# AADL工具集中的行为附件的实现Osate2

Abstract-AADL是一种用于设计和分析高分辨率分布式和实时系统的建模语言。 作为AADL附件出版的嵌入式子语言扩展了AADL模型以加强分析。 行为附件指定了AADL应用程序模型的行为。 因此，本附件的植入允许执行行为分析。 另外，由于有几个AADL附件，实施通用机制来支持其中的每一个都是具有挑战性的。 行为附件是通过组合几种子语言来说明这些挑战的有效候选人。 在本文中，我们公开我们的实验来支持参考AADL工具集OSATE2中的行为附件。 这一个，通过提供前端和一组分析插件来分析AADL模型来支持AADL版本2。  
指标术语 - AADL;AADL-BA; 行为; 附件;Osate2;MDD。

## 一，引言

架构分析和设计语言（AADL [1]）是一个域特定建模

针对高完整性（HI）分布式和实时（DRE）系统的语言（DSML）。它允许对分布式，可重新配置或分区系统进行建模，分析和生产软件系统组件[2]。

AADL核心语言旨在可扩展，并为用户或工具提供功能，以便使用附加属性集或附件语言（如错误建模附件或行为附件）来优化AADL应用程序模型的语义和分析功能。他们集成了批准的外部子语言来加强DRE系统分析。

AADL行为附件[3]是指定AADL应用程序模型行为的扩展。它通过将行为规范（例如状态机）附加到每个AADL组件来优化语言核心中指定的隐含行为。因此，本附件的实施允许对HI-DRE系统执行行为分析。由于有几个附件来扩展AADL应用程序模型的分析功能，因此实施通用机制以支持其中的每一个都是具有挑战性的。

他的行为附件是一个有效的候选人，以说明许多挑战。 1）它需要解析和分析几个子语言，而不仅仅是一个。 2）不同的AST生产需要连接进行分析。 3）完成分析，要求确保与核心语言的一致性。 4）附件的内部表示需要符合核心语言内部表示，以便在分析中提供独特的代表作为输入。

为了协助开发具有AADL的HI-DRE系统，软件工程研究所（SEI）开发了AADL工具集OSATE2。这是一个可扩展的开源平台，它包括AADL前端，架构分析功能和扩展机制，将外部后端作为插件进行集成。  
本文介绍了我们的实验来支持行为附件作为参考AADLtoolset OSATE2的扩展插件[4]。挑战涉及到选择用于指定AADL-BA元模型的策略以及将其集成到OSATE2中的AADL-BA编译器的设计。我们解释我们如何重用多个OSATE2模块来驱动用AADL-BA元素完成的AADL模型的分析。  
本文共分五部分。第二部分显示OSATE2和子语言扩展过程的概述;第三部分介绍了执行行为附件的简要概述和挑战;第四节介绍了我们制定AADL-BA元模型的策略;第五节描述了行为附件的实施及其在OSATE2中的整合;第六节介绍结语和未来的作品。

II。 AADL工具OSATE2概述

A.工具架构  
开源AADL工具环境（OSATE [5]）已经被开发为ECLIPSE平台的一组插件。 OSATE2是支持AADL版本2的版本。其主要组件如图1所示。  
OSATE2前端包括一个解析器，名称解析器和语义检查器来处理文本的AADL模型并将它们转换为内部表示。这种内部表示可以通过一个更加粗略的或者以基于XMI的格式（AAXL2文件）保存的方式被序列化成文本AADL。文本AADL模型仅声明组件，必须实例化（由AADL实例化器）进行分析。实例模型也以XMI格式持续存在。  
为了分析AADL模型，OSATE2提供了一组分析插件。分析插件是ECLIPSE插件，它使用OSATE2基础架构来访问模型，报告错误等。分析插件通常在Java中实现，但可以选择其他实现语言，例如Groovy。

