ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.5, May 2010, pp.899−915
doi: 10.3724/SP.J.1001.2010.03700

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

复杂嵌入式实时系统体系结构设计与分析语言:AADL*

杨志斌[1+], 皮 磊[2], 胡 凯[1], 顾宗华[3], 马殿富[1]

[1](北京航空航天大学 计算机学院,北京 100191)

[2](Toulouse Institute of Computer Science Research, Toulouse, France)

[3](浙江大学 计算机科学与技术学院,浙江 杭州 310027)

AADL: An Architecture Design and Analysis Language for Complex Embedded Real-Time Systems

YANG Zhi-Bin[1+], PI Lei[2], HU Kai[1], GU Zong-Hua[3], MA Dian-Fu[1]

[1](School of Computer Science and Engineering, BeiHang University, Beijing 100191, China)

[2](Toulouse Institute of Computer Science Research, Toulouse, France)

[3](College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China)

+ Corresponding author: E-mail: kenney@cse.buaa.edu.cn

Yang ZB, Pi L, Hu K, Gu ZH, Ma DF. AADL: An architecture design and analysis language for complex embedded real-time systems. *Journal of Software*, 2010,21(5):899−915. http://www.jos.org.cn/1000-9825/3700.htm

Abstract: This paper firstly presents a summary of AADL (architecture analysis and design language), including its progress over the years and its modeling elements. Then, it surveys the research and practice of AADL from a model-based perspective, such as AADL modeling, AADL formal semantics, model transformation, verification and code generation. Finally, the potential research directions are discussed.

Key words: complex embedded real-time system; system architecture; model driven; AADL (architecture analysis and design language)

摘 要: 首先归纳了 AADL(architecture analysis and design language)的发展历程及其主要建模元素.其次,从模型驱动设计与实现的角度综述了 AADL 在不同阶段的研究与应用,总结了研究热点,分析了现有研究的不足,并对 AADL 的建模与分析工具、应用实践进行了概述.最后,探讨了 AADL 的发展与研究方向.

关键词: 复杂嵌入式实时系统;系统体系结构;模型驱动;AADL(architecture analysis and design language)

中图法分类号: TP311 **文献标识码**: A

复杂嵌入式实时系统广泛应用于航空电子、航天器、汽车控制等领域,这些系统具有资源受限、实时响应、

* Supported by the National High-Tech Research and Development Plan of China under Grant Nos.2007AA01A127, 2006AA01Z19A (国家高技术研究发展计划(863)); the Aviation Science Foundation of China under Grant Nos.20081951027, 2007ZC51032, 20060151003 (航空科学基金); the RGC CERG of Hong Kong of China under Grant No.613506 (香港研究资助局面上项目); the TOPCASED Project in France (法国 TOPCASED 项目)

Received 2009-01-08; Revised 2009-04-27; Accepted 2009-07-09

容错、专用硬件等特点,对实时性、可靠性等性质有较高的要求,文献[1]称其为性能关键系统(performance critical systems,简称 PCS).由于计算精度、实时响应的要求,这类系统变得越来越复杂,如何设计与实现高质量的复杂嵌入式实时系统,并有效控制开发时间与成本,是学术界和工业界共同面临的难题.

模型驱动开发方法(model driven development,简称 MDD)能够在早期阶段对系统进行分析与验证,有助于保证系统的质量属性,并有效控制开发时间与成本.而质量属性是由系统体系结构决定的[1].因此,基于体系结构模型驱动(model-based architecture-driven)的设计与开发方法成为复杂嵌入式系统领域的重要研究内容.其中一个重要的方面就是研究合适的体系结构描述语言.

常用的体系结构描述语言主要有 UML(unified modeling language)和 ADL(architecture description language).UML 侧重描述系统的软件体系结构,为了支持嵌入式实时系统的非功能属性分析,OMG(Object Management Group)先后定义了 UML Profile for SPT(schedulability, performance, and time,简称 SPT)[2],UML Profile for Qos/FT(quality of service and fault tolerance,简称 Qos/FT)[3]以及 UML Profile MARTE(modeling and analysis of real-time and embedded systems)[4],它们继承了 UML 的多模型多分析方法,因此模型之间可能存在不一致性;而 C2,Darwin,Wright,Aesop,Unicon,Rapide 等 ADL 都是通用领域的软件体系结构描述语言,难以满足软硬件协同设计、实时响应、资源受限等特定需求;MetaH 是面向航空电子系统的 ADL,可以用于嵌入式实时系统体系结构描述与分析,但 MetaH 在支持运行时体系结构描述、可扩展、与其他 ADL 兼容以及复杂系统设计等方面有所欠缺.

2004 年,美国汽车工程师协会 SAE(society of automotive engineers)在 MetaH,UML 的基础上,提出嵌入式实时系统体系结构分析与设计语言 AADL(architecture analysis & design language)[5],并发布为 SAE AS5506 标准,目的是提供一种标准而又足够精确的方式,设计与分析嵌入式实时系统的软、硬件体系结构及功能与非功能性质,采用单一模型支持多种分析的方式,将系统设计、分析、验证、自动代码生成等关键环节融合于统一框架之下.AADL 具有语法简单、功能强大、可扩展的优点.由于具有广阔的应用前景,AADL 得到了欧美工业界,特别是航空航天领域(如 Airbus,Lockheed Martin,Rockwell Collins,Honeywell,Boeing)的支持;CMU(Carnegie Mellon University),MIT(Massachusetts Institute of Technology),UIUC(University of Illinois at Urbana-Champaign),Pennsylvania 大学,NASA 以及法国 IRIT(Toulouse Institute of Computer Science Research),INRIA,Verimag 等研究机构对 AADL 展开了深入研究与扩展.TOPCASED[6],SPICES[7],AVSI-SAVI[8]等是欧美工业界和学术界共同参与的 AADL 重大研究项目,涉及 AADL 标准扩展、建模工具、形式语义、验证、可靠性分析、可调度分析以及自动代码生成等方面的研究,这些也是目前 AADL 的研究热点.

AADL 被认为是基于模型驱动的嵌入式实时系统设计与实现的基础,但 AADL 还有很多问题需要进一步研究与完善.本文试图从模型驱动设计与开发的角度,综述近几年 AADL 的研究成果,剖析一些重要概念,如执行模型(execution model),总结研究热点以及分析现有研究的不足.第 1 节简介 AADL 的发展历程及其建模元素.第 2 节分述 AADL 语言的研究、基于 AADL 的建模方法、AADL 模型的验证与分析以及基于 AADL 模型的自动代码生成等内容.第 3 节介绍 AADL 现有的建模与分析工具.第 4 节概述 AADL 的应用实践.最后讨论 AADL 可能的研究发展方向.

1 AADL 简介

1.1 AADL发展历程

AADL 创始人 Peter H. Feiler 和 Bruce A. Lewis 总结了 AADL 的发展历程[1],提出 AADL 是在 MetaH,ADL 以及商业建模语言 UML 和 HOOD(hierarchical object-oriented design)[9]的基础上发展起来的.1991 年,在美国 DARPA 计划 DSSA(domain specific software architectures)项目支持下,Honeywell 实验室提出体系结构描述语言 MetaH,专用于航空电子、飞行控制等系统的体系结构描述与分析.到 1999 年,学术界和工业界对 MetaH 进行了大量研究与应用,如支持多处理器系统建模、Ada95 和 POSIX 中间件配置、MetaH-ACME 的转换、可靠性建模等.2001 年,SAE 提出基于 MetaH 定义一个航空电子体系结构描述语言标准,即 Avionics Architecture

Description Language(AADL),支持描述标准的航空电子控制与数据流机制及实时、容错、安全等非功能性质.2004年,AADL标准化得到了进一步的扩展.SAE,CMU,Honeywell等共同提出建立嵌入式实时系统体系结构描述语言标准,并正式命名为 Architecture Analysis & Design Language(AADL).2004 年 11 月,发布了 AADL1.0 版本,以文本的形式给出核心语言的语法和语义.2006 年发布了一些扩展附件(annex),如 Graphical AADL Notation Annex[10],AADL Meta model and XML/XMI Interchange Format Annex[10],Error Model Annex[10], Behavior Annex[11],UML Profile Annex.2009 年 1 月发布了 AADL 2.0 版本[12].

1.2 AADL建模元素

AADL 通过构件、连接等概念描述系统的软、硬件体系结构;通过特征、属性描述系统功能与非功能性质;通过模式变换(mode change)描述运行时体系结构演化;通过用户定义属性和附件支持可扩展;对于复杂系统建模,AADL 通过包(package)进行组织.AADL 提供了 3 种建模方式:文本、XML 以及图形化.如图 1 所示,本文以汽车巡航控制系统(cruise control system)的 AADL 模型为例,介绍 AADL 的基本建模元素.

