Software Engineering Group
Department of Computer Science
Nanjing University
http://seg.nju.edu.cn

Technical Report No. NJU-SEG-2009-CJ-001

基于 MDE 的异构模型转换:
从 MARTE 模型到 FIACRE 模型

张天, Frédéric JOUAULT, Christian ATTIOGBÉ, Jean BÉZIVIN, 李宣东

Postprint Version. Originally Published in: Journal of Software, Vol.20, No.2, February 2009, pp.214−233.

ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.2, February 2009, pp.214−233
doi: 10.3724/SP.J.1001.2009.03423

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

基于MDE的异构模型转换:从MARTE模型到FIACRE模型*

张 天[1,2,3+], Frédéric JOUAULT[3], Christian ATTIOGBÉ[4], Jean BÉZIVIN[3], 李宣东[1,2]

[1](南京大学 计算机科学与技术系,江苏 南京 210093)

[2](南京大学 计算机软件新技术国家重点实验室,江苏 南京 210093)

[3](ATLAS Team, INRIA & LINA, Nantes University, Nantes 44300, France)

[4](COLOSS Team, LINA, Faculty of Science, Nantes University, Nantes 44300, France)

MDE-Based Mode Transformation: From MARTE Model to FIACRE Model

ZHANG Tian[1,2,3+], Frédéric JOUAULT[3], Christian ATTIOGBÉ[4], Jean BÉZIVIN[3], LI Xuan-Dong[1,2]

[1](Department of Computer Science and Technology, Nanjing University, Nanjing 210093, China)

[2](State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China)

[3](ATLAS Team, INRIA & LINA, Nantes University, Nantes 44300, France)

[4](COLOSS Team, LINA, Faculty of Science, Nantes University, Nantes 44300, France)

+ Corresponding author: E-mail: ztluck@gmail.com

Zhang T, Jouault F, Attiogbé C, Bézivin J, Li XD. MDE-Based mode transformation: From MARTE model to FIACRE model. *Journal of Software*, 2009,20(2):214−233. http://www.jos.org.cn/1000-9825/3423.htm

Abstract: This paper presents a representative case study of bridging UML/MARTE (unified modeling language/ modeling and analysis of real time and embedded systems) to FIACRE (intermediate format for the architectures of embedded distributed components), and discusses in detail two sub-problems of semantic mapping and syntactic transformation respectively. At the semantic level, the semantic mapping rules are developed using model transformation technology so as to implement the transformation between metamodels. At the syntactic level, the concrete syntax rules are built on the metamodels so that textual programs could be generated. Based on the case study, the general framework and corresponding key techniques are discussed. In addition, both the advantages and deficiencies of the work are concluded.

Key words: MDE (model-driven engineering); formal method; MARTE (modeling and analysis of real time and embedded systems); FIACRE (intermediate format for the architectures of embedded distributed components); heterogeneous

摘 要: 通过研究一个具有代表性的 UML/MARTE(unified modeling language/modeling and analysis of real time and embedded systems)模型向 FIACRE(intermediate format for the architectures of embedded distributed components)

* Supported by the National Natural Science Foundation of China under Grant No.60425204 (国家自然科学基金); the National Basic Research Program of China under Grant No.2002CB312001 (国家重点基础研究发展计划(973)); the National High-Tech Research and Development Plan of China under Grant No.2007AA010302 (国家高技术研究发展计划(863)); the Natural Science Foundation of Jiangsu Province of China under Grant No.BK2007714 (江苏省自然科学基金)

Received 2008-04-02; Accepted 2008-07-03

形式模型的转换实例,探讨了异构模型之间在语义和语法层的相互转换问题.在语义层,通过模型转换技术构造语义映射规则,实现元语言之间的转换;在语法层,通过构造元模型的具体语法,反映元语言的语法规则,从而产生目标模型的程序实体.基于此实例研究,探讨了通用转换途径的相关框架和关键技术,并讨论了转换工作的优缺点和实用性.

关键词: 模型驱动工程;形式化方法;MARTE(modeling and analysis of real time and embedded systems);FIACRE (intermediate format for the architectures of embedded distributed components);异构性

中图法分类号: TP311 **文献标识码:** A

近 10 年来,形式化领域和以 UML(unified modeling language)[1]为代表的可视化建模领域都产生和发展出了大量不同的建模方法和模型.形式化方法(formal method)是基于数学的手段为软件和硬件系统提供精确规约、开发和验证的方法,它以严格的形式模型为中心,为系统的可靠性提供有力的保证.UML 则强调建模的灵活性和实用性,它基于核心元模型构造建模语言,同时支持对系统静态和动态的多个侧面进行建模,目前已经成为事实上的工业标准.然而,不同领域的异构模型彼此无法互用,相应的支撑和验证工具也难以共享和复用.异构模型之间的转换,尤其是以 UML 为主的非/半形式化模型向严格的形式化模型的映射,成为学术界和工业界的关注焦点.

本文选择目前在形式化领域和可视化建模领域备受关注的 FIACRE(intermediate format for the architectures of embedded distributed components)[2]形式模型和 UML/MARTE(UML/modeling and analysis of real time and embedded systems)[3]模型,进行了异构模型转换的实例研究.通过构造从 UML/MARTE 到 FIACRE 的转换桥,探讨了异构模型相互转换的两个重要子问题的解决方案,即如何实现异构模型间的语义映射,以及如何在语法层面形成完整的转换.此外,我们还通过一个咖啡壶系统(coffee maker system,简称 CMS)的应用示例,讨论了 UML/MARTE 到 FIACRE 转换桥的应用范围和效果.

针对异构模型间的语义映射问题,我们首先抽象出源模型和目标模型的元模型,然后构造元模型间的语义映射,并基于模型转换技术定义相应的转换规则,最后通过模型转换在语义层实现相互转化.针对语法层面的转换问题,我们为元模型构造具体语法规则,将在语义层转换得到的目标模型进一步转换为程序(或模型)实体,从而形成完整的异构模型间的相互转换.以上两个层面所采用的转换方法是受到了模型驱动工程(model-driven engineering,简称 MDE)[4]的影响,因此在实现上我们也采用了相应的支撑平台 AMMA(ATLAS(Atlantic data systems) model management architecture)[5]所提供的技术.

MDE 是软件工程领域新兴的一种软件开发模式.它以模型为首要软件制品,通过(元)建模和模型转换来驱动软件的开发,能够较好地解决异构平台间的相互转换问题.随着 MDA™(model driven architecture)[6]的提出,MDE 逐渐受到了学术界和工业界的广泛关注,产生了大量的元元模型体系、建模方法和相应的支撑工具.其中具有代表性的是 OMG(Object Management Group)推动的 MDA 框架标准和 ATLAS 研究组提出的 AMMA 开发平台.

基于 MDE 的异构模型语义映射和语法转换的具体实现如下:首先基于 AMMA 平台的元元模型体系 KM3(kernel metametamodel)[7],通过元建模(metamodeling)抽象出源模型与目标模型的 KM3 元模型,将异构模型引入到 MDE 中,从而实现异构模型的同构化;然后利用平台的模型转换语言 ATL(ATLAS transformation language)[8]针对元模型构造转换规则,通过将对应的实例模型进行相互转换,实现在 MDE 下同构化模型之间的相互转换;最后,基于 TCS(textual concrete syntax)[9]为目标模型的元模型构造具体语法,搭建一个从模型域(modelware)[10]到文本域(grammarware)[11]的转换桥.从而将引入到 MDE 中的异构模型映射回传统的文本表示,以产生最终的程序(或模型)实体.在本文中,“文本”主要指代码化的程序描述方式,尤其是形式模型的代码表示法,用以区分 MDE 中“模型”的概念.有关 Grammarware 术语的详细介绍参见文献[12].

基于以上 UML/MARTE 模型到 FIACRE 形式模型的实例研究,本文针对通用转换途径讨论了相关的框架和关键技术.该途径以 AMMA 平台为技术实现平台,可同时应用于模型域和文本域,具有较好的实用性.本文以

实例的方式贯穿全文,一方面易于对抽象内容的感性把握,另一方面也初步展示了本文工作的有效性和实用性.

本文第 1 节简要介绍 UML/MARTE 模型和 FIACRE 形式模型.第 2 节详细描述 UML/MARTE 到 FIACRE 转换桥的构造过程,重点体现语义层和语法层的转换细节.第 3 节以一个嵌入式应用系统为例,讨论转换桥的应用范围和效果.第 4 节针对通用转换途径讨论相关的框架和关键技术.第 5 节进行相关工作比较.第 6 节总结全文并给出结论.

1 MARTE模型和FIACRE模型

1.1 UML/MARTE模型

MARTE 是 OMG 于 2007 年底发布的一个新 UML Profile,用以取代原有的针对调度、性能和时间的 UML Profile(UML profile for schedulability, performance and time),作为 UML 对实时和嵌入式系统进行建模的正式规范,因而备受关注.

MARTE 中的概念主要分为基础(foundation)、建模(modeling)和分析(analyzing)3 个部分,分别封装在基础模型、设计模型和分析模型 3 个包中.其中,基础模型包主要用于对实时和嵌入式系统中的一些基本概念,如优先级、响应时间、执行周期、受限资源等,进行建模和规约设计.它是构建设计和分析模型的基础,也是整个 MARTE 规范的核心.设计模型包提供了较高抽象层的建模元素,用于对并发和实时的活动主体和行为进行建模,例如,具有独立控制能力的实时模块、对受限资源的保护模块以及响应并发或实时调用的服务和动作等.分析模型包封装了用于对系统的性能和可靠性等进行分析的模型元素,主要针对能量消耗、内存占用以及系统的可用性、可靠性和安全性等.设计模型和分析模型均是通过复用基础模型构建起来的,因此,实例研究中的 MARTE 模型虽然主要针对设计模型包中的相关概念,但仍需涉及一些基础包所定义的模型.

