软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(2):234−253 [doi: 10.13328/j.cnki.jos.004533]

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

可组合嵌入式软件建模与验证技术研究综述*

王 博[1], 白晓颖[1], 贺 飞[2], Xiaoyu SONG[3]

[1](清华大学 计算机科学与技术系,北京 100084)

[2](清华大学 软件学院,北京 100084)

[3](Maseeh College of Electrical and Computer Engineering, Portland State University, Portland, USA)

通讯作者: 王博, E-mail: harvicflyhigh@gmail.com, http://www.tsinghua.edu.cn

摘 要: 可组合嵌入式软件以构件开发技术为基础,研究嵌入式构件的建模、组合性质、构件间组合机制以及组合验证等理论、方法和技术.从组合理论、建模与验证技术这 3 个方面对可组合嵌入式软件的研究现状进行调研分析.组合理论研究给出构件可组合性的乐观定义和悲观定义,从组合操作、组合规则两个方面定义构件间的组合机制.针对嵌入式构件的特点,着重调研了非功能特性和异构构件的建模与组合技术,分析了非功能特性约束、面向多特性的模型等方法.分析了基于契约的验证、基于不变量的验证、基于模型检查的验证等多种嵌入式软件组合验证技术.最后,探讨了需要进一步研究的问题.

关键词: 可组合嵌入式软件;可组合性;相容性;组合模型;组合机制;组合验证

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 王博,白晓颖,贺飞,Song XY.可组合嵌入式软件建模与验证技术研究综述.软件学报,2014,25(2):234−253. http://www.jos.org.cn/1000-9825/4533.htm

英文引用格式: Wang B, Bai XY, He F, Song XY. Survey on modeling and verification techniques of composable embedded software. Ruan Jian Xue Bao/Journal of Software, 2014,25(2):234−253 (in Chinese). http://www.jos.org.cn/1000-9825/4533.htm

Survey on Modeling and Verification Techniques of Composable Embedded Software

WANG Bo[1], BAI Xiao-Ying[1], HE Fei[2], Xiaoyu SONG[3]

[1](Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

[2](School of Software, Tsinghua University, Beijing 100084, China)

[3](Maseeh College of Electrical and Computer Engineering, Portland State University, Portland, USA)

Corresponding author: WANG Bo, E-mail: harvicflyhigh@gmail.com, http://www.tsinghua.edu.cn

Abstract: Based on CBSE (component-based software engineering), this research on composable embedded software investigates the theory, methods and technologies for modeling and verification of embedded components. The paper surveyes the state-of-the-art research and practices on composable embedded system from three perspectives: composite theory, modeling, and verification techniques. It introduces the optimistic and pessimistic definitions of component compatibility, and composition mechanisms including operations and rules. In modeling techniques, the paper particularly addresses the issues of composition of non-functional attributes and heterougeneous components, which are important to embedded components design and verification. It analyzes non-functional attribute constraints and multi-attributes oriented model. The paper also investigates three typical verification techniques of component composition including contract-based, invariants-based, and model checking techniques. It discusses future works in the end.

Key words: composable embedded software; compatibility; composability; composition model; composition mechanism; composition verification

在现代嵌入式系统中,越来越多的复杂、核心任务由软件负责,如算法执行、系统调度等,软件的设计与开

* 基金项目: 国家自然科学基金(61073003, 91218302)
收稿时间: 2013-05-07; 修改时间: 2013-09-29; 定稿时间: 2013-12-05

发已成为嵌入式系统设计与开发的主要问题.大规模、复杂嵌入式软件体系结构呈现出网络化、层次化、分布式的发展趋势,软件规模迅速扩大,大量同构、异构的软件模块间存在复杂的交互关系和协同运行方式,使得嵌入式软件系统设计与开发的复杂性、兼容性、完整性、易变性、重用性等问题日益突出.传统的以人工为主的综合集成过程效率低下,且难以保证软件系统质量.国际上的统计资料表明,由于不能在有效的时间内完成设计并保证系统质量,约有 11%的嵌入式软件项目取消.因此,改进嵌入式软件的设计与开发过程,提高开发效率、提升产品质量、实现敏捷开发,成为嵌入式软件研究的一个关注点.一些学者开始研究具有可组合特性的构件化设计方法,嵌入式软件组合理论由此出现并逐步发展起来.

Kopetz 等人提出了实时嵌入式软件组合构建理论的发展路线图[1],明确了构件化设计、组合构造的思想和原则.嵌入式软件组合理论涉及的研究问题有组合模型、构件可组合性、组合机制、组合操作及其动态语义、组合验证、非功能特性建模与组合、异构构件建模与组合以及组合开发方法与工具等.美国国防先进技术研究管理局(Defense Advanced Research Projects Agency,简称 DARPA)开展了可组合高可信系统(composable high-assurance trustworthy system,简称 CHATS)研究,对影响构件可组合性的因素(如不详细的构件间交互描述、隐性构件属性、不规范的构件封装等)以及改善构件可组合性的方法进行研究,对大规模可信软件系统组合构建进行研究,并给出了软件系统组合架构的一般设计原则[2−4].

组合理论起源于面向对象技术(object oriented programming,简称 OOP),研究类和对象间的组合机制,目的是解决大规模、复杂软件开发问题.在 OOP 领域,类与对象均是对现实世界的抽象描述,具有封装性、继承性等特性,不同对象之间可以同步、异步方式并发运行,由此引出类、对象之间组合的概念以及相关组合理论研究问题.

与通用软件构件相比,嵌入式软件构件具有以下特点:

(1) 应用领域:面向特定应用,专用性强,具有特定应用领域的需求.

(2) 构件形态:嵌入受控设备内部,不仅可以软件模块形态存在,也可以与硬件紧密结合的固件形态存在,后者需要固化存储.

(3) 分布性:大规模、复杂嵌入式软件系统内的构件,随受控设备的分布而具有较强分布性.

(4) 运行环境:可按需定制,差异性大,既可以直接运行于硬件平台上,也可以运行于嵌入式操作系统上,资源有限.

(5) 可信属性:除功能特性外,往往更侧重实时性、资源有限性、安全性等非功能特性,软件特性与硬件特性紧密相关.

(6) 异构性:不同的应用领域、设计语言与范型、接口协议、运行环境,使得嵌入式系统内常常包含异构构件.

针对上述特点,Kopetz 等人提出了实时嵌入式系统组合设计的一般性原则,包括构件应独立开发、构件应提供稳定外部服务、构件间应实现通过通信接口的增量式组合以及构件在系统内应可复制[5].Richling 等人针对实时嵌入式系统提出了架构可组合性的概念[6],关注在特定架构下、按特定规范和流程设计与开发的构件是否都自然地具有可组合性,并通过架构可组合性概念说明了组合设计与开发过程的一般性原理.架构可组合性需要满足以下 3 个要求:第一是可达可组合性,即,对于每一项系统功能/非功能需求,在系统内至少有 1 个构件来实现;第二是安全可组合性,即,参与组合构造的所有构件是正确的,构件间的交互与协同是安全的;第三是实效可组合性,即,在同一架构下,设计与开发的软件构件均满足前两个特性.Richling 等人将架构可组合性应用到嵌入式系统软、硬件协同设计过程中[7].

构件接口是嵌入式构件与组合功能建模的重要依据.接口保证了构件的封装性,通过接口可对构件外部行为特性进行描述,并可屏蔽软件构件、运行平台的结构差异和技术细节.部分学者从构件接口方面开展了可组合嵌入式软件的相关研究,关注通过构件接口组装,实现构件间组合,并对构件接口间在语法、语义上的可组合性进行研究.例如,Kopetz 等人讨论了嵌入式系统信息流接口的分类,将信息流接口分为基本接口和组合接口,研究了基于基本接口、组合接口的构件间组合问题,并进一步讨论了利用事件触发接口、时间触发接口、服务

接口组合构建嵌入式系统的机制[8].其中,在基本接口上,数据信息和控制信息均单向流动;在组合接口上,数据信息为单向流动,而控制信息为双向流动.Kopetz 等人还研究了利用构件间连接接口实现分布式实时系统可组合设计的方法,指出构件间组合可通过接口上的信息交互实现,并对相关构件描述、接口描述方式进行研究,建立了可组合设计的理论基础[9].Li 等人针对嵌入式系统高复杂度、安全苛求、有实时性要求等特点,以通信序列描述语言(communicating sequential language,简称 CSP)为描述语言,对嵌入式构件组合操作的语义进行分类描述,并从构件间接口类型的角度,将组合操作语义归纳为方法调用、数据交换、事件触发这 3 类[10].

非功能特性之间(如实时性等)的组合也是研究的一个关键问题.例如,Kopetz 等人对实时嵌入式系统的时间特性可组合性进行研究,提出了时序防火墙接口[9].时序防火墙接口以时间逻辑描述构件间的实时交互关系,发送者和接收者均有独立缓存,数据从一个发送缓存到另一个接收缓存的传递过程以时间触发方式启动.通过时序防火墙接口,可对不同构件间行为的时间特性进行组合.

本文总结并对比分析了近年来在组合模型、构件可组合性、组合机制、非功能特性建模与组合、组合验证等方面的重要研究工作,并分析后续研究中需要解决的若干关键问题.本文第 1 节介绍嵌入式软件组合理论的基本概念,分析构件可组合性、组合机制.第 2 节讨论组合模型及若干建模问题,包括非功能特性建模与组合、异构构件建模与组合.第 3 节介绍组合验证技术.第 4 节分析探讨需要进一步研究的问题,并对全文进行总结.

1 嵌入式软件组合理论

组合理论起源于 OOP 技术.一些学者对 OOP 领域内的组合模型、组合机制以及组合设计方法进行研究,得到一系列成果,建立了相关理论基础.

Bergmans 等人针对面向对象模型在描述软件并发、同步方面的不足,提出 Composition-Filters 模型[11],将对象模型分为接口、操作集两部分,接口部分负责根据外部消息,判断需要执行的操作集.Composition-Filters 模型实现对象接口与操作的分离,由此解决由于继承异常造成的模型操作与构件并发、同步要求相冲突的问题.Aksit 等人针对不同对象、不同应用领域关注点不同的情况,在 OOP 技术和 Composition-Filters 模型的基础上进一步提出关注点组合以及系统视图的概念[12],其中,视图由不同关注点组合构建,系统由不同视图组合构建,由此为不同系统特性间组合研究做了理论铺垫.Bergmans 和 Aksit 等人也对具有多重关注点的复杂软件组合问题进行了研究[13].随着面向方面编程(aspect oriented programming,简称 AOP)技术的出现与兴起,Havinga 等人进一步对功能/非功能特性间的组合问题也进行了研究,分析了 AOP 领域内组合机制可能面临的问题,包括编程语言间的语法冲突(如 Aspectj 中类的循环继承定义)、不同方面行为特性间的冲突(如 Aspectj 中对源代码内方法的重定义)等[14].