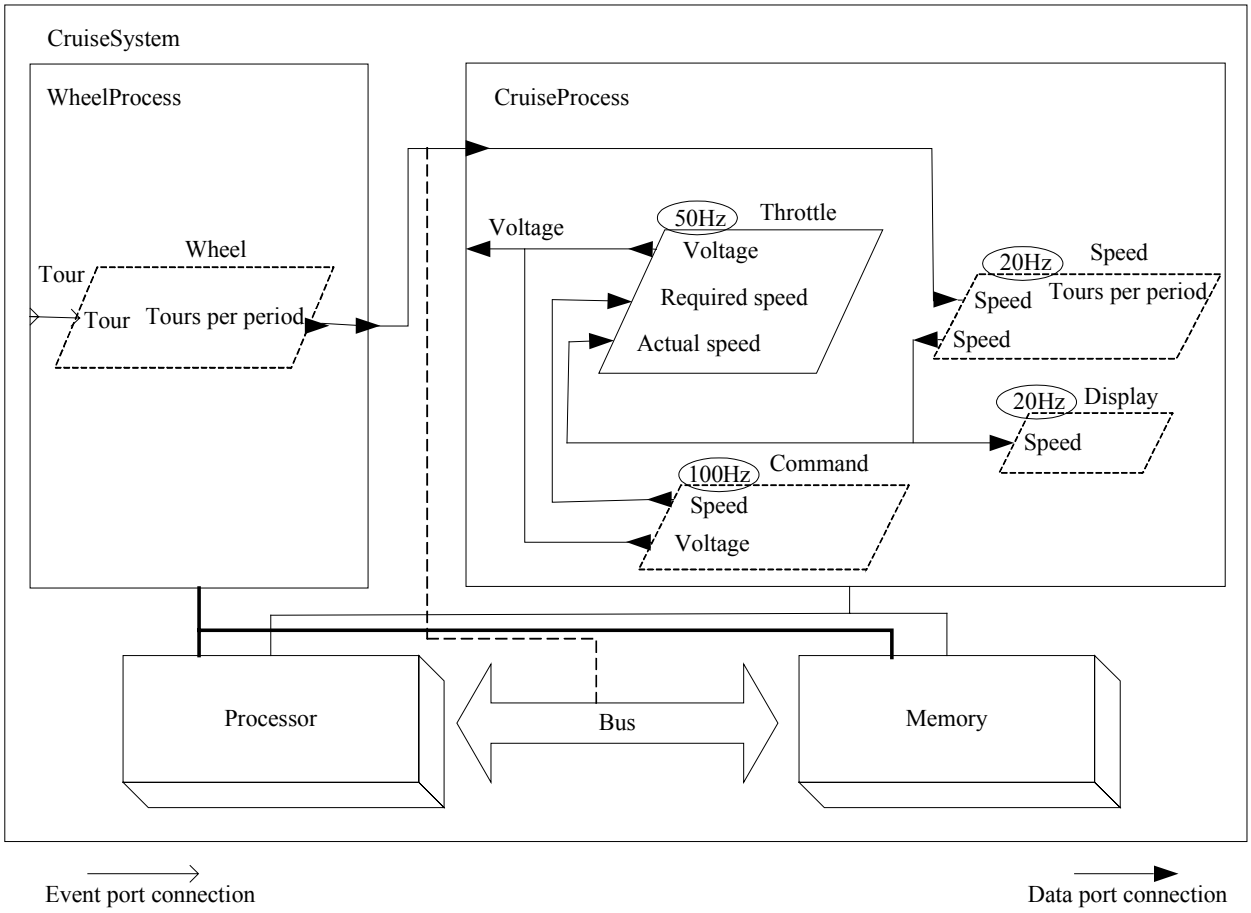


Fig.1 Graphic AADL model of an automotive cruise-control system

图 1 汽车巡航控制系统的 AADL 图形化模型

(1) 构件种类

AADL 定义了 3 类构件:软件构件、执行平台构件以及系统构件.软件构件用于软件体系结构建模,包括数据(data)、线程(thread)、线程组(thread group)、进程(process)、子程序(subprogram)构件;执行平台构件用于硬件体系结构建模,包括处理器(processor)、虚拟处理器(virtual processor)、存储器(memory)、总线(bus)、虚拟总线(virtual bus)、外设(device)构件;系统构件组合所有的构件,层次化地建立系统的体系结构.在图 1 中,采用系统构件(CruiseSystem)、进程构件(WheelProcess,CruiseProcess)、线程构件(Wheel,Throttle,Speed,Command, Display)、处理器构件、存储器构件以及总线构件构造系统的体系结构模型.

(2) 构件描述

AADL 构件被定义为两部分:类型(component type)和实现(component implementation).一个构件拥有一个类型以及对应的 0 个、1 个或多个实现.构件类型描述对外的功能接口(输入输出端口等).构件实现则描述构件的内部结构(子构件、连接等),例如,进程构件 CruiseProcess 由 4 个线程子构件及其连接来实现.构件类型和实现都是可继承的.

(3) 构件特征

特征(feature)是构件类型定义的一部分,用于描述构件的对外接口,主要包括 4 类:端口、子程序、参数以及子构件访问.端口用于定义构件之间的数据、事件交互接口,分为数据、事件、数据事件端口,图 1 中主要采用了数据和事件端口进行交互.子程序用于定义子程序共享访问接口,分为子程序访问者(required subprogram access)和子程序提供者(provides subprogram access),前者表示需要访问其他构件内部的子程序,后者表示提供子程序给其他构件来访问,可以支持远程子程序调用.参数用于定义子程序被访问时输入、输出的合法数据类型.子构件访问分为数据构件访问和总线构件访问,前者用于共享数据或共享资源描述,后者用于硬件平台构件之间的连接.

(4) 连接、流

AADL 采用连接(connection)来描述构件之间的交互行为,与构件特征对应,AADL 支持 3 种连接方式:端口连接、参数连接及访问连接.端口连接用于描述并发执行构件之间的数据与控制交互;参数连接描述一个线程构件访问的所有子程序的参数所形成的数据流;访问连接又分为数据访问连接、总线访问连接以及子程序访问连接,分别描述数据共享、总线共享以及子程序共享.

连接是点对点的,为了简化对体系结构的分析过程,AADL 引入了流(flow)的概念,用于描述系统中信息传输的逻辑路径.流的完整描述包括流规约(flow specification)和流实现(flow implementation).流规约在构件类型中定义,包含外部可见的源结点(flow source)、目标结点(flow sink)及路径(flow path),源结点和目标结点分别是构件的特征,路径则是构件中从一个特征到另一个特征的连接.流实现在构件实现中定义,包括构件中流路径的具体实现或整个系统的端到端流(end-to-end flow)定义.端到端流的描述可以用于支持端到端延迟分析和可靠性分析.

(5) 构件属性

属性(property)用于描述体系结构中的约束条件,即非功能属性约束,进而支持验证与分析系统的可靠性、安全性、可调度性等性质,如子程序的执行时间、线程的周期、数据或事件端口的等待队列协议、安全层次等.AADL 提供了标准的属性集,用户也可以根据需要定义新的属性.属性和特征的区别在于,特征主要是描述构件功能接口,而属性则是描述系统非功能性质的约束.

(6) 模式

AADL 通过模式(mode)来描述运行时体系结构的动态演化.模式就是系统或构件的操作状态,如汽车巡航控制系统可能包括初始化、人工控制、自动巡航等模式.它们对应了系统功能行为的不同配置,模式变换体现系统体系结构的变化,能够描述体系结构重构及容错等需求.

(7) 扩展附件

当定义新的属性不能满足用户需要时,AADL 引入了附件的概念.它拥有独立的语法和语义,但必须与 AADL 核心标准保持语义一致.如故障模型附件(error model annex)[10],支持构件、连接的故障事件、故障概率等属性建模;行为附件(behavior annex)[11]增强了 AADL 对构件实际功能行为的详细描述能力,以更好地支持功能行为验证和自动代码生成.

另外,时间正确性是实时系统重要的特征,不仅与 AADL 属性中定义的时间约束(时限、最坏执行时间等)有关,而且与调度算法、调度属性有关.AADL 支持描述周期、非周期、偶发等任务模型,支持抢占与非抢占式调度策略,支持多种固定优先级、动态优先级调度算法,如 RM(rate monotonic),DM(deadline monotonic),EDF(earliest deadline first)等.这些调度算法定义在处理器构件的属性中.

2 AADL 的研究现状

2.1 AADL语言的研究

为了适应不同的应用需求,AADL 语言本身还需要进一步完善和扩展.AADL 语言扩展及其语义的形式化目的是为了更好地支持系统体系结构建模与分析.

2.1.1 AADL 语言扩展

AADL 提供两种扩展方式:引入新属性或符号以及子语言(sub language)扩展.前者与具体应用相关,允许用户和工具提供商为各种构件引入新的属性集,或专用于特殊分析的符号,如可调度分析工具 Cheddar[13],通过定义新的属性集扩展 AADL 对更复杂调度算法的支持;后者则更严格,需要提议、发展以及被核准,才能成为 AADL 语言的一部分,一般以 AADL 附件的形式给出,需要提供子语言的语法和语义.已经被核准的附件有[10]: Graphical AADL Notation Annex,AADL Meta-Model and Interchange Formats Annex,Language Compliance and Application Program Interface Annex,Error Model Annex 以及 UML Profile Annex.正在发展中的附件有:Behavior Annex[11],ARINC653 Annex.

下面对 AADL 扩展附件进行概述.

AADL 核心标准是文本形式的,Graphical AADL Notation Annex 为 AADL 定义了一系列图形符号,构件、端口通信、数据共享访问、子程序调用、模式变换、端到端的数据流、属性、附件等元素都能用图形符号表示.但 Behavior Annex 还没有图形化.目前,AADL 的主要建模工具 OSATE/TOPCASED[14],STOOD[15]都支持图形化建模,而且能够方便地与文本表示相互转换.

Meta Model and Interchange Formats Annex 定义了 AADL 元模型以及基于 XML 的模型交互格式.元模型定义了 AADL 语言的结构,也就是 AADL 模型的对象表示.这些对象表示采用 XML 标准的交互格式保存,以支持不同工具之间的互操作性.

自动代码生成是模型驱动开发的关键环节之一.Language Compliance and Application Program Interface Annex 为用户提供了 AADL 到 Ada,C 语言代码的转换规则.主要涉及线程、进程、数据等软件构件以及端口通信和子程序调用.