MARTE 中的设计模型主要封装在实时和嵌入式的计算和通信模型(RTE model of computation and communication,简称 RTEMoCC)包中.图 1 从域概念的角度展示了相关的模型元素.在图 1 中,RtUnit 和 PpUnit 表示实时和嵌入式系统中的活动对象(active object),主要用来对系统的活动主体进行建模.其中,RtUnit 是一个包含对象及其行为模式的并发单元,通过支持对消息的异步处理实现并发控制.RtUnit 分为主 RtUnit(main RtUnit)和非主 RtUnit 两种,通过 isMain 属性标示.主 RtUnit 提供了 main 操作,作为系统的执行入口点.每个系统中只能有 1 个主 RtUnit 作为系统的控制单元,并且其 main 操作的生命期和系统的生命期相同.PpUnit 是一个被动受护单元(protected passive unit),它为共享数据提供并发访问策略.PpUnit 主要用来对系统中的受限资源进行建模,往往由 RtUnit 所拥有并进行控制.

RtSevrice 通过定义并发策略、异步方式、响应时间和优先级等属性来支持实时服务,它与 RtAction 非常类似.但 RtService 继承自 MARTE::GRM::ResourceCore::Service,主要对服务进行建模;而 RtAction 则继承自 CommonBehavior::BasicBehaviors::InvocationAction,主要用来对原子行为进行建模,如 UML 状态机(UML state machine)中的 Invocation Action 等.

图 2 是 RTEMoCC 包里所提供的域模型在 MARTE profile 中作为构造型(stereotype)的定义.域模型的定义与构造型的定义侧重点不同,前者强调特定领域的概念,是概念模型的直接反映;后者则强调特定模型在 UML 中的应用对象.在 UML 的 profile 定义中,构造型必须通过扩展(extension)关系应用到某个/些元类型(metaclass)上,表示该构造型在实例化时将会作为附加特性绑定到某个指定元模型的实例上.正是基于以上原因,本文的实例研究部分将会分别针对域模型和构造型设计相关的查询以及转换规则(相关细节见第 2 节).

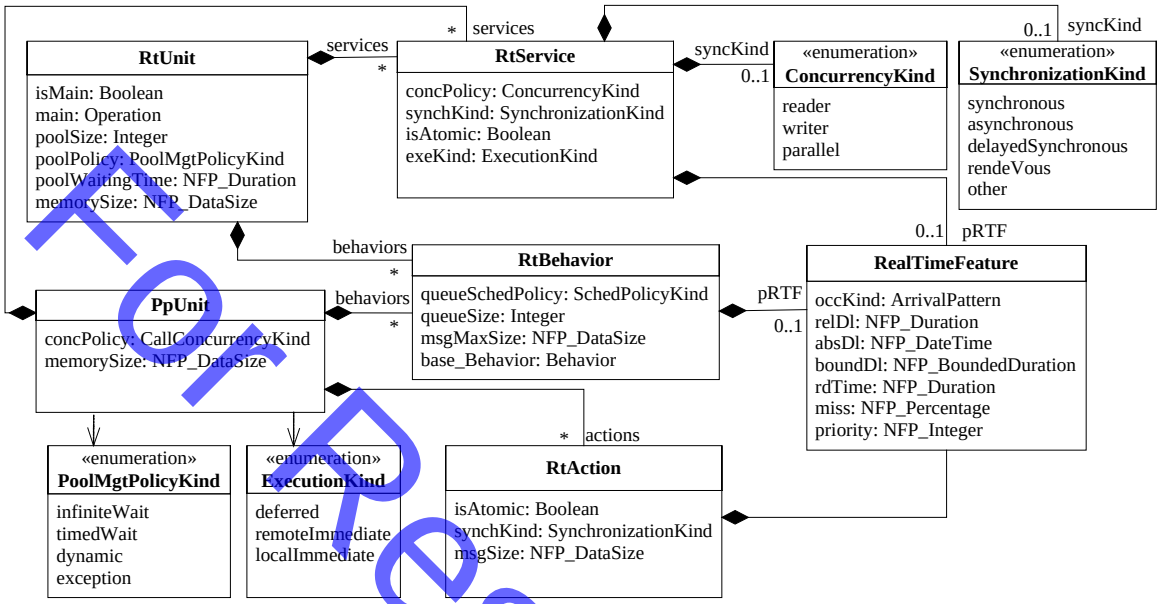


Fig.1 RTEMoCC models in MARTE profile

图 1 MARTE profile 中的 RTEMoCC 模型

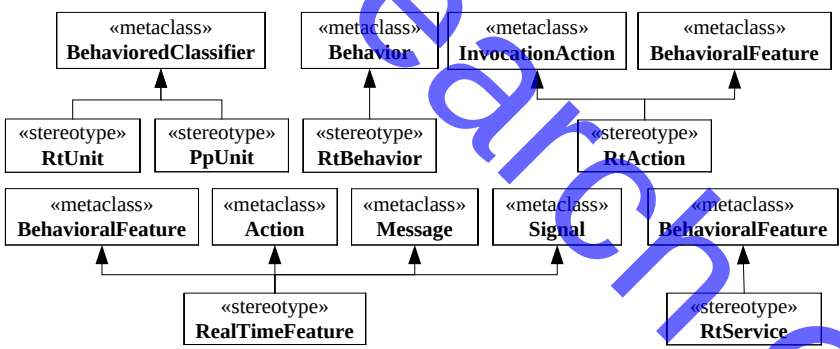


Fig.2 Sereotypes in RTEMoCC package

图 2 RTEMoCC 包中的构造型

1.2 FIACRE形式模型

FIACRE 是由法国国家计算机科学与控制研究所(INRIA(National Institute for Research in Computer Science and Control))[13]提出的一种形式模型,主要用于对嵌入式和分布式系统进行形式化的验证(verification)和仿真(simulation).INRIA 还计划将其作为在未来支持 MDE 的一种主要形式化规约和验证方法.

FIACRE 对并发行为的支持是基于 LOTOS(language of temporal ordering specification)[14]和自动机(automata)进行开发和设计的,由两个主要的行为实体 Process 和 Component 构成.其中,Process 是最基本的行为单元,用于对顺序行为进行描述.每个 Process 通过一组控制状态(control states)进行定义,这些状态也可以关联到程序片断上.通信消息(communication events)发生在端口(ports)上,使系统从一个状态跃迁到另一个状态.Component 通过对 Process 进行组合,如同步、并发等,用以实现对复杂行为的描述.Component 是层次结构的,它还可以再包含子 Component,但 Process 则不能包含子 Process.Component 之间的组合就构成了一个完整的系统,由 Program 表示.图 3 给出了简化的 FIACRE 核心元模型.

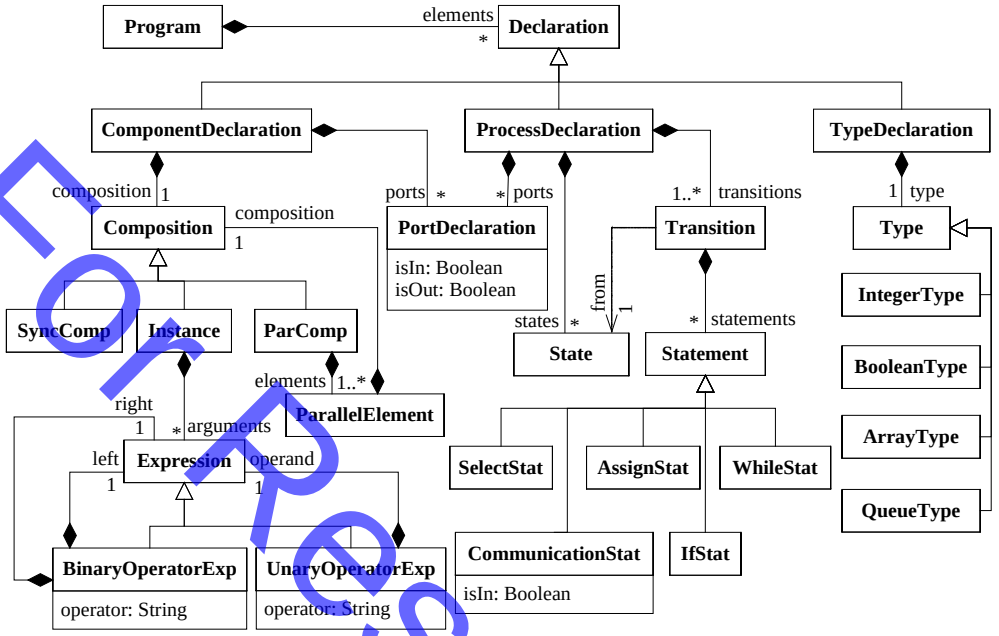


Fig.3 Core metamodel of FIACRE

图 3 FIACRE 的核心元模型

在描述时间方面,FIACRE 借鉴 Time Petri Nets 使用时间间隔(time interval)来定义行为的时间约束.由于在 FIACRE 中基本行为是通过 Process 来定义的,因此时间约束则被定义在 Process 之间进行交互的端口上.时间间隔的单位使用整型表示由多少个时间单元组成.

目前,FIACRE 已经在 INRIA 参与推动的 Topcased 开源项目中使用.作为一种中间语言支持各种模型到形式验证工具的过渡,如 CADP(construction and analysis of distributed processes)[15],Tina(time petri net analyzer)[16].

2 模型转换:从MARTE模型到FIACRE模型

从 MDE 的视角去理解异构模型(尤其是到形式模型)的相互转换问题,虽然与传统的形式化方法领域的理解方式不尽相同,但从本质上讲仍有一定的相似之处.它们都要解决语义层的映射以及语法层的形式转换问题.本节就从这两个层次来介绍如何通过 MDE 的支撑工具 AMMA 平台来实现 MARTE 模型到 FIACRE 形式模型的转换.