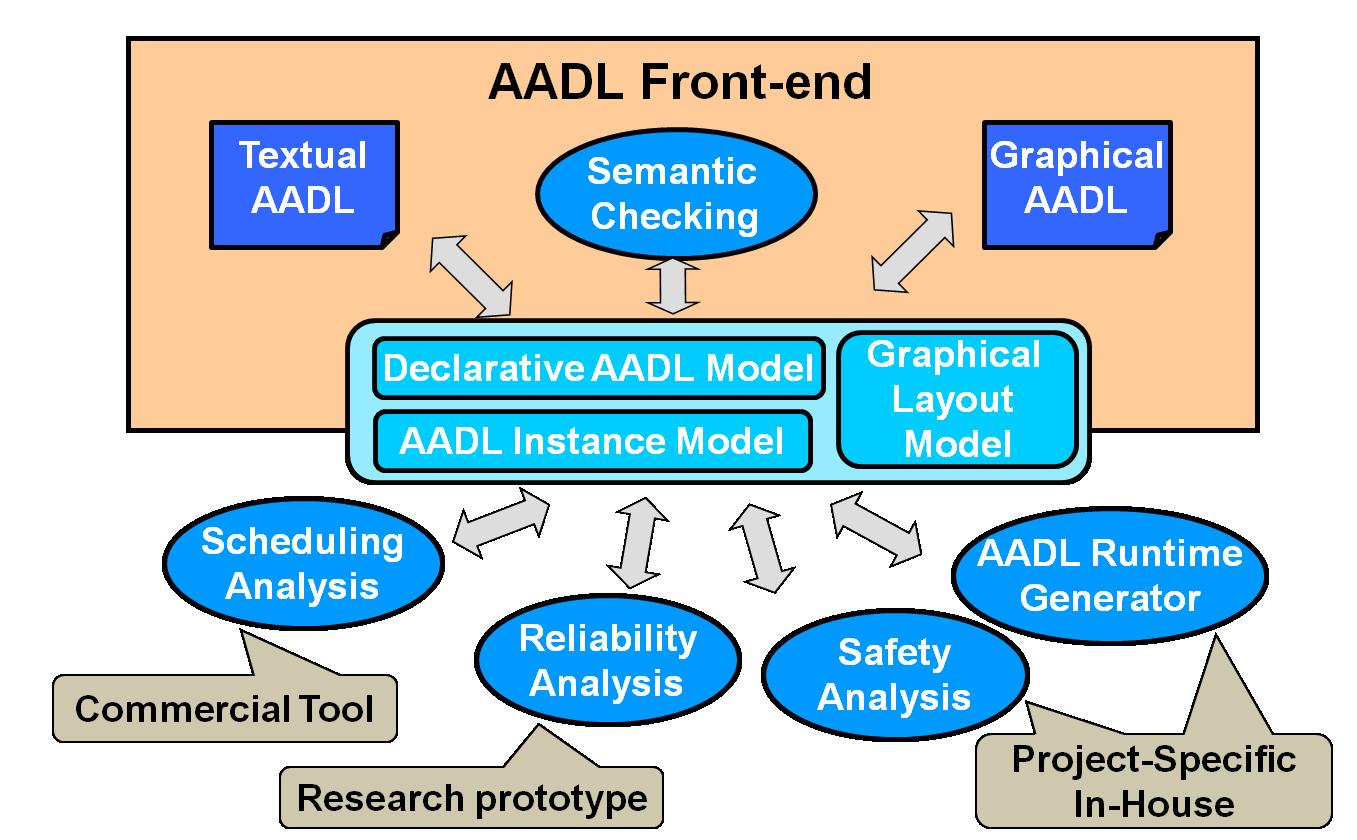


图1：OSATE2架构

外部分析工具可以集成到  
OSATE2通过创建从AADL模型中提取工具所需的数据并调用该工具的插件。分析工具也可以直接访问XMI交换格式。  
OSATE2将AADL2元模型用于AADL模型的内部表示（声明和实例）。该元模型在UML2中定义，并使用ECLIPSE / UML2与ECLIPSE建模框架（EMF）一起实现。这将自动提供XMI格式和代码来读写从AADL型号构建的相应的AAXL2文件。  
B. AADL中的子语言  
AADL是可扩展的（a）用户定义的属性集和（b）嵌入式子语言。用户可以将新属性定义为模型的一部分，并将其与建模元素相关联。可以在模型分析过程中评估这些属性。属性集机制主要用于将这些特征添加到可以以结构化值形式表示的建模元素，例如字符串，枚举值，带有或单位的数值，对其他模型元素的引用，以及由这些价值观。子语言允许将更复杂的结构添加到AADL模型中。子语言可以被标准化，并作为AADL附件出版。已经定义了几个这样的附件，例如，错误建模子语言用于定义AADL模型的错误状态和故障传播，并且行为附件允许将详细的组件行为建模为状态机。  
子语言片段作为附件或附件子句包含在文本AADL模型中。附件库直接包含在AADL包中。它载有可用于附件图书馆和小节的附件要素的声明。