Aksit 等人还进一步利用 Composition-Filters 模型对 Java 和 Smalltalk 语言中类之间的组合进行研究,由此对 OOP 领域内不同类之间的组合机制进行探讨[15].Weiher 等人也对 OOP 领域内的组合机制进行了研究,认为在面向对象编程中,无论继承关系或多态关系,都是类、对象间的组合形式[16].在此基础上,Weiher 等人对组合机制中的接口技术支持、支持大规模软件组合构建的软件架构等问题进行了研究.Nierstrasz 等人认为,OOP 领域内的构件组合存在两个主要缺陷:一是面向对象编程语言缺少一般性组合机制定义;二是面向对象编程语言的一些特征之间存在语义干扰,如父类的修改会影响子类,由此会影响组合机制的正确执行.为了解决以上问题,他们提出设计模式语言的思路[17,18].模式是软件中任何可以重用的语法对象,如类、对象、函数等.通过模式,使面向对象应用程序的开发成为采用预设计模式以及模式间组合构建的过程.Nierstrasz 等人还对 OOP 领域的研究成果做了总结,并对面向对象编程中涉及的组合机制进行了讨论[19].Keller 等人也对基于模式语言的通用软件组合理论进行了研究[20],与 Nierstrasz 等人的研究思路基本相同,都将模式作为构建应用系统的基本对象.

嵌入式软件组合理论是指在特定应用下,针对系统功能/非功能应用需求,描述如何采用系统特性分解和组装技术构建嵌入式软件的概念、原理,以支持嵌入式软件的构件化设计、组合构造、组合验证等工作.

可组合嵌入式系统的研究提出了多种软件构件模型,包括 de Alfaro 等人提出的接口自动机(interface automata,简称 IA)模型[21]、Lynch 等人提出的 I/O 自动机(I/O automat,简称 IOA)模型[22,23]、欧洲 Verimag 实验

室提出的行为-交互-优先级(behavior/interaction/priority,简称 BIP)模型[24]、Alur 等人提出的时间自动机(timed automata,简称 TA)模型[25]以及欧盟 SPEEDS 项目研究提出的混杂特性(heterogeneous rich component,简称 HRC)模型[26−29]、概率自动机(probabilistic automata,简称 PA)模型[30,31]等.Lee 等人从控制系统角度提出嵌入式系统行为建模思想,并利用状态机(finite state machine,简称 FSM)组合理论讨论嵌入式系统的形式化分析和设计[32].

通过组合模型对构件行为特性、组合关系进行描述,是可组合嵌入式软件研究的基础.嵌入式软件组合模型是指在嵌入式构件模型上,增加清晰的可组合性定义,增加清晰的组合操作、组合规则定义,使其适用于描述嵌入式软件构件的组合构造过程.组合模型不仅应支持构件结构、数据与接口协议的静态语义描述,还应支持构件行为、构件间交互关系的动态语义描述.而构件间的组合关系不仅包括接口上的交互关系,还包括构件间的协同运行方式.因此,对可组合性的研究不仅需要对构件接口间的可组合性进行研究,还需要对组合后构件之间的正确协同运行进行研究.

嵌入式软件组合机制是指定义在构件模型上的组合操作、组合规则和约束条件的集合,是组合设计与开发方法的核心内容.组合操作应具有清晰的动态语义,针对不同形式的组合关系,可定义不同形式的组合操作.组合操作可具有多种描述形式,如,在设计建模阶段,可定义模型组合的形式化语言;在运行阶段,可定义构件之间动态绑定的组合关系.组合关系同样具有多种实现方式,如,通过开发框架内构件设计与开发模板间的依赖关系实现,通过中间件、或粘合代码实现等.

组合规则针对不同的组合模型以及不同形式的组合操作分别制定,为构件间的组合操作提供指导,以规范组合操作具体执行过程,辅助推导组合结果的外部行为特性,并为组合结果的验证提供依据.因此,组合规则应包括组合操作适用规则、构件接口信息预处理规则、组合操作执行规则、组合结果处理规则等部分.在实际操作中,可将组合规则与组合操作的形式化描述相结合,从而给出完备的组合操作定义.

约束条件同样针对不同的组合模型描述方式,由专业人员根据领域知识、设计要求制定,以对组合构建过程及结果进行约束,避免非预期状态与行为.这需要针对不同模型元素以及组合操作中的各项原子操作,设计约束条件的一般描述形式,或定义专用约束描述语言,并规定约束条件在组合过程中的语义以及对组合操作的作用方式.

1.1 构件可组合性

可组合性定义是判断不同构件在特定组合关系下是否可进行组合构造的准则[21],只有相互间具有可组合性的构件才可进行组合构造.广义可组合性定义考虑行为特性的保持性,这包含两层含义:一是每个参与组合构造的构件在组合构造前后,在不同外部环境下,是否可保持其外部行为特性;二是组合构造结果的外部行为特性是否满足系统需求.其中,构件运行环境指平台硬件和基础软件资源、与其有交互关系的构件等构成的集合.因此,依据广义可组合性判断构件间的组合关系时,要求对构件接口上的交互关系、构件间的协同运行情况进行全面考察.但由于构件外部环境的复杂性和易变性,对广义可组合性的定义与验证很困难,在验证过程中难以为构件建立充分的外部环境,因而难以对其外部行为特性保持性进行充分的验证.

狭义可组合性定义仅考虑构件间形式上的可组合性,而不考虑行为特性的保持性.它描述软件构件间在形式上的兼容性,多以构件接口行为间的兼容性体现.在目前的研究工作中,各类嵌入式构件模型上的可组合性定义多属于狭义可组合性.构件间狭义可组合性可划分为语法可组合、语义可组合、以及服务可组合这 3 个层次[33−35].语法可组合性描述交互协议、交互行为签名的兼容性,具体包括行为名称、行为参数个数、类型和排列顺序的一致性.语义可组合性描述交互行为的前置条件、后置条件的兼容性,具体包括行为参数值域、参数取值约束、行为时序关系的一致性.服务可组合性还需描述时间约束等非功能特性的可组合性等.

狭义可组合性定义又可分为乐观定义[21,35]和悲观定义[22].在乐观定义中,只要组合状态空间不为空,则判定构件间是可组合的;在悲观定义中,如果组合状态空间中存在非法状态,则判定构件间是不可组合的,即,要求可组合构件在任意运行环境中的行为特性均符合设计要求.

在可组合性乐观定义下,在组合前后,参与组合的构件的外部行为特性将受到不同外部环境的影响,可能造

成构件间交互行为不同步、构件行为不符合行为间时序关系等问题,影响构件间协同运行.因此,正确的构件组合构建的软件系统不一定正确,由此进一步引出了相容性概念.相容性描述软件构件间协同运行的可能性,构件间是相容的,意味着组合结果存在合法状态空间及特定运行环境,在运行环境限制下,可保证各构件在合法状态空间中协同运行.对相容性的验证同样困难,这是因为证明合法环境的存在非常困难.因此,对合法状态空间的求解方法常用于相容性验证.在组合过程中,裁剪组合结果中所有不可达状态、非法状态,如果裁剪后的状态空间不为空,则意味着组合结果可能具有合法运行环境.

构件间可组合性需要在构件模型上进行清晰、准确的定义,并以与构件模型相同的语言进行描述,以指导组合验证工作.构件模型之上的可组合性定义多采用形式化定义,以便作为可组合性验证规则,但具体验证工作还需要验证方法与工具的支持.下面以 IA 模型和 IOA 模型为例,分别说明可组合性的乐观与悲观定义.

可组合性乐观定义更适用于体系结构易变的可伸缩的开放式系统,便于大规模、复杂嵌入式软件的组合构建与动态演化,但具有可组合性的构件间需要进一步验证其相容性,以确定是否存在合法状态空间.悲观定义则更适用于体系结构、构件组织结构相对固定的封闭系统.

IA 是一种轻量级的软件构件接口模型.IA 将接口行为描述与状态机描述相结合,通过确定输入、输出行为的激活状态,建立构件行为间的时序关系[36−39].IA 上的可组合性定义即为一种乐观可组合性定义.

定义 1(接口自动机 IA)[21]. 构件 P 的 IA 定义为一个六元组 $(V_p, V_p^{init}, A_p^I, V_p^O, A_p^H, T_p)$, 其中,

- V_p 是一个有限状态集合;
- $V_p^{init} \subseteq V_p$ 表示初始状态集合,每一个 IA 可包含一组初始状态,如果 $V_p^{init}=\varnothing$,则称 P 为空;
- A_p^I, V_p^O, A_p^H 是 3 个互不相交的集合,分别表示输入行为、输出行为及内部行为集合, $A_p = A_p^I \cup V_p^O \cup A_p^H$ 表示 IA 全部行为的集合;
- T_p 是一组状态间的映射,$T_p \subseteq V_p \times A_p \times V_p$ 表示 IA 状态间的迁移关系集合.

定义 2(IA 可组合性)[21]. 构件 P 和 Q 对应 IA 为 A_P 和 A_Q,A_P 和 A_Q 是可组合的,当且仅当:

$$A_P^H \cap A_Q = \varnothing,\ A_P^I \cap A_Q^I = \varnothing,\ A_P^O \cap A_Q^O = \varnothing,\ A_P \cap A_Q^H = \varnothing.$$

即,A_P 与 A_Q 之间是可组合的,当且仅当除互为输入、输出行为的共享行为之外,A_P 与 A_Q 的其他行为之间都是正交的.由于采用乐观可组合性定义,IA 提出了相容性概念,定义如下:

定义 3(IA 相容性)[21]. 构件 P 和 Q 对应 IA 模型为 A_P 和 A_Q,A_P 和 A_Q 是可相容的,当且仅当 A_P 和 A_Q 非空、可组合,且对于 $A_P \otimes A_Q$,存在一个合法的环境 E.其中,$A_P \otimes A_Q$ 表示 A_P 和 A_Q 之间的组合运算.

IOA 是另一种轻量级的接口模型,与 IA 不同,IOA 是输入使能的,即,在所有状态上、所有输入行为均可被激活,IOA 仅能限制输出行为和内部行为的执行时机.因此,IOA 在描述反应式系统时的形式更复杂.IOA 上的可组合性定义是一种悲观定义.

定义 4(I/O 自动机)[22]. 构件 P 的 IOA 定义为一个五元组($Sig(IOA)$,$states(IOA)$,$start(IOA)$,$steps(IOA)$,$part(IOA)$),其中,

- $Sig(IOA)$表示 IOA 行为的一个划分,包括 3 个互不相交的集合,分别为输入行为集合 $in(IOA)$、输出行为集合 $out(IOA)$及内部行为集合 $int(IOA)$;
- $states(IOA)$是一个有限状态集合;
- $start(IOA) \subseteq states(IOA)$表示初始状态集合;
- $steps(IOA)$是一组状态间映射,$steps(IOA) \subseteq states(IOA) \times acts(IOA) \times states(IOA)$表示 IOA 状态间的迁移关系集合,其中,$acts(IOA)=in(IOA) \cup out(IOA) \cup int(IOA)$;
- $part(IOA)$是等价关系,将 $local(IOA)$分解为可数等价类,其中,$local(IOA)=out(IOA) \cup int(IOA)$.

定义 5(IOA 可组合性)[22]. 构件 P 和 Q 对应 IOA 为 A_P 和 A_Q,A_P 和 A_Q 是可组合的,当且仅当:

$$out(A_P) \cap out(A_Q)=\varnothing,\ int(A_P) \cap acts(A_Q)=\varnothing,\ int(A_Q) \cap acts(A_P)=\varnothing.$$

即,A_P 与 A_Q 之间是可组合的,当且仅当除互为输入、输出的共享行为以及共同的输入行为之外,A_P 与 A_Q 的其他

行为之间都是正交的.IOA 可组合性定义在形式上与 IA 可组合性定义类似,不同的是,由于 IOA 是输入使能的,因此在组合前后的任意运行环境中均应能正常响应环境的所有输入,并保证其行为特性符合设计要求,而不允许进入非法状态.因此,IOA 的可组合性定义是一种悲观定义.