下面首先简要介绍 AMMA 平台,然后重点从语义映射和语法转换两个层次详细讨论 MARTE 模型到 FIACRE 形式模型转换的实现过程.在 AMMA 平台上的语义映射和语法转换实现过程均是基于 KM3 元模型进行的,MARTE 和 FIACRE 的 KM3 元模型可以通过 KM3 对图 1 和图 3 的抽象元模型直接建模而获得(限于篇幅,文中不再列出具体建模结果).

2.1 AMMA平台

AMMA 是 ATLAS 研究组提出的一个 MDE 开发平台,支持(元)建模、模型转换、模型编织和模型管理.根据平台各个模块的功能特点,文献[5]将平台上的工具分为两个集合,分别对应于 modeling in the small 和 modeling in the large.前者主要针对(元)建模、模型转换和编织,包含 ATL,AMW(ATLAS model weaver)和 TCS;后者则主要用来对(元)模型进行管理,包含 AM3(ATLAS megamodel management).

AMMA 平台所使用的元元模型体系的根是 KM3.其层次结构非常类似于 OMG 的元对象设施(meta object facility,简称 MOF)[17].但与 MOF 相比,KM3 具有更加简单、实用的特点.AMMA 平台上的所有的语言和工具都

遵循 KM3 体系,并最终可由 KM3 解释.本文的工作主要是基于 AMMA 平台的 KM3,ATL 以及 TCS 实现的,因此下面分别介绍这 3 个部分.

2.1.1 KM3 元元模型

KM3 的设计理念是将 AMMA 上的各种语言(如 ATL,AMW,TCS 以及 AM3 等)视为 DSLs,通过 KM3 为这些 DSLs 提供一个共同的元元模型根.这一点与 MOF 非常相似.并且,如果用 MOF 的四层元模型体系来看待 KM3,那么 KM3 和 MOF 是位于同一个层次的.不过,MOF 是 OMG 为了支撑 MDA 而制定的一个规范,因而非常强调与其他相关规范的一致性和体系性,如对 UML 的解释和支持等.KM3 则更加强调元元模型的实用性,它省略了基于包(package)的复用机制,强化类与关系这两个概念,以非常简洁而实用的方式构造各种 DSLs 的元模型.同时,KM3 自身的元模型也是通过 KM3 定义的.此外,KM3 还定义了文本化的具体语法,并且通过 TCS 以 Eclipse 插件的形式在 Eclipse 官方站点进行发布.图 4 给出了 KM3 的元模型结构.

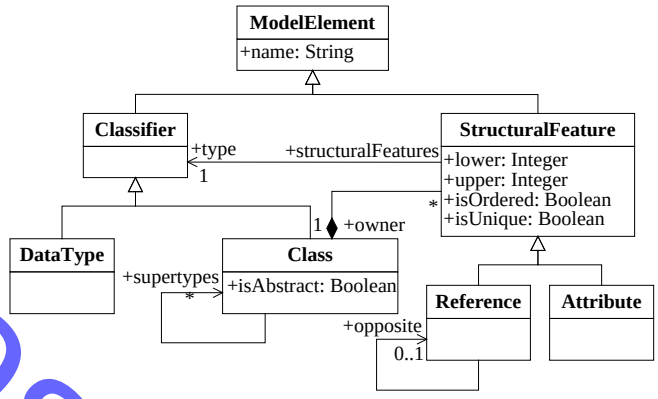


Fig.4 KM3 metamodel structure

图 4 KM3 的元模型结构

除了定义 AMMA 平台自身的各种语言之外,KM3 的另一个重要作用是为各种 DSLs 构造元模型,即所谓的元建模(metamodeling).本文对形式化语言进行元建模的部分就是基于 KM3 完成的.

2.1.2 ATL 模型转换语言

ATL 是 AMMA 平台最重要的一个部分.它支持对模型转换规则进行声明式(declarative)和命令式(imperative)的设计,是一种混合式的转换规则设计语言[18].ATL 最初被设计用来响应 QVT 提案需求(query/view/transformation RFP,简称 QVT RFP)[19].因此,其语法结构是基于 OCL2.0(object constraint language)[20]定义的.在 QVT 成为 MOF 规范的一个组成部分发布后,ATL 被作为 QVT 的一个标准实现体向 OMG 提出了正式申请,目前已进入 OMG 的官方审批流程.

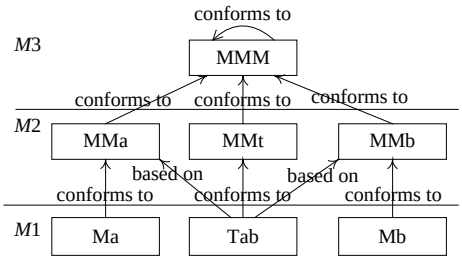


Fig.5 ATL model transformation

图 5 ATL 遵循的模型转换视图

图 5 给出了基于 ATL 的模型转换视图.从图中可以看出,ATL 下的模型转换与目前的 QVT 规范中定义的模型转换层次和方式极为吻合.其中,MMa 和 MMb 是源模型(source model)和目标模型(target model)的元模型,Ma 和 Mb 是基于此元模型定义的实例模型.MMt 是模型转换语言(此处为 ATL)的元模型,而 Tab 则是针对 MMa 和 MMb 定义的转换规则,通过 MMt 的执行引擎(此处为 ATL 虚拟机)对 Tab 规则的执行,就可以将源模型 Ma 转换为目标模型 Mb 了.

ATL 虚拟机(ATL virtual machine)是 ATL 的执行引擎.在设计上,ATL 虚拟机借鉴了 Java 虚拟机的很多设计思路,具有很好的可移植性,目前已经集成到 ADT(ATL development tool)中,并与 ADT 一起在 Eclipse 的官方站点作为顶层项目(top-level project)发布.

2.1.3 TCS 文本化语法

TCS 是 AMMA 平台上的一个较为独特的部分.其他部分,如 ATL,AMW 等,都是针对 MDE 领域内部应用设计的,而 TCS 则是专门为沟通模型域和文本域而设计的.TCS 的设计思路最早来源于 OMG 的 MOF 模型到文本

转换语言提案需求(MOF model to text transformation language RFP)[21],目前也已经成为 Eclipse 的一个开源项目.

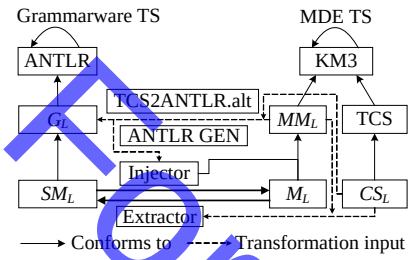


Fig.6 Overview of TCS application

图 6 TCS 应用视图

TCS 在语言上和 ATL 位于同一层次,通过 KM3 定义其元模型.TCS 在语法解析方面利用了 ANTLR(another tool for language recognition)[22]技术,一种类似于 Yacc 的文本语法分析器生成器(parser generators).图 6 给出了 TCS 的应用视图.假定我们要构造一种 DSL 语言 L 的模型域和文本域转换桥,图中 MM_L 表示 L 在模型域中的元模型,G_L 表示 L 在文本域中的语法表达式.通过为 MM_L 构造对应的文本语法规则,就可以利用 ANTLR 生成相应的模型到文本桥(extractor)和文本到模型桥(injector),具体细节参见文献[9].通过对语言 L 的模型 M_L 执行 Extractor,可以生成符合语法规则 G_L 的文本 SM_L;通过对 SM_L 执行 Injector,可以生成符合元模型 MM_L 的模型 M_L.

2.2 语义层的映射

在 MDE 中,一种语言的语义及抽象语法都是通过该语言的元模型所定义的.例如,UML 的语义是通过 MOF 对其在 $M2$ 层的元模型定义而确定的.假定 M 是 MARTE 的元模型,F 是 FIACRE 的元模型,关系 Φ 定义了一个从 M 到 F 的映射,则 FIACRE 的语义可以通过关系式 $\Phi(M)=F$ 表示.其中,Φ 的定义是通过一组由 MARTE 元模型到 FIACRE 元模型的映射规则(MARTE-to-FIACRE,简称 M2F)给出的.对每一个映射规则 $\Phi(m)=f$ 而言,m 表示 1 个或多个 MARTE 的元模型,f 表示一个 FIACRE 的元模型.

映射规则 M2F 的定义分别从结构、行为以及时间约束 3 个方面给出,每个方面都包含了一组 M2F 规则.

2.2.1 类型结构的映射规则

- M2F 1. 基本数据类型.

  MARTE 中的基本数据类型 Real,Integer,Unlimited Natural,Boolean 和 String 映射为 FIACRE 的 RealType,IntegerType,NaturalType,BooleanType 和 StringType.DataTime 类型映射为 IntegerType.NFP_Duration,NFP_DateTime,NFP BoundedDuration,NFP_Integer 以及 NFP_Percentage 类型映射为 IntegerType.

MARTE 中使用的数据类型主要在 MARTE 模型库(MARTE model library)中定义.基本数据类型有 Real,Integer,Unlimited Natural,Boolean,String 以及 DateTime.其中,Real,Integer,Unlimited Natural,Boolean 和 String 可以分别直接映射为 FIACRE 中的 RealType,IntegerType,NaturalType,BooleanType 和 StringType.虽然 FIACRE 中没有 Date Time 类型可以直接与 DateTime 对应,但根据 DateTime 的语义可以直观地将其映射为 IntegerType.

在 RETMoCC 包中,与实时特征(real-time feature)相关的数据类型有 NFP_Duration,NFP_DateTime,NFP BoundedDuration,NFP_Integer 以及 NFP_Percentage.根据它们的语义,可以映射为 IntegerType.

- M2F 2. 顶层 Package.

  顶层 Package 是 UML/MARTE 模型中不被其他 Package 所包含的 Package,它被映射到 FIACRE 中的 Program 上.顶层 Package 所包含的模型元素(PackagedElement)也根据自身类型被映射到 Program 所包含的元素上.