相比之下，附件小节包含在分类器声明中，这使得在组件的子部分中定义的元素。 这类似于在包中声明核心AADL分类器并在其他分类器的声明中使用的方式，例如作为子组件。

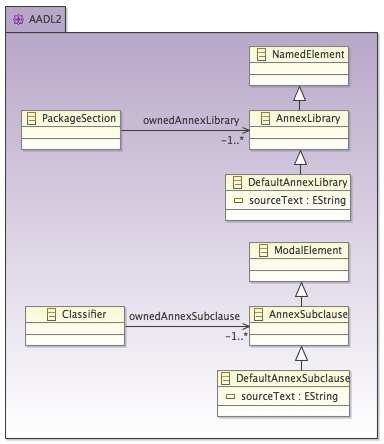


图2：OSATE2附件扩展

在以下意义上，附件与核心AADL是分开的：如果从AADL模型中删除所有附件库，子条目和附件相关的属性关联，则生成的模型是有效的核心AADL模型。此外，假设不同的附件彼此独立。  
C. OSATE2中的子语言  
通过创建一个或多个实现必要组件的插件，可以将支持子语言添加到OSATE2。通常，这些包括（a）附件库和子句的内部表示，以及（b）处理子语言的组件，例如解析器和较为粗糙的名称解析器和语义检查器。  
核心AADL元模型包含两个代表顶级附件元素的抽象UML类：AnnexLibrary和AnnexSubclause（见图2）。这些类必须通过子语言的元模型进行扩展。当附件在ECLIPSE中实现时，附件元模型可以使用UML2定义并引用核心AADL元模型。然后，ECLIPSE中的UML2 / EMF基础设施可用于生成附件元模型的实现，该模型定义附件库和子句的内部表示，并支持XMI中的序列化和解串行化。  
OSATE2为子语言处理组件提供了注册表，插件可以注册解析器，名称解析器等。核心语言处理函数在需要时调用附件特定的功能，例如，核心AADL解析器每当到达时调用行为附件解析器AADL源文本中的行为附件条款。由于不同的附件被视为彼此独立和核心语言，因此在处理模型时可以忽略附件。 OSATE2通过定义一个默认的附件元模型，解析器和不起眼来支持这一点。默认的附件元模型将附件库或子条款的源文本简单地存储为字符串，默认的解析器和不起眼的内容不重要，因为它们不处理附件内容。即使未安装特定附件的插件，也允许使用附件元素处理AADL模型。



778/5000

III。 AADL-BA和模式驱动开发  
A. AADL行为概述附录  
AADL行为附件允许通过五个子语言附加到每个AADL组件的“行为规范”。 每个子语言使用以下语法定义行为概念：语法; 一组用于验证行为规范的命名和合法性规则; 一组语义规则来验证其语义的规范; 和一套规则，以确保与核心语言的一致性。  
图3显示了如何组合五种子语言来描述零星线程接收器的行为。 在接收到Data\_Sink端口上的数据时，零星线程会将接收到的最后一个值递增，并将结果存储在共享数据中。



图3：零星线程行为规范

第一种语言定义了具有变量的状态/转换自动机，行为条件（例如，警卫）和描述接收器组件执行行为的动作。它将不同种类的状态指定为“完整”状态stExec（第17行），表示挂起/恢复状态，线程被调度。  
转换表示从当前状态到目标状态的变化，并且当其行为条件被评估为真时被激活。然后执行附加到转换的动作块（'{'...'}'）。  
行为条件表示为执行或调度条件。调度条件由第二种语言定义，并改进了在AADL线程调度行为[1]中作为触发器使用的端口和子程序调用的使用。在我们的示例中，在AADL线程界面（第3行）中定义的Data\_Sink端口用于指定行为规范中的调度条件（第20行）。  
第三种语言定义交互操作来优化通过AADL共享数据，端口，子程序调用等描述的AADL组件交互。通过行为动作（第四语言）表达转换触发时处理的动作。这将使用交互操作语言来定义组件之间的动作，以调用由共享数据提供的子程序doUpdate（第22行）来更新其内部值。  
最后一种语言是一种表达式语言，它提供逻辑，关系和算术表达式来操作AADL数据组件和行为变量，因为lastValue增加了Data\_Sink端口上接收的值（第21行）。  
最后，已经提出了用于定义或解释AADL-BA的语义的几种方法[7]，[8]。还开发了基于AADL 1.0的旧AADL-BA规范的实现[9]。然而，对AADL元模型进行的几项修改，以支持其版本2.0和当前的AADL-BA草案不允许重新使用此第一个实验。  
B.实施AADL-BA的组成部分  
a）MDD方法：像AADL这样的DSML简洁地表达其域的常见概念。它提供了几个功能：一个由元模型描述的抽象语法，它定义了域的概念及其关系;具体的语法（文本或图形）定义概念如何表示;和通过定义转换描述的语义，将概念与正式方法或目标编程语言映射为Ada等。在这种方法中，元模型被提升为“中心”和“管理”角色。  
AADL-BA是表示HIDRE系统行为的DSML，并提供相关元素以使用MDD方法实现。由于附件定义了几个子语言，我们需要有效地设计其元模型。