BIP 模型、PA 模型上的可组合性定义也是一种乐观定义,与 IA 模型之上的可组合性定义类似.在组合理论研究中,多利用 PA 模型描述构件行为和状态迁移的不确定性以及组合构造结果的不确定性[40−43].

IA 模型、IOA 模型、BIP 模型、PA 模型上的可组合性定义,均试图在构件模型上给出构件可组合性的一般形式化定义.在后续研究中,针对构件间不同类型的组合关系,针对不同的功能/非功能特性,可进一步制定构件间可组合性的分解定义.

Mouelhi 等人提出了语义接口自动机(semantical interface automata,简称 SIA)模型[34].在 SIA 中,定义了行为参数(parameter)、共享变量(shared variable)和局部变量(local variable).通过局部变量描述构件所维护的内部变量,通过共享变量描述构件与环境或其他构件间共享的变量,以对构件之间的一致性状态进行统一描述,并通过行为与对应参数定义行为签名.由此,通过 SIA 描述构件接口行为语义,并在此基础上讨论构件间语义可组合性问题.

定义 6(语义接口自动机 SIA)[34]. 构件 P 的 SIA 定义为一个九元组 $(S_p, Init_p, \Sigma_p^I, \Sigma_p^O, \Sigma_p^H, \delta_p, L_p, V_p, \Psi_p)$, 其中,

- S_p 是一个有限状态集合;
- $Init_p$ 表示初始状态集合;
- $\Sigma_p^I, \Sigma_p^O, \Sigma_p^H$ 是 3 个互不相交的集合,分别表示输入行为、输出行为及内部行为集合, $\Sigma_p = \Sigma_p^I \cup \Sigma_p^O \cup \Sigma_p^H$ 表示 SIA 全部行为的集合;
- $\delta_p \subseteq V_p \times \Sigma_p \times V_p$ 表示状态间的迁移关系集合;
- L_p 表示 SIA 使用的局部变量集合;
- V_p 表示 SIA 使用的共享变量集合;
- Ψ_p 表示 SIA 中构件行为 $a \in \Sigma_p$ 的前置条件 $Pre_p(a)$与后置条件 $Post_p(a)$的集合,其中,$Pre_p(a)$= $Preds(V_p \cup P_a^i)$,而 $Post_p(a)$=$Preds(V_p \cup P_a^i \cup P_a^o)$.其中,$Preds(X)$表示变量集合 X 内变量构成的一阶逻辑表达式, P_a^i 与 P_a^o 分别表示行为 a 的输入与输出参数集合.

定义 7(SIA 可组合性)[34]. 构件 P 和 Q 对应 SIA 为 A_P 和 A_Q,A_P 和 A_Q 是可组合的,当且仅当:

$$A_P^H \cap A_Q = \varnothing,\ A_P^I \cap A_Q^I = \varnothing,\ A_P^O \cap A_Q^O = \varnothing,\ A_P \cap A_Q^H = \varnothing.$$

- $\forall a \in shared(A_P, A_Q)$,相应输入行为与输出行为在共享变量集合 $V_P \cap V_Q$ 上的作用一致;
- $\forall a \in shared(A_P, A_Q)$,如果 a 在 A_P 中为输出行为,对应行为签名为 $a(o_1,\ldots,o_n)$,在 A_Q 中为输入行为,对应行为签名为 $a(i_1,\ldots,i_n)$,则 $D_{o_k} \subseteq D_{i_k}$,$1 \leqslant k \leqslant n$,其中,$D_\chi$表示变量$\chi$的合法取值空间;

其中,$shared(A_P, A_Q)$表示 A_P 与 A_Q 的共享行为集合,$shared(A_P, A_Q) = (A_P^I \cap A_Q^O) \cup (A_P^O \cap A_Q^I)$.通过以上定义,可对构件间接口行为上的参数取值约束及行为执行约束进行考察,判断构件间是否具有语义层面的可组合性.

1.2 组合机制

组合机制是组合理论研究的核心内容,也是嵌入式软件组合开发方式与传统开发方式最大的区别.组合机制通过制定统一的组合操作、组合规则与约束条件,规范软件构建过程,可有效解决开发过程中的复杂性、兼容性、重用性等问题.

组合机制需要针对构件间不同形式的组合关系,采用与组合模型一致的方式描述.嵌入式软件构件之间的组合关系多样化,例如,构件间可以是有序的前驱与后继关系,也可以完全独立运行或并发运行.针对不同的组合关系,组合操作将具有不同的形式.因此,组合操作的定义可分为完备定义与分解定义.其中,完备定义支持在所有类型组合关系下进行组合构造,而分解定义仅支持在特定类型组合关系下进行组合构造.

1.2.1 组合操作

构件间的组合关系可分为顺序组合关系、平行组合关系、并行组合关系这 3 类[41,44,45],其中,平行组合关系最简单,而并行组合关系最复杂.在顺序组合中,各构件根据数据流、控制流关系,按照一定的先后次序嵌入相应位置进行组合,其先后次序固定.平行组合较为简单,由于各构件独立运行,构件间无交互关系,其组合仅仅是将不同构件平行放置在一起.在并行组合中,构件间有交互关系和协同运行需求,其组合需要完整构建各构件之间的交互关系,并在必要时从外部环境对构件的运行进行协调与控制.目前,主要组合模型上的组合操作定义均是完备定义,而缺乏针对不同组合关系的组合操作分解定义,可在后续工作中研究制定.如,在 IA 模型、BIP 框架模型中的构件模型、PA 模型以及 HRC 模型上,均给出了完备的组合操作形式化定义.

各类模型上的组合操作定义类似,通常以笛卡尔积运算为基础,对不同构件模型的状态集合、初始状态、行为集合、迁移关系集合等执行积运算,并依据特定规则对运算结果进行处理.

IA 模型之上的组合乘积定义如下:

定义 8(IA 组合乘积)[21]. 构件 P 和 Q 对应 IA 为 A_P 和 A_Q,则其组合乘积为 $A_P \otimes A_Q$,定义如下:

$$V_{P\otimes Q}=V_P\times V_Q;V^{init}_{P\otimes Q}=V^{init}_P\times V^{init}_Q;$$

$$A^I_{P\otimes Q}=(A^I_P\cup A^I_Q)\setminus shared(A_P,A_Q);A^O_{P\otimes Q}=(A^O_P\cup A^O_Q)\setminus shared(A_P,A_Q);A^H_{P\otimes Q}=A^H_P\cup A^H_Q\cup shared(A_P,A_Q);$$

$$\begin{aligned}T_{P\otimes Q}=(&\{((p,q),a,(p',q))\mid(p,a,p')\in T_P\wedge a\notin shared(A_P,A_Q)\}\cup\\&\{((p,q),a,(p,q'))\mid(q,a,q')\in T_Q\wedge a\notin shared(A_P,A_Q)\}\cup\\&\{((p,q),a,(p',q'))\mid(p,a,p')\in T_P\wedge(q,a,q')\in T_Q\wedge a\in shared(A_P,A_Q)\}).\end{aligned}$$

其中,$shared(A_P,A_Q)$表示 A_P 与 A_Q 的共享行为集合,$shared(A_P,A_Q)=(A^I_P\cap A^O_Q)\cup(A^O_P\cap A^I_Q)$, $P\otimes Q$ 表示组合构建得到的 IA.首先,组合结果的输入行为集合、输出行为集合应分别包括子系统与环境之间的所有输入行为、输出行为,且不包括构件间共享行为;其次,对于互为共享行为的输入行为与输出行为,其在组合结果中共同完成子系统内的构件间交互,因此合并为一项内部行为;再次,互为共享行为的输入行为与输出行为,在对应构件中引起的状态迁移也将被合并.

当进行语义、服务层面的组合时,需要进一步考虑构件间接口参数取值范围、接口行为前置与后置条件之间的组合,并针对组合结果讨论可能产生的异常与风险类型.例如,对于互为输入、输出行为的共享行为,如果输入行为的输入参数χ_i的取值范围为D_{χ_i},则对应输出行为的输出参数χ_o的取值范围为D_{χ_o}.当$D_{\chi_i}\subset D_{\chi_o}$时,对于超出$D_{\chi_i}$取值范围的数据,输出行为可发送,输入行为却无法接收.对于此类异常情况,需要依据特定组合规则对组合结果进行处理,并采取必要措施,保证系统运行不进入异常状态.在后续研究工作中,需要对此类问题进行更深入的探讨.

1.2.2 组合规则

Reussner 等人首次将契约理论引入基于构件的设计与开发过程中[46],将契约从传统 C/S 架构设计领域的方法契约推广到组合理论领域的组合契约,使基于契约的设计思想在组合设计与开发过程中得到应用.将契约理论引入组合理论的目的是:通过建立构件契约、明确构件职责的划分,规范构件的设计与开发,并通过制定构件契约组合规则指导构件间的组合构造,辅助推导组合结果的外部行为特性,并提供构件间组合验证的理论基础.

契约是软件构件外部行为特性的抽象描述,定义如下:

定义 9(构件契约). 一个构件契约具有如下描述:

$$Contract=(Assume,Guarantee),即,C=(A,G),$$

其中,A 为假设部分,描述构件对运行环境的需求;G 为保证部分,描述当运行环境需求被满足时,构件可保证的外部行为特性,即,构件承担的职责.

对于一个软件构件,契约中的保证部分即是其设计任务.一个构件契约的假设部分被满足时,构件实现的外部行为特性满足构件契约的保证部分,则称构件实现符合构件契约,实现了其所担负的职责.

在进行构件间组合的过程中,构件间的组合对应构件契约的组合,且构件组合结果应符合构件契约组合结

果.在以前的研究工作中,针对各类组合关系提出了各种形式的假设/保证(A/G)规则,示例如下:

定理 1(顺序组合 A/G 规则). 两个构件 C_1 与 C_2 顺序组合,前驱构件契约的保证为后继构件契约的假设,即,C_1 对应构件契约的保证为 C_2 对应构件契约的假设:

$$\frac{\begin{array}{c}C_1\models A\\(A)C_2(G)\end{array}}{C_1\,\|\,C_2\models G},$$

其中,"$\models$"表示构件满足相关契约假设或保证,"$\|$"表示构件契约之间的组合.

定理 1 给出的 A/G 规则含义如下:如果构件 C_1 满足契约 1 的保证 A,而契约 1 的保证是契约 2 的假设,则构件 C_1 与 C_2 组合后,所得子系统满足契约 2 的保证 G.

定理 2(平行组合 A/G 规则). 两个构件 C_1 与 C_2 平行组合:

$$\frac{\begin{array}{c}(A_1)M_1(G_1)\\(A_2)M_2(G_2)\end{array}}{(A_1,A_2)M_1\,\|\,M_2\models(G_1\vee G_2)}.$$

定理 2 给出的 A/G 规则含义如下:如果构件 C_1 与 C_2 平行组合,则构件 C_1 与 C_2 组合后,所得子系统在契约 1 与契约 2 的假设(A_1,A_2)下,满足契约 1 或契约 2 的保证 $G_1\vee G_2$.