Error Model Annex 定义了构件和连接建立故障模型的声明规则及语义.故障模型也由两部分组成:类型(error model type)和实现(error model implementation).故障类型可以声明故障状态、故障事件以及故障传播等;故障实现则定义故障状态的变迁,即构件在故障事件发生和故障传播过程中,故障状态是如何变化的.故障状态变迁本质上是一个随机自动机,附件提供 Occurrence 属性定义故障的发生概率.Error Model Annex 可以与多种分析方法结合,如 MA(Markov analysis)、DD(dependency diagrams)、ETA(event tree analysis)、FTA(fault tree analysis)、随机 Petri 网等,从而对系统的可靠性、完整性、可用性以及可维护性进行定量分析.

系统行为描述与验证是非常重要的.文献[16]提出,系统行为不仅依赖于 AADL 构件和连接所描述的静态体系结构,而且依赖于运行时环境.因此 AADL 提供了执行模型[16,17]的概念来描述运行时环境,用于管理和支持构件的执行,分为同步和异步两种,包括构件分发(dispatch)、同步/异步通信、调度、模式变换等行为.AADL 核心标准的默认模式为带同步通信的抢占式调度,在这种模式下,执行模型和系统体系结构的结合能够保证系统行为的可预见性(predictability).线程、子程序构件是最小的执行和调度单元,AADL 标准仅描述两者的对外功能接口,而构件内部具体执行行为无法描述.这样就导致在异步执行模型情况下,系统行为的可预见性难以保证.为了更好地分析异步模式下的系统行为,法国 IRIT 提出了 Behavior Annex,对线程和子程序构件的具体行为进行详细描述.Behavior Annex 通过状态、变迁来描述构件行为,变迁可以定义触发条件及变迁后的动作,条件和动作主要包括接收/发送数据、子程序调用、异步访问、执行时间、延迟时间等,且通过层次、并发状态来支持更复杂的行为描述.目前,AADL 建模工具都支持 Behavior Annex 的描述.

我们认为 Behavior Annex 与执行模型有密切的关系[18].执行模型的分发机制定义了线程、子程序周期性或非周期性地读取数据、计算、发送数据,Behavior Annex 则是对计算状态内的执行行为进行详细刻画.也就是说,

执行模型定义了 Behavior Annex 何时执行、哪些数据被执行,而 Behavior Annex 处于构件内部,对线程、子程序构件的执行进行更精确的描述.因此,静态体系结构、执行模型及 Behavior Annex 组成一个完整的 AADL 模型.以线程构件为例,图 2 给出了三者的关系.

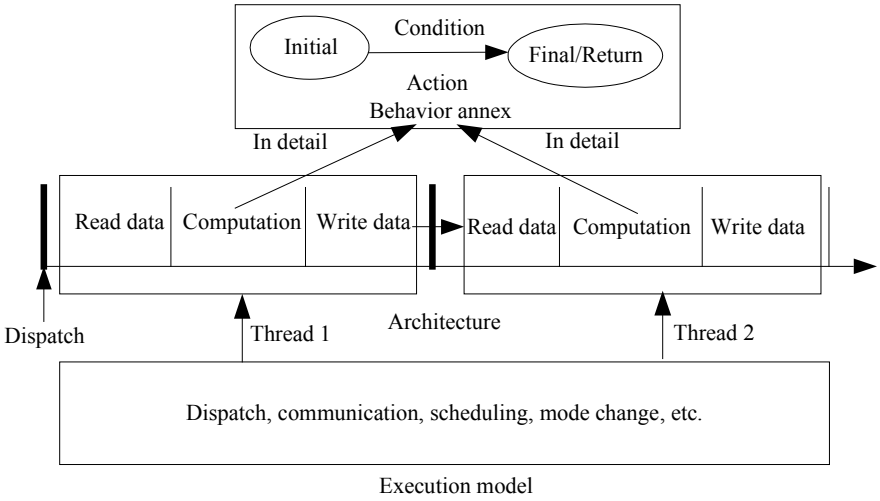


Fig.2 Structure, execution model and behavior annex

图 2 体系结构、执行模型及行为附件的关系

随着 AADL 在工业界的广泛应用,对 AADL 扩展的需求会更多.在航空航天领域,ARINC653 是重要的航空电子应用软件接口标准,文献[19,20]研究了 AADL 与 ARINC653 标准之间的结合.在自动控制领域,AADL 可以与该领域的 OSEK[21],AutoSAR[22],EAST-ADL[23]等标准结合.

2.1.2 AADL 形式语义

形式化方法能够对语义进行更精确的刻画,有助于体系结构验证与分析.体系结构、执行模型和行为描述构成一个完整的 AADL 模型,因此 AADL 语义也涉及这 3 个方面.AADL 采用混成自动机(hybrid automata)对线程、进程、处理器、虚拟处理器构件的执行状态和动作进行了语义描述,但对通信、模式变换、同步/异步访问等执行模型语义以及 Behavior Annex 语义没有形式化,主要采用自然语言和例子进行解释.因此,AADL 形式语义研究主要针对后两者的内容.

目前,AADL 形式语义描述主要采用转换的方式(translational semantics),大致可以分为两类:采用一种具有精确语义的形式语言来定义 AADL 语义,再依据语义进行转换;直接将 AADL 模型转换到另一种形式化模型.我们称前者为显式描述(explicit),可以将不精确的 AADL 语义形式化,能够更完整地描述语义,这种方式类似于操作语义.我们称后者为隐式描述(implicit),目的是为了直接使用语义模型的现有形式化分析工具,但这种方式有不足之处:仅仅假设转换的语义是一致的,语义描述可能不够精确;模型转换是依据 AADL 已有语义,而那些用自然语言和例子给出的语义不够精确,可能导致语义转换不够完整.

在显式语义描述方面,文献[24]采用 UML Marte 对同步执行模型中周期性线程之间基于数据端口的即时与延迟通信(immediate and delayed communication)机制进行了语义描述,UML Marte 具有显式的时间模型,能够同时描述逻辑时间和物理时间,方便定义线程的时间属性和通信协议.同时,AADL 的通信语义本质上就是实时任务的时钟约束关系,对系统调度和可调度分析有直接的影响.但 UML Marte 难以支持形式化验证和分析.文献[25]则对 AADL 数据端口通信语义进行了描述,提出端口连接可以看作对共享数据变量的访问,采用消息缓冲区模型来定义任务之间的交互,并提出了基于共享缓冲区的分析框架,对端口通信进行优化.文献[26]则采用 UML Marte 对异步执行模型中事件端口通信进行语义描述,事件端口都带有队列以支持异步访问,但这样可能会导致数据或事件的传输延迟,因此该文献基于语义描述对端到端的延迟进行了分析.文献[27,28]采用动作时序逻辑描述语言 TLA+(temporal logic of actions plus)对 AADL 执行模型的部分语义做了初步研究,包括端口通

信、共享变量的构件间通信、抢占式调度策略以及模式的语义,但 TLA+的模型检测工具分析能力不足.文献[29]采用 Fiacre 和时间抽象状态机(timed abstract state machine,简称 TASM)两种方法从不同角度对 AADL 同步执行模型进行语义描述,并进行分析与比较.Fiacre[30]是一种能够描述实时系统行为和时间的形式化中间语言,目的是为了支持多种高层建模语言(如 AADL,UML,SDL)到不同验证工具(如 CADP,TINA)的建模语言的转换.TASM[31]则是抽象状态机的实时扩展,能够支持时间、资源、同步、并发等行为的描述.文献[18]采用时间抽象状态机对 AADL 同步执行模型以及 Behavior Annex 的语义进行形式描述,并采用 UPPAAL[32]进行时间属性验证.

在隐式语义描述方面,法国 Verimag 实验室提出 AADL 到 BIP(behavior interaction priority)的语义转换[33],如 AADL 线程转换到 BIP 原子构件、数据端口连接转换到 BIP 的连接器,仅描述了 AADL 线程、进程、处理器构件以及 Behavior Annex 的部分语义,采用 Aldebaran 工具进行死锁、安全性的验证.BIP[34]是 Verimag 提出的实时系统建模语言,采用自动机的方式描述行为,并支持异构构件之间的组合.英国 Leicester 大学研究 AADL 到重写逻辑 Maude 语言的模型转换[35],支持行为验证,并实现了 MOMENT2-AADL 工具.文献[36]研究 AADL 转换到 IF 语言进行形式化验证,采用 Kermeta 元模型语言作为转换工具,支持静态结构、Behavior Annex、执行模型到 IF 的转换,但目前只进行了部分转换.IF 语言是 Verimag 实验室提出的实时系统描述语言,扩展了通信时间自动机来描述实时系统的功能与非功能行为[37].文献[38]提出 AADL 到时间进程代数 ACSR(algebra of communicating shared resources)[39]的语义转换,然后进行可调度性分析,但其分析工具 VERSA[40]只能支持简单的分析.文献[41]研究了 AADL 到实时演算(real time calculus,简称 RTC)[42]的语义转换,以支持端到端的时间延迟分析.文献[43]研究 AADL 到随机 Petri 网的转换.

模型转换是 AADL 语义描述的关键.文献[44]采用 B 语言和高阶逻辑 HOL(higher-order logic)形式刻画 AADL 元模型,以支持 AADL 模型转换的正确性.我们认为,可以采用定理证明的方式来保证模型转换的语义一致性.我们对部分 AADL 形式语义描述语言进行比较,见表 1.