我们观察到每个AADL-BA子语言都需要使用核心语言描述的AADL组件。一些语言作为状态/转换语言（或行为动作）需要使用其他AADL-BA子语言，例如调度条件（或交互操作）子语言。这些要求表示为附件的不同子语言与AADL核心语言之间的依赖关系。这些依赖关系对于选择设计第四节详述的AADL-BA元模型的策略有影响。  
b）规则实施：该标准定义了验证行为概念“合法”并验证其语义的规则。由于在子语言中定义的行为概念可以与在另一个子语言中定义的行为概念相结合，因此可以在两个子语言部分中指定一些规则。为了避免代码复制并简化维护工作，我们选择了一个验证整个行为规范的规则的子集。  
在以下部分中，我们公开了AADL-BA元模型和AADL-BA编译器的体系结构。

IV。 AADL-BA元模型  
在上一节中，我们简要介绍了行为附件，它提供了使用MDD方法实现所需的所有功能。  
为了设计AADL-BA元模型，我们重新使用用于指定AADL元模型的EMF框架。一方面，它有助于在依赖性和嵌入式Java API方面将AADL-BA集成到OSATE2中。另一方面，使用相同的形式来指定两个元模型可以简化对象依赖关系的表达，并简化它们之间的导航。  
在下文中，我们描述了将AADL-BA概念映射到EMF元模型元素的策略和规则。  
A.单个AADL-BA元模型  
在经典的元模型开发过程中，自然的解决方案是为每个AADL-BA语言定义元模型。因此，它需要在不同的元模型之间表达几个依赖关系，并且使用AADL元模型来复制依赖关系。  
为了简化行为规范的对象与核心语言之间的导航，并减少不同的依赖关系，我们选择为AADL-BA附件中指定的所有语言开发单个元模型。

B.将AADL-BA概念映射到元模型元素

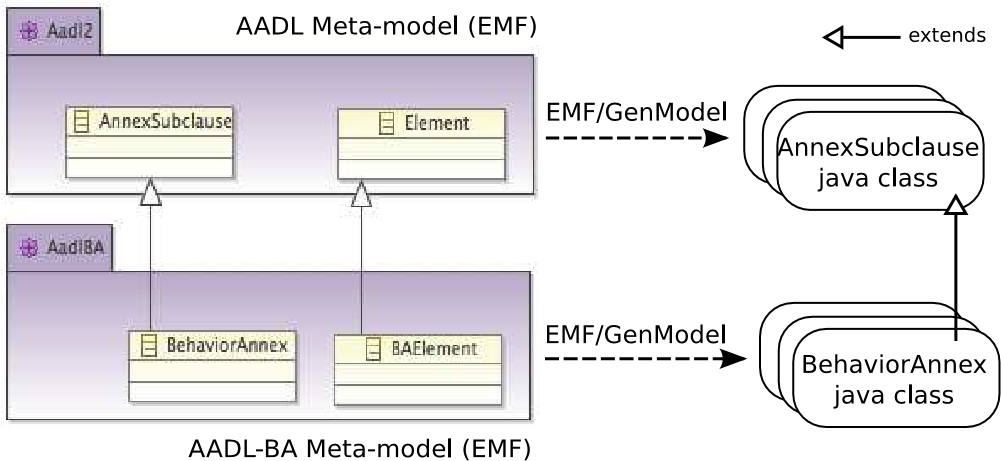


图4：AADL-BA元模型依赖关系

AADL-BA EMF元模型描述了行为规范的结构（例如，AADL-BA模型），并且明确了所有由标准表达的概念。然后，EMF生成与元模型对象相对应的Java实现类（见图4）。  
要指定AADL-BA元模型，我们定义一些将AADL-BA概念映射到元模型对象的规则。这些规则通过关注1）限制生成的类的数量来简化元模型的复杂性和生成类，2）仅保留用于定义概念的相关元素，以及3）促进其在外部的重用分析工具。  
在下面我们给出主要规则：  
R1：具有强语义和具体文本表示（例如，BNF）的行为概念被映射到元模型EClass（例如，java类）。几个AADL-BA概念作为图5中描述的BehaviorAnnex具有具体的文本表示和规则来精确描述其语义。因此，将这些概念映射为EMF元模型的EClass是很自然的。  
R2：具有弱或无语义但用于阐明概念层次结构的行为概念映射到抽象EClass。作为BehaviorCondition的一些概念（见图5）没有具体的文本表示和真正的语义，而是表达为简化和澄清对附件的理解。在这种情况下，我们将这些概念映射为抽象类。它们的使用有助于简化分类器中的类层次结构和访问者的实现以及基于外部模型的分析工具。  
R3：属于同一系列的行为概念相对于Java的继承机制进行映射。 AADL-BA中的一些概念作为循环语句（例如for / forall，while等）共享共同属性。在这种情况下，它们的映射将父类定义为抽象EClass，子类为EClass，使用Java的继承机制。该规则允许将代码分解并简化编译器开发及其维护。  
R4：表示一个行为概念需要另一个行为概念映射到EReferences的链接。作为EMF EReference表示两个类之间的关联的一端，它被用于链接（例如，参考）两个行为概念。