定理 3(并发组合 A/G 规则). 两个构件 C_1 与 C_2 并发组合:

$$M_1\,\|\,M_2=(A,G)\text{ where }\left\{\frac{A=(A_1\wedge A_2)-(G_1\wedge G_2)}{G=(G_1\wedge G_2)}\right\}.$$

定理 3 给出的 A/G 规则含义如下:如果构件 C_1 与 C_2 并发组合,则构件 C_1 与 C_2 组合后,所得子系统在契约 1 与契约 2 的假设($A_1\wedge A_2$)–($G_1\wedge G_2$)下,满足契约 1 与契约 2 的保证 $G_1\wedge G_2$.其中,($A_1\wedge A_2$)–($G_1\wedge G_2$)表示从契约 1 与契约 2 的假设中剔除由另一构件契约的保证提供的假设.

根据各种形式的 A/G 规则,由构件契约组合可得到子系统、系统契约.设计人员可对子系统、系统契约进行检查,确认其是否满足设计要求,并据此对构件组合结果进行验证.

如果嵌入式构件模型的定义支持对契约的充分描述,则称该模型为基于契约的模型.基于契约的构件模型蕴含了构件契约描述,建立构件模型即相当于建立了构件契约,而构件组合也自然地对应构件契约的组合,有相应的组合规则支持.当前,主要的组合模型,如 IA 模型、BIP 模型、PA 模型、HRC 模型等,都是基于契约的构件模型.其中,在 SPEEDS 框架内,可基于 HRC 模型,针对构件功能/非功能特性分别建立构件契约,并组合构建子系统、系统契约,并由构件契约、子系统契约和系统契约构成针对某种特性的层次化视图.

此外,Schmidt 等人提出的 RADL 模型也是一种基于契约的模型[47].其设计目的就是通过对契约进行描述,为复杂分布式实时系统提供职责分解与分配支持,并为构件建模、组合机制、组合验证工作提供契约描述支持.在 RADL 模型中,定义了请求 Gate 和服务 Gate:请求 Gate 描述构件的外部环境需求,对应契约中的假设部分;服务 Gate 描述构件提供的服务,对应契约中的保证部分.Schmidt 等人在使用 RADL 模型对分布式实时系统的层次化组合构建进行描述、验证的基础上,还讨论了基于契约的构件可组合性推导问题.

2 组合模型及若干建模问题

2.1 组合模型

基于组合模型的嵌入式软件组合设计过程可分为 4 个阶段:一是系统体系结构设计,即,规划系统、子系统内的构件组织结构,进行构件间职责分解与分配,设计构件间交互关系,设计构件间调度与控制策略,确定构件运行平台;二是构件模型设计,即,根据构件设计要求,使用特定组合模型,在可组合性定义指导下建立可组合构件模型;三是构件模型组合与验证,即,通过科学的组合机制,将构件模型组合构建为子系统、系统模型,并根据组合规则与约束条件对组合构建结果进行验证;四是依据构件、子系统模型进行的后续设计与开发工作.

在基于组合模型的组合设计过程中,根据设计要求选用适当的组合模型建立构件模型,通过组合构建子系

统、系统模型,并通过仿真与验证检查模型正确性.层次化的构件、子系统和系统模型为后续设计与开发工作提供可视化的分解设计方案,并支持基于模型的系统动态测试.

从模型描述对象的角度,组合模型可分为接口模型、行为模型、框架模型等:

- 接口模型关注对构件接口行为的描述,如 IA 模型、IOA 模型即为一种轻量级的软件构件接口模型[21,22].其中,IA 模型与 IOA 模型都可对构件接口进行建模,将接口行为描述与状态机描述相结合,建立构件行为间时序关系,支持构件模型之间的可组合性与相容性验证.所不同的是,IOA 模型是输入使能的.
- 行为模型关注对构件行为的描述,如 TA 模型、PA 模型即为软件构件行为模型[25,30].其中,TA 模型通过为行为增加时间约束,支持构件行为时间特性建模,支持构件行为时间特性的分析与验证.PA 模型通过不同行为分支上的概率分布,可对构件行为、状态迁移的不确定性进行量化描述.
- 框架模型是一组模型的集合,在框架内利用不同模型,对嵌入式软件系统内各构成要素分别进行建模,并对各要素之间的组合方式进行描述,以构建完整的软件系统框架模型.BIP 模型即为一种软件设计框架模型,通过对软件构件、构件间交互关系和调度策略、定时机制、资源管理机制等进行建模,可为嵌入式软件组合设计提供完整的框架模型支持[24].MADES 框架模型同样包含一系列底层模型,如扩展行为模型、扩展状态机、扩展行为序列、扩展用例模型以及提供系统时基的时钟模型等,通过配合使用,以完整地描述软件系统行为特性[48−53].

组合模型具有图形描述、形式化语言描述等描述方式,几乎所有的组合模型都具有专用形式化描述语言.如,IA 模型、IOA 模型、BIP 模型、TA 模型、PA 模型、HRC 模型等均具有专用形式化描述语言,其中,不同的 BIP 框架模型还可使用类 C++的 BIP 语言描述,而 MADES 框架模型则在 MARTE 语言的基础上开发其底层模型专用的形式化描述语言.此外,IA 模型、IOA 模型、TA 模型、PA 模型等还具有相应图形描述方式,其中,BIP 框架模型还具有 EMF 扩展图形描述.此外,如果组合模型支持对契约的充分描述,则该组合模型为基于契约的模型.在基于契约的模型之上,可开展基于契约的组合验证(见表 1).

Table 1 Specification of composition model

表 1 组合模型描述方式

模型名称	IA 模型	IOA 模型	BIP 模型	TA 模型	PA 模型	HRC 模型	MADES 框架模型
形式化语言	IA 形式化描述	IOA 形式化描述	类 C++的 BIP 语言; Petri 网; TA	TA 形式化描述	PA 形式化描述	CSL(contract specification language) 形式化语言; HRC 形式化描述	基于 MARTE 形式化语言; Modelica 语言
图形描述	IA 图形描述	IOA 图形描述	EMF 元模型扩展图形描述	TA 图形描述	PA 图形描述	HRC 图形描述	SysML 模型元素; UML 绘图元素
契约描述	组合契约描述	组合契约描述	组合契约描述	组合契约描述	组合契约描述	组合契约描述	–

从可组合性的角度,组合模型可分为具有乐观可组合性定义的模型、具有悲观可组合性定义的模型等.其中,IA 模型、BIP 模型、TA 模型、PA 模型、HRC 模型等均是具有乐观可组合性定义的模型,而 IOA 模型则是具有悲观可组合性定义的模型.

从组合机制的角度,组合模型可分为具有组合操作完备定义的模型、具有组合操作分解定义的模型等.其中,IA 模型、IOA 模型、BIP 框架模型中的构件模型、TA 模型、PA 模型、HRC 模型等均是具有组合操作完备定义的模型.此外,在 BIP 框架模型中,可利用连接器模型对构件间交互关系和调度策略进行建模,以描述构件间不同类型的组合关系.因此,通过不同的连接器模型,可支持不同组合关系类型下组合操作的分解定义.

从针对的行为特性类型的角度,组合模型可分为功能模型、时间模型、平台资源模型等,其中,几乎所有的组合模型或其扩展都支持功能特性、时间特性的描述;BIP 框架模型还支持对资源特性进行描述,可对系统资源有限性、管理机制以及构件与目标平台资源之间的映射关系进行描述;HRC 模型也支持对资源特性的描述;

MADES 框架模型中的资源分配模型也描述了功能单元到资源之间的映射关系.此外,PA 模型还支持采用概率方法,对软件构件、子系统的安全性、可靠性进行描述与分析(见表 2).

Table 2 Behavior property of composition model

表 2 组合模型行为特性

模型名称	IA 模型	IOA 模型	BIP 模型	TA 模型	PA 模型	HRC 模型	MADES 框架模型
特性支持	功能特性; 时间特性	功能特性; 时间特性	功能特性; 时间特性; 资源特性	功能特性; 时间特性	功能特性; 时间特性; 安全特性	功能特性; 时间特性; 资源特性	功能特性; 时间特性; 资源特性

2.2 非功能特性建模

嵌入式软件由于其特定的应用领域和应用环境,不仅关注功能特性,更关注非功能特性,如实时性、资源有限性、安全性、功耗等.如对于具有强实时性要求的嵌入式软件,实时性指标是其核心指标,系统中某些构件任务的超时可能造成系统任务的失败,甚至对软件系统造成破坏性影响.此外,由于嵌入式系统安装空间、运行环境所限,嵌入式软件运行平台的运算和处理能力、存储空间、通信信道带宽等资源通常是有限的.而对于安全苛求系统而言,安全性则是最重要的系统特性.伴随着处理器等电子元器件性能的提升,器件的功耗水平也在逐步提升,对于机载、车载系统等供电能力有限的嵌入式系统,功耗问题将更加突出.因此,嵌入式软件的设计与开发应充分考虑各类特性,以构建完整的软件系统.

非功能特性建模与组合同样需要组合模型、组合机制、组合验证理论和方法支持.功能特性的描述方式和组合机制通常不适用于非功能特性,因此需要针对不同非功能特性,在构件模型中定义其描述方式和组合机制,并制定不同功能/非功能特性间的横向组合机制.

非功能特性的建模与组合主要有两种技术途径:

- 第一,在构件模型上添加非功能特性约束,对模型描述与组合机制进行扩展,使其适用于非功能特性建模与组合;
- 第二,建立面向多特性的组合模型,并制定相应组合机制.

同样,可利用组合契约对构件、子系统的非功能特性进行描述.此时,契约保证部分描述的外部行为特性是特定非功能特性.利用契约可描述构件间非功能特性的组合规则,以指导非功能特性组合过程,并为非功能特性组合结果的验证提供依据.

在构件模型上为构件行为添加非功能特性约束,是最常见的获得非功能特性支持的方法[54].构件行为的激活条件或执行结果均应满足相关约束,构件间的组合构造同时包括这些约束间的组合构造.例如,针对资源有限性,在构件模型描述中增加目标平台资源约束条件的描述,在组合构造时不仅对其功能特性进行组合,对相应平台资源约束条件也进行组合,并可依据物理资源量对组合资源约束条件进行验证,检验物理资源量是否满足需求.

在 TA 模型中,通过定义时钟变量集合和时间约束集合对自动机描述进行扩展,使其适用于描述嵌入式系统时间特性[25].

定义 10(时间自动机 TA)[25]. 时间自动机(TA)定义为一个六元组(L,L^0,Σ,X,I,E),其中,

- L 是一个有限状态集合;
- $L^0\subseteq L$ 表示初始状态集合;
- Σ是一个有限行为集合;
- X 是一个有限时钟变量集合;
- I 是一个映射关系集合,将$\forall s\in L$ 与其时间约束ϕ联系起来;
- $E\subseteq L\times\Sigma\times\Phi(X)\times 2^X\times L$ 是一个迁移关系集合,$\langle s,a,\phi,\lambda,s'\rangle$表示一个迁移,其中,$s$ 与 s'分别表示源状态和目标状态;a 表示引起迁移的行为;ϕ表示状态 s 上的时间约束,以时间变量取值约束的形式描述(如 $value_1\leqslant\chi\leqslant value_2,\chi\in X$);$\lambda\in X$ 表示迁移完成后、需要置零的时钟.通过时间约束描述行为执行以及相应状态迁

移过程需要满足对应时间约束,从而描述构件行为的时间特性.

在执行 TA 间组合时,需要对时钟变量集合以及时间约束集合进行组合,即,进行构件时间特性的组合.

