(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104932905 A
(43)申请公布日 2015.09.23

(21)申请号 201510413384.4

(22)申请日 2015.07.14

(71)申请人 北京神舟航天软件技术有限公司
地址 100094 北京市海淀区永丰路28号

(72)发明人 刘姝 王晓晗

(74)专利代理机构 北京世誉鑫诚专利代理事务所(普通合伙) 11368
代理人 孙国栋

(51)Int.Cl.
G06F 9/44(2006.01)

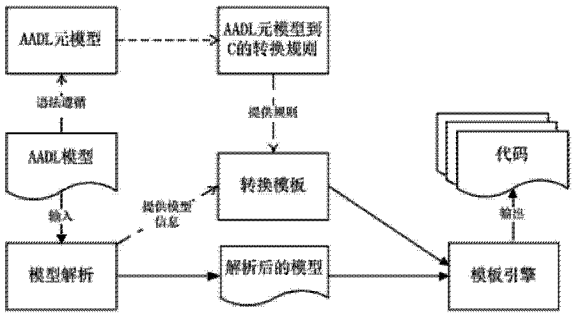
权利要求书1页 说明书5页 附图1页

(54)发明名称

一种AADL到C语言的代码自动生成方法

(57)摘要

本发明涉及一种AADL到C语言的代码自动生成方法,其包括以下步骤:1)根据AADL元模型的语义、目标运行环境的语义和C语言语法语义定义AADL元模型到C语言的转换规则;2)根据AADL模型能够获取的所使用的元模型及其语义、AADL元模型到C语言的转换规则制定代码自动生成模板;3)根据代码自动生成模板编写模板转换引擎;4)输入AADL模型后,自动对输入的AADL模型解析,提取AADL元模型信息,按照代码自动生成模板,通过模板转换引擎将AADL元模型信息生成C语言代码。本发明具有以下有益效果:1定义了AADL模型到C语言的转换规则,有效解决AADL与C语言语义映射;2通过模板引擎提高代码转换效率。



CN 104932905 A

1. 一种 AADL 到 C 语言的代码自动生成方法，其特征在于，包括以下步骤：

1) 根据 AADL 元模型的语义、目标运行环境的语义和 C 语言语法语义定义 AADL 元模型到 C 语言的转换规则；

2) 根据 AADL 模型能够获取的所使用的元模型及其语义、步骤 1) 得到的 AADL 元模型到 C 语言的转换规则制定代码自动生成模板；

3) 根据步骤 2) 得到的代码自动生成模板编写模板转换引擎；

4) 输入 AADL 模型后，自动对输入的 AADL 模型解析，提取 AADL 元模型信息，按照步骤 2) 中的代码自动生成模板，通过步骤 3) 得到的模板转换引擎将 AADL 元模型信息生成 C 语言代码。

2. 根据权利要求 1 所述的一种 AADL 到 C 语言的代码自动生成方法，其特征在于，步骤 1) 中 AADL 元模型到 C 语言的转换规则包括：

11)AADL 数据类型 Base_Type 生成 C 语言程序中的 *.h 头文件；

12) 系统构件不对应具体的可执行代码，仅将系统构件的功能接口转换为函数的输入参数和输出参数，且将所定义的数据构件转换为 C 语言程序中的变量定义，系统构件的名称作命名空间使用；

13) 线程构件、线程组构件以及子程序构件对应具体的可执行代码，线程构件、线程组构件以及子程序构件均转换为 C 语言程序中的函数定义；线程构件的功能接口转换为函数的输入参数和输出参数，线程组构件的功能接口转换为函数的输入参数和输出参数，子程序构件的功能接口转换为函数的输入参数和输出参数，子程序构件定义中的数据构件转换为局部变量定义；

14) 处理器构件中定义调度算法，调度算法转换为 C 语言程序中的函数定义；

15) 进程构件转换为 C 语言程序中的主函数，进程构件所包含的子构件之间的连接关系转换为源端函数和目标端函数，并将进程构件所包含的子构件之间的连接名称转换为变量定义，用于源端函数和目标端函数之间的数据传递；

16) 对于系统调用，需在 AADL 模型中进行定义。

一种 AADL 到 C 语言的代码自动生成方法

技术领域

[0001] 本发明涉及软件开发工具，特别涉及一种 AADL 到 C 语言的代码自动生成方法。

背景技术

[0002] 基于模型的软件开发是继面向对象技术后，又一次软件设计理念和方法的跨越。基于模型的软件开发思想是：将“模型”作为软件设计和开发的核心要素，通过模型清晰地刻画软件系统的功能、性能和安全性等关键特征，准确描述系统的解决方案；通过形式化方法对系统模型加以验证，以确保软件设计的正确性；通过模型驱动方法生成代码，以保证软件设计与实现的一致性。