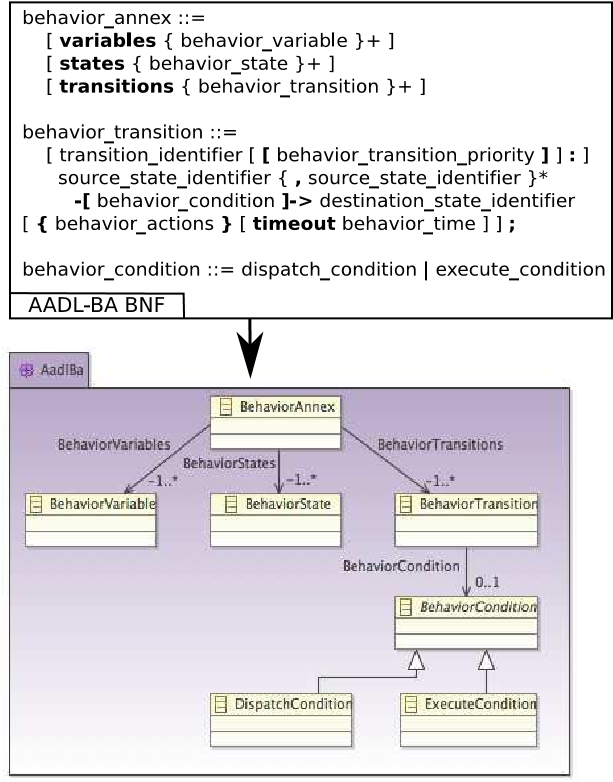


图5：AADL-BA：元模型和BNF

C.AADL元模型依赖  
行为附件要求将行为规范附加到AADL组件并将AADL-BA对象与AADL对象进行链接。它通过实施OSATE2子语言扩展来加强，这需要链接a  
BehaviorAnnex对象到AADL组件的AnnexSubclause对象（见图4）。  
此外，该标准提供了若干规则，以确保AADL架构组件与其行为规范之间的一致性。这些规则由我们的不同的AADL-BA分析仪进行验证，需要通过AADL-BA元模型和AADL元模型进行浏览。  
图4显示了我们如何使用EMF扩展（例如，Java的继承机制）来表达两个元模型之间的依赖关系。因此，一个BehaviorAnnex扩展了一个附件子句，一个BAElement扩展了一个元素。最后一个简化了从AADL-BA模型到AADL模型的导航，可以在行为规范中引用AADL对象，并在分析过程中轻松检索相应的AADL对象。  
在下一节中，我们将描述我们如何使用AADL-BA元模型作为骨干来构建涉及到AADL-BA编译器架构的几个模块。  
V.实施和整合  
AADL-BA编译器  
OSATE2中行为附件的集成允许重新使用实现OSATE2提供的子语言扩展机制的AADL元模型，AADL前端和附件插件。  
在本节中，我们详细介绍了AADLBA编译器作为集成到OSATE2的ECLIPSE插件的实现，以及AADL-BA元模型如何作为构建编译器的几个模块的骨干网。  
编译器架构  
图6显示了“经典”AADL-BA编译器架构，其中包括两个部分：前端和后端。从AADL-BA元模型和EMF，我们生成AADL-BA构建器工厂来构建和操作编译器中使用的AADL-BA对象。前端包含两个模块：解析器和分析器。错误处理由OSATE2错误管理器管理。