Table 1 Comparison of AADL semantics models

表 1 AADL 语义模型比较

Methods	Fundamental formalism	Verification and analysis tools	Strengths	Weaknesses
TLA+	Logic	Model checker: TLC	General and abstract expressions	The verification tool support simple simulation
TASM	Abstract state machine	Model checker: UPPAAL Analysis tool: TASMToolset	Describe real time, resource and concurrency well readability	Less time and data types of expression
Fiacre	Process algebra	Model checker: TINA, CADP	Offers a rich set of structuring techniques and concurrency patterns	Less powerful construct, such as resource expression
Maude	Rewrite logic	Proving tool: ITP Analysis tool: MTT	General and abstract expressions	Don't support real time description
BIP	Automata	Model checker: Aldebaran	Powerful for components interaction and priority	The verification tool only support deadlock detection
ACSR	Process algebra	Analysis tool: VERSA	Describe real time, resource strongly	The verification tool only support simple analysis
UML Marte	UML	None	Explicit time model	Don't support formal verification

AADL 能够描述系统功能行为、非功能属性以及运行时的体系结构动态演化,尤其是 AADL 的可扩展性,使得 AADL 建模机制越来越复杂,很多语义还需要形式化.本文认为可以对多种语义描述进行比较,从不同角度来刻画 AADL 的语义与性质.

2.2 AADL建模方法的研究

AADL 是嵌入式实时系统领域的体系结构设计与分析标准,而 MDD 正在成为主流的开发过程.文献[45]提出 AADL 与 MDD 结合,能够使 AADL 得到更广泛的使用.MDD 定义了 3 个层次的抽象:平台独立模型(platform independent model,简称 PIM)、平台特定模型(platform specific model,简称 PSM)以及应用代码.PIM 是模型驱动设计的初始抽象模型,是设计过程中最为重要的环节之一.因此,AADL 建模方法的研究主要涉及 AADL PIM 的

构造.我们归纳为 3 类:

(1) 从需求模型到 AADL 模型的转换

由于 UML,SysML,xUML(executable UML)[46],HOOD 等描述语言已经广泛应用于需求的描述,因此结合这些描述语言的优势是必要的,从需求模型到 AADL 模型的转换就成为重要的研究内容.文献[47]采用 UML 作为需求模型,提出需求模型到 AADL 模型的转换规则,如 UML 类转换为 AADL 构件类型,类之间的组合连接转换为 AADL 子构件,类之间的直接连接转换为 AADL 端口等.Feiler 提出将 xUML 和 AADL 集成到统一的模型驱动开发过程,通过一些经验规则来完成需求模型到 AADL 模型的转换[48],其中 xUML 对 UML 增加了更精确的执行语义.文献[49]采用 SysML 作为需求模型,并研究了 SysML 到 AADL 的模型转换.文献[50]则采用 UML,HRT-HOOD(hard real time HOOD)[51]作为需求模型,提出将带有属性和操作的 UML 类图转换到 AADL 的包,每个包由数据构件类型、数据构件实现、子程序构件类型组成,前两者表示属性,后者表示操作,同时将 HRT-HOOD 的周期性对象、非周期性对象、保护对象分别转换到 AADL 的周期性线程、非周期性线程、数据构件,并在 STOOD[15]工具中实现了这些转换.其中,HOOD 是欧洲航空防务航天公司(European Aeronautic Defense and Space,简称 EADS)提出的复杂系统建模方法,HRT-HOOD 则是 HOOD 在嵌入式实时系统领域的扩展,而 STOOD 是建立在 HOOD,AADL 之上的商业建模工具.

(2) AADL 常用于构造系统的 PSM 模型,因为 AADL 能够同时描述系统的软、硬件体系结构,两者映射即可构成 PSM 模型.同时,也可以采用 AADL 单独建立系统的软件体系结构模型,作为 AADL PIM 模型.

(3) 采用 AADL 与 UML 结合的方式,文献[52]基于 AADL UML Profile Annex,将 AADL 原型(stereotype)标注在 UML 模型上,作为 AADL PIM 模型.如,构件用 UML 类图表示,并标注 AADL 构件类型.

模型驱动设计的建模一般在需求分析之后进行,为了保持需求规约与系统设计之间的一致性,可以研究基于 AADL 的需求建模.文献[53,54]提出了面向软件体系结构的需求工程方法,这些可以借鉴到基于 AADL 的需求规约当中,而且 AADL V2.0 的新概念将更有助于 AADL 直接应用于需求工程.需求阶段的 AADL 研究与应用,为整个生命周期提供完整的需求和概念.

2.3 AADL模型转换

模型转换是 MDD 的核心,同时也是 AADL 模型验证与分析的重要基础.根据嵌入式实时系统的需求,文献[52]给出了基于 AADL 的模型驱动开发过程:(1) 获得 AADL PIM 模型,如,基于 AADL UML Profile Annex,将 AADL 原型(stereotype)标注在 UML 模型上,作为 AADL PIM;(2) 利用功能转换规则,转换成 AADL PSM;(3) 利用非功能转换规则,将 PSM 转换成 AADL 分析模型,进行可调度、性能等分析;(4) 基于正确的 AADL 模型转成可执行代码.

模型转换关键在于转换规则及工具的研究.文献[55]对常见的模型转换方法进行了分类,按转换规则分为变量型、模式型、关系型以及逻辑型;按转换方法可以分为模型到模型的转换以及模型到代码的转换,前者包括手动转换、关系转换、图转换、结构转换、元模型转换等方法,后者则包括基于访问的转换以及基于模板的转换.在模型转换工具方面,QVT(query/view/transformation)[56]是由 OMG 定义的模型转换标准;ATL(Atlas transformation language)[57]是由法国 INRIA 开发的一种基于元模型转换方法的模型转换语言;Kermeta[58]是法国 INRIA Triskel Team 开发的模型转换语言,能够对元模型的结构、行为进行描述;GreAT(graph rewriting and transformation)[59]则是一种将图形转换和重写技术应用到模型转换过程中的方法.ATL 和 Kermeta 在 AADL 模型转换研究中比较常用.同时,本文认为 AADL 模型转换的正确性验证与证明也是需要重点研究的内容.

2.4 AADL模型的验证与分析

在早期阶段进行体系结构验证与分析,能够尽早发现系统设计的潜在错误.目前,AADL 模型验证与分析主要采用仿真和形式化方法.

仿真方法需要设定体系结构模型中的初始参数,通过执行体系结构模型,对每个元素的行为进行分析与评价.ADeS(architecture description simulation)[60]是一个事件驱动的 AADL 行为仿真工具,支持执行时间、时限、

调度策略、模式的分析.但目前还不支持子程序、Behavior Annex 以及 AADL V2.0 一些新建模元素的行为仿真.文献[61]采用 SCoPE(system co-simulation and performance estimation)[62]工具对 AADL 中的软硬件划分(HW/SW partition)、硬件构件行为等进行仿真.AADL 建模工具 OSATE/TOPCASED 也通过仿真的方式,支持端到端的流延迟、安全层次检测等分析.法国 Brest 大学 LISYC Team 采用仿真的方法对 AADL 模型进行可调度分析,并开发了 Cheddar 工具[13].Cheddar 支持经典实时调度算法、资源共享、分布式多处理器、端到端任务等多种情况的可调度分析,但不支持模式变换、Behavior Annex 的可调度分析,同时还可以基于 Cheddar 对资源(带宽、存储等)[63,64]、性能[65]进行分析.Furness[66]也提供仿真的方式对 AADL 模型进行可调度分析,但目前只支持简单的调度模型.低功耗是嵌入式实时系统的一个重要性质,文献[67]研究基于 AADL 的功耗分析与优化.

为了更精确的体系结构验证与分析,形式化方法是重要的手段.AADL 属于高层建模语言,因此需要将 AADL 模型转换到更低层次的形式化模型.在行为验证方面,主要基于第 2.1.2 节所述的语义模型进行验证,如 Fiacre,TLA+,TASM,BIP 等.在可靠性分析方面:文献[68−70]采用 AADL Error Model Annex 构建故障模型,与体系结构模型结合并转换成随机 Petri 网模型,然后进行可靠性分析,实现了原型工具 ADAPT(AADL architectural models to stochastic Petri nets through model transformation)[71];文献[72]提出了一个可扩展的可靠性评价框架 Arcade(architectural dependability evaluation),能够支持包括 AADL,UML 在内的多种建模语言,将输入模型转换到 IO-IMC(input/output interactive Markov chains)模型,然后基于 CADP 工具进行可靠性分析,并可以进行组合可靠性分析(compositional analysis),以支持复杂系统的需求,其中,IO-IMC 是 I/O 自动机与马尔可夫链结合的形式化模型;文献[73]提出将故障树与 AADL 模型相结合,进行可靠性和安全性分析.在可调度分析方面:Pennsylvania 大学提出将 AADL 模型转换成时间进程代数 ACSR,并基于工具 VERSA 进行可调度分析,但主要针对单处理器环境.由 Telecom Paris 研究的 Ocarina[74]工具,能够将 AADL 模型转换到 Petri 网进行形式化验证、基于 Cheddar 进行可调度分析以及自动生成 Ada 和 C 代码.随着嵌入式实时系统的网络化、模块化,安全性(security)受到关注,系统会受外部攻击、恶意代码的影响.目前,AADL 对安全性建模能力还较弱,只能对安全性层次进行简单的检测.文献[75]提出将安全策略 Bell-LaPadula 模型和 AADL 模型结合起来,对端口和连接的安全性、构件安全层次进行分析.文献[76]提出在设计阶段就要考虑系统的安全性,通过对 AADL 进行安全性建模扩展,包括基于角色的访问控制、安全连接、公平交易等,以支持在早期进行安全性验证与分析.