定义 11(TA 组合乘积)[25].设 $A_1=\langle L_1, L_1^0, \Sigma_1, X_1, I_1, E_1\rangle$和 $A_2=\langle L_1, L_2^0, \Sigma_2, X_2, I_2, E_2\rangle$是两个时间自动机,当 X_1 与 X_2 相对独立,即,没有公共时钟时,A_1 与 A_2 的组合 $A_1\|A_2$ 表示为

$$A_1 \| A_2 = \langle L_1 \times L_2, L_1^0 \times L_2^0, \Sigma_1 \cup \Sigma_2, X_1 \cup X_2, I, E\rangle,$$

其中,时间约束集合的组合结果表示为

$$I(s_1,s_2)=I(s_1)\wedge I(s_2).$$

迁移关系集合的组合结果表示为:

$$a \in \Sigma_1 \cap \Sigma_2, \langle (s_1,s_2), a, \phi_1 \wedge \phi_2, \lambda_1 \cup \lambda_2, (s_1',s_2')\rangle \in E_{A_1\|A_2};$$
$$a \in \Sigma_1 \setminus \Sigma_2, \langle (s_1,s_2), a, \phi_1 \wedge \phi_2, \lambda_1 \cup \lambda_2, (s_1',s_2)\rangle \in E_{A_1\|A_2};$$
$$a \in \Sigma_2 \setminus \Sigma_1, \langle (s_1,s_2), a, \phi_1 \wedge \phi_2, \lambda_1 \cup \lambda_2, (s_1,s_2')\rangle \in E_{A_1\|A_2}.$$

de Alfaro 等人参考 TA 模型对时间特性的描述,进一步对 IA 模型进行扩展,增加时钟变量集合,增加输入行为、输出行为时间约束描述,使其适用于描述构件时间特性,并对组合操作进行扩展,以支持时间特性的组合[55].

定义 12(时间接口自动机 TIA)[55]. 时间接口自动机(TIA)定义为一个八元组$(Q,q^{init},X,Acts^I,Acts^O,Inv^I,Inv^O,\rho)$,其中,

- Q 是一个有限状态集合.
- q^{init} 表示初始状态.
- X 是一个有限时钟变量集合.
- $Acts^I$ 和 $Acts^O$ 表示有限输入行为集合、输出行为集合,且两个集合相互独立.
- Inv^I 和 Inv^O 表示两个映射关系集合,分别将 Q 中的每个状态 s 与其输入行为的时间约束、输出行为的时间约束联系起来.
- $\rho\subseteq Q\times[X]\times Acts\times 2^X\times Q$ 是一个迁移关系集合,代表状态间的迁移关系,其中,$[X]$表示时钟变量取值约束集合.$\langle q,g,a,r,q'\rangle$表示一个迁移,其中,$q$ 和 q'表示源状态和目标状态;a 表示引起迁移的行为;$r\in X$ 表示需要置零的时钟;$g\in[X]$表示时钟变量值的监视条件,由一组时钟变量上的时间约束组成,当约束条件满足时执行相应行为,进行状态迁移.

定义 13(TIA 可组合性)[55]. 设 A_1 和 A_2 是两个 TIA,A_1 和 A_2 是可组合的,当且仅当 $Acts_1^0 \cap Acts_2^0 = X_1 \cap X_2 = \varnothing$.

同样,可基于 TIA 进行时间特性的组合.

定义 14(TIA 组合乘积)[55]. 设 A_1 和 A_2 是两个可组合 TIA,其组合乘积 $A_1\otimes A_2$ 表示为

$$Q_{A_1\otimes A_2} = Q_{A_1} \times Q_{A_2};$$
$$q^{init}_{A_1\otimes A_2} = (q^{init}_{A_1}, q^{init}_{A_2});$$
$$X_{A_1\otimes A_2} = X_{A_1} \cup X_{A_2};$$
$$Acts^I_{A_1\otimes A_2} = (A^I_{A_1} \cup A^I_{A_2}) \setminus shared(A_1,A_2);$$
$$Acts^O_{A_1\otimes A_2} = (A^O_{A_1} \cup A^O_{A_2}) \setminus shared(A_1,A_2);$$
$$Inv^I_{A_1\otimes A_2}(p,q) = Inv^I_{A_1}(p) \wedge Inv^I_{A_2}(q);$$
$$Inv^O_{A_1\otimes A_2}(p,q) = Inv^O_{A_1}(p) \wedge Inv^O_{A_2}(q).$$

$\rho_{A_1\otimes A_2}$ 表示迁移集合,其中任意一个迁移表示为

$$((q_1,q_2), g_1 \wedge g_2, a, \{r_1\} \cup \{r_2\}, (q_1',q_2')),$$

其中,$shared(A_1,A_2)=Acts_A\cap Acts_B$ 表示 A_1 与 A_2 的共享行为集合.

此外,一些学者还研究了 IOA 模型、BIP 模型、PA 模型之上的时间约束描述与组合问题.如,David 等人和 Kaynar 等人提出的 Timed I/O Automata 同样通过在构件行为上添加时间约束对时间特性进行描述[56,57].

Kopetz 等人针对嵌入式系统实时性提出了 TT(time-triggered)模型[58].该模型通过实时性实体和实时性映像描述构件间实时交互关系.其中,实时性实体是一组变量,反映原构件的属性或行为参数度量;实时性映像是原构件的实时性实体在与其存在交互关系的目标构件中的映像.在建模过程中,根据构件间交互的实时性,规定在一定时间域内,实时性映像相对实时性实体是精确的,可准确反映原构件的相应度量.因此,利用实时性映像可描述特定时间域内、目标构件对原构件的各类度量的约束条件,或描述目标构件从原构件感知的属性或行为参数度量,从而为构件间交互行为建立时间约束,对构件间实时交互关系进行描述.

在 UML 中,同样可采用约束条件对模型的非功能特性进行描述.针对 UML 图形符号无法描述模型元素约束条件的细节,OMG 从 UML1.1 版本起正式采用模型约束语言(object constraint language,简称 OCL).OCL 是一种声明性、形式化、无二义性的语言,可对模型元素属性、行为前置和后置条件进行描述[59,60].OCL 是一种表达式语言,包括类型、常量与变量、操作、表达式、语句等要素.通过 OCL 语言,可为构件模型元素、行为建立各类功能/非功能特性约束条件,以准确描述构件行为特性.针对 UML 在描述嵌入式实时系统时表达能力不足的问题,OMG 进一步制定了 MARTE 建模规范.MARTE 引入了时间过程、时间事件、资源等概念,提供一套通用的非功能特性描述机制,提供用于非功能特性建模的基本元素以及用于系统特性描述的高级元素,构成嵌入式实时系统建模框架.

在欧盟和瑞士政府的资助下,欧洲多家科研机构和企业合作开展了 PECOS(pervasive component system)项目研究工作[61−64],致力于为嵌入式系统开发提供基于构件的设计技术支持.PECOS 项目研究团队认为,正是非功能特性限制了基于构件的设计技术在嵌入式软件开发中的应用,因此特别关注对嵌入式系统非功能特性的研究.在 PECOS 框架内,可使用 Timed Petri Net 描述构件模型,通过添加时间约束,支持对行为时间特性的描述,并可在工具支持下进行基于时间特性的行为预测以及生成实时调度策略等.

另一种获得非功能特性支持的途径是建立面向多特性的组合模型,支持在同一模型中对功能/非功能特性进行描述和组合构建.在面向多特性的组合模型之上,嵌入式软件具有多个视图,不同视图对应不同功能/非功能特性,从不同角度对软件进行描述,如实时性视图描述软件时间特性.通过视图内组合构造以及视图间组合构造,得到完整多视图系统.通过这种方式获得非功能特性支持的主要有 HRC 模型等.

在欧盟第六框架协议下,空中客车公司等 20 余家企业与科研机构共同开展了 SPEEDS(speculative and exploratory design in system engineering)项目研究工作[26−29].SPEEDS 项目旨在为嵌入式系统设计与开发提供基于模型的设计技术支持,并提供构件化设计、开发、测试、集成、验证技术支持.HRC 模型是 SPEEDS 框架的底层模型,SPEEDS 研究工作在 HRC 模型上提供了一套设计、建模、组合构建和分析验证技术,并将相关技术与工具进行集成.

SPEEDS 项目研究的一个主要问题就是:在系统设计与开发过程中,如何克服多特性约束.如,在实时性、资源有限性、可靠性、安全性等非功能特性的多维约束下,如何得到健壮、灵活的系统设计方案.SPEEDS 项目定义的混杂特性(heterogeneous rich component,简称 HRC)模型如下:

定义 15(HRC 模型)[26]. HRC 模型定义为一个 8 元组$(V,P,G,init,inv,flow,final,trans)$,其中,

- $V=V_d+V_c$ 是一个有限变量集合,其中,V_d 是离散变量集合,V_c 是连续变量集合.
- P 是一个有限接口集合.
- G 是一个状态转换图,$G=(L,E)$,其中,L 是一个有限控制位置集合,E 是一个有限开关集合.
- $Init,inv,flow,final,trans$ 为 5 个函数,定义如下:
  - $inv(l)$表示有一些事件,可以引起从控制位置 l 发生迁移,并迁移到其他控制位置;
  - $flow(l)$表示在控制位置 l 上,发生状态变量的连续变化;
  - 如果控制位置状态对(s,l)可达,且 $init(s,l)$=true,则控制位置状态对(s,l)为初始状态;
  - 如果控制位置状态对(s,l)可达,且 $final(s,l)$=true,则控制位置状态对(s,l)为终止状态;
  - $trans(e)\subseteq S\times\Lambda\times S$ 表示由开关 e 触发的离散状态迁移,其中,$\Lambda=_{def}(P\rightharpoonup D)$,表示一个离散迁移过程.

在 HRC 模型中,不同功能/非功能特性的度量可以模型内特定离散、连续变量的值描述.由此,描述不同特

性的变量可以被组合在同一个 HRC 模型之中,从而解决了不同特性间横向组合的问题.在 HRC 模型基础上,研究人员设计了混合特性软件系统建模解决方案.在方案中,针对不同功能/非功能特性分别建立视图,并建立对应系统契约.在特定视图内对系统契约进行分解,生成对应构件契约,描述构件相关特性.再根据不同视图内的构件契约,建立构件 HRC 模型.由此,SPEEDS 框架不仅建立了软件系统与构件之间纵向的组合关系,也建立了不同系统特性之间横向的组合关系.

2.3 异构构件建模

现代嵌入式软件体系结构多样化,如工业控制系统具有分布性、层次化特点,而新一代机载航电系统则具有综合模块化结构等.在嵌入式系统内,设备平台之间存在很大差异,软件构件间也存在异构性.嵌入式软件构件间的异构性通常由以下因素造成:第一,应用领域、设计要求不同.针对不同的设计要求,构件具有不同的设计方案,具有不同形式的控制流与数据流.第二,设计范型不同.依据不同范型设计的构件,其内部结构通常不同.第三,描述语言不同.使用不同描述语言设计的构件,其描述的形式不同.第四,接口协议不同.不同构件之间有时采用不同的接口协议.第五,运行平台不同.为了适应不同的运行环境,构件的特征、行为特性通常有差异.