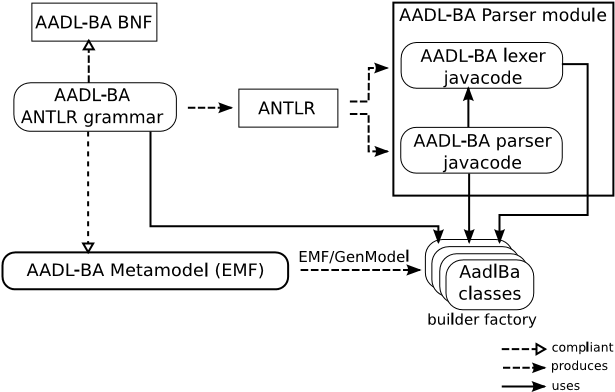


图7：AADL-BA解析器

1. 解析器：AADL-BA BNF到ANTLR语法：图7表示解析器的架构。 我们使用ANTLR框架根据AADL-BA BNF构建解析器。 图8显示了BNF定义的规则很容易映射到ANTLR语法。  
   ANTLR允许在语法规则中附加Java代码声明。 由于我们的规则R1，R2，R3和R4（见第四节）来指定AADL-BA元模型，我们仅使用ANTLR语法中的AADL-BA构建器工厂的方法来指定如何构建抽象语法树 （AST）与AADL-BA对象。 它确保AST符合AADL-BA元模型。 最后，我们使用ANTLR框架从定义的AADL-BA ANTLR语法生成解析器和词法分析器的Java类。

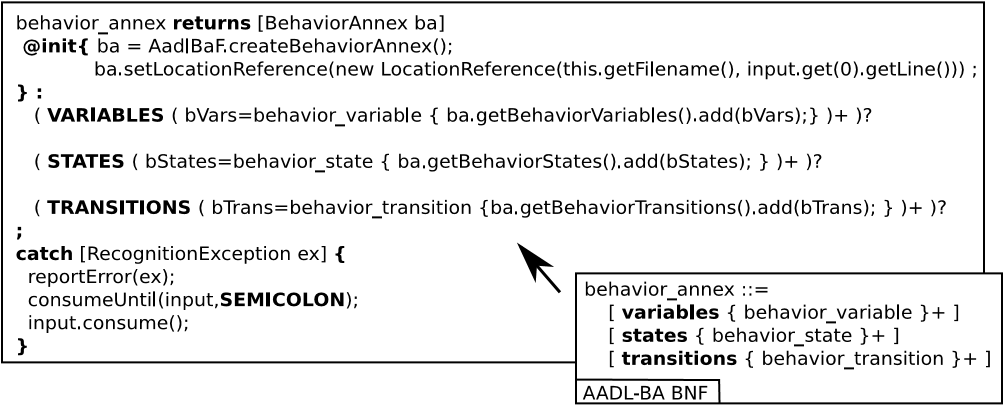


图8：AADL-BA BNF到ANTLR语法

1. 命名，合法性，一致性和语义分析：分析器模块扫描AST并检查AADL-BA模型的语义。首先，它进行到将AADL-BA对象链接到它们对应的AADL对象或AADL-BA对象的解析阶段（例如命名解析器）（参见图3，源标识符stExec和transition stExec\_stExec）。为了实现这个阶段，我们使用OSATE2提供的访问者（例如，java类）来检索AADL对象。对于AADL-BA，我们开发了访问AADL-BA AST所需的访问者。此阶段将信息添加到AST中，使其更容易使用。  
   其次，我们已经实施了足以验证整个行为规范的规则子集。第二阶段进行了对合法性，语义和一致性规则这一子集的验证。  
   为了验证与AADL组件的一致性，我们开发了一些特定的访问者来浏览AADL模型。该分析的结果导致AST符合AADL-BA语义并与AADL模型一致。该AST作为附件的内部表示，可用作后端部件或外部工具的输入。我们选择了相同的策略来实现我们的分析仪。我们考虑整个行为规范（见图3）。首先，我们分析变量和状态。然后，我们分析过渡的结构，行为条件，最后分析过渡过程中的动作块。
2. 后端集成：AADL-BA是描述AADL应用程序模型行为的形式主义。标准的实现只关注行为规范的语法和语义的验证。因此，可以描述具有死锁的“合法”线程行为规范。为了验证这些类型的属性，我们需要使用模型检查器。  
   为了实现这一目标，我们提供一个后端注册表。由于ECLIPSE插件扩展点，我们允许将外部基于ECLIPSE的插件注册为后端，并将其用作从AADL-BA前端生成的AST的输入。  
   我们还开发了一个不起眼的后端，从AADL-BA AST生成相应的文本行为规范。它允许我们通过构建中间模型来验证我们编译器的不同模块。它还用于生成由我们团队开发的标准和AADL-BA模型示例构成的测试套件的不同输出。