我们认为,AADL 应该与实时系统已有研究结合起来,以促进 AADL 的广泛使用.在形式验证方面,如基于时间自动机的验证工具 UPPAAL、基于时间 Petri 网的模型检测工具 TINA,以及可扩展的模型检测工具 BOGOR[77]等;在可调度分析方面,基于处理器利用率、最坏响应时间的可调度分析、基于模型检测的可调度分析以及调度分析工具 TIMES[78],MAST[79];在安全性分析方面,UML 的安全性扩展 UMLsec[80]及其相关工具都可以与 AADL 结合起来.同时,由于验证分析方法的多样性,应该研究可扩展的验证与分析框架,将各种方法和工具的优势集成在一起.特别是将验证与分析结果直接反馈到 AADL 模型,对系统设计的改进将有重要的作用.

2.5 基于AADL模型的自动代码生成

自动代码生成主要研究从 AADL 模型到可执行代码的生成规则、方法和工具.其优点在于,减少由手动编程所带来的错误,保证系统质量属性以及降低系统开发时间.目前基于 AADL 模型可以生成的程序语言有 Ada,C,C++,Java-RTSJ(real time specification for Java)等.同时,针对分布式嵌入式实时系统,可以生成实时中间件及其上的分布式应用代码.本文将基于 AADL 模型的代码生成归纳为以下几类:

(1) 基于 AADL 各类软件构件,生成对应的源代码.如线程构件到 C 代码的转换.STOOD 工具支持 AADL 到 C,C++,Ada 代码生成以及 HTML,RTF,MIF 等文档的生成,还支持 Ada,C 代码到 AADL 描述的逆向工程.IRIT 研究了 AADL 到 RTSJ 的代码转换[81],RTSJ 被认为是未来航天应用系统的执行内核.Verimag 实验室研究 AADL 到同步语言 Lustre 的代码转换问题[82],Lustre 广泛的应用于航空航天领域,其转换的难点在于如何将非确定性和异步的 AADL 描述转换成 Lustre 的同步描述.ARC(Ada to Ravenscar converter)[83]则是基于 Eclipse 的 AADL 到 Ada 代码生成工具.

(2) 生成符合不同领域标准的应用代码,例如,生成符合汽车电子控制系统标准 AutoSAR、铁路控制系统标

准 EN-50128 的 C 代码.文献[21]采用模型转换语言 ATL,将 AADL 模型转换到基于汽车电子实时操作系统 OSEK 之上的 C 代码.

(3) 实时中间件及其上的分布式应用代码生成.文献[84]提出了基于 AADL 的中间件生成技术(automatic middleware generation).Ocarina 工具支持从 AADL 模型生成运行在 PolyORB[85],PolyORB-HI(PolyORB-high integrity)中间件之上的 Ada 分布式应用代码,同时支持生成 POSIX 和 RTEMS 平台上的 C 代码,其中 PolyORB 是一个通用中间件,目标是为不同分布式应用提供统一的解决方案.SPICES 项目[7]研究从 AADL 模型到实时 CORBA 中间件及其上的 CCM(CORBA component model)应用构件生成.

(4) 基于 AADL 的设备构件,研究设备驱动代码的生成技术.

(5) 实时操作系统本身也可以用 AADL 来描述.文献[21]研究了生成实时操作系统代码的方法.

3 AADL 建模与分析工具

AADL 建模、分析、自动代码生成等方面有相应的工具支持,OSATE,TOPCASED,STOOD 属于集成开发工具;Furness,Cheddar 属于可调度分析工具;Ocarina 属于自动代码生成工具.但 AADL 属于一个新的研究领域,其建模和分析工具还不够成熟.下面对学术界比较常用的几个工具进行概述:

(1) OSATE/TOPCASED

OSATE(open source AADL tool environment)[14]是由 CMU 开发的 AADL 开源集成开发环境,是在 Eclipse 平台上的一套插件,用于 AADL 建模、编译和分析.在 OSATE 上开发了多种分析插件,进行可调度性分析、安全性分析、时间延迟分析等.而 TOPCASED 提供了 AADL 的图形化编辑器,因此,一般将 OSATE 和 TOPCASED 一起作为建模与分析工具来使用.

(2) Cheddar

Cheddar[13]是由法国 Brest 大学开发的实时系统可调度分析工具,采用仿真的方式对 AADL 模型进行可调度分析,支持单处理器、多处理器、共享资源等多种调度情况.可以作为 TOPCASED,STOOD 的插件使用.

(3) Ocarina

Ocarina[74]是由法国 Telecom Paris 开发的 AADL 分析套件,通过模型转换到 Petri 网进行行为验证;通过集成 Cheddar 进行可调度分析;实现了 AADL 到实时中间件 PolyORB,PolyORB-HI 及其上的分布式 Ada,C 代码的自动生成.

表 2 给出 AADL 已有建模与分析工具的比较.

Table 2 Comparison of AADL modeling and analysis tools

表 2 AADL 建模与分析工具比较

Tools	Modeling	Schedulability analysis	Verification	Code generation
OSATE	Textual XML-based	Scheduling analysis plug-in	Architecture consistency checks	MetaH
TOPCASED	Textual, graphic and XML-based	Scheduling analysis plug-in	Model transformation to serveral model checker	C/C++,Ada,RTSJ
STOOD	Textual, graphic and XML-based	Connection with Cheddar	Legality rules checker	C/C++,Ada,HTML
Cheddar	No	Using simulation methods	No	No
Furness	No	Using model checking methods	No	No
Ocarina	No	Connection with Cheddar	Model transformation to Petri net	C,Ada,PoyORB,PolyORB-HI

4 AADL 应用实践概述

AADL 自提出以来,一直是学术界和工业界共同关注的对象.目前,AADL 研究成果主要应用于航空、航天领域,如航空电子、卫星系统等.下面对几个典型的项目和应用进行概述.

(1) TOPCASED

TOPCASED(toolkit in open source for critical applications & systems development)[6]是由空中客车公司提出

并实际应用的项目,采用模型驱动开发和形式化验证结合、多种开源工具集成的思想,支持复杂嵌入式实时系统的设计、开发与实现,以保证系统的质量属性并降低开发时间和成本.目前支持 AADL,UML,SDL 等多种建模语言,支持 TINA,CADP,UPPAAL 等模型检测工具,支持 C,Ada,RTSJ 等语言的自动代码生成,并支持 AADL 图形化建模.

(2) SPICES

SPICES(support for predictable integration of mission critical embedded systems)[7]是由欧盟 EUREKA-ITEA 支持的项目.针对航空工业界具体应用,提出对 AADL 语言进行扩展,以支持一些新概念和属性的描述,如功耗约束、对 ARINC653 标准的支持等,并标准化为 AADL V2.0;研究基于扩展 AADL 的验证和分析方法,如 AADL 行为仿真工具 Ades,Osate 和 TOPCASED 工具集成、AADL 建模工具 ADELE 等;研究综合化航空电子系统(integrated modular avionics,简称 IMA)、ARINC653 平台之上的安全关键嵌入式系统的可预见性实现;研究 AADL 自动生成到实时 CORBA 中间件之上分布式应用.研究成果应用于空中客车的下一代航空电子原型系统、法国地球观测卫星的嵌入式控制器、Thales 公司的某个航空设备等系统中.

(3) ASSERT

ASSERT(automated proof based system and software engineering for real-time applications)[86]是欧盟支持的项目,目标是对传统嵌入式实时系统的开发过程进行改进,生命周期中每一个环节,都需要进行严格地确认和验证(validation and verification,简称 VV)才能进入下一个环节.AADL 被用来描述所开发系统的体系结构.在该项目的支持下,法国达索公司(Dassault-Aviation)采用 AADL 对关键任务系统进行建模,并转换到时间 Petri 网模型进行行为验证,以检验 AADL 的建模能力.

5 总结与进一步的研究工作

AADL 是针对系统工程(system engineering)的体系结构描述语言,而不仅仅是软件体系结构描述语言.AADL 的研究与传统软件体系结构研究一样,首先关注设计阶段,然后过渡到设计之后的实现、集成阶段,最后再关注设计之前的需求分析阶段,从而覆盖全生命周期.表 3 从系统全生命周期的角度对 AADL 研究进行分类与总结.

Table 3 Research and practice of AADL in the system lifecycle

表 3 系统全生命周期中 AADL 的研究与应用

Requirement phase	• System-Level requirements modeling using AADL: Functionality requirements, non-functionality requirements, etc. • Transformation from requirement model to AADL model.
Design phase	• Functionality extension: Behavior annex, error model annex, ARINC653 annex and UML profile annex, etc. • AADL formal semantics • Analysis and verification: Dependability analysis, schedulability analysis, security analysis, model checking and simulation, etc.
Implementation phase	• Model transformation • Code generation: C, Ada, etc. • Testing and Integration
Integration phase	• System deployment • Dynamic architecture • Architecture reconstruction

在工业界和学术界的共同努力下,AADL 研究已经取得了较好的成果,并将模型驱动思想和形式化方法应用推进到一个新台阶.本文归纳了 AADL 研究热点并分析了现有研究的不足,剖析了诸如执行模型、Behavior Annex 等重要概念,希望能够为 AADL 的进一步研究提供参考.

同时,随着系统规模及复杂性的增加,模型驱动方法在嵌入式实时系统设计与开发中的作用更加显著,AADL 的研究与应用也会进一步得到加强.我们认为还需要从以下几个方面对 AADL 进行深入探讨:

(1) AADL 语言研究

- AADL 执行模型涉及构件分发、调度、通信、模式变换等多个方面,其形式语义还仅涉及其中部分内容,因此 AADL 形式语义需要进一步完善.