- M2F 3. Sub-Package 仅映射其中包含的模型到 FIACRE.

  顶层 Package 所包含的 Sub-Package 不直接映射到 FIACRE 中的模型,但是该 Sub-Package 所包含的模型元素和主 Package 中的模型进行相同的转化.Sub-Package 所包含的 Sub-Package 仍遵循相同的规则.

由于 MARTE 是 UML 的一个 Profile,因此在使用 MARTE 构造系统模型时需要同时使用 UML 作为建模基础.M2F2 所定义的映射规则确定了所映射的 FIACRE 模型的根,也就是 Program.这里假定通过 UML/MARTE 进行系统建模时仅有 1 个顶层 Package.其他 Package 都必须作为顶层 Package 的 Sub-Package 而存在,并且通过其 PackagedElement 进行定义.

由非顶层 Package 所包含的模型元素被映射到 FIACRE 的 Program 中,而不再拥有层次结构.这是由于 FIACRE 的 Program 本身不再具有子 Program 的层次划分.对于出现命名冲突的问题,可以通过引入 Package 的名字来解决.

- M2F 4. 主 RtUnit 映射为主 ComponentDeclaration.
  主 RtUnit 是 isMain 属性值为真的 RtUnit,它被映射到一个 FIACRE 的 ComponentDeclaration,该 ComponentDeclaration 直接由 FIACRE 的 Program 所拥有,为 FIACRE 程序执行的入口点.
- M2F 5. 非主 RtUnit 映射为非主 ComponentDeclaration.
  非主 RtUnit 被映射到一个 FIACRE 的 ComponentDeclaration,并且该 ComponentDeclaration 设置为非程序执行入口点.
- M2F 6. PpUnit 映射为主 ProcessDeclaration.
  一个 PpUnit 被映射为一个 FIACRE 的 ProcessDeclaration,并且该 ProcessDeclaration 由拥有该 PpUnit 的 ComponentDeclaration 通过 Composition 进行调用.
- M2F 7. PpUnit 映射为主 ProcessDeclaration.
  一个 PpUnit 被映射为一个 FIACRE 的 ProcessDeclaration,并且该 ProcessDeclaration 由拥有该 PpUnit 的 ComponentDeclaration 通过 Composition 进行调用.

以上 M2F 4~M2F 7 定义了 MARTE 的 RtUnit 和 PpUnit 模型转换到 FIACRE 上的规则.但事实上,由于 MARTE 模型是以构造型的形式绑定到 UML 模型上的,因此在构造相应的 ATL 转化规则时必须首先对构造型的绑定模型进行判断.从图 2 的 MARTE 构造型到 UML 元类型扩展中可以看到,RtUnit 和 PpUnit 这两个构造型都是应用在 UML 中的 BehavioredClassifier 上.因此,要首先对 BehavioredClassifier 类型进行检查,判断是否应用了 RtUnit 或 PpUnit 构造型.对 UML 的 BehavioredClassifier 类型进行构造型判断和获取的 ATL 规则如下:

ATL 规则 1. 针对构造型的 Helper 操作.

```
1.  helper context UML2!Element def: hasStereotype(s: String): Boolean =
2.      self.getAppliedStereotypes()->select(e| e.name = s)->notEmpty();
3.  helper context UML2!Element def: getAppliedStereotype(s: String): UML2!Stereotype =
4.      if self.hasStereotype(s) then
5.          self.getAppliedStereotypes()->select(e| e.name = s)->first()
6.      else
7.          OclUndefined
8.      endif;
```

ATL 规则 1 的第 1~2 行和第 3~8 行分别定义了两个 Helper 操作,专门对 UML 模型所应用的构造型情况进行查询和判断.Helper 是 ATL 语言的用于对模型进行查询和判断的操作模块.第 1 个 Helper 定义了名为 hasStereotype 的操作,它有一个 String 类型的参数 *s*,返回值为 Boolean 类型.该操作定义在 Element 类型上,判断在其实例模型是否应用了以 *s* 为名字的构造型.第 2 个 Helper 定义了名为 getAppliedStereotype 的操作,该操作用于返回应用在 Element 上的构造型模型.在 UML2 中,Element 被定义在 Kernel 包中,是其他模型的根.由于定义在 Element 上的操作也可以在其子类上进行调用,因此以上两个 Helper 具有很强的通用性,可以在 BehavioredClassifier,Behavior 以及 Signal 等各种应用 MARTE 构造型的 UML2 模型上进行调用.

ATL 规则 2. 顶层 Package 的转化规则.

```
1.  helper context UML2!Package def: isTopLevelPackage: Boolean=
2.  if self.refImmediateComposite().oclIsUndefined() then
3.      true
4.  else
5.      false
```

```
6.  endif;
7.  rule Package {
8.      from
9.          p: UML2!Package(
10.             p.isTopLevelPackage
11.         )
12.     to
13.         s: FIACRE! Program (
14.             name<-p.name+'_Program',
15.             elements<-compDec
16.         ),
17.         compDec: FIACRE!ComponentDeclaration (
18.             name<-p.packagedElement.name,
19.             composition<-p.classes->collect(b|thisModule.resolveTemp(b,'p'))
20.         )
21. }
```

ATL 规则 2 定义了 M2F 2 所对应的 ATL 转换规则.第 1~6 行定义了一个 Helper 以判断 Package 的实例模型是否为顶层 Package.第 8~11 行通过该操作匹配符合条件的源模型.第 12~20 行定义了目标模型.第 15 行使用了一个嵌套性的模型匹配,并且指向在第 17~22 行定义的新目标模型 compDec.以上的模型转换规则使用了 ATL 的声明式(declarative)规则定义风格.

ATL 规则 3. 主 RtUnit 到 ComponentDeclaration 的转化规则.

```
1.  helper context UML2!Classifier def: isMainOfRtUnit(): Boolean=
2.      let s: String='RtUnit' in
3.          if self.hasStereotype(s) then
4.              if self.getValue(self.getAppliedStereotype(s),'isMain').oclIsKindOf(Boolean) then
5.                  self.getValue(self.getAppliedStereotype(s),'isMain').toBoolean()
6.              else
7.                  false
8.              endif
9.          else
10.             false
11.         endif;
12. rule MainRtUnit2Component {
13.     from s: UML2!Class(
14.         s.isRtUnit() and s.isMainOfRtUnit()<>true
15.     )
16.     to t: FIACRE!ComponentDeclaration(
17.         name<-s.name+'_Comp',
18.         ports<-portDec,
19.         parameters<-parm,
20.         variables<-varDec
21.     ),
```

```
22.         portDec: FIACRE!PortDeclaration(
23.                 ...
24.         ),
25.         param: FIACRE!Parameters(
26.                 ...
27.         ),
28.         varDec: FIACRE! VariablesDeclaration(
29.                 ...
30.         )
31.         ...
32.   }
```

ATL 规则 3 给出了将一个 RtUnit 模型映射到 Component 的转换规则.其中,第 1~11 行定义了用来判断 RtUnit 是否为主 RtUnit 的 Helper 操作.在这个定义中,操作的调用类型为 BehavioredClassifier 而不是 Element.这是因为应用 RtUnit 构造型只能应用到 BehavioredClassifier 上(如图 2 所示).

ATL 声明式转换规则定义的默认情况是 1:1 映射,也就是一个源模型映射到一个目标模型.但是,通过嵌套的定义目标模型的匹配规则就可以产生层次性的多个目标模型.如第 18 行 ports 的初始值设定就用到了第 22 行定义的 PortDeclaration;第 19 行 parameters 指向了第 25 行;第 20 行 variables 则指向了第 28 行.从而产生了层次性的目标模型 ComponentDeclaration.

XMI 1. 简化的 XMI 格式 RtUnit 源模型.

```
1.  〈?xml version="1.0" encoding="UTF-8"?〉
2.  〈xmi:XMI xmi:version="2.1" xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"〉
3.    ...
4.    〈packagedElement xmi:type="uml:Class" xmi:id="_2qAcoJa3EdyZVPrsnpgvdA"
5.                          name="CoffeeMakerController"〉
6.      ...
7.    〈/packagedElement〉
8.    ...
9.    〈RTEMoCC:RtUnit xmi:id="_7N5ekJa3EdyZVPrsnpgvdA" isMain="true"
10.     main="_DDapUJa4EdyZVPrsnpgvdA" base_BehavioredClassifier="_2qAcoJa3EdyZVPrsnpgvdA"/〉
11.   ...
12. 〈/xmi:XMI〉
```

XMI 1 所示的 RtUnit 源模型是使用 XMI2.1 格式表示的 UML 模型.第 2 行指定了该 XMI 文件遵照的 XMI 标准版本以及相应的 Schema 规范版本.第 4~7 行是一个名为 CoffeeMakerController 的 Class 实例,它被定义为 Package 的一个 packagedElement 属性.第 9~10 行定义了一个 RtUnit 构造型的实例,它被应用到 xmi:id 值为"_2qAcoJa3EdyZVPrsnpgvdA"的模型实例上(通过 BehavioredClassifier 指定该值).在 XMI 中,每一个模型实例都通过 xmi:id 值进行唯一标示,因此 RtUnit 构造型的实例就应用到 CoffeeMakerController 上了.通过 ATL 规则 3 中定义的 Helper 操作 isMainOfRtUnit 就可以判断出 CoffeeMakerController 上应用了 RtUnit 构造型,并且为一个主 RtUnit 实例.

通过执行 ATL 规则 3,可以将 XMI 1 所示的 RtUnit 源模型转换为 FIACRE 目标模型,如 XMI 2 所示.与源模型不同的是,生成的模型使用了 XMI 标准的 2.0 版本,这是因为目前 ATL 虚拟机对生成模型所使用的 XMI 标准仍是 2.0 版本.但这并不影响目标模型在 AMMA 平台上的进一步使用,如通过 TCS 产生 FIACRE 文本程序(这一部分内容将在第 2.3 节作进一步介绍).

