基于IMA平台的嵌入式软件设计模型仿真及实时性分析方法

孙 磊 杨海燕 吴 际
(北京航空航天大学计算机软件与理论系 北京 100191)

摘 要 如何确保机载软件满足其实时性需求一直是一个引人关注的研究问题。根据工业界的报告结果,缺陷发现得越早,用于修复缺陷以提高机载软件不超时的可能性的代价就越小。对于运行在由ARINC653标准所描述的综合模块化航电系统(IMA)上的机载软件可采用以下方法:将机载软件的设计模型(UML模型)转化成仿真模型(Simulink模型),通过在Simulink平台上运行仿真模型来发现潜在的实时性问题。由于机载软件可能与IMA平台之间具有大量的交互(例如接口层和操作系统层)用来申请使用资源或者与其它的应用程序进行通讯,因此设计了一个仿真内核来仿真IMA平台的行为。最后,使用一个工业案例来论证上述方法的有效性。

关键词 机载软件,实时性,综合模块化航电,模型转换,Simulink仿真
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.12.021

Simulation and Real-time Analysis for Embedded Software Design Model with Consideration of Integrated Modular Avionics Platform

SUN Lei YANG Hai-yan WU Ji
(Department of Computer Software and Theory, Beihang University, Beijing 100191, China)

Abstract How to ensure an avionics software satisfies its real-time requirement is always a hot research problem. According to the results reported in industry, the earlier the defects found, the less cost to fix them to improve the chances of not missing deadlines. For avionics software running on integrated modular avionics (IMA) which is standardized by the ARINC653, people can use the following method. The design model (in UML) of avionics software is transformed into simulation model (in Simulink), and the potential real time problems are investigated by executing the simulated models in the platform of Simulink. Since avionics software may have plenty of interactions with IMA platform (i.e., interface layer and operating system layer) to apply resources to use, or to communicate with other applications, a respective simulation module was designed to simulate the behavior of IMA platform. An industrial case study was used to demonstrate the effectiveness of the proposed approach.

Keywords Avionics software, Real-time, IMA, Model transformation, Simulink simulation

1 引言

航空电子系统(简称航电系统)是安全关键系统(Safety Critical System),实时性是该系统的一个重要属性。本文研究的对象是部署于综合模块化航电(Integrated Modular Avio-nics, IMA)平台上的航电应用软件。IMA系统是对应用软件、嵌入式操作系统和底层硬件集成体的统称。IMA平台的规范和特性在ARINC653[1]标准中进行了详细描述。

目前,对机载软件实时性的测试主要集中在编码的后期,通过代码测试或者仿真实验的手段,根据得到的结果来改善设计方案,如此往复。但是软件实时性问题发现得越晚,修复代价就越大,越临近开发后期,发现软件中潜在实时性问题的困难也越大。本文提出一种在软件设计阶段发现潜在实时性问题的方法,把软件设计模型转化为仿真模型,并开展仿真实验。

本文第1节介绍涉及的核心概念和研究问题;第2节详细描述本文研究的方法;第3节结合具体的飞控软件案例来分析本文研究方法;最后对研究结果进行总结。

2 基于模型仿真的实时性能分析

如图1所示,本文研究的方法有2个输入和3个输出。其中软件设计模型基于UML定义了设计逻辑,应用配置文件则按照XML规范提供了关于该应用软件未来集成到IMA系统的相关配置。基于ATL(ATLAS Transformation Language)[4]设计相应的模型转换规则,从而转换得到符合Simulink模型语言规范的仿真模型。更进一步,在

到稿日期:2015-02-10 返修日期:2015-03-17
孙 磊(1989—),男,硕士,主要研究领域为软件建模仿真,E-mail:sunlei198911@163.com;杨海燕(1975—),女,讲师,主要研究领域为软件可靠行、软件测试,E-mail:yhy@buaa.edu.cn;吴 际(1974—),男,副教授,主要研究领域为软件安全性与可靠性、嵌入式软件设计与验证、软件测试。

Simulink 仿真工具中，根据一定的仿真策略运行转换生成的仿真模型，提取符合实时性约束的成功运行场景和不符合实时性约束的失败运行场景，并对两类运行场景进行差异分析。