[0003] 2004 年，美国汽车工程师协会 SAE(society of automotive engineers) 基于 15 年的研究和工业实践，提出嵌入式实时系统体系结构分析与设计语言 AADL(architecture analysis&design language)，提供了一种标准而又足够精确的方式设计与分析嵌入式实时系统的软、硬件体系结构及功能与非功能性质。但是基于 AADL 进行嵌入式系统代码自动生成非常具有挑战性，因为生成的代码要运行在不同的目标平台，满足各平台有不同的特征，如软硬件体系结构、编程接口等。模型较为抽象地表达了在特定平台上各要素的组织方式和特征，易于理解和验证，但与实现之间有语义鸿沟，给代码生成增加了负担，而且建模语言的抽象语义不一定能够与目标平台匹配。

[0004] AADL 模型到代码的自动生成，能够有助于提高嵌入式实时系统的软件开发的自动化水平，缩短软件开发周期，减少人工写代码的工作量和编码过程中出错的可能性。目前欧美模型驱动软件开发环境的代码自动生成主要支持 Ada 语言、Java 语言的代码生成，面向符合 ANRINC 653 规范的 POK 操作系统、汽车行业标准的 OSEK 操作系统，各领域在使用 AADL 进行代码生成时，需根据自身平台的运行时环境进行代码生成工具开发。我国航天领域广泛采用 SZOS 等嵌入式实时操作系统，但是尚未出现 AADL 到 SZOS 等特定运行时环境的 C 语言的代码自动生成方法。

发明内容

[0005] 发明目的：本发明针对上述现有技术存在的问题做出改进，即本发明公开了一种 AADL 到 C 语言的代码自动生成方法，使用该方法可以面向特定运行时环境实现 AADL 到 C 语言的代码自动生成。

[0006] 技术方案：一种 AADL 到 C 语言的代码自动生成方法，包括以下步骤：

[0007] 1) 根据 AADL 元模型的语义、目标运行环境的语义和 C 语言语法语义定义 AADL 元模型到 C 语言的转换规则；

[0008] 2) 根据 AADL 模型能够获取的所使用的元模型及其语义、步骤 1) 得到的 AADL 元模型到 C 语言的转换规则制定代码自动生成模板；

[0009] 3) 根据步骤 2) 得到的代码自动生成模板编写模板转换引擎；

[0010] 4) 输入 AADL 模型后，自动对输入的 AADL 模型解析，提取 AADL 元模型信息，按照步

骤 2) 中的代码自动生成模板，通过步骤 3) 得到的模板转换引擎将 AADL 元模型信息生成 C 语言代码。

[0011] 进一步地，步骤 1) 中 AADL 元模型到 C 语言的转换规则包括：

[0012] 11)AADL 数据类型 Base_Type 生成 C 语言程序中的 *.h 头文件；

[0013] 12) 系统构件不对应具体的可执行代码，仅将系统构件的功能接口转换为函数的输入参数和输出参数，且将所定义的数据构件转换为 C 语言程序中的变量定义，系统构件的名称作命名空间使用；

[0014] 13) 线程构件、线程组构件以及子程序构件对应具体的可执行代码，线程构件、线程组构件以及子程序构件均转换为 C 语言程序中的函数定义；线程构件的功能接口转换为函数的输入参数和输出参数，线程组构件的功能接口转换为函数的输入参数和输出参数，子程序构件的功能接口转换为函数的输入参数和输出参数，子程序构件定义中的数据构件转换为局部变量定义；

[0015] 14) 处理器构件中定义调度算法，调度算法转换为 C 语言程序中的函数定义；

[0016] 15) 进程构件转换为 C 语言程序中的主函数，进程构件所包含的子构件之间的连接关系转换为源端函数和目标端函数，并将进程构件所包含的子构件之间的连接名称转换为变量定义，用于源端函数和目标端函数之间的数据传递；