XMI 2. 生成的 FIACRE 目标模型.

```
1. ⟨?xml version="1.0" encoding="ISO-8859-1"?⟩
2. ⟨xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
3.          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="FIACRE"⟩
4. ⟨Program name="CoffeeMaker_Program"⟩
5.   ⟨elements⟩
6.     ... ...
7.     ⟨declarations xsi:type="ComponentDeclaration" name="CoffeeMaker_Comp" isMain="true"⟩
8.       ⟨ports xsi:type="PortsDeclaration" name="CM_Ports"⟩
9.         ⟨port⟩CoffeeMaker_req⟨/port⟩
10.        ⟨port⟩CoffeeMaker_resp⟨/port⟩
11.      ⟨/ports⟩
12.      ⟨parameters xsi:type="Parameters"⟩
13.        ⟨parameter⟩CoffeeMaker_in⟨/parameter⟩
14.        ⟨parameter⟩CoffeeMaker_out⟨/parameter⟩
15.      ⟨/parameters⟩
16.      ... ...
17.    ⟨/declarations⟩
18     ... ...
19.  ⟨/elements⟩
20. ⟨/Program⟩
21. ⟨/xmi:XMI⟩
```

在生成的 FIACRE 目标模型中,第 7~17 行定义了由 RtUnit 映射所产生的 ComponentDeclaration,它所拥有的 PortsDeclaration 和 Parameters 分别由 ATL 规则 3 中的第 22~24 行以及第 25~27 行所产生.由于源模型中没有变量定义,因此,第 28~30 行转换规则也就没有发生作用,而生成的目标模型里也没有出现 VariablesDeclaration 的定义.

2.2.2 行为的映射规则

- M2F 8. RtUnit 对应的简单 State Machine 映射为 ProcessDeclaration.
  如果对应的状态机没有子状态机状态(submachine state)或复合状态(composite state)的话,则直接映射为 ProcessDeclaration.
- M2F 9. RtUnit 对应的复合 State Machine 映射为 ProcessDeclaration.
  如果对应的状态机包含了子状态机状态或复合状态,那么,
  a) 首先,将子状态机状态或复合状态视为简单状态,从而使当前状态机仅包含简单状态,然后映射为 ProcessDeclaration.
  b) 然后,将子状态机或复合状态所在的 Region 映射为 ProcessDeclaration.因为在 UML 状态图(UML state machine diagram)中,子状态机状态和复合状态实质上都包含了一个子状态机(sub state machine),因此这个过程是可以迭代进行的.
  c) 最后,创建一个 ComponentDeclaration,将同一状态机中产生的 Process 通过 ParComp 设置为其并发组合.其中,ParComp 定义为 FIACRE 中并发组合操作符的元模型.
- M2F 10. RtAction 映射为 Statement.
  在进行 RtAction 的转换时我们作了如下的应用限制:首先,RtAction 构造型仅用来定义 UML 状态机中的动作(action);其次,UML 状态机中的动作均为 CallOperationAction.在这种限制下的建模活动中,UML 状

态图就被主要用来进行系统的详细设计,其中的动作都可以对应到类中的方法.因此,可以将 RtAction 构造型对应的动作类映射到 FIACRE 中的行为表达式 Statement.

- M2F 11. PpUnit 映射为 Statement.
  PpUnit 的语义与 RtUnit 非常相似,可以采用类似于映射 RtUnit 的策略将 PpUnit 映射为 Process.另外,由于 PpUnit 主要用来建模对共享资源实现并发访问,其中的服务一般都使用 RtService 构造型来定义,所以,PpUnit 中的 RtService 可以映射为 PortDeclaration.
- M2F 12. 在 UML 状态图中如果没有对应的 RtService,则将不被转换.
  我们假定 RtService 构造型只应用于 UML 中的 Operation,因此,对 RtService 的映射实际上就是对 Class 以及 Interface 中的 Operation 的映射.在前面转换 RtAction 时,我们限定了 UML 状态机中的动作都可以对应到具体操作,因此,在转换 RtAction 的同时也就完成了对 RtService 的转换.这种情况的出现主要是因为 UML 支持对系统的多视角(viewpoint)的建模.

由于基于 UML 的建模方式具有高度的灵活性,因此我们不得不在构造转换桥时对 MARTE 的应用细节作一些限制.通过第 3 节的应用示例将可以看到,这种限制并不会影响到 MARTE 对系统的建模.

2.2.3 时间约束的映射规则

在 RETMoCC 包中,实时特征,如并发、优先级等,可以直接通过 RtUnit,PpUnit,RtService 以及 RtAction 的属性进行建模.但是,更加细致的时间约束一般通过 RealTimeFeature 构造型进行建模.

RealTimeFeature 的属性 relDl,absDl,boundDl 和 miss 都与响应时间相关属性 occKind 描述了服务、动作、消息以及信号的到达模式.属性 priority 用来描述实时特征之间的优先级.在我们的转换桥中,主要关注其中几种常见的时间约束,如 occKind,miss,priority 以及 absDl.因为 FIACRE 是使用时间间隔(time interval)描述时间约束的,我们将以上时间属性转换为以整型为上下界的时间间隔,并在行为交互的端口(port)上进行定义.

2.2.4 ATL 规则总结

ATL 模型转换规则针对源模型和目标模型的元模型进行定义,通过 ATL 虚拟机的执行将源模型的实例模型转换为目标模型的实例模型.因此,构建 MARTE 到 FIACRE 的转换,就是针对 MARTE 和 FIACRE 的 KM3 元模型定义具体的 ATL 规则.我们主要采用声明式的规则定义法,通过给出源/目标模型元模型之间的对应关系构建转换规则.

图 7 给出了在 ATL 下,MARTE 到 FIACRE 转换规则定义和应用的视图.这里需要特别注意的是,图 7 中位于 *M*1 层的模型.其中,MARTE Profile 也被视作和 User Model 同一等级的模型.这似乎有点令人迷惑.其实,MARTE Profile 中所有的构造型都是基于 UML2 的元模型(如 Class,Operation 以及 Property 等)进行定义的.因此,从 ATL 的角度来看,MARTE Profile 也相当于是 *M*1 层的模型.

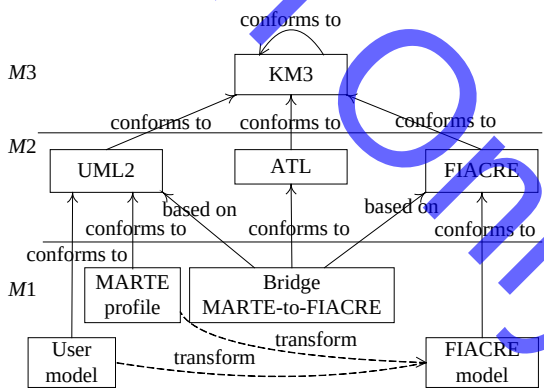


Fig.7 Overview of MARTE-to-FIACRE in ATL

图 7 ATL 下的 MARTE 到 FIACRE 转换视图

MARTE 到 FIACRE 转换规则的定义分为两个部分:1) MarteHelper.atl 定义了一些必要的基本操作(Helper),包括对 MARTE 模型的查询、判断等,非常类似于 C/C++ 中的库文件;2) Marte2Fiacre.alt 定义了具体的 MARTE 到 FIACRE 元模型之间的映射规则.

表 1 列出了 MarteHelper.alt 中所定义的部分 Helper.我们定义了如下 3 种类型的操作:

1) 专为其他 Helper 调用的底层操作,适用于所有 stereotype 模型.

2) 为特定 stereotype 定义的操作,如 RtUnit,PpUnit.

3) 专门为处理 MARTE 中的基本数据类型而定义的操作.

表 2 列出了 Marte2Fiacre.atl 中定义的部分转换规则.

Table 1 Main helpers in MarteHelper.atl

表 1 MarteHelper.atl 中主要的 helper 操作

Name	Helper description
Boolean: hasStereotype	Returns true/false regarding UML model has applied the stereotype with name or doesn't have.
UML2!Stereotype: getAppliedStereotype	Returns the stereotypes applied on a UML model.
Boolean: isRtUnit	Returns true/false regarding the model is applied with RtUnit stereotype or not.
Boolean: isMainOfRtUnit	Returns true/false regarding user models is the main RtUnit or not.
UML2!Operation: mainOfRtUnit	Returns the value of the "main" property of RtUnit.
Boolean: isPpUnit	Returns true/false regarding the model is applied with PpUnit stereotype or not.
UML2!EnumerationLiteral: concPolicyOfPpUnit	Returns the value of the "concPolicy" property of PpUnit.
Boolean: isRtService	Returns true/false regarding the model is applied with RtService stereotype or not.
UML2!EnumerationLiteral: concPolicyOfRtService	Returns the value of the "concPolicy" property of RtService.
UML2!EnumerationLiteral: synchKindOfRtService	Returns the value of the "synchKind" property of RtService.

Table 2 Rules in Marte2Fiacre.atl

表 2 Marte2Fiacre.atl 中的规则

Name	Transformation rule description
Integer2IntegerType	Match MARTE.Integer to FIACRE.IntegerType.
NFP_Duration2IntegerType	Match MARTE.NFP_Duration to FIACRE.IntegerType.
NFP_DateTime2IntegerType	Match MARTE.NFP_DateTime to FIACRE.IntegerType.
MainRtUnit2Component	Match main MARTE.RtUnit to FIACRE.ComponentDeclaration.
RtUnit2Process	Match MARTE.RtUnit to FIACRE.ProcessDeclaration.
StateMachine2Process	Match UML2.StateMachine to FIACRE.ProcessDeclaration.
State2State	Match UML2.State to FIACRE.State.
PpUnit2Process	Match MARTE.PpUnit to FIACRE.ProcessDeclaration.
RtService2Port	Match MARTE.RtService to FIACRE.PortDeclaration.
RtAction2Statement	Match MARTE.RtAction to FIACRE.Statement.