(2) 全生命周期各阶段中的 AADL 研究

- 基于 AADL 的需求建模.常见的需求规约语言有 UML,SysML[87].UML 主要针对软件系统采用用例图、类图、顺序图等来描述需求,这对需要描述硬件需求的嵌入式实时系统还是不够的[50];SysML 对 UML 进行了扩展和修改,用于描述包含软、硬件的复杂系统,能够对系统的需求、行为、结构以及参数进行描述,但主要针对通用系统.对 AADL 进行扩展以直接应用于需求分析,有助于保证需求和系统设计之间的一致性和可追踪性.AADL V2.0 新增加了抽象构件(abstract component)、原型(prototype)等建模元素,可以进一步支持需求阶段概念模型、抽象数据类型的定义.
- AADL 模型验证与分析.在 AADL 可靠性分析方面,目前只依赖于 Error Model Annex,研究更丰富的可靠性建模与分析方法是必要的;在 AADL 可调度性分析方面,目前支持一些经典调度情况,而多处理器环境下的资源共享、模式变换、Behavior Annex 对可调度分析也会有影响.
- 在自动代码生成方面,由于嵌入式系统中的资源受限,可定制的轻量化中间件生成以及代码优化都是应该考虑的.
- AADL 模型驱动测试研究.模型驱动开发方法和软件测试技术的结合,使得能够尽早开始测试,减少中间环节.基于 AADL 模型的测试用例生成以及模型驱动的测试过程、方法、工具是需要重点研究的内容.如,基于 AADL 语义模型,采用模型检测方法进行测试用例生成.
- 集成部署阶段的 AADL 研究.随着分布式实时嵌入式系统的发展,系统的部署和配置受到重视.部署即将分布式应用构件绑定到对应的物理硬件并准备运行,而中间件的使用有利于系统的部署;配置则是部署阶段构件及其参数的选择.文献[88]提出基于体系结构的部署有益于以下几个方面:(1) 提供高层的体系结构视图描述部署阶段的软硬件模型;(2) 基于体系结构模型可以分析部署方案的质量属性,从而选择合理的部署方案;(3) 通过体系结构记录系统部署的经验,以便下次复用.AADL 通过属性来描述详细的配置与部署信息,文献[89]利用 Ocarina 工具由 AADL 定义的属性自动生成可配置的中间件,并将分布式应用部署在中间件之上.根据 AADL 体系结构模型中的配置和部署信息,可以分析与评估部署方案.
- 基于 AADL 的体系结构演化研究.研究表明,系统体系结构由于内部执行或外部环境会导致在运行时发生改变.问题是,如何在设计阶段获取体系结构这种动态性,并指导系统在运行时实施这些变化,从而达到系统的自动演化或自适应.AADL 构件定义都包括类型和实现,一个类型可以对应多个版本的运行时实现,支持多个版本之间的演化;模式变换可以方便地描述运行时体系结构变化;AADL 还提供 Plug-replacement,Recursive-replacement,Interceptor 等机制,支持构件替换、构件递归替换以及增加新构件等变化的描述.随着系统规模的扩大,动态体系结构研究会面临更多的新问题.因此,AADL 对动态体系结构的描述机制需要进一步加强,并采用形式化方法对其语义进行精确刻画.
- 体系结构恢复与重构主要是针对遗留系统(legacy systems)升级的需要,这类系统大部分没有体系结构描述或体系结构不合理.Boeing,Honeywell 采用 AADL 对遗留系统进行体系结构重构,分析与评价系统的性质,从而达到替换一些关键构件的目的;STOOD 工具支持 C,Ada 代码到 AADL 的逆向工程(reverse engineering),以实现系统重构.目前基于 AADL 的体系结构恢复和重构还处于探索阶段,缺乏必要的自动重构工具和方法支持.
- 支持 AADL V2.0 的验证与分析方法也有待研究.

(3) AADL 可以与其他领域研究结合起来,这需要扩展 AADL 的属性集.Virginia 大学的 Prasad 等人将 AADL 用于无线传感器网络系统的建模与分析,并基于 OSATE 工具开发了无线传感器网络分析与设计工具 ANDES (analysis-based design tool for wireless sensor networks)[90].文献[91]利用 AADL 对 UDP/IP 协议栈的实现进行层次化的体系结构描述.文献[92]提出将 SOA 的概念引入嵌入式实时系统,并采用 AADL 对面向服务的嵌入式系

统体系结构进行建模与分析.

(4) 将 AADL 应用于实际工业系统,尤其是大规模系统设计,如何保证建模与分析工具的有效性和实用性.同时,AADL 对不同系统的专用需求进行了抽象和简化,如何满足实际系统需求,也是需要进一步探讨的内容.

致谢 感谢法国 Toulouse Institute of Computer Science Research 的 Mamoun Filali 教授和 Jean-Paul Bodeveix 教授,他们是 AADL Behavior Annex 的提出者,在 AADL 执行模型、形式语义、验证和分析方面给予本文很大的支持与建议,并合作发表论文[18,29],在此表示感谢.同时感谢中国科学院软件研究所的李广元副研究员、朱雪阳老师给予的宝贵意见.希望本文能够给研究和应用 AADL 的同行提供一些参考.

References:

[1] Feiler PH, Lewis BA, Vestal S. The SAE architecture analysis & design language (AADL)—A standard for engineering performance critical systems. In: Proc. of the 2006 IEEE Conf. on Computer Aided Control Systems Design. Washington: IEEE Computer Society Press, 2006. 1206−1211.

[2] OMG. UML Profile for Schedulability, Performance, and Time V1.1. 2005. http://www.omg.org/technology/documents/formal/schedulability.htm

[3] OMG. UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms Specification V1.1. 2008. http://www.omg.org/technology/documents/formal/QoS_FT.htm

[4] OMG. UML profile for modeling and analysis of real-time and embedded systems Beta 2. 2008. http://www.omgmarte.org/

[5] SAE Aerospace. SAE AS5506: Architecture Analysis and Design Language (AADL). SAE Int'l, 2004. http://www.aadl.info/aadl/currentsite/

[6] Farail P, Gaufillet P, Canals A, Le Camus C, Sciamma D, Michel P, Crégut X, Pantel M. The TOPCASED project: A toolkit in open source for critical aeronautic systems design. In: Proc. of the 3rd European Congress Embedded Real Time Software. Toulouse: French Society of Automobile Engineers, 2006. 55−59.

[7] Gaufillet P, Heim S, Bonnin H, Dissaux P. ITEA SPICES AADL experimentation at airbus. In: Proc. of the 14th Int'l Conf. on Reliable Software Technologies, Ada-Europe. Washington: IEEE Computer Society Press, 2009. http://public.enst-bretagne.fr/~kermarre/RST2009/p.gaufillet_airbus.pdf

[8] Lewis BA, Feiler PH. Multi-Dimensional model based engineering using AADL. In: Proc. of the 19th IEEE/IFIP Int'l Symp. on Rapid System Prototyping. Washington: IEEE Computer Society Press, 2008. xv−xviii.

[9] HOOD User Group. HOOD Reference Manual Release 4.0. HUG, 1995. http://www.esa.int/TEC/Software_engineering_and_standardisation/TECKLAUXBQE_0.html

[10] SAE Aerospace. Architecture Analysis and Design Language (AADL) Annex Volume 1. Document Number: AS5506/1, SAE International, 2006. http://www.sae.org/technical/standards/ AS5506/1

[11] SAE Aerospace. SAE AS5506 Annex: Behavior_Specification V1.6. SAE International, 2006. http://www.aadl.info/aadl/documents/Behaviour_Annex1.6.pdf

[12] SAE Aerospace. SAE AS5506A: Architecture Analysis and Design Language V2.0. 2009. http://www.sae.org/technical/standards/AS5506A

[13] Singhoff F, Legrand J, Nana L, Marcé L. Cheddar: A flexible real time scheduling framework. ACM Ada Letters, 2004,24(4):1−8. [doi: 10.1145/1046191.1032298]

[14] The SEI AADL Team. An extensible open source AADL tool environment (OSATE). Software Engineering Institute, CMU, 2006. http://www.aadl.info/aadl/downloads/osate13/AADLToolUserGuide1.3.0%202006-06-02.pdf

[15] Dessaux P, Singhoff F. Stood and Cheddar: AADL as a pivot language for analysing performances of real time architectures. In: Proc. of the 4th European Congress on Embedded Real-Time Software. Toulouse, 2008. http://www.ea2215.univ-brest.fr/publications/par_chercheur/singhoff/ERTS08.pdf

[16] França RB, Bodeveix JP, Filali M, Rolland JF, Chemouil D, Thomas D. The AADL behaviour annex—Experiments and roadmap. In: Proc. of the 12th IEEE Int'l Conf. on Engineering Complex Computer Systems. Washington: IEEE Computer Society, 2007. 377−382.

[17] Bodeveix JP, Filali M, Rached M, Chemouil D, Gaufillet P. Experimenting an AADL behavioural annex and a verification method.

In: Proc. of the DASIA 2006—DAta Systems in Aerospace. Berlin, 2006. ftp://ftp.estec.esa.nl/pub/wm/wme/Web/AADLExperience2006.pdf

[18] Yang ZB, Hu K, Ma DF, Pi L. Towards a formal semantics for the AADL behavior annex. In: Proc. of the IEEE/ACM Design, Automation and Test in Europe 2009. Nice: EDAA, 2009. 1166−1171.

[19] Singhoff F, Plantec A. AADL modeling and analysis of hierarchical schedulers. In: Proc. of the 2007 ACM Int'l Conf. on SIGAda Annual Int'l Conf. New York: ACM Press, 2007. 41−50.