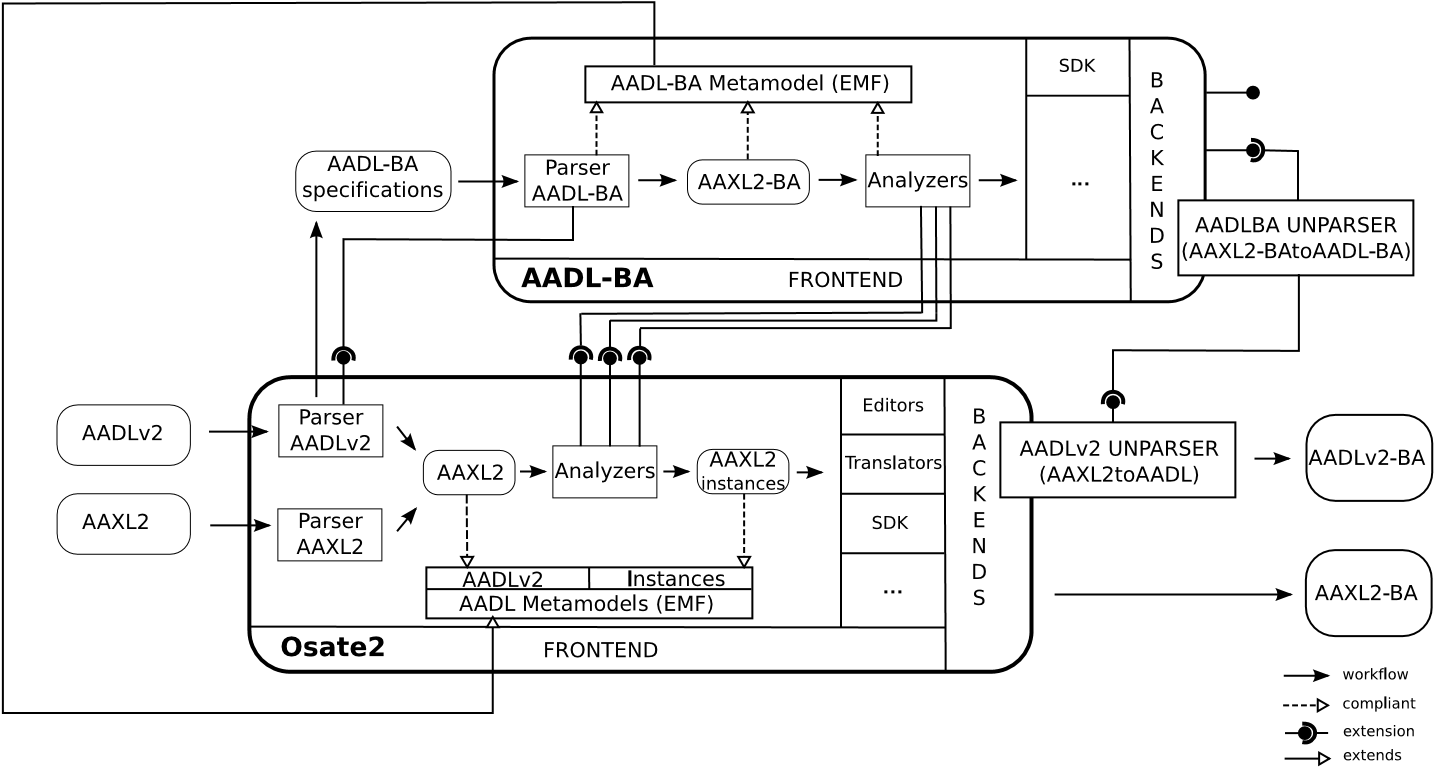


图6：集成到OSATE2的AADL-BA插件

整合OSATE2  
图6显示了我们的行为附件插件在OSATE2中的集成。  
1）插件扩展：AADL-BA插件的集成是两个步骤的过程。首先，我们将AADLBA插件连接到OSATE2附件插件。附件插件定义了允许插件连接在一起的ECLIPSE扩展点。插件通过声明扩展扩展另一个。在我们的AADL-BA插件中，我们声明扩展对应于由OSATE2声明的扩展点，以扩展AADL解析器和分析器。  
其次，我们必须在OSATE2附件注册表中注册我们的分析器（分析器）。由于AADL-BA是AADL描述的一部分，AADL-BA插件不是后端，而是由OSATE2直接集成和驱动。  
2）产出和后端扩展：AADL-BA AST是行为附件的内部表示。由于AADL-BA元模型使用与AADL元模型相同的形式主义，因此有助于将该表示集成在AADL模型的内部表示中。  
我们还支持按照相同的XML模式在XMI中进行序列化和解除序列化。它允许产生独特的AAXL2表示，包括序列化行为附件对象。因此，我们提供一个持续表现，包括整个描述AADL架构+行为附件。它可以用作不同后端的输入。图6描述了OSATE2如何使用AADL较前端的后端从AAXL2文件生成AADL文本描述。它提供了扩展点，以扩展这个后端与不同的附件解析器。我们使用这个扩展点来链接我们的AADL-BA，并且生成包含文本行为规范的AADL文本模型。  
VI。结论和未来的工作  
本文介绍了AADL行为附件作为ECLIPSE插件的实现[4]。我们展示了我们如何指定用作骨干的AADL-BA元模型来开发我们的编译器的几个模块。由于行为附件定义了几个简单的相互关联的子语言，单个元模型的设计允许产生和分析唯一的AST。使用AADL-BA模型的内部表示作为外部后端的输入，使用后端注册表进行集成。  
我们的插件集成在参考AADL工具集中  
OSATE2。这一个提供了AADL前端和附件插件，驱动行为附件分析（解析器+分析器）。用于定义AADL和AADL-BA元模型的相同技术可轻松跨两个元模型的导航，以及生成独特的持久XMI表示（AADL模型+行为元素），有助于将其用作外部后台的输入，结束。

我们未来的工作将集中在通过外部后端的集成来分析行为自动机属性，例如作为模型检查器来验证死锁和基于模型的工具，以通过改进WCET估计和阻止共享资源的时间来增强调度分析，这得益于 行为附件功能。

参考  
[1] SAE，AADL v2.0（AS5506），2008年9月。  
[2] J.Hugues，B.Zalila，L.Pautet和F.Kordon，“从原型到使用Ocarina AADL工具套件的最终嵌入式系统”，ACM Transactions in Embedded Computing Systems（TECS），vol。 7，2008年7月。[3] SAE，附件X行为附件（AS5506-X draft-2.13），2010年8月。  
[4]电信巴黎科技“AADL-BA项目”，http：//eve.enst.fr/aadlba，2010。  
[5] SAE AADL，“OSATE”，http://www.aadl.info，2010。  
[6] D. Koening，Groovy in Action。曼宁出版社，2007。  
[7] Y. Ma，J.-P. Talpin和T. Gautier，“Interpretation of AADL Behavior Annex in Synchronous Formalism Using SSA”，国际计算机和信息技术会议，第2361-2366页，2010年。  