2.3 语法层的转换

2.3.1 模型到文本转换视图

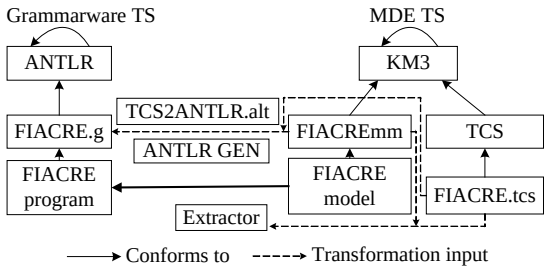


Fig.8 FIACRE model-to-text bridge

图 8 FIACRE 模型到文本桥

使用 TCS 为 FIACRE 的 KM3 元模型定义相应的具体文本语法,然后通过 TCS 工具产生 FIACRE 的模型到文本桥,就可以将模型化的 FIACRE 重新映射到文本域.图 8 给出了 TCS 下 FIACRE 模型到文本桥的构造视图.

在图 8 中,FIACREmm 表示 FIACRE 的 KM3 元模型,针对其定义的具体文本语法保存在 FIACRE.tcs 中.我们的目的是要通过 FIACREmm 和 FIACRE.tcs 获得:

1) FIACRE.g 语法规则.它是基于 ANTLR 定义的 FIACRE 标注语法(annotated grammar),即最终生成的 FIACRE Program 必须遵循的语法规则.

2) Extractor 模型到文本转换器.它是可以直接将 FIACRE 模型转换为 FIACRE Program 的桥.

FIACRE.g 是通过 TCS 中预定义的 TCS2ANTLR.atl 转换器产生的.TCS2ANTLR.atl 以 FIACREmm 和

FIACRE.tcs 为输入,将生成的语法规则和标注存到 FIACRE.g 中.Extractor 的产生也采用同样的方法,但其中还用到了 ANTLR 语法分析器生成器(ANTLR parser generator),在图中表示为 ANTLR GEN.

2.3.2 FIACRE 的文本语法

TCS 主要使用 PrimitiveTemplates 和 ClassTemplates 两种基本元素为元模型定义文本语法规则.规则的表示方式类似于 C++中的模板(template).其中,PrimitiveTemplates 主要用来为数据类型定义语法规则,而 ClassTemplates 则主要为复杂类型或类(class)定义语法规则.下面给出针对 FIACRE 中的 ComponentDeclaration 定义的 ClassTemplate.

TCS 1. ComponentDeclaration 的定义.

```
1.  template ComponentDeclaration context addToContext
2.      :   "component" name
3.          (isDefined(ports)? "[" ports{separator=","}"]")
4.          (isDefined(parameters)? "(" parameters{separator=","}")")
5.          "is" [
6.              (isDefined(variables)? "vars" [ variables{} ])
7.              (isDefined(localPorts)? "port" [ localPorts{} ])
8.              composition
9.          ]
10.     ;
```

在 ComponentDeclaration 的文本语法定义中,语法关键字 template 表示该模板是针对 Class 类型的元模型.第 2 行的"component"表示一个 ComponentDeclaration 在程序中的定义是通过关键字 component 标识的.紧跟在后面的 name 表示在 FIACRE 程序中,关键字 component 之后应该是 ComponentDeclaration 的名字,即属性 name 的值.第 3 行的关键字 isDefined 表示根据实例模型的某个属性值的情况定义相应的语法.该行表示如果 ComponentDeclaration 实例的 ports 属性被赋值,则将具体值用","分割显示.

由于 ports,parameters,variables,localPorts 和 composition 的类型分别为 PortsDeclaration,Parameters,VariablesDeclaration,PortsDeclaration 和 Composition,因此在生成 FIACRE Extractor 时,该模板还会进一步调用相应的 ClassTemplates 进行语法构造.限于篇幅,文中不再列出所有语法定义的细节.对第 2.2.1 节的 FIACRE 模型应用 ComponentDeclaration 语法表示所产生的结果如下:

FIACRE 1. 生成的 FIACRE 程序片断.

```
1.  component CoffeeMaker_Comp [CoffeeMaker_in: data,CoffeeMaker_out: data] is
2.      port CoffeeMaker_req in,
3.           CoffeeMaker_resp out
4.      par send in
5.          par
6.              startBoiling[CoffeeMaker_req,CoffeeMaker_resp]
7.           ||
8.              startHeating[CoffeeMaker_req,CoffeeMaker_resp]
9.          end
10.     end
```

3 示 例

本节在第 2 节实例研究的基础上给出 MARTE 到 FIACRE 转换桥的一个具体应用实例.通过该应用实例讨论转换桥的应用范围和效果,进一步反映出本文方法的实用价值.

3.1 问题描述

咖啡壶系统是一种常见的嵌入式实时系统,主要用于对咖啡壶的烹制咖啡过程的控制.一个典型的咖啡壶设备及其烹制过程如下:咖啡壶一次可烹制 10 杯咖啡.用户首先将滤纸放到过滤网上方,然后在上面放适量咖啡粉,之后再向煮水壶(boiler)中加满水.这时,用户还要将空的咖啡壶(coffee pot)放在煮水壶的出水口下方,以便将煮好的咖啡接入壶中.然后,用户就可以按下电源开关,开始煮咖啡了.在咖啡的煮制过程中不断有煮好的咖啡流到下面的咖啡壶中,等到全部煮好后,机器会自动关闭.在煮制过程中,用户可以将咖啡壶拿出来以便品尝咖啡,这时咖啡壶底座感应器就会感应到,上面的出水阀(pipe valve)会关闭,咖啡就不会继续流出来,直到咖啡壶放回底座上.底座同时还有一个加热板(plate heater),可以保持咖啡壶中咖啡的温度.煮水壶中除了加热器(boiler heater)外还有水位感应器(water sensor)和壶感应器(boiler sensor),可以反映壶中的水位以及沸腾程度.

3.2 使用MARTE进行系统建模

使用 MARTE 对 CMS 进行建模的过程与普通的 UML 建模过程并没有太大的区别.这里,我们略去对用例图(use case diagram)等的描述,主要关注详细设计阶段所涉及的结构和行为模型.

CMS 的类图如图 9 所示,其中有 3 个使用 RtUnit 构造型定义的控制类.系统的主 RtUnit 是 CoffeeMakerController,它提供了系统行为开始执行的入口点.CoffeeMakerController 有两个以 RtService 构造型定义的服务,其中的 exeKind 属性值被设置为 localImmediate.这表示 CoffeeMakerController 能够提供两个本地定义的服务,并且其响应方式为立即响应.BrewingController 控制类提供了对煮水壶感应器、加热器和出水阀的控制.其中,对用于控制出水阀的服务为 startPouring()和 stopPouring(),它们必须满足在一个限定时间内作出响应的要求,否则可能会发生咖啡溢出的情况.因此,对这两个服务同时使用 RtService 和 Rft 构造型进行定义.HeatingController 控制类用于控制咖啡壶底座上的加热板,它只提供对加热板的开和关两个服务.

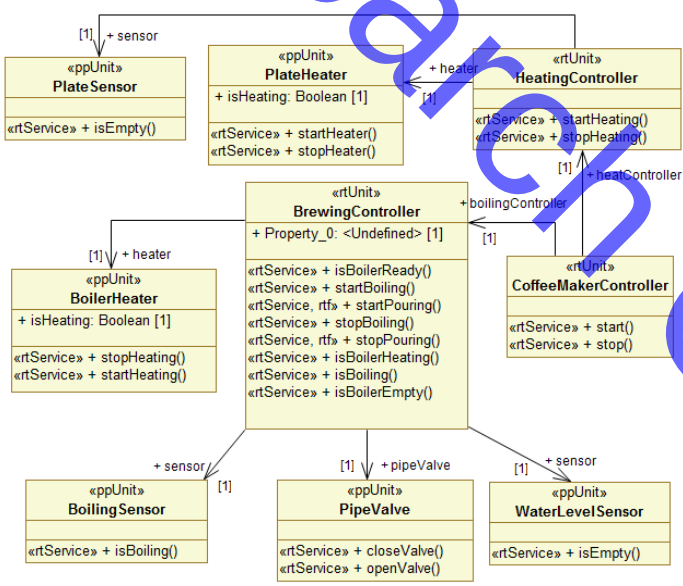
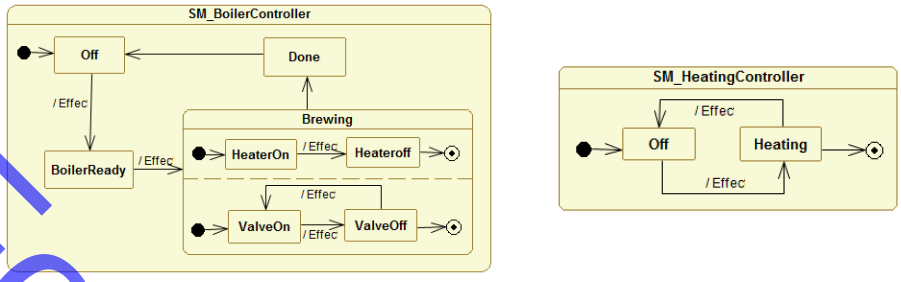


Fig.9 Overview of the classes in the CMS

图 9 CMS 的类视图

CMS 的控制类对应的状态图如图 10 所示.虽然 CoffeeMakerController 是主 RtUnit,但其状态非常简单,出于篇幅上的考虑,在图 10 中略去了对该控制类的描述.另外,本节中列出的 UML 类图和状态图均使用开源的 UML 开发环境 Papyrus[23]进行构建,其模型以 XMI 格式存放在 Coffee-Maker.uml 文件中.AMMA 平台提供了对 XMI 格式模型的操作接口,因此可以直接在平台的各个模块中使用.