[20] Ma Y, Talpin JP, Gautier T. Virtual prototyping AADL architectures in a polychronous model of computation. In: Proc. of the 6th ACM/IEEE Int'l Conf. on Formal Methods and Models for Co-Design. Washington: IEEE Computer Society Press, 2008. 139−148.

[21] Brun M, Delatour J, Trinquet Y. Code generation from AADL to a real-time operating system: An experimentation feedback on the use of model transformation. In: Breitman K, Wookcock J, Sterrit R, Hinchey MJ, eds. Proc. of the 13th IEEE Int'l Conf. on Engineering of Complex Computer Systems. Washington: IEEE Computer Society Press, 2008. 257−262.

[22] Autosar Web Site. http://www.autosar.org

[23] Cuenot P, Chen DJ, Gérard S, Lönn H, Reiser MO, Servat D, Sjöstedt CJ, Kolagari RT, Törngren M, Weber M. Managing complexity of automotive electronics using the EAST-ADL. In: Proc. of the 12th IEEE Int'l Conf. on Engineering Complex Computer Systems. Washington: IEEE Computer Society Press, 2007. 353−358.

[24] Andreé C, Mallet F, de Simone R. Modeling of immediate vs. delayed data communications: From AADL to UML MARTE. In: Proc. of the Forum on Specification & Design Languages (FDL). Barcelona: ECSI, 2007. 249−254.

[25] Feiler PH. Efficient embedded runtime systems through port communication optimization. In: Breitman K, Wookcock J, Sterrit R, Hinchey MJ, eds. Proc. of the 13th IEEE Int'l Conf. on Engineering of Complex Computer Systems. Washington: IEEE Computer Society Press, 2008. 294−300.

[26] Lee SY, Mallet F, de Simone R. Dealing with AADL end-to-end flow latency with UML MARTE. In: Breitman K, Wookcock J, Sterrit R, Hinchey MJ, eds. Proc. of the 13th IEEE Int'l Conf. on Engineering of Complex Computer Systems. Washington: IEEE Computer Society Press, 2008. 228−233.

[27] Rolland JF, Bodeveix JP, Chemouil D, Filali M, Thomas D. Towards a formal semantics for AADL execution model. In: Proc. of the 4th European Congress on Embedded Real-Time Software. 2008. http://www.citeulike.org/user/DavidChemouil/article/3482660

[28] Rolland JF, Bodeveix JP, Filali M, Chemouil D, Thomas D. Modes in asynchronous systems. In: Breitman K, Wookcock J, Sterrit R, Hinchey MJ, eds. Proc. of the IEEE Int'l Conf. on Engineering Complex Computer Systems. Washington: IEEE Computer Society Press, 2008. 282−287.

[29] Pi L, Yang ZB, Bodeveix JP, Filali M, Hu K, Ma DF. A comparative study of FIACRE and TASM to define AADL real time concepts. In: Proc. of the 14th IEEE Int'l Conf. on Engineering of Complex Computer Systems. Washington: IEEE Computer Society Press, 2009. 347−352.

[30] Berthomieu B, Bodeveix JP, Farail P, Filali M, Garavel H, Gaufillet P, Lang F, Vernadat F. Fiacre: An intermediate language for model verification in the TOPCASED environment. In: Proc. of the 4th European Congress on Embedded Real-Time Software. Toulouse, 2008. ftp://ftp.inrialpes.fr/pub/vasy/publications/others/Berthomieu-Bodeveix Farail-et-al-08.pdf

[31] Ouimet M, Lundqvist K. The Timed abstract state machine language: Abstract state machines for real-time system engineering. Journal of Universal Computer Science, 2008,14(12):2007−2033.

[32] Behrmann G, David A, Larsen KG. A tutorial on UPPAAL. In: Proc. of the 4th Int'l School on Formal Methods for the Design of Computer, Communication, and Software Systems. Bertinoro: Springer-Verlag, 2004. 200−236.

[33] Chkouri MY, Robert A, Bozga M, Sifakis J. Translating AADL into BIP—Application to the verification of real-time systems. In: Proc. of the ACESMB 2008 Workshop Conjunction with MODELS 2008. Berlin, Heidelberg: Springer-Verlag, 2009. 5−19.

[34] Basu A, Bozga M, Sifakis J. Modeling heterogeneous real-time components in BIP. In: Hung DV, Pandya P, eds. Proc. of the SEFM 2006. Washington: IEEE Computer Society Press, 2006. 3−12.

[35] Benammar M, Belala F, Latreche F. AADL behavioral annex based on generalized rewriting logic. In: Rolland C, Collard M, Pastor O, Flory A, Cavarero JL, eds. Research Challenges in Information Science. Washington: IEEE Computer Society Press, 2008. 1−8.

[36] Abdoul T, Champeau J, Dhaussy P, Pillain PY, Roger JC. AADL execution semantics transformation for formal verification. In:

Breitman K, Wookcock J, Sterrit R, Hinchey MJ, eds. Proc. of the 13th IEEE Int'l Conf. on Engineering of Complex Computer Systems. Washington: IEEE Computer Society Press, 2008. 263−268.

[37] Bozga M, Graf S, Mounier L. IF-2.0: A validation environment for component-based real-time systems. In: Proc. of the Conf. on Computer Aided Verification. Berlin: Springer-Verlag, 2002. 630−640.

[38] Sokolsky O, Lee I, Clark D. Schedulability analysis of AADL models. In: Proc. of the 20th Int'l Parallel and Distributed Processing Symp. 2006. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1639421

[39] Lee I, Philippou A, Sokolsky O. A family of resource-bound real-time process algebras. Electronic Notes in Theoretical Computer Science, 2006,162:221−226. [doi: 10.1016/j.entcs.2005.12.085]

[40] Clarke D, Lee I, Xie HL. VERSA: A tool for the specification and analysis of resource-bound real-time systems. Journal of Computer and Software Engineering, 1995,3(2):185−215.

[41] Sokolsky O, Chernoguzov A. Analysis of AADL models using real-time calculus with applications to wireless architectures. Technical Report, No.MS-CIS-08-25, Department of Computer and Information Science, University of Pennsylvania, 2008. http://repository.upenn.edu/cis_reports/887/

[42] Thiele L, Chakraborty S, Naedele M. Real-Time calculus for scheduling hard real-time systems. In: Int'l Symp. on Circuits and Systems ISCAS 2000. Washington: IEEE Computer Society Press, 2000. 101−104.

[43] Rugina PAE. Dependability modeling and evaluation—From AADL to stochastic Petri nets [Ph.D. Thesis]. Toulouse: LAAS-CNRS, 2008.

[44] Bodeveix JP, Filali M, Martin S. Towards formalising AADL in proof assistants. Electronic Notes in Theoretical Computer Science. 2005,141:153−169. [doi: 10.1016/j.entcs.2005.05.008]

[45] Gérard S, Feiler PH, Rolland JF, Filali M, Reiser MO, Delanote D, Berbers Y, Pautet L, Perseil I. UML&AADL '2007 Grand challenges. ACM SIGBED Review, 2007,4(4):1−17. [doi: 10.1145/1366546.1366547]

[46] Fuentes L, Sánchez P. Towards executable aspect-oriented UML models. In: Proc. of the 10th Int'l Workshop on Aspect-Oriented Modeling. New York: ACM Press, 2007. 28−34.

[47] Delanote D, van Baelen S, Joosen W, Berbers Y. Using AADL in model driven development. In: Canals A, Gerard S, Perseil I, eds. Proc. of the Int'l Conf. on Engineering Complex Computer Systems. Washington: IEEE Computer Society Press, 2007. 1−10.

[48] Feiler PH, de Niz D, Raistrick C, Lewis BA. From PIMs to PSMs. In: Proc. of the 12th IEEE Int'l Conf. on Engineering Complex Computer Systems. Washington: IEEE Computer Society Press, 2007. 365−370.

[49] Fetzer C. Systems engineering modeling languages: SysML & AADL. TU Dresden, 2004. http://wwwse.inf.tu-dresden.de/data/courses/SE1/SE1-2004-lec13.pdf

[50] Dissaux P. AADL model transformations. In: Ouwehand L, ed. Proc. of the DASIA 2005 Conf. in Edinburgh. 2005. http://www.ellidiss.com/dasia05.pdf

[51] Burns A, Wellings AJ. HRT-HOOD: A structured design method for hard real-time systems. Real-Time Systems, 1994,6(1): 73−114. [doi: 10.1007/BF01245300]

[52] Delanote D, van Baelen S, Joosen W, Berbers Y. Using AADL in model driven development. In: Canals A, Gerard S, Perseil I, eds. Int'l Conf. on Engineering Complex Computer Systems. Washington: IEEE Computer Society Press, 2007. 1−10.

[53] Mei H. A complementary approach to requirements engineering-software architecture orientation. Software Engineering Notes, 2000,25(2):40−45. [doi: 10.1145/346057.346070]

[54] Svetinovic D. Architecture-Level requirements specification. In: Proc. of the 2nd Int'l Software Requirements to Architectures Workshop. 2003. 14−19. http://se.uwaterloo.ca/~straw03/ProceedingsSTRAW03.pdf

[55] Czarnecki K, Helsen S. Classification of model transformation approaches. In: Proc. of the OOPSLA 2003 Workshop on Generative Techniques in the Context of Model-Driven Architecture. 2003. http://www.swen.uwaterloo.ca/~kczarnec/ECE750T7/czarnecki_helsen.pdf

[56] OMG. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. 2007. http://www.omg.org/docs/ptc/07-07-07.pdf

[57] Jouault F, Kurtev I. Transforming models with ATL. In: Proc. of the Model Transformations in Practice Workshop at MoDELS 2005. Berlin: Springer-Verlag, 2006. 128−138.