基于模型的设计技术已在嵌入式软件设计中得到广泛应用,在以前的研究中,提出了很多嵌入式软件模型,模型语法、语义之间存在很大差别.此外,复杂、大规模嵌入式软件的设计过程往往需要使用多种面向领域的模型语言与相应支撑工具,以有针对性地开展子系统构建、分析和仿真.不同模型之间同样具有不同的语法和语义,相互之间难以进行有效的转换与集成,进一步增加了软件设计工作的复杂度.因此,只有在嵌入式软件组合构建过程中充分考虑构件间异构性,制定处理异构性的有效方法,才能在各类异构构件间进行正确组合构造.

连接模型可屏蔽异构软件构件的结构差异和技术细节,通过连接模型的连接与调度,实现异构构件间的组合,其在本质上是一种模型中间件.

在 BIP 框架内,不仅可利用原子构件模型对软件构件进行建模,还可利用连接器模型对构件间交互关系和调度策略进行建模,以描述构件间的组合关系.其中,原子构件模型既可使用类 C++的 BIP 语言描述,也可使用 Petri 网、FSM 描述.连接器模型包含构件间交互集合与交互优先级控制策略,每个交互均与一个标志位和数据通信函数相关联,当标志位有效时执行相应函数,进行构件间数据通信.连接器具有优先权调度机制,在交互集合之上规定了构件间交互的动态优先级调度策略.当多个交互被同时使能时,根据相关策略,选择具有较高优先级的交互予以执行.由此,通过连接器模型可将不同构件模型进行有效组合,以构建系统模型.

Lee 等人领导开发的 Ptolemy II 平台是一种开源软件设计与仿真平台,主要用于对嵌入式混杂应用系统进行建模,并进行基于模型的系统特性分析与验证[65−67].Ptolemy II 平台模型是一种由上层主管构件、下层执行者构件构成的层次化模型,主管构件可实现不同种类的计算模型,利用计算模型对执行者构件间连接关系与调度策略进行建模.由此,可针对不同应用领域,面向不同执行者构件,使用相应计算模型将不同执行者构件组合在一起.

基于元模型的模型转换和集成研究,是实现异构模型建模与组合的另一种途径,这通常需要有效的基于模型的设计框架支持.

在 BIP 框架内集成有相关工具链,可将以 MATLAB/Simulink,AADL(architecture analysis and design language),GeNoMIP,NesC/TinyOS,C 等语言描述的构件模型转换为 BIP 模型,以支持异构构件间组合设计.

Balasubramanian 等人针对不同类型构件模型之间难以组合的问题,利用 GME(generic modeling environment)元模型工具,开发了一种面向领域的系统集成建模语言(system integration modeling language,简称 SIML)[68].通过将不同类型的构件模型转化为 SIML 语言形式的统一描述,在抽象层面上进行异构构件间组合.SIML 语言继承 GME 元模型间组合机制,并需要继承相关模型语言中的所有元素,并对组合机制进行相应扩展,由此实现构件间数据结构、接口协议描述的一致,完成不同类型构件模型的组合.

同样在欧盟推动下,研究人员开展了 GENESYS(generic embedded system platform)项目研究工作,致力于建立一套跨领域嵌入式系统参考体系结构,以满足对系统可组合性、健壮性、可维护性等方面的要求[69−72].GENESYS 参考体系结构的一个显著特点就是其跨领域特性,GENESYS 参考体系结构包含逻辑视图和物理视

图,提供一套基于平台独立模型(platform independent model,简称 PIM)和平台相关模型(platform specific model,简称 PSM)的设计、开发方法,支持抽象层面、物理层面的嵌入式软件系统设计.GENESYS 参考体系结构内有共享通信基础架构,不同逻辑、物理视图中的构件模型可通过基础架构进行通信,从而完成异构构件间的交互与组合.通信基础架构支持时间触发、事件触发、同步数据流等多种构件间通信方式,并支持多种通信协议.

3 组合验证技术

组合验证主要包括两方面内容:一是对嵌入式软件构件可组合性、相容性进行验证;二是对软件系统、构件功能/非功能特性进行验证.通过组合验证,将艰巨的后期测试任务分解为设计与开发过程中不同阶段的验证任务,以验证系统设计方案是否符合设计要求,为目标软件开发提供先验证据.组合验证工作需要相应方法与工具的支持.

构件可组合性验证的目的是:判断不同构件间协同运行的可能性,从而确定相关构件是否可进行组合构造.对于狭义可组合性,在可组合性验证之后还应开展相容性验证;而对功能/非功能特性的验证,需要根据设计要求推导预期行为特性,并以预期行为特性的形式化描述为验证目标,验证软件行为特性是否满足设计要求.

在软件系统设计与开发过程中开展同步验证工作,可有效提高验证工作效率,尽早发现设计缺陷.此外,还可根据构件重要度级别对每个构件采取匹配的验证策略,从而进一步降低验证工作的成本.

在欧盟第七框架协议下,研究人员开展了 MADES 项目的研究[48−53],MADES 项目同样旨在通过基于模型的设计技术应用,针对航空、防务领域嵌入式软件,开发一套模型驱动的实时嵌入式软件开发技术与工具,并将其整合为 MADES 框架.在 MADES 框架内,验证工作贯穿在设计全过程中,伴随系统设计与开发工作同步开展.某阶段建模完成后,通过 MADES 模型转换工具,将相关模型转化为基于数理逻辑的形式化描述,导入工具即可进行同步验证.

而在 SPEEDS 框架内,各类建模与验证工具通过相应的适配器层连接在 SPEEDS BUS 上,并在底层通过统一文件格式交换数据和指令信息.如,The Mathworks 公司的 Simulink、Esterel 公司的 Scade 等工具均可连接在 SPEEDS BUS 上.由此,在设计与开发过程的不同阶段,均可按需调用相应工具进行同步验证.

3.1 基于契约的验证

基于契约的组合验证方法是一种常用的组合验证方法.在基于契约的验证方法中,契约间的支配和替代关系是核心.支配契约定义如下:

定义 16(支配契约). 对于契约 $C_1=(A_1,G_1)$和 $C_2=(A_2,G_2)$,如果 $A_1\supseteq A_2$,且 $G_1\subseteq G_2$,那么契约 C_1 支配契约 C_2,记作 $C_1\preccurlyeq C_2$,即,父契约支配子契约.

子契约 C_2 相对于父契约 C_1 的前置条件较弱,而后置条件较强.前置条件更弱意味着对环境的要求更弱,而后置条件更强则意味着提供更多的外部行为保证.在软件系统中,构件 A 可以被构件 B 替代,即,符合子契约的构件可替代符合父契约的构件,需要满足的条件是:“如果构件 A 符合契约 C_1,构件 B 符合契约 C_2,则构件 B 必也符合契约 C_1”,而上述的契约支配关系保证了该条件被满足.这种替代不会造成系统特性的改变,在替代后无需再次对组合契约与系统契约的一致性进行验证.以上关系构成了基于契约的组合验证方法的理论基础.

基于契约的构件模型间可组合性验证分为 3 个步骤:一是系统契约对组合契约的支配关系验证;二是构件实现对构件契约符合性验证;三是构件契约间相容性验证.

首先,通过契约间组合操作得到组合契约,并验证组合契约是否是系统契约的子契约:如果是,则可使用组合契约替代系统契约.由于组合契约的保证部分包含系统契约的全部保证部分,因此组合契约保持了系统契约的全部外部行为特性.其次,验证构件的实现是否符合相关构件契约.最后,验证有交互关系的构件间,每个构件契约的行为保证是否可被其他构件的环境假设全部接纳,且其他构件的行为保证是否不违反该构件的环境假设,即,进行构件契约间相容性验证.目前,基于契约的验证方法的最大困难仍然在于如何为每个构件建立合理的构件契约,并在契约中充分描述每个构件的外部环境假设.

当前,主要的组合模型如 IA 模型、BIP 模型、PA 模型、HRC 模型等都是基于契约的模型,均可使用基于契约的组合验证法进行验证.

考虑到基于契约的组合验证的困难,Atkinson 等人提出了一种组合时动态契约检查方法[73].他们设计了一种 Built-in Test 构件,分为 Testing 构件和 Tester 构件,内建测试任务.其中,Tester 构件负责测试构件是否实现了其外部行为特性,Testing 构件负责测试外部环境提供的服务是否满足构件的需求.通过在软件中植入检测构件,提取相关运行时信息进行分析,以确定构件可组合性.由此,将构件契约与其运行环境契约间可组合性验证过程转化为一种动态检查过程,绕过静态的形式化验证,从测试的角度对构件的可组合性进行实际检验.

3.2 基于不变量的验证

另一种常用的组合验证方法是基于不变量的组合验证法,所谓不变量,就是其度量在软件系统运行过程中不发生变化的量.不变量组合验证法采用不变量描述对构件某项特性的约束,在软件组合构建过程中,如果不变量的值不发生变化,则该项特性满足相关约束条件.因此,在验证过程中仅需要验证不变量的保持性,特别适用于存在复杂交互关系的构件间组合构造.

Verimag 实验室在 BIP 模型的可组合性验证中引入了不变量验证技术,其验证过程主要基于如下规则:

定理 4(基于不变量的可组合性验证规则)[74]. 对于一组构件$(B_1,B_2,\ldots,B_n)$,其组合构件不变量根据以下规则证明:

$$\frac{\{B_i\langle\Phi_i\rangle\}_i^n,\Psi\in\prod(\gamma(B_1,B_2,\ldots,B_n),\{\Phi_i\}_i^n),(\wedge_i^n\Phi_i)\wedge\Psi\Rightarrow\Phi}{\gamma(B_1,B_2,\ldots,B_n)\langle\Phi\rangle},$$

其中,共定义了 3 类不变量,分别是构件不变量Φ_i、交互不变量Ψ、系统不变量Φ.构件不变量描述各原子构件上某项特性的约束.交互不变量描述组合构造时,一组构件交互具有的某项特性的约束.系统不变量由构件不变量和交互不变量通过公式"$(\wedge_i^n\Phi_i)\wedge\Psi\Rightarrow\Phi$"计算得到,即,各构件特性的约束与交互具有的特性约束做与运算后,如果可推导出系统特性的约束,则在该组交互关系下,相应构件对应特性间具有可组合性.

同样地,可以通过基于不变量的验证技术来检验软件构件时间特性是否满足时间约束[75].在 TA 与 TIA 模型中,TA 或 TIA 的某个状态上的所有时间约束构成一个一阶逻辑表达式,称为一个时间不变量.当所有相关时钟变量的一次取值满足这个时间不变量,即使其取值为真时,构件的时间特性满足对应时间约束.此外,在一些模型中,支持通过求解时间约束一致性问题,以验证软件时间特性是否满足时间约束.所谓时间约束一致性问题,是指对于软件系统存是否在一个运行场景,在该场景中,软件运行满足所有的时间约束,具体的求解方法包括基于时间约束图的负环计算等.因此,多个构件模型间的时间约束相容性,在本质上就是不同构件间的时间约束一致性问题.

3.3 基于模型检查技术的验证

模型检查技术也是一种常用的组合验证方法.应用模型检查技术通常需要工具支持.验证规则需要预先制定,并进行形式化描述.在进行验证时,分别将组合模型、验证规则的形式化描述导入工具,通过工具内核进行计算,以验证模型特性对验证规则的符合性.

PA 模型的验证工作有多种模型检查工具支持,如 PRISM,FMEA 等.