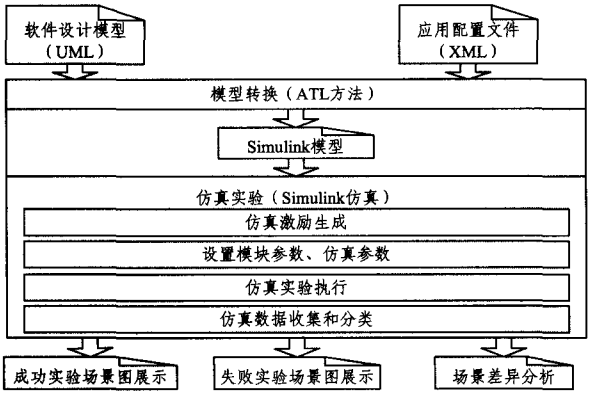

图 1　方法框架

2.1　面向 IMA 平台规范的仿真系统设计

AIRNC653 标准[1]制定了位于应用程序和 RTOS(Real-Time Operating System)之间的 APEX 接口。目前诸多厂商都发布了支持 APEX 接口的操作系统，风河开发的 VxWorks[5,6]是其中得到广泛应用的系统之一。

本文所设计的仿真系统架构包括两层。下层是符合 ARINC653 标准的仿真模块 Library 层，为上层的仿真模型提供包括分区、分区调度、进程调度、健康监控以及分区通讯等服务。上层是待分析的应用软件仿真模型，包括被仿真的应用 SUS(System Under Simulation)、仿真激励模块、故障注入模块、仿真数据收集模块。本文关注 SUS 在逻辑上部署于分区模块中，分区调度模块“调度”分区模块。SUS 由模型转换自动生成，应用模块层的其它模块(仿真激励模块、故障注入模块、仿真数据记录模块)具有一定的通用性，但是需要针对 SUS 模块进行一定的配置。ARINC653 模型库中的模块与 SUS 模块没有逻辑依赖关系，具有通用性。

2.2　基于模型转换的 SUS 模块自动生成

在 MDE(Model-Driven Engineering)领域，模型转换是一种重要的技术手段，可以把相关信息从一个模型(源模型)提取转换生成到另外一个模型(即目标模型)。为了保证转换的严谨性，OMG 组织提出了在元模型层次定义和实现转换规则，目前已有多种语言和工具支持这种转换方法，如 QVT[7]和 ATL[4,8]等。

本文研究的源模型是基于 UML2.0[9]的软件设计模型以及符合 XML 规范的配置信息，目标模型是符合 Simulink[10] V7.9 的仿真模型。目前官方并没有发布 Simulink 模型的标准元模型。本文针对实时性仿真分析，对仿真模型涉及的要素进行整理，提取出 Simulink 元模型，如图 2 所示。另外，XML 文件提供了平台操作系统的配置信息，包括分区、任务的基本信息，元模型结构如图 3 所示。本文直接使用了 OMG 官方所提供的 UML2.0 标准元模型，限于篇幅，不再介绍。本文使用 ATL 来设计模型转换，ATL 转换引擎使用所设计的转换规则，把输入的 UML 设计模型和配置文件生成相应的 Simulink 仿真模块。本文研究关注 UML 模型中的类图、组件图、序列图和活动图。