[0017] 16) 对于系统调用，需在 AADL 模型中进行定义。

[0018] 有益效果：本发明公开的一种 AADL 到 C 语言的代码自动生成方法具有以下有益效果：

[0019] 1、详细描述了代码自动生成的实施步骤，为特定运行时环境实现 AADL 到 C 语言代码自动生成提供解决方案；

[0020] 2、定义了 AADL 模型到 C 语言的转换规则，有效解决 AADL 与 C 语言语义映射；

[0021] 3、使用 Xtend 语言定义模板，并通过封装等方式提供灵活的模板操作；

[0022] 4、通过模板引擎提高代码转换效率。

附图说明

[0023] 图 1 为本发明公开的一种 AADL 到 C 语言的代码自动生成方法的流程图。

具体实施方式：

[0024] 下面对本发明的具体实施方式详细说明。

[0025] 如图 1 所示，一种 AADL 到 C 语言的代码自动生成方法，包括以下步骤：

[0026] 1) 根据 AADL 元模型的语义、目标运行环境的语义和 C 语言语法语义定义 AADL 元模型到 C 语言的转换规则；

[0027] 2) 根据 AADL 模型能够获取的所使用的元模型及其语义、步骤 1) 得到的 AADL 元模型到 C 语言的转换规则制定代码自动生成模板；

[0028] 3) 根据步骤 2) 得到的代码自动生成模板编写模板转换引擎；

[0029] 4) 输入 AADL 模型后，自动对输入的 AADL 模型解析，提取 AADL 元模型信息，按照步骤 2) 中的代码自动生成模板，通过步骤 3) 得到的模板转换引擎将 AADL 元模型信息生成 C 语言代码。

[0030] 下面对 AADL 元模型到 C 语言的转换规则、AADL 元模型到 C 语言的转换规则制定代码自动生成模板、模板转换引擎、模板解析分别进行说明：

[0031] 关于步骤 1) 中 AADL 元模型到 C 语言的转换规则的说明：

[0032] 生成运行于特定运行环境的 C 语言应用程序代码时，软件体系结构模型生成应用软件代码框架，运行时模型则主要考虑调度算法，并生成相应的调度程序。

[0033] AADL 语言通过自顶向下的方式对系统进行建模，一般可以分为四个层次：最高层为系统构件，对应一个系统或子系统；一个系统构件包括多个进程构件及连接（用于表达任务执行的虚拟地址空间）、处理器构件（用于表达调度）以及其它硬件构件（如外设构件、存储构件、总线构件等）；一个进程构件代表任务的虚拟地址空间，可以执行多个线程构件或线程组构件及连接（用于表达任务及任务之间的通信），一个线程可以访问子程序构件，多个线程可以共享访问一个数据构件；线程构件、子程序构件的内部行为可以用行为附件来表达，用户也可以直接给出源代码。

[0034] AADL 元模型到 C 语言的转换规则如下：

[0035] 步骤 1) 中 AADL 元模型到 C 语言的转换规则包括：

[0036] 11)AADL 数据类型 Base_Type 生成 C 语言程序中的 *.h 头文件；

[0037] 12) 系统构件不对应具体的可执行代码，仅将系统构件的功能接口转换为函数的输入参数和输出参数，且将所定义的数据构件转换为 C 语言程序中的变量定义，系统构件的名称作命名空间使用；

[0038] 13) 线程构件、线程组构件以及子程序构件对应具体的可执行代码，线程构件、线程组构件以及子程序构件均转换为 C 语言程序中的函数定义；线程构件的功能接口转换为函数的输入参数和输出参数，线程组构件的功能接口转换为函数的输入参数和输出参数，子程序构件的功能接口转换为函数的输入参数和输出参数，子程序构件定义中的数据构件转换为局部变量定义；

[0039] 14) 处理器构件中定义调度算法，调度算法转换为 C 语言程序中的函数定义；

[0040] 15) 进程构件转换为 C 语言程序中的主函数，进程构件所包含的子构件之间的连接关系转换为源端函数和目标端函数，并将进程构件所包含的子构件之间的连接名称转换为变量定义，用于源端函数和目标端函数之间的数据传递；

[0041] 16) 对于系统调用（如创建任务、信号量、时钟定时器等），因为 AADL 模型中没有显式表达，需在 AADL 模型中进行定义。

[0042] 关于步骤 2) 中 AADL 元模型到 C 语言的转换规则制定代码自动生成模板的说明如下：

[0043] 代码自动生成模板是代码转换规则的载体，模板是由静态文本和占位符组成的文件，静态文本是直接输出到代码文件中的文本，占位符是用模型中的信息替换后输出到代码文件中的文本。占位符中的模型信息通过模型解析获取。

[0044] 代码生成模板使用 Xtend 语言编写，Xtend 语言是一种基于 Java 静态模板语言，通过编译成 Java 语言然后运行在 Java 虚拟机上执行。

[0045] 模板形式如下：

[0046] switch 语句示例：

[0047]

```
def toText(Node n){
    switch n{
    Contents: n.text
    A:'''<a herf="<<n.href>>"><<n.applyContents>></a>'''
    default: '''
        <<<n.tagName>>>
            <<n.applyContents>>
        </<<n.tagName>>>
    }
}
```