PRISM 工具可计算 PA 模型的可达状态空间,定量计算 PA 模型在可达状态下对规则的符合程度,并计算行为、状态迁移集合上的概率分布[76].PRISM 验证方法采用二进制决策表(BDD)、多终端二进制决策表(MTBDD)、稀疏矩阵等数据结构支持验证计算,在本质上是一种基于符号的模型检查技术,其可采用基于 MTBDD 的方法、基于稀疏矩阵的方法以及混合方法等进行系统行为、状态迁移的概率分布计算.其中,基于 MTBDD 的方法可有效处理大规模系统;稀疏矩阵方法对行为概率的计算效率较高,但牺牲了 MTBDD 方法在大规模系统状态空间遍历搜索方面的好处;混合方法采用类 MTBDD 的数据结构存储 PA 模型相关信息,计算效率较高,兼顾前二者的长处,但使用较复杂.

FMEA 工具则关注软件缺陷检测,主要用于在 PA 模型中识别高激活概率的软件缺陷[42].在 FEMA 验证工

具中,缺陷分析主要分为两步:风险分析和危险分析.其中,

- 风险分析根据系统状态和外部环境的组合条件以及既往工程经验,预测系统运行时可能出现的危险情况以及危险程度.
- 危险分析首先需要识别构件行为缺陷;随后,将行为缺陷与可能出现的危险情况关联起来,即,通过模型状态空间内的前向/后向搜索,确定构件行为缺陷与危险情况之间的因果关系;最后,通过模型中的行为、状态迁移概率分布,计算危险情况的发生概率.

此外,MADE 框架内的验证工作同样主要采用模型检查技术进行,通过开发模型仿真与验证工具,支持构件的自动形式化验证.

4 总结与展望

在嵌入式软件组合理论后续研究工作中,仍有一些亟待解决的问题,如构件间复杂调度策略的协同设计、组合状态空间爆炸、增量式组合等.

由于嵌入式软件构件间存在复杂的跨平台交互关系,构件间以不同方式协同运行,不同构件间往往存在着多种调度、控制策略.嵌入式软件与其调度、控制策略之间相互依存,当设计要求、使用环境发生变化时,相应构件的行为特性也必将发生变化.由于上述依存关系,任何构件的变化都要求对关联构件以及软件调度、控制策略进行调整,造成软件的频繁修改,开发效率难以提高.同时,由于构件间的交互关系和协同运行方式也就是构件间的组合关系,调度、控制策略的调整也意味着构件间组合方式的变化.如果对构件间调度、控制策略进行独立设计,则难以在软件设计过程中及时对其进行更新.此外,脱离软件设计过程的独立设计缺乏针对性,容易造成调度、控制策略的设计缺陷.这些因素都将造成构件间的不兼容,使得构件间无法正确组合,并增加了组合验证工作量.因此,在嵌入式软件组合理论研究中,需要研究复杂调度、控制策略与软件构件之间的协同设计方法,实现调度、控制策略设计与软件设计的同步,避免独立设计带来的风险.

每个构件均具有其运行状态空间,例如,构件 P 具有 n 个运行状态,构件 Q 具有 m 个运行状态,当构件 P 与构件 Q 进行组合时,组合状态空间可能包含的状态数为 $n\times m$.当多个构件组合时,组合状态空间可能包含的状态数将呈几何级数上升,造成组合状态空间爆炸.因此,在组合构造过程中,如何解决组合状态空间爆炸问题,依据各类组合规则、约束条件构建有限、可达和合法的组合状态空间,成为需要研究的问题.

此外,嵌入式软件的设计要求、使用环境有时会发生变化,因而构件实现可能发生变化,并可能在软件系统中引入新的构件.增量式组合是指在软件组合构建过程中,将已完成组合构建的部分与新的构件进行组合,不会引起已完成部分的功能/非功能特性改变,不需要进行额外的验证与测试.因此,需要研究嵌入式软件增量式组合构建方法,使嵌入式软件具有可伸缩性,以应对软件系统的持续调整与变化.

本文总结了嵌入式软件组合理论的基本概念,介绍了其主要研究问题,总结并分析对比了近年来在组合模型、构件可组合性、组合机制、非功能特性建模与组合、异构构件建模与组合、组合验证等方面的研究工作.嵌入式软件组合理论的研究在近年来取得了一定的进展,但还存在很多尚未解决的问题,组合设计与开发方法在嵌入式软件设计工作中的实际应用也处于探索阶段,这些都需要开展进一步的研究与实践工作.基于组合设计与开发方法在大规模、复杂嵌入式软件开发中的良好应用前景,通过对嵌入式软件组合理论进行进一步的深入研究,可更好地应对未来嵌入式软件设计与开发任务.

致谢 在此,我们向对本文的工作给予支持和建议的同行,尤其是清华大学计算机科学与技术系陈文光教授以及实验室内的各位老师和同学表示感谢.

References:

[1] Kopetz H. Software engineering for real-time: A roadmap. In: Proc. of the Conf. on the Future of Software Engineering. New York: ACM Press, 2000. 201−211. [doi: 10.1145/336512.336555]

[2] Irvine CE, Levin T, Wilson JD, Shifflett D, Pereira B. An approach to security requirements engineering for a high assurance system. Springer Requirements Engineering, 2002,7(4):192−206. [doi: 10.1007/s007660200015]

[3] Neumann PG. Achieving principled assuredly trustworthy composable systems and networks. Proc. of the DARPA Information Survivability Conf and Expo, Vol.2. Los Alamitos: IEEE Computer Society Press, 2003. 182−187.

[4] Neumann PG. Principled assuredly trustworthy composable architectures (final report for task). Technical Report, 2004. http://www.csl.sri.com/users/neumann/chats4.pdf

[5] Kopetz H, Obermaisser R. Temporal composability. Computing & Control Engineering Journal, 2002,13(4):156−162.

[6] Richling J, Popova-Zeugmann L, Werner M. Verification of non-functional properties of a composable architecture with Petri nets. Fundamenta Informaticae, 2002,51(1-2):185−200. [doi: 10.1049/cce:20020401]

[7] Werner M, Richling J, Milanovic N, Stantchev V. Composability concept for dependable embedded systems. In: Proc. of the Int'l Workshop on Dependable Embedded Systems at the 22nd Symp. on Reliable Distributed Systems. Washington: IEEE Computer Society, 2003. 20−25.

[8] Kopetz H. Elementary versus composite interfaces in distributed real-time systems. In: Proc. of the 4th Int'l Symp. on Autonomous Decentralized Systems-Integration of Heterogeneous Systems. Los Alamitos: IEEE Computer Society Press, 1999. 26−33. [doi: 10.1109/ISADS.1999.838362]

[9] Kopetz H, Suri N. Compositional design of RT systems: A conceptual basis for specification of linking interfaces. In: Proc. of the 6th IEEE Int'l Symp. on Object-Oriented Real-Time Distributed Computing. Los Alamitos: IEEE Computer Society Press, 2003. 51−60. [doi: 10.1109/ ISORC.2003.1199236]

[10] Li CD, Zhou XS, Dong YW, Zhang TT. Composition semantics for component-based embedded software. In: Proc. of the Int'l Conf. on Computational Intelligence and Software Engineering. Los Alamitos: IEEE Computer Society Press, 2009. 1−6. [doi: 10.1109/CISE.2009.5367025]

[11] Bergmans L, Aksit M, Wakita K. An object-oriented model for extensible concurrent systems: The composition-filters approach. Technical Report, 1992. http://doc.utwente.nl/19653/1/Bergmans92object.pdf

[12] Aksit M. Composition and separation of concerns in the object-oriented model. ACM Computing Surveys, 1996,28A(4):Article No.148.

[13] Bergmans L, Aksit M. Composing software from multiple concerns: A model and composition anomalies. In: Proc. of the Multi Dimensional Separation of Concerns in Software Engineering Workshop (ICSE 2000). 2000. http://doc.utwente.nl/18812/1/composing.pdf

[14] Havinga W, Nagy I, Bergmans L. An analysis of aspect composition problems. In: Proc. of the 3rd European Workshop on Aspects in Software. 2006. http://doc.utwente.nl/66822/1/HavingaNagyBergmans-EWAS2006.pdf

[15] Aksit M, Tekinerdogan B. Component composability issues in object-oriented programming. Xootic Magazine, 1997,5(2):15−20.

[16] Weiher M. Approaches to Composition and Refinement in Object-Oriented Design. Berlin: Technische Unversitat, 1997.

[17] Nierstrasz O. Composing active objects. In: Proc. of the Research Directions in Concurrent Object-Oriented Programming. Cambridge: MIT Press, 1993. 151−171.

[18] Nierstrasz O, Meijler TD. Research directions in software composition. ACM Computing Surveys, 1995,27(2):262−264. [doi: 10.1145/210376.210389]

[19] Nierstrasz O, Tsichritzis D. Object-Oriented Software Composition. Englewood Cliffs: Prentice Hall, 1995.

[20] Keller RK, Schauer R. A compositional approach to software design. In: Proc. of the 31st Hawaii Int'l Conf. on System Sciences, Vol.5. Los Alamitos: IEEE Computer Society Press, 1998. [doi: 10.1109/HICSS.1998.648334]

[21] de Alfaro L, Henzinger TA. Interface automata. In: Proc. of the 9th Annual Symp. on Foundations of Software Engineering. New York: ACM Press, 2001. 109−120. [doi: 10.1145/503271.503226]

[22] Lynch NA, Tuttle MR. An introduction to input/output automata. MIT Technical Memo MIT/LCS/TM-373, Cambridge: MIT Press, 1988.

[23] Lynch N, Segala R, Vaandrager F. Hybrid I/O automata. Information and Computation, 2003,185(1):105−157. [doi: 10.1016/S0890-5401(03)00067-1]

[24] Basu A, Bensalem S, Bozga M, Combaz J, Jaber M, Nguyen TH, Sifakis J. Rigorous component-based system design using the BIP framework. IEEE Software, 2011,28(3):41−48. [doi: 10.1109/MS.2011.27]

[25] Alur R. Timed automata. In: Computer Aided Verification. Berlin: Springer-Verlag, 1999. 8−12.

[26] Benveniste A, Caillaud B, Passerone R. Multi-Viewpoint state machines for rich component models. In: Mosterman P, Nicolescu G, eds. Proc. of the Model-Based Design of Heterogeneous Embedded Systems. Boca Raton: CRC Press, 2009.

[27] The SPEEDS Consortium. SPEEDS methodology—A white paper. 2006. http://www.speeds.eu.com/downloads/SPEEDS_WhitePaper.pdf

[28] The SPEEDS Consortium. D.2.5.4 contract specification language (CSL). 2006. http://www.speeds.eu.com/downloads/D_2_5_4_RE_Contract_Specification_Language.pdf

[29] The SPEEDS Consortium. HRC meta-model implementation user guide. 2006. http://www.speeds.eu.com/downloads/HRC_implementation_guide.pdf

[30] Wu SH, Smolka SA, Stark EW. Composition and behaviors of probabilistic I/O automata. Theoretical Computer Science, 1997,176(1):1−38.

[31] Sokolova1 A, de Vink EP. Probabilistic automata: System types, parallel composition and comparison. In: Validation of Stochastic Systems. Berlin: Springer-Verlag, 2004. 1−43. [doi: 10.1007/978-3-540-24611-4_1]

[32] Lee EA, Seshia SA. Introduction to embedded system. UC Berkeley, 2011. http://LeeSeshia.org

[33] Mouelhi S, Chouali S, Mountassir H. Refinement of interface automata strengthened by action semantics. In: Proc. of the 6th Int'l Workshop on Formal Engineering Approches to Software Approaches to Software Components and Architectures. London: Elsevier, 2009. 111−126. [doi: 10.1016/j.entcs.2009.09.031]

[34] Mouelhi S, Chouali S, Mountassir H. Invariant preservation by component composition using semantical interface automata. In: Proc. of the 6th Int'l Conf. on Software Engineering Advances. Barcelona: ThinkMind, 2011. 305−311.