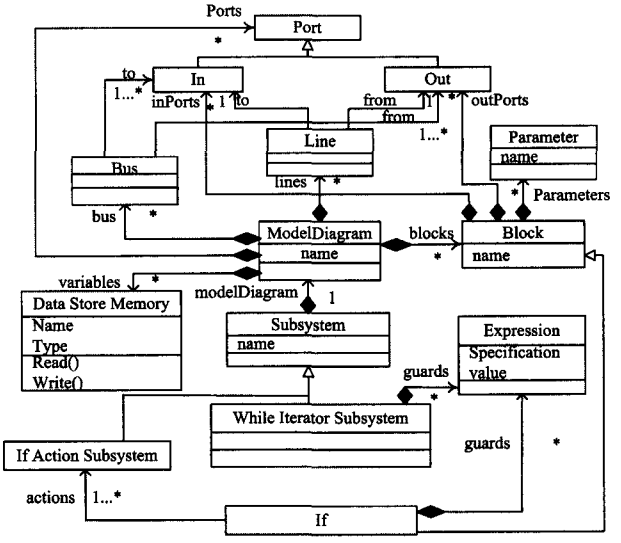

图 2　Simulink 元模型结构

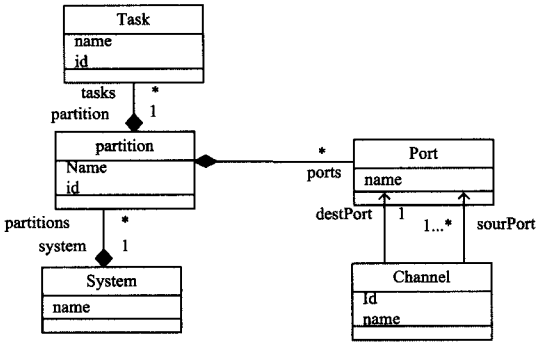

图 3　应用配置文件元模型结构

为了论述方便，设输入和输出分别为 I 和 O，则模型转换规则可定义为二者之间的映射，即 $R: I \rightarrow O$。本文设计了 3 类转换规则：①一对一转换规则 $I_i \rightarrow O_j$；②一对多转换规则 $I_i \rightarrow O_j + \cdots + O_k$；③多对一转换规则 $I_i + \cdots + I_j \rightarrow O_j$。本文共设计了 20 条转换规则，每一类规则选取一条，如表 1 所列。由于表格空间有限，有的模型名称使用了简称，如 SD(Sequence Diagram)。

表 1　转换规则整理表

规则类别	规则编号	输入模型元素	输出模型元素
①	R1	System	ModelDiagram
②	R3	Channel	Subsystem, Bus
③	R5	Task, SD (Task. name= SD. name)	Subsystem

对于上述这 20 条规则，首先对每一条规则单独做了测试，类似于单元测试，保证每条规则的正确转换。最后，对这些规则整体做了集成测试。通过这些测试，确保其正确性。对于输入的 UML 模型，转换规则覆盖了序列图、类图以及活动图，其中序列图表示任务流程，由类图中类的对象完成其中的交互，类图中类的操作的逻辑又由活动图来表述。输入模型和输出模型的覆盖情况反映了转换规则的全面性。

3　工业案例分析

3.1　案例简介

本文研究的方法在一个真实的工业案例中得到了验证。

该案例为自动驾驶仪系统(Auto Pilot,AP),用于控制飞机的自动飞行和辅助驾驶员操纵飞机。在每个周期,传感器都会检测飞机的飞行状态数据(如飞行姿态角数据)和环境状态数据(如大气数据),当飞机偏离预定姿态时,计算机根据传感器所获得的数据,通过一定算法算出修正舵偏量,伺服系统将舵面操纵到所需位置、航向角和速度。此外,该系统还具有多种自检功能,如飞行中的自检功能等。系统在分区层次采用基于时间片的调度策略,该系统的功能由 5 个任务来实现,分别部署于两个分区 P1 和 P2,其中 P1 分区分配的时间为 14ms,部署的任务为 T1、T2 和 T3;P2 分区分配的时间为 6ms,部署的任务为 T4 和 T5。

3.2 仿真模型转换和仿真系统配置

通过应用本文所设计的 20 个转换规则,得到 Simulink 仿真模型,如图 4 所示。在开展仿真实验之前,对得到的Simulink仿真系统进行了必要的仿真参数配置,包括组件故障概率和组件运行预期时长。其中组件的故障名称、故障处理策略如表 2 所列。

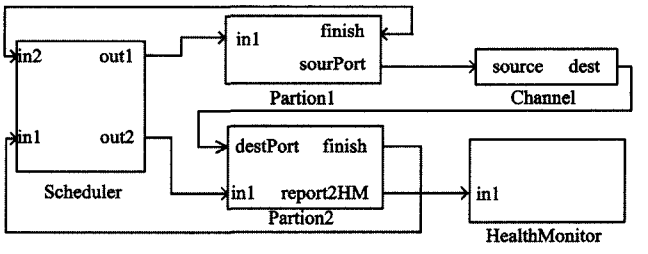

图 4 Simulink 模型顶层图

表 2 给出了该案例所采用的故障处理策略,故障分为瞬时故障和永久故障。表 2 中涉及了 9 个故障,举例说明故障的注入过程,如采样组件的 InvalidSampler_fault 故障,在从采样组件获得数据之后,会对其进行一个判断的过程,如果某个数据超出了正常范围,表现为不可能出现的值,说明采样组件出现故障,故障注入则是将采样组件获得的某个数据值进行修改,使其判定为出现故障;继而引发表 2 的故障处理策略。

