on Computer Assurance. Gaithersburg: IEEE, 1996. 237–249. [doi: 10.1109/CMPASS.1996.507891]

[33] Booch G, Rumbaugh J, Jacobson I. The Unified Modeling Language User Guide. Reading: Addison-Wesley, 1999.

[34] Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. 3rd ed., Reading: Addison-Wesley, 2003.

[35] Harel D. Statecharts: A visual formalism for complex systems. Science of Computer Programming, 1987, 8(3): 231–274. [doi: 10.1016/0167-6423(87)90035-9]

[36] Delligatti L. SysML Distilled: A Brief Guide to the Systems Modeling Language. Addison-Wesley, 2013.

[37] Gery E, Harel D, Palachi E. Rhapsody: A complete life-cycle model-based development system. In: Proc. of the 3rd Int'l Conf. on Integrated Formal Methods. Turku: Springer, 2002. 1–10. [doi: 10.1007/3-540-47884-1_1]

[38] Harel D, Lachover H, Naamad A, Pnueli A, Politi M, Sherman R, Shtull-Trauring A, Trakhtenbrot M. STATEMATE: A working environment for the development of complex reactive systems. IEEE Trans. on Software Engineering, 1990, 16(4): 403–414. [doi: 10.1109/32.54292]

[39] Dabney JB, Harman TL. Mastering Simulink. Upper Saddle River: Pearson/Prentice Hall, 2004.

[40] Bemporad A, Morari M, Ricker NL. Model Predictive Control Toolbox™: Getting Started Guide. Natick: The MathWorks, Inc., 2012.

[41] Berry G, Sentovich E. Multiclock esterel. In: Proc. of the 11th Advanced Research Working Conf. on Correct Hardware Design and Verification Methods. Livingston: Springer, 2001. 110–125. [doi: 10.1007/3-540-44798-9_10]

[42] Nielsen M, Havelund K, Wagner KR, George C. The RAISE language, method and tools. Formal Aspects of Computing, 1989, 1(1): 85–114. [doi: 10.1007/BF01887199]

[43] Yang ZB, Hu K, Zhao YW, Ma DF, Bodeveix JP. Verification of AADL models with timed abstract state machines. Ruan Jian Xue Bao/Journal of Software, 2015, 26(2): 202−222 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4776.htm [doi: 10.13328/j.cnki.jos.004776]

[44] Wang F, Yang ZB, Huang ZQ, Zhou Y, Liu CW, Zhang WB, Xue L, Xu JM. Approach for generating AADL model based on restricted natural language requirement template. Ruan Jian Xue Bao/Journal of Software, 2018, 29(8): 2350−2370 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5530.htm [doi: 10.13328/j.cnki.jos.005530]

[45] Feiler PH, Gluch DP, Hudak JJ. The architecture analysis & design language (AADL): An introduction. Pittsburgh: Carnegie Mellon University, 2006.

附中文参考文献:

[12] 王戟, 詹乃军, 冯新宇, 刘志明. 形式化方法概貌. 软件学报, 2019, 30(1): 33−61. http://www.jos.org.cn/1000-9825/5652.htm [doi: 10.13328/j.cnki.jos.005652]

[13] 陈钢, 于林宇, 裘宗燕, 王颖. 基于逻辑的形式化验证方法: 进展及应用. 北京大学学报(自然科学版), 2016, 52(2): 363–373. [doi: 10.13209/j.0479-8023.2015.131]

[14] 石梦烨. 基于形式化模型的系统需求与设计安全性分析方法研究 [硕士学位论文]. 南京: 南京航空航天大学, 2020. [doi: 10.27239/d.cnki.gnhhu.2020.000101]

[15] 胡建成, 胡军, 汪文轩, 康介祥, 王辉, 高忠杰. 一种面向领域自然语言需求的形式化需求模型生成方法研究. 小型微型计算机系统, 2021, 42(8): 1639–1648. [doi: 10.3969/j.issn.1000-1220.2021.08.012]

[18] 霍曼, 邓中卫. 国外军用飞机航空电子系统发展趋势. 航空电子技术, 2004, 35(4): 5–10. [doi: 10.3969/j.issn.1006-141X.2004.04.002]

[19] 石娇洁. 基于四变量模型的系统安全性建模与分析方法 [硕士学位论文]. 南京: 南京航空航天大学, 2016.

[21] 张漾, 胡军, 王立松, 康介祥, 王辉, 高忠杰. 一种面向SCR需求模型的形式化验证方法研究. 小型微型计算机系统, 2022, 43(1): 193−202.

[43] 杨志斌, 胡凯, 赵永望, 马殿富, Bodeveix JP. 基于时间抽象状态机的AADL模型验证. 软件学报, 2015, 26(2): 202−222. http://www.jos.org.cn/1000-9825/4776.htm [doi: 10.13328/j.cnki.jos.004776]

[44] 王飞, 杨志斌, 黄志球, 周勇, 刘承威, 章文炳, 薛垒, 许金淼. 基于限定自然语言需求模板的AADL模型生成方法. 软件学报, 2018, 29(8): 2350−2370. http://www.jos.org.cn/1000-9825/5530.htm [doi: 10.13328/j.cnki.jos.005530]

胡军(1973－), 男, 博士, 副教授, CCF 专业会员, 主要研究领域为模型驱动的系统安全性分析, 软件验证.

康介祥(1969－), 男, 研究员, 主要研究领域为航空电子技术, 软件工程, 嵌入式操作系统.

吕佳润(1998－), 男, 硕士生, CCF 学生会员, 主要研究领域为需求建模与分析.

王辉(1979－), 女, 研究员, 主要研究领域为软件工程, 软件验证.

王立松(1969－), 男, 博士, 副教授, 主要研究领域为航空电子安全分析, 分布式环境中的数据管理, 形式化方法, 基于模型的安全分析, 无线传感网络.

高忠杰(1982－), 男, 研究员, 主要研究领域为机载显示控制系统开发.