[58] Drey Z, Faucher C, Fleurey F, Mahé V, Vojtisek D. Kermeta Language Reference Manual. 2009. http://www.kermeta.org/docs/KerMeta-Manual.pdf

[59] Agrawal A. Graph rewriting and transformation (GReAT): A solution for the model integrated computing (MIC) bottleneck. In: Proc. of the 18th IEEE Int'l Conf. on Automated Software Engineering. Washington: IEEE Computer Society Press, 2003. 364−368.

[60] Axlog. ADeS, a simulator for AADL. 2007. http://www.axlog.fr/aadl/ades_en.html

[61] Varona-Gómez R, Villar E. AADS: AADL simulation and performance analysis in SystemC. In: IEEE/ACM Design, Automation and Test in Europe 2009. Nice: EDAA, 2009. http://www.teisa.unican.es/gim/pub_files/file_375.pdf

[62] Posadas H, Quijano D, Villar E, Martínez M. SCope: SoC co-simulation and performance estimation in SystemC. In: IEEE/ACM Design, Automation and Test in Europe 2007. Nice: EDAA, 2007. http://www.teisa.unican.es/gim/pub_files/file_376.pdf

[63] Singhoff F, Plantec A, Dissaux P. Can we increase the usability of real time scheduling theory? The Cheddar project. In: Proc. of the 13th Int'l Conf. on Reliable Software Technologies, Ada-Europe. Berlin: Springer-Verlag, 2008. 240−253.

[64] Singhoff F, Legrand J, Nana L, Marcé L. Scheduling and memory requirement analysis with AADL. ACM Ada Letters, 2005, 25(4):1−10. [doi: 10.1145/1104011.1103847]

[65] Singhoff F, Plantec A, Dissaux P, Legrand J. Investigating the usability of real-time scheduling theory with the Cheddar project. Real-Time Systems, 2009,43(3):1−37. [doi: 10.1007/s11241-009-9072-y]

[66] Furness Toolset v1.6 User Guide. 2007. http://www.furnesstoolset.com/

[67] Senn E, Laurent J, Juin E, Diguet JP. Refining power consumption estimations in the component based AADL design flow. In: Forum on Specification, Verification and Design Languages. Washington: IEEE Computer Society Press, 2008. 173−178.

[68] Feiler PH, Rugina A. Dependability modeling with the architecture analysis & design language. Technical Note, CMU/SEI-2007-TN-043, 2007. http://www.sei.cmu.edu/publications/documents/07.reports/07tn043.html

[69] Rugina AE, Kanoun K, Kaâniche M. An architecture-based dependability modeling framework using AADL. In: Proc. of the 10th Int'l Conf. on Software Engineering and Applications. Washington: IEEE Computer Society Press, 2006. 222−227.

[70] Rugina AE, Kanoun K, Kaâniche M. A system dependability modeling framework using AADL and GSPNs. In: de Lemos R, *et al*., eds. Architection Dependable Systems IV. LNCS 4615, 2007. 14−38. [doi: 10.1007/978-3-540-74035-3_2]

[71] Rugina AE, Kanoun K, Kaâniche M. The ADAPT Tool: From AADL architectural models to stochastic Petri nets through model transformation. In: Proc. of the 7th European Dependable Computing Conf. Washington: IEEE Computer Society Press, 2008. 85−90.

[72] Boudali H, Crouzen P, Haverkort BR, Kuntz M, Stoelinga M. Arcade—A formal, extensible, model-based dependability evaluation framework. In: Breitman K, Wookcock J, Sterrit R, Hinchey MJ, eds. Proc. of the 13th IEEE Int'l Conf. on Engineering of Complex Computer Systems. Washington: IEEE Computer Society Press, 2008. 243−248.

[73] Sun HY, Hauptman M, Lutz R. Integrating product-line fault tree analysis into AADL models. In: Cukic B, Dong J, eds. Proc. of the 10th IEEE High Assurance Systems Engineering Symp. Washington: IEEE Computer Society Press, 2008. 15−22.

[74] Hugues J, Zalila B, Pautet L, Kordon F. From the prototype to the final embedded system using the ocarina AADL tool suite. ACM Trans. on Embedded Computing Systems, 2008,7(4):42.1−42.25.

[75] Hansson J, Feiler PH. Data quality attributes in network-centric systems. 2006. http://la.sei.cmu.edu/aadl/documents/JHansson_QualityAttributesInAADL_April_2006.pdf

[76] Eby M, Werner Jan, Karsai G, Ledeczi A. Integrating security modeling into embedded system design. In: Proc. of the Int'l Conf. and Workshop on the Engineering of Computer Based Systems. Washington: IEEE Computer Society Press, 2007. 211−218.

[77] Dwyer MB, Hatcliff J, Hoosier M, Robby. Building your own software model checker using the Bogor extensible model checking framework. In: Etessami K, Rajamani SK, eds. Proc. of the CAV 2005. LNCS 3576, Berlin: Springer-Verlag, 2005. 148−152. [doi: 10.1007/11513988_15]

[78] Fersman E, Pettersson P, Wang Y. Timed automata with asynchronous processes: Schedulability and decidability. In: Proc. of the TACAS 2002. Berlin: Springer-Verlag, 2002. 67−82.

[79] Harbour MG, García JJG, Gutiérrez JCP, Moyano JMD. MAST: Modeling and analysis suite for real-time applications. In: Proc. of the 13th Euromicro Conf. on Real-Time Systems. Washington: IEEE Computer Society Press, 2001. 125−134.

[80] Jürjens J. Towards development of secure systems using UMLsec. In: Proc. of the 4th Int'l Conf. on Fundamental Approaches to Software Engineering. London: Springer-Verlag, 2001. 187−200.

[81] Bodeveix JP, Cavallero R, Chemouil D, Filali M, Rolland JF.A mapping from AADL to Java-RTSJ. In: Proc. of the 5th Int'l Workshop on Java Technologies for Real-Time and Embedded Systems. New York: ACM Press, 2007. 165−174.

[82] Jahier E, Halbwachs N, Raymond P, Nicollin X, Lesens D. Virtual execution of AADL models via a translation into synchronous programs. In: Proc. of the 7th ACM & IEEE Int'l Conf. on Embedded Software. New York: ACM Press, 2007. 134−143.

[83] Hamid I, Zalila B, Najm E, Hugues J. Automatic framework generation for hard real-time applications. Innovations in Systems and Software Engineering: A NASA Journal, 2008,4(1):107−122. [doi: 10.1007/s11334-008-0044-5]

[84] Zalila B, Pautet L, Hugues J. Towards automatic middleware generation. In: Proc. of the 11th IEEE Int'l Symp. on Object-Oriented Real-Time Distributed Computing. Washington: IEEE Computer Society Press, 2008. 221−228.

[85] Vergnaud T, Hugues J, Pautet L, Kordon F. PolyORB: A schizophrenic middleware to build versatile reliable distributed applications. In: Proc. of the 9th Int'l Conf. on Reliable Software Technologies Ada-Europe 2004. Berlin: Springer-Verlag, 2004. 106−119.

[86] The assert-project final report. 2007. http://www.assert-project.net

[87] OMG. Systems Modeling Language V1.0. 2007. http://www.omg.org/cgi-bin/doc?formal/2007-09-01

[88] Mei H, Shen JR. Progress of research on software architecture. Journal of Software, 2006,17(6):1257−1275 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1257.htm [doi: 10.1360/jos171257]

[89] Hugues J, Zalila B, Pautet L. Combining model processing and middleware configuration for building distributed high-integrity systems. In: Miguel M, Kalogeraki V, Kim DH, eds. Proc. of the 10th IEEE Int'l Symp. on Object-Oriented Real-Time Distributed Computing. Washington: IEEE Computer Society Press, 2007. 307−312.

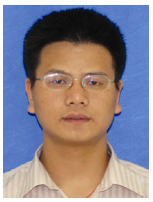
[90] Prasad V, Yan T, Jayachandran P, Li ZZ, Son SH, Stankovic JA, Hansson J, Abdelzaher T. ANDES: An analysis-based design tool for wireless sensor networks. In: Proc. of the 28th IEEE Int'l Real-Time Systems Symp. Washington: IEEE Computer Society Press, 2007. 203−213.

[91] Delanote D, van Baelen S, Joosen W, Berbers Y. Using AADL to model a protocol stack. In: Breitman K, Wookcock J, Sterrit R, Hinchey MJ, eds. Proc. of the 13th IEEE Int'l Conf. on Engineering of Complex Computer Systems. Washington: IEEE Computer Society Press, 2008. 277−281.

[92] Hudak JJ. Analyzable Architectural models of service-based embedded systems. Software Engineering Institute, Carnegie Mellon University, 2008. http://www.cs.kent.ac.uk/events/conf/2008/wads/Slides/hudak.pdf

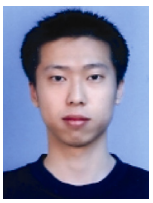
附中文参考文献:

[88] 梅宏,申峻嵘.软件体系结构研究进展.软件学报,2006,17(6):1257−1275. http://www.jos.org.cn/1000-9825/17/1257.htm [doi: 10.1360/jos171257]



杨志斌(1982−),男,江西吉安人,博士生,CCF 学生会员,主要研究领域为实时系统,形式化验证.

顾宗华(1972−),男,博士,副教授,CCF 会员,主要研究领域为实时嵌入式系统.



皮磊(1980−),男,博士生,主要研究领域为实时系统,形式化方法.

马殿富(1960−),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为服务计算,实时系统.

胡凯(1963−),男,博士,副教授,主要研究领域为分布式计算,实时系统.