表 2 组件及其处理的故障信息

组件名称	故障名称	分区	瞬时故障处理	永久故障处理
同步	Sync_funcfault	P1,P2	报告、重试	关闭 AP 系统、切换人工驾驶
采样	InvalidSampler_fault	P1	报告、重试、使用上周期数据	关闭 AP 系统、切换人工驾驶
交叉传输	CrossData_funcfault,	P1	报告、重试、使用上周期数据	关闭 AP 系统、切换人工驾驶
输入输出表决	Invoter_fault	P1	报告、使用上周期数据	关闭 AP 系统、切换人工驾驶
控制律计算	Comput_fault	P1	报告、使用上周期数据	关闭 AP 系统、切换人工驾驶
故障审计	Faulthandler_fault	P2	报告、使用上周期数据	关闭 AP 系统、切换人工驾驶
IFBIT	IFBIT_fault	P2	报告、使用上周期数据	关闭 AP 系统、切换人工驾驶
模态选择	Modeselection_fault	P1	报告、使用上周期数据	关闭 AP 系统、切换人工驾驶
作动器控制	Actuator_fault	P1	报告、使用上周期数据	关闭 AP 系统、切换人工驾驶

3.3 仿真实验结果分析

仿真实验设定为 3000 周期,共进行 100 次,所以各分区和任务被调度了 300000 次。实验中 P2 分区没有发生超时现象,这里不作分析;P1 分区共发生了 400 次超时现象。本文分别抽取一段失败场景和一段成功场景,如图 5 和图 6 所示。

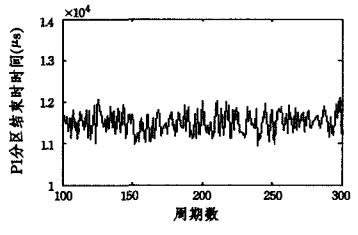

图 5 P1 分区成功场景图展示

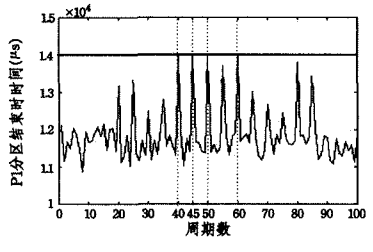

图 6 P1 分区失败场景图示

从失败实验场景的展示图上可以看出,P1 分区在第 40、45、50、60 周期时发生了超时现象,达到了 14μs 的截止时间,其执行被分区调度模块终止。P1 分区失败和成功场景的差异分析如表 3 所列。表中的"T1"表示任务 T1,"重试"表示发生故障采取了重试操作,"T1+T2 重试"表示在 T1 和 T2 任务中都发生了故障并且都采取了重试操作,其它同理。可以看出,失败场景的 4 个周期集中发生的 InvalidSamplerData_fault、CrossData_fault 较多。可以分析得出 InvalidSamplerData_fault、CrossData_fault 故障发生导致采样组件和交叉传输组件重试对系统的时间性能影响较大,在一定的组合情况下甚至发生"超时"现象。该结果提示设计人员需要考虑调整故障的处理策略,以减少故障处理所消耗的时间资源;或者考虑减少故障发生概率,这将对 IMA 平台的硬件和系统资源提出更高的要求。

表 3 失败场景与成功场景的差异分析

失败场景(周期数)	差异分析结果
40	CrossData_fault(T1+T2 重试)
45	InvalidSamplerData_fault(重试)、CrossData_fault(T1 重试)
50	InvalidSamplerData_fault(重试)、CrossData_fault(T1+T2 重试)
60	Sync_funcfault(T1 重试)、InvalidSamplerData_fault(重试)、CrossData_fault(T1+T2 重试)

结束语 本文提出了一种模型转换方法,并在仿真系统中通过对 IMA 平台关键特征的模拟来实现对机载嵌入式软件的实时性仿真。首先,在仿真系统中模拟实现了 IMA 平台的时间分区和空间分区能力,在此基础上提供了进程通讯、分

(下转第 135 页)

实验结果表明，当实时数据被不断加载时，本文采用的基于动态镜像的实时数据存取技术在不同的事务更新频率下，查询效率可靠，如果给予动态存储区域更大的内存空间则性能会更好。

结束语 本文提出的基于动态镜像的实时数据仓库预存取技术是一种空间换时间的技术。通过在实时数据仓库外部创建一个由多个镜像组成的实时存储区域，有效地解决了实时数据仓库的查询竞争问题，并且提升了实时数据查询结果的精度。另一方面，在海量数据的背景下，如何进一步提高实时数据查询的精度，以及保证实时数据新鲜度，将在后续的研究中展开。

参 考 文 献

[1] Mohammad R, Klanmehr K, Alhajj R, et al. Data warehouse architecture and design[C]// Proc. of 2008 IEEE Int'l Conf. on Information Reuse and Integration. 2008:58-63

[2] Vassiliadis P, Simitsis A. Near real time ETL[M]// New Trends in Data Warehousing and Data Analysis. 2009:1-31