[0048] IF 和 FOR 语句示例：

[0049]

```
def someHTML(List<Paragraph> paragraphs) '''
    <html>
        <body>
            <<FOR p: paragraphs>>
                <<IF p.headLine != null>>
                    <h1><<p.headline>></h1>
                <<ENDIF>>
                <p>
```

[0050]

```
                        <<p.text>>
                    </p>
                <<ENDFOR>>
            </body>
        </html>
    '''
```

[0051] 使用 Xtend 语言的模板表达式和一组封装的文件字符操作函数以及 Xtend 扩展函数编辑实现。Xtend 模板表达式允许可读的字符串连接，模板内容由三重单引号包围(''' …''')，在模板中使用法语引号"《》"来进行处理插入的表达式。

[0052] 模板表达式支持《IF…》…《END IF》的判断语句，《FOR…》…《ENDFOR》的循环语句以及可以向下转型的 switch 语句处理实例模型。同时，Xtend 支持扩展方法，可以在不改变变量方法的前提下改变变量。模板可以灵活的在方法中交替出现，既可以作为方法

体，也可以作为方法的语句。

[0053] 为了便于生成代码过程中模板操作更加灵活，可封装一些常用的文件操作和字符处理方法。

[0054] 关于步骤 3) 中的模板引擎的说明如下：

[0055] 模板引擎依据代码转换规则，将代码模板中的占位符用模型中的信息替换，输出到代码文件中。

[0056] 关于步骤 4) 中的模板解析的说明如下：

[0057] AADL 标准提供了 AADL 的文本、XML、图形表示规范，为便于操作，本方法对 XML 文件进行解析。

[0058] EMF(Eclipse Modeling Framework) 对 AADL 语言标准进行了精确描述（即 Ecore 模型），通过 Ecore 模型自动生成一套封装了 AADL 所有信息的 Java 数据对象，将 AADL 模型转换成高效的、正确的和易于定制的 Java 代码。

[0059] 使用 EMF 封装的 AADL 元模型数据结构来封装模型信息，将提取的模型信息保存到此数据结构中。数据结构以系统实现为根节点，组件为核心节点，特征属性等为叶节点呈树状结构。

[0060] 上面对本发明的实施方式做了详细说明。但是本发明并不限于上述实施方式，在所属技术领域普通技术人员所具备的知识范围内，还可以在不脱离本发明宗旨的前提下做出各种变化。

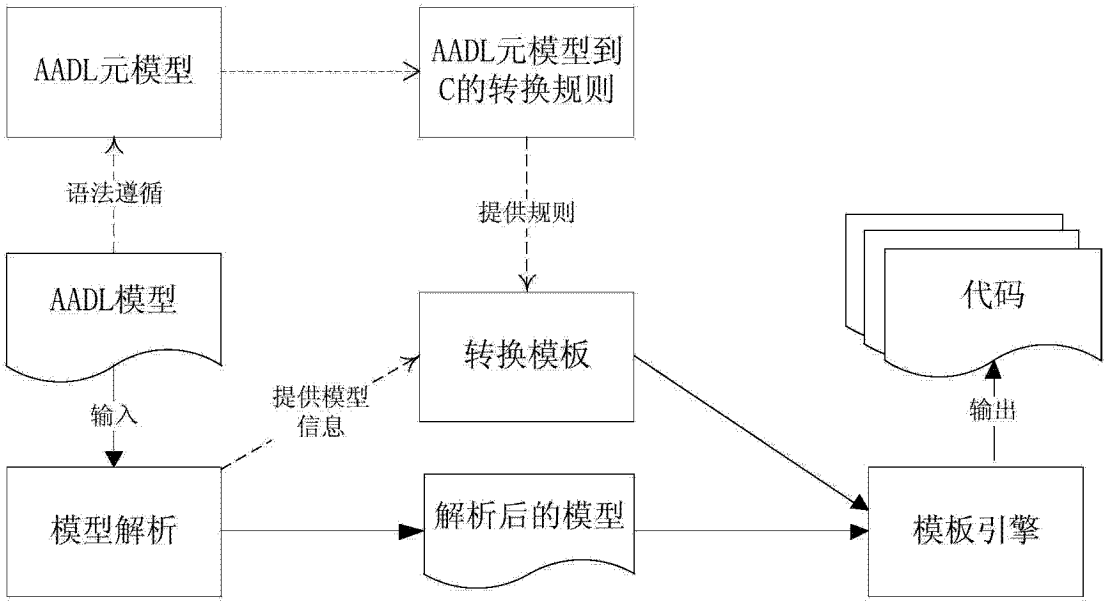

AADL元模型
AADL元模型到C的转换规则
语法遵循
提供规则
AADL模型
转换模板
代码
输入
提供模型信息
输出
模型解析
解析后的模型
模板引擎

图 1