[35] Rowson JA, Sangiovanni-Vincentelli A. Interface-Based design. In: Proc. of the 9th Annual Symp. on Foundations of Software Engineering. New York: ACM Press, 1997. 178−183.

[36] Bhaduri P. Synthesis of interface automata. In: Proc. of the 3th Int'l Symp. on Automated Technology for Verification and Analysis. Berlin: Springer-Verlag, 2005. 338−353. [doi: 10.1007/11562948_26]

[37] Larsen KG, Nyman U, Wasowski A. An interface theory for input/output automata. Technical Report, 2006. http://www.brics.dk/RS/06/11/BRICS-RS-06-11.pdf

[38] Doyen L, Henzinger TA, Jobstmann B. Interface theories with component reuse. In: Proc. of the 8th ACM Int'l Conf. on Embedded Software. New York: ACM Press, 2008. 79−88. [doi: 10.1145/1450058.1450070]

[39] Leduc RJ. Hierarchical interface-based supervisory control with data events. Int'l Journal of Control, 2009,82(5):783−800. [doi: 10.1080/00207170802291411]

[40] Lynch N, Segala R, Vaandrager F. Compositionality for probabilistic automata. In: Proc. of the CONCUR 2003—Concurrency Theory. Berlin: Springer-Verlag, 2005. 208−221. [doi: 10.1007/978-3-540-45187-7_14]

[41] Edwards SH, Gibson DS, Weide BW, Zhupanov S. Software component relationships. In: Proc. of the 8th Int'l Workshop on Software Reuse. 1997. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.36.545&rep=rep1&type=pdf

[42] Grunske L, Colvin R, Winter K. Probabilistic model-checking support for FMEA. In: Proc. of the 4th Int'l Conf. on the Quantitative Evaluation of Systems. Los Alamitos: IEEE Computer Society Press, 2007. 119−128. [doi: 10.1109/QEST.2007.34]

[43] Katoen JP, Zapreev IS, Hahn EM, Hermanns H. The ins and outs of the probabilistic model checker MRMC. Performance Evaluation, 2011,68(2):90−104. [doi: 10.1016/j.peva.2010.04.001]

[44] Leduc RJ, Brandin BA, Lawford M, Wonham WM. Hierarchical interface-based supervisory control—Part I: Serial case. IEEE Trans. on Automatic Control, 2005,50(9):1322−1345. [doi: 10.1109/TAC.2005.854586]

[45] Leduc RJ, Lawford M, Wonham WM. Hierarchical interface-based supervisory control—Part II: Parallel case. IEEE Trans. on Automatic Control, 2005,50(9):1336−1348. [doi: 10.1109/TAC.2005.854612]

[46] Reussner RH, Poernomo IH, Schmidt HW. Contracts and quality attributes of software components. In: Proc. of the 8th Int'l Workshop on Component-Oriented Programming, Vol.29. 2003. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1.3941&rep=rep1&type=pdf

[47] Schmidt HW. Trustworthy components-compositionality and prediction. The Journal of Systems and Software, 2003,65(3): 215−225. [doi: 10.1016/S0164-1212(02)00045-6]

[48] Bagnato A, Sadovykh A, Paige RF, Kolovos DS. MADES: Embedded systems engineering approach in the avionics domain. In: Proc. of the 1st Workshop on Hands-on Platforms and Tools for Model-Based Engineering of Embedded Systems. http://www.txtgroup.com/newsletter/attachment/MADES_HoPES2010_0.4.pdf

[49] Josey A, Hansen S, Dobson I. D1.2 standards survey. 2010. http://www.mades-project.org/

[50] Brosse E, Matragkas N, Rossi M. D2.1 modelling language specification. 2010. http://www.mades-project.org/

[51] Baresi L, Casella F, Donida F, Ferretti G, Leva A, Rossi M. D3.2 models and methods for systems' environment. 2011. http://www.mades-project.org/

[52] Baresi L, Morzenti A, Motta A, Rossi M. D3.3 formal dynamic semantics of the modelling notation. 2011. http://www.mades-project.org/

[53] Matragkas N, Gray I, Kolovos D, Paige R, Audsley N, Indrusiak LS. D4.1 model transformation and code generation tools specification. 2011. http://www.mades-project.org/

[54] Genßler T, Zeidler C. Rule-Driven component composition for embedded systems. In: Proc. of the 4th ICSE Workshop on Component-Based Software Engineering: Component Certification and System Prediction. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.9380&rep=rep1&type=pdf

[55] de Alfaro L, Henzinger TA, Stoelinga M. Timed interface. In: Proc. of the 2nd Int'l Workshop on Embedded Software. Berlin: Springer-Verlag, 2002. 108−122. [doi: 10.1007/3-540-45828-X_9]

[56] David A, Larsen KG, Legay A, Nyman U, Wasowski A. Timed I/O automata: A complete specification theory for real-time systems. In: Proc. of the 13th ACM Int'l Conf. on Hybrid Systems: Computation and Control. New York: ACM Press, 2010. 91−100. [doi: 10.1145/1755952.1755967]

[57] Kaynar DK, Lynch N, Segala R, Vaandrager F. The theory of timed I/O automata. In: Proc. of the Synthesis Lectures on Distributed Computing Theory. San Rafael: Morgan and Claypool, 2010. 1−137.

[58] Kopetz H, El Salloum C, Huber B, Obermaisser R, Paukovits C. Composability in the time-triggered system-on-chip architecture. In: Proc. of the IEEE Int'l SOC Conf. Los Alamitos: IEEE Computer Society Press, 2008. 87−90. [doi: 10.1109/SOCC.2008.4641485]

[59] Object Management Group. Object Constrait Language (version2.0). OMG, 2006. http://www.omg.org/spec/UML/2.2/

[60] Object Management Group. Unified Modeling Language, Superstructure 2.2. OMG, 2009. http://www.omg.org/spec/OCL/2.0/

[61] Müller P, Zeidler C, Stich C, Stelter A. PECOS—Pervasive component systems. In: Proc. of the Workshop on Open Source Technologie in der Automatisierungstechnik. 2001. http://scg.unibe.ch/archive/pecos/public_documents/Muel01a.pdf

[62] Winter M, Zeidler C, Stich C. The PECOS software process. In: Proc. of the Workshop on Components-Based Software Development Processes. 2002. http://scg.unibe.ch/archive/pecos/public_documents/Wint01a.pdf

[63] Genssler T, Christoph A, Schulz B, Winter M, Stich CM, Zeidler C, Müller P, Stelter A, Nierstrasz O, Ducasse S, Arevalo G, Wuyts R, Liang P, Schonhage B, van den Born R. PECOS in a nutshell. 2002. http://scg.unibe.ch/archive/pecos/public_documents/pecosHandbook.pdf

[64] Nierstrasz O, Arévalo G, Ducasse S, Wuyts R, Black AP, Müller PO, Zeidler C, Genssler T, van den Born R. A component model for field devices. In: Proc. of the IFIP/ACM Working Conf. on Component Deployment. LNCS 2370, Berlin: Springer-Verlag, 2002. 200−209. [doi: 10.1007/3-540-45440-3_14]

[65] Brooks C, Lee EA, Liu XJ, Neuendorffer S, Zhao Y, Zheng HY. Heterogeneous concurrent modeling and design in Java—Vol.1: Introduction to ptolemy II. 2008. http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-28.pdf

[66] Brooks C, Lee EA, Liu XJ, Neuendorffer S, Zhao Y, Zheng HY. Heterogeneous concurrent modeling and design in Java—Vol.2: Ptolemy II software architecture. 2008. http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-29.pdf

[67] Brooks C, Lee EA, Liu XJ, Neuendorffer S, Zhao Y, Zheng HY. Heterogeneous concurrent modeling and design in Java—Vol.3: Ptolemy II domains. 2008. http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-37.pdf

[68] Balasubramanian K, Schmidt DC, Molnar Z, Ledeczi A. Component-Based system integration via (meta) model composition. In: Proc. of the 14th Annual IEEE Int'l Conf. and Workshops on the Engineering of Computer-Based Systems. Los Alamitos: IEEE Computer Society Press, 2007. 93−102. [doi: 10.1109/ECBS.2007.24]

[69] Obermaisser R, Salloum CE, Huber B, Kopetz H. Fundamental design principles for embedded systems: The architectural style of the cross-domain architecture GENESYS. In: Proc. of the IEEE Int'l Symp. on Object/Component/Service-Oriented Real-Time Distributed Computing. Los Alamitos: IEEE Computer Society Press, 2009. 3−11. [doi: 10.1109/ISORC.2009.12]

[70] Kopetz H. GENESYS—A cross-domain architecture for dependable embedded systems. In: Proc. of the 2011 Latin-American Symp. on Dependable Computing. Los Alamitos: IEEE Computer Society Press, 2011. 1−6. [doi: 10.1109/LADC.2011.9]

[71] The GENESYS Consortium. Report on analysis for architectural requirements D6.1. 2008. http://www.genesys-platform.eu/Genesys_Del_D6-1_31-03-2008.pdf

[72] The GENESYS Consortium. Report describing the cross-domain architectural style D6.2. 2008. http://www.genesys-platform.eu/GENESYS_Del_D6-2_30-06-2008.pdf

[73] Atkinson C, Gross HG. Built-In contract testing in model-driven, component-based development. In: Business Component-Based Software Engineering. Berlin: Springer-Verlag, 2003. 65−82.

[74] Bensalem S, Bozga M, Nguyen TH, Sifakis J. Compositional verification for component-based systems and application. In: Proc. of the 6th Int'l Symp. on Automated Technology for Verification and Analysis. LNCS 5311, Berlin: Springer-Verlag, 2008. 64−79. [doi: 10.1007/978-3-540-88387-6_7]

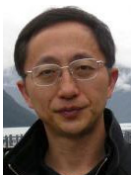
[75] Mouelhi S, Chouali S, Mountassir H. Invariant preservation by component composition using semantical interface automata. In: Proc. of the 6th Int'l Conf. on Software Engineering Advances. Barcelona: IARIA, 2011. 305−311.

[76] Kwiatkowska M, Norman G, Parker D. PRISM: Probabilistic symbolic model checker. In: Proc. of the 12th Int'l Conf. on Computer Performance Evaluation for Modelling Techniques and Tools. Berlin: Springer-Verlag, 2002. 200−204. [doi: 10.1007/3-540-46029-2_13]

王博(1979－),男,陕西西安人,博士生,主要研究领域为软件工程,嵌入式系统.
E-mail: harvicflyhigh@gmail.com

贺飞(1980－),男,博士,副教授,CCF 会员,主要研究领域为形式化方法,自动机理论.
E-mail: hefei@tsinghua.edu.cn

白晓颖(1973－),女,博士,副教授,CCF 会员,主要研究领域为软件测试,服务计算.
E-mail: baixy@tsinghua.edu.cn



Xiaoyu SONG(1963－),男,博士,教授,博士生导师,主要研究领域为形式化方法,嵌入式计算系统.
E-mail: song@ece.pdx.edu