[3] White C. Intelligent business strategies: Real-time data warehousing heats up[D]. DM Review, 2012

[4] 徐俊刚 裴莹. 数据 ETL 研究综述[J]. 计算机科学, 2011, 38(4)
Xu J, Pei Y, Overview of data extraction, transaction and loading [J]. Computer Science, 2011, 38(4)

[5] Heman S, Zukowski M, Nes N J, et al. Positional update handling in column stores[C]// SIGMOD. 2010:543-554

[6] Kuo T W, Kao Y T, Kuo C F. Two-version based concurrency control and recovery in real-time client/server databases[J]. IEEE Transaction on Computer, 2003, 52(4):506-524

[7] Stonebraker M, Abadi D J, Batkin A, et al. C-store: a column-oriented DBMS[C]// VLDB. 2005:553-564

[8] Langseth J. Real-time data warehousing: challenges and solutions [OL]. http://dssresources.com/papers/features/langseth/langsth02082004.html

[9] Ankorion I. Change Data Capture-Efficient ETL for Real-Time BI[J]. Article published in DM Review Magazine, January 2005

[10] Italiano I C, Ferreira J E. Synchronization Options for Data Warehouse Designs[J]. IEEE Computer Magazine, 2006, 8(4): 167-172

[11] Lin Z, Yang D, Song G, et al. Dealing with Query Contention Issue in Real-time Data Warehouses by Dynamic Multi-level Caches[C]// Seventh International Conference on Computer and Information Technology. 2012

[12] TPC-H decision support bench mark, Transaction Processing Council[OL]. http://www.tpc.com

[13] 朱阅岸，张延松，周烜，等. 一个基于三元组存储的列式 OLAP 查询执行引擎[J]. 软件学报, 2014, 25(4):753-767
Zhu Y A, Zhang Y S, Zhou X, et al. Column-Oriented query execution engine for OLAP based on triplet[J]. Journal of Software, 2014, 25(4):753-767

[14] Lahman S. The Lahman Baseball database[OL]. http://www.baseball.com

(上接第 97 页)

区通讯和调度等服务；其次，有效区分了应用软件执行所消耗的时间和与 IMA 平台交互所消耗的时间；最后，通过故障注入模拟了应用软件执行过程中平台所出现的相应故障，使得可以分析故障处理对实时性的影响，并开展了成功场景和失败场景的差异分析。为了验证本文研究的有效性，在与工业合作伙伴的合作过程中使用了一个真实的工业项目进行案例研究。结果表明，本文研究成果可有效地把 UML 设计模型转换为仿真模型，并支持开展相应的仿真实验。目前，该方法仅适用于采用 ARINC653 标准的机载嵌入式软件系统。本文后续研究将进一步把 IMA 物理平台带入到仿真环境，支持运行平台硬件在环的半实物仿真实验。

参 考 文 献

[1] Airlines Electronic Engineering Committee[S]. Avionics Application Software Standard Interface. Aeronautical Radio, 1997

[2] Wilson A, Preyssler T. Incremental Certification and Integrated Modular Avionics[J]. Aerospace and Electronic Systems Magazine, IEEE, 2009, 24(11):10-15

[3] Parkinson P, Kinnan L. Safety-critical software development for integrated modular avionics[J]. Embedded System Engineering, 2003, 11(7):40-41

[4] Jouault F, Allilaire F, Bézivin J, et al. ATL: a QVT-like transformation language[C]// Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications. ACM, 2006:719-720

[5] 罗国庆. VxWorks 与嵌入式开发[M]. 北京：中国机械工业出版社，2003

[6] http://www.windriver.com/products/platforms/safety_critical_arinc_653/index.html

[7] OMG. QVT Query/views/Transformations RFP, OMG Document ad[OL]. http://www.omg.org/cgi-bin/doc? ad/2002-4-10

[8] Van Amstel M, Bosems S, Kurtev I, et al. Performance in model transformations: experiments with ATL and QVT[M]// Theory and Practice of Model Transformations. Springer Berlin Heidelberg, 2011:198-212

[9] OMG. Unified Modeling Language Specification, Version 2.0, OMG Document formal [OL]. http://www.omg.org/spec/UML/2.0

[10] Vanderperren Y, Dehaene W. From UML/SysML to Matlab/Simulink: current state and future perspectives[C]// Proceedings of the Conference on Design, Automation and Test in Europe(DATE 2006). Munich, Germany, 2006:93-93

[11] 耿素云，屈婉玲. 离散数学[M]. 北京：清华大学出版社，2004