(a) BoilerController state machine
(a) BoilerController 的状态图

(b) HeatingController state machine
(b) HeatingController 的状态图

Fig.10 State machine diagrams of the controllers
图 10 CMS 控制类的状态图

3.3 MARTE到FIACRE桥的转换应用

通过应用 MARTE 到 FIACRE 的转换桥,将基于 MARTE 开发的 CMS 系统模型转换为 FIACRE 模型,这相当于对 MARTE 到 FIACRE 的 ATL 规则的一次实例化.在 ADT 中需要提供相应的配置信息,以便 ATL 虚拟机得以执行.图 11 给出了该配置信息的定义界面,其内容将保存在以“.launch”结尾的 ATL 配置文件中.

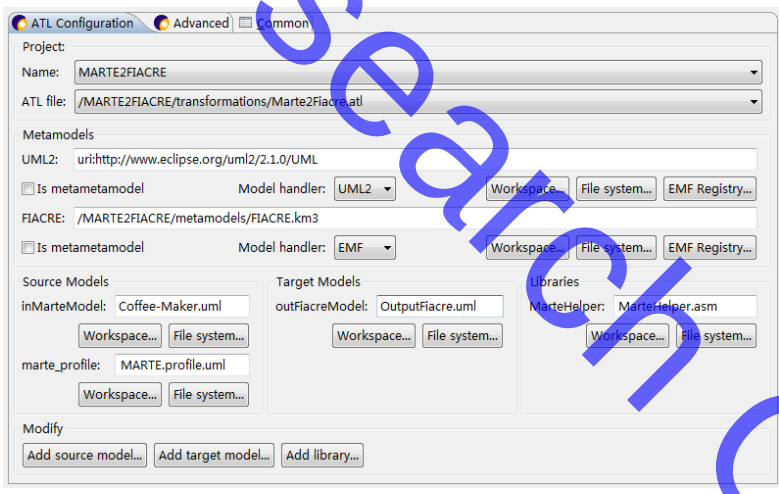


Fig.11 ATL configuration for the bridge application
图 11 转换桥应用的 ATL 配置视图

在完成 MARTE 模型到 FIACRE 模型的转换之后,就可以通过 Extractor 将这些模型转换为符合 FIACRE 语法的文本表达式.这样,针对 FIACRE 的相关验证工作和支撑工具都可以应用到 MARTE 的模型上.限于篇幅,这里不再列出所有的模型及文件内容,我们将会在 AMMA 平台的站点发布该项目,并提供相关文档以及源码的下载.

3.4 应用效果和范围

通过对 CMS 的开发以及转换桥的具体应用可以看出,本文在实例研究中所给出的 MARTE 到 FIACRE 转换桥是在实际应用中有效的.同时,我们对 MARTE 建模所作的相关限制也并未影响到 MARTE 的实际应用.

实例研究中的 MARTE 到 FIACRE 转换桥可以非常方便地应用到使用 MARTE Profile 的 UML 模型上.但这并不意味着该转换桥仅支持 UML 的建模.实际上,CMS 系统开发中所用到的 UML 可以被替换为任意一种建

模语言.用户只要在 ATL 配置中提供该建模语言的 KM3 元模型就可以了.

4 通用转换途径的讨论

通过文中 MARTE 模型到 FIACRE 形式模型转换的实例研究我们发现,在异构模型相互转换的问题上,重点是要解决异构模型之间的语义映射和语法转换两个子问题.本节总结文中实例研究的经验,针对通用转换途径讨论相关的框架和关键技术.

异构模型相互转换的工作也可以称作为异构模型构造转换桥的工作.基于 AMMA 的转换桥构造框架主要分为如下 3 个层次:

层次1. 通过 KM3 元建模将异构的模型引入到 AMMA 平台上,从而实现异构模型的同构化;

层次2. 为同构化之后的模型构建元模型之间的语义映射规则,并根据语义映射定义 ATL 转换规则,从而实现模型之间的转换;

层次3. 使用 TCS 为元模型定义具体语法,将通过 ATL 转换生成的模型进一步转换为相应的文本程序.

以上 3 个层次中的层次 2 和层次 3 分别解决了语义映射和语法转换的问题.下面分别针对这两个子问题讨论通用的转换途径.

4.1 语义层的映射

通过实例研究中的语义映射部分可以看出,各种不同的异构模型可以被理解为不同的 DSLs(domain specification languages).这些 DSLs 之间的语义对应关系可以通过它们的元模型之间的转换关系给出.

在 DSLs 元模型的基础上,可以通过 ATL 构建语义之间的映射关系,即所谓的 ATL 转换规则.从 MDE 的角度来讲,ATL 转换规则就是描述源(source)模型的元模型和目标(target)模型的元模型之间的对应关系或者转换过程.这里,源模型的元模型和目标模型的元模型实际上就是源 DSLs 语言和目标 DSLs 语言的 KM3 元模型.每一条具体的 ATL 转换规则都是单向的,但通过为正、反两个方向分别构造转换规则,就可以实现模型间的双向转换,从而实现异构模型在语义层的相互转换.

ATL 转换规则的执行是通过 ATL 虚拟机完成的,因为转换规则的定义中只涉及到了元模型层的内容,所以,具体的转换还需要对规则的具体输入模型和输出模型进行配置(见第 3 节示例).

4.2 语法层的形式转换

对异构模型的语义层转换是整个转换得以实现的基础,但仅停留在这一层仍是不够的.在实例研究的转换桥构造过程中可以看出,要真正实现异构模型的支撑工具能够互用,还必须提供语法层上的形式转换.

模型域(modelware)和文本域(grammarware)是目前异构模型的两个最主要的表现领域.在模型域中,(半)形式模型主要体现为图形化的表示法,如 UML,Petri Net,MSC(message sequence chart)等.在文本域,模型则表现为文本式的表示法,如 CCS(calculus of communicating systems)/CSP(communicating sequential processes),LOTOS(language of temporal ordering specification)等.因此,对语法层的形式转换应该充分考虑到这两个不同的表现域,以便支持不同域中的转换,甚至是跨域的转换.

在模型域中,考虑到 UML 已经在工业界成为可视化建模语言的事实标准,因此可以将 UML 作为标准的中间图形格式,利用 AMMA 平台中对 XMI(XML metadata interchange)[24]的相关操作接口将模型直接保存为 XMI 格式,从而实现不同图形化方式模型之间的交互.

对于文本域中以及模型域和文本域之间的语法转换,可以基于 AMMA 平台中的 TCS 模块实现.当实现模型域和文本域之间的语法转换时,可以通过构造 Injector 和 Extractor 直接完成相应模型和文本的形式转换(见第 1.2.3 节关于 TCS 的介绍).

5 相关工作比较

随着形式化方法和可视化建模技术的发展,出现了大量的非形式化模型、半形式化模型以及形式化模型.

这些异构的模型被广泛应用在不同领域中,却彼此难以交互和复用.因此,异构模型之间的相互转换工作逐渐成为学术界和工业界的关注焦点.

传统的解决方案大多数是 ad-hoc 式的,即针对特定的异构模型建立专门的转换,如文献[25]中提到的 SMV 到 PVS 的转换、Murphi 到 PVS 的转换、SMV 到 Spin 的转换、Automata 到 Petri Net 的转换,以及文献[26]中提到的 Interface Automata 到 I/O Automata 的转换等.还有一些常见的基于消息序列、工作流的转换,如文献[27]给出了从 UML 顺序图和状态图到 Petri Nets 的转换,文献[28]定义了从 Message Sequence Charts(MSCs)到 Queueing Network Model 的转换,文献[29]将 MSC 模型转换到 Statecharts 模型,文献[30]将 UML EDOC profile 模型转换为 BPEL 模型,而文献[31]将 BPEL 模型转换为 annotated deterministic finite state automata 模型,等等.

这种方式逐渐暴露出了很多问题.首先是语义匹配和语法映射常常交织在一起.这一方面使转换关系显得复杂和难以理解;另一方面,在原有模型增加新的语言成分时,原来的转换关系往往需要重新构建.其次,作为转换工作的基础,异构模型的元模型应该具有高度的抽象性,能够独立于语义匹配和语法映射而存在.但大部分转换工作往往将元建模的工作隐含在语义匹配中,这也使得语义匹配和语法映射很难分隔开.另外,由于转换规则是针对特定的异构模型而构建的,因此很难得到复用.这使得不同研究机构的学者在开发新的转换规则时,不得不从头开始.甚至在同一个研究机构中,也难以直接基于现有的语义匹配和语法映射结果,开发向新异构模型的转换.最后,由于缺乏标准的支持,不同的转换支撑工具之间往往无法交互和集成.这也在很大程度上制约了异构模型相互转换工作的进一步发展.

为了避免 ad-hoc 式的异构模型转换方式,S. Katz 等学者在文献[25,32]中提出了一个通用转换框架及支撑工具 VeriTech,M. Gallardo 等学者在文献[33]中提出一个基于 XML 的分析工具集成框架 PiXL.他们的工作在一定程度上解决了 ad-hoc 式的工作所带来的重复性开发、通用性差以及复用程度低等问题,但仍然不能很好地处理抽象元模型以及分离语义匹配和语法映射的问题.

在总结了上述工作所存在问题的基础上,我们从 MDE 的视角和思维方式来重新审视异构模型的转换问题,进行了从 UML/MARTE 模型向 FIACRE 形式模型转换的实例研究.通过该实例研究,我们较好地分离了语义映射和语法转换两个子问题,并基于 AMMA 平台较为完整地实现了异构模型的同构化、基于模型转换技术的语义映射以及基于 TCS 语法构造技术的文本语法转换.