[8] Z. Yang，K. Hu，D. Ma，and Pi Pi，“Towards a Formal Semantics for the AADL Behavior Annex”，in Design，Automation Test in Europe Conference Exhibition - DATE'09。，2009年4月，pp。1166 -1171。  
[9] R. Frana，J.-P. Bodeveix，M. Filali和J.-F. Rolland，“AADL Behavior Annex-Experiments and Roadmap”，Engineering Engineering Computer Systems，第12届IEEE国际会议，2007年7月，pp。  
377 -382。

窗体顶端

Cānkǎo  
[1] SAE,AADL v2.0(AS5506),2008 nián 9 yuè.  
[2] J.Hugues,B.Zalila,L.Pautet hé F.Kordon,“cóng yuánxíng dào shǐyòng Ocarina AADL gōngjù tàojiàn de zuìzhōng qiànrù shì xìtǒng”,ACM Transactions in Embedded Computing Systems(TECS),vol. 7,2008 Nián 7 yuè.[3] SAE, fùjiàn X xíngwéi fùjiàn (AS5506-X draft-2.13),2010 Nián 8 yuè.  
[4] Diànxìn bālí kējì “AADL-BA xiàngmù”,http://Eve.Enst.Fr/aadlba,2010.  
[5] SAE AADL,“OSATE”,http://Www.Aadl.Info,2010.  
[6] D. Koening,Groovy in Action. Màn níng chūbǎn shè,2007.  
[7] Y. Ma,J.-P. Talpin hé T. Gautier,“Interpretation of AADL Behavior Annex in Synchronous Formalism Using SSA”, guójì jìsuànjī hé xìnxī jìshù huìyì, dì 2361-2366 yè,2010 nián.  
[8] Z. Yang,K. Hu,D. Ma,and Pi Pi,“Towards a Formal Semantics for the AADL Behavior Annex”,in Design,Automation Test in Europe Conference Exhibition - DATE'09.,2009 Nián 4 yuè,pp.1166 -1171.  
[9] R. Frana,J.-P. Bodeveix,M. Filali hé J.-F. Rolland,“AADL Behavior Annex-Experiments and Roadmap”,Engineering Engineering Computer Systems, dì 12 jiè IEEE guójì huìyì,2007 nián 7 yuè,pp.  
377 -382.

点击图标下载 App

窗体底端