与上述工作相比,本文工作的区别和优势在于:

- 有框架性标准规范的支撑.本文实例研究使用的转换技术和语法构造技术都是遵循 OMG 提出的 MDA 框架性标准,从而避免了因为使用不同方法而出现转换桥的异构问题,使基于本文思想构造的转换桥具有更好的重用性.例如,可以采用类似于 Web Service 的服务发布与搜索方式将不同组织和研究机构开发的转换桥进行共享和重用.
- 在构建异构模型转换桥的过程中,每一步的结果都可以重用.通过实例研究可以看出:KM3 元建模得到的异构模型的元模型可以在其他转换桥的构建中重用;ATL 转换规则本身也是一种模型,并支持通过继承(inheritance)的方式复用;针对元模型的 TCS 具体语法也可单独在模型的文本化工作中使用.
- 易于实现对语义转换和语法转换两个子问题的划分.实例转换的过程明显地区分出了语义映射和语法转换两个子问题.这样,不仅实现了对复杂问题进行分而治之的处理,同时还有利于广泛支持模型域和文本域的异构模型转换.

6 结 论

异构模型的相互转换问题是目前学术界和工业界的研究热点.本文尝试改变对该问题的研究思路,从 MDE 的视角入手,通过复用现有的模型驱动方法和工具,实现异构模型的相互转换.本文通过 UML/MARTE 模型向 FIACRE 形式模型转换的实例研究,讨论了如何分离语义映射和语法转换这两个子问题并通过 AMMA 平台实现转换桥的构造.最后,在总结实例研究的基础上,我们针对通用转换途径讨论了相关的框架和关键技术.我们认为,本文工作的主要贡献在于:

- 具备较强的实用性.通过文中给出的实例研究展示了如何具体地进行异构模型的同构化,然后分别实现语义映射和语法转换.我们基于 AMMA 平台给出了实现过程的细节,具有较强的实用价值.
- 展示了如何以分离的方式解决语义和语法两个转换子问题.文中实例研究和方法总结部分都强调了如何分别在语义层和语法层实现转换.这种方式有利于以“分而治之”的思想解决问题,可以有效降低解决问题的复杂度.
- 为转换规则的重用建立基础.通过 AMMA 平台构造的转换桥具有很强的重用性,可直接应用于 KM3 下的模型.因此,已开发的转换桥可以通过 AM3 注册和管理,并在 Internet 上发布,以便由其他用户搜索获取,进而实现重用.

我们的工作有助于在软件开发过程中集成多种模型及开发方法,重用不同开发方法的验证途径和支撑工具.通过将一种软件系统规约形式转换到不同的模型上,就可以利用该模型的验证工具对现有规约进行验证.

本文的工作需要进一步完善的地方是,本文是通过一个典型的实例研究探讨和解决异构模型转换问题,在此基础上还需要进行更多的实例研究工作,以便总结出完善的通用转换途径.此外,本文工作对使用者的要求也较高.用户必须既要具备特定可视化/形式化建模方法领域的相关知识,又要掌握 MDE 领域的理论和 AMMA 平台的使用技术.因此,尽可能地提供有效的相关支撑工具,也是我们进一步的工作.

致谢 在此,我们向对本文的工作给予支持和帮助的同行,尤其是 INRIA&ATLAS 的工程师 Fréddy Allilaire, Hugo Bruneliere,博士生 Michaël Barbero 以及北京大学软件研究所的博士后张岩表示感谢.

References:

[1] OMG. Unified modeling language: Superstructure v2.0. 2005. http://www.omg.org/docs/formal/05-07-04.pdf

[2] FIACRE Home Page. http://www-sop.inria.fr/oasis/fiacre/

[3] OMG. A UML profile for MARTE Beta 1. 2007. http://www.omg.org/cgi-bin/doc?ptc/2007-08-04

[4] France R, Rumpe B. Model-Driven development of complex software: A research roadmap. In: Knight J, ed. Future of Software Engineering (FoSE) on the 29th Int'l Conf. on Software Engineering. Washington: IEEE CS Press, 2007. 37−54.

[5] Bezivin J, Jouault F, Rosenthal P, Valduriez P. Modeling in the large and modeling in the small. In: Aβann U, Aksit M, Rensink A, eds. Proc. of the Model Driven Architecture: Foundations and Applications 2003/2004. Berlin: Springer-Verlag, 2004. 33−46.

[6] Miller J, Mukerji J. MDA guide version 1.0.1. OMG, 2003. http://www.omg.org/docs/omg/03-06-01.pdf

[7] Jouault F, Bezivin J. KM3: A DSL for metamodel specification. In: Gorrieri R, Wehrheim H, eds. Proc. of the 8th IFIP Int'l Conf. on Formal Methods for Open Object-Based Distributed Systems. Berlin: Springer-Verlag, 2006. 171−185.

[8] ATLAS Team. ATLAS Transformation Language (ATL) Home Page. http://www.eclipse.org/gmt/atl/

[9] Jouault F, Bezivin J, Kurtev I. TCS: A DSL for the specification of textual concrete syntaxes in model engineering. In: Jarzabek S, ed. Proc. of the 5th Int'l Conf. on Generative Programming and Component Engineering (GPCE 2006). New York: ACM Press, 2006. 249−254.

[10] Modelware Home Page. http://www.modelware-ist.org/index.php

[11] Kort J, Klint P, Klusener S, Lammel R, Verhoef C, Verhoeven E. Engineering of grammarware. 2005. http://www.cs.vu.nl/grammarware/

[12] Klint P, Lammel R, Verhoef C. Toward an engineering discipline for grammarware. ACM Trans. on Software Engineering and Methodology (TOSEM), 2005,14(3):331−380.

[13] INRIA Home Page. http://www.inria.fr/index.en.html

[14] ISO. ISO 8807. LOTOS Standard, 1989. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=16258

[15] CADP Home Page. http://www.inrialpes.fr/vasy/cadp/

[16] TINA Home Page. http://www.laas.fr/tina/

[17] OMG. Meta object facility core specification v2.0. 2006. http://www.omg.org/cgi-bin/apps/doc?formal/06-01-01.pdf

[18] Jouault F, Kurtev I. Transforming models with ATL. In: Bruel J, ed. Proc. of the Satellite Events on the 8th Int'l Conf. on Model Driven Engineering Languages and Systems. Berlin: Springer-Verlag, 2006. 128−138.

[19] OMG. MOF 2.0 Query/Views/Transformations RFP. 2002. http://www.omg.org/cgi-bin/apps/doc?ad/2002-04-10.pdf

[20] OMG. Object constraint language v2.0. 2006. http://www.omg.org/cgi-bin/apps/doc?formal/06-05-01.pdf

[21] OMG. MOF model to text transformation language request for proposal. 2004. http://www.omg.org/cgi-bin/apps/doc?@ad/04-04-07.pdf

[22] Parr TJ, Quong RW. ANTLR: A predicated-*LL*(*k*) parser generator. Software—Practice and Experience, 1995,25(7):789−810.

[23] CEA LIST Team. Papyrus (open source tool for graphical UML2 modeling). 2007. http://www.papyrusuml.org/

[24] OMG. XML metadata interchange v2.0. 2003. http://www.omg.org/cgi-bin/apps/doc?formal/03-05-02.pdf

[25] Katz S, Grumberg O. A framework for translating models and specifications. In: Butler M, ed. Proc. of the 3rd Int'l Conf. on Integrated Formal Methods (IFM 2002). Berlin: Springer-Verlag, 2002. 145−164.

[26] Zhang Y. Interface automata based components behavior derivation [Ph.D. Thesis]. Nanjing: Nanjing University, 2006 (in Chinese with English abstract).

[27] Bernardi S, Donatelli S, Merseguer J. From UML sequence diagrams and statecharts to analysable petri net models. In: Balsamo S, Inverardi P, eds. Proc. of the 3rd Int'l Workshop on Software and Performance. New York: ACM Press, 2002. 35−45.

[28] Andolfi F, Aquilani F, Balsamo S, Inverardi P. Deriving performance models of software architectures from message sequence charts. In: Woodside M, ed. Proc. of the 2nd Int'l Workshop on Software and Performance. New York: ACM, 2000. 47−57.

[29] Kruger I, Grosu R, Scholz P, Broy M. From MSCs to statecharts. In: Rammig FJ, ed. Proc. of the IFIP WG10.3/WG10.5 Int'l Workshop on Distributed and Parallel Embedded Systems. Norwell: Kluwer Academic Publishers, 1999. 61−71.

[30] Yu X, Zhang Y, Zhang T, Wang L, Zhao J, Zheng C, Li X. Towards a model driven approach to automatic BPEL generation. In: Akehurst DH, Vogel R, Paige RF, eds. Proc. of the 3rd European Conf. of Model Driven Architecture-Foundations and Applications (ECMDA-FA 2007). Berlin: Springer-Verlag, 2007. 204−218.

[31] Wombacher A, Fankhauser P, Neuhold E. Transforming BPEL into annotated deterministic finite state automata for service discovery. In: Zhang LJ, ed. Proc. of the IEEE Int'l Conf. on Web Services. Washington: IEEE CS, 2004. 316−323.

[32] Katz S. Faithful Translations among models and specifications. In: Oliveira JN, Zave P, eds. Proc. of the Int'l Symp. of Formal Methods Europe on Formal Methods for Increasing Software Productivity. Berlin: Springer-Verlag, 2001. 419−434.

[33] del Mar Gallardo M, Martinez J, Merino P, Nuñez P, Pimentel E. PiXL: Applying XML standards to support the integration of analysis tools for protocols. Science of Computer Programming, 2007,65(1):57−69.

附中文参考文献:

[26] 张岩.基于接口自动机的构件行为获取[博士学位论文].南京:南京大学,2006.

张天(1978－),男,江苏徐州人,博士生,主要研究领域为模型驱动软件工程.

Frédéric JOUAULT(1979－),男,博士,主要研究领域为模型驱动工程.

Christian ATTIOGBÉ(1959－),男,博士,教授,博士生导师,主要研究领域为形式化方法和规约,形式化开发及支撑工具.

Jean BÉZIVIN(1953－),男,博士,教授,博士生导师,主要研究领域为模型驱动工程.

李宣东(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件建模与分析,软件测试与验证.