计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.201100173

基于 AADL 的自主无人系统可成长框架

丁 嵘[1] 于千惠[2]
1 北京航空航天大学人工智能研究院 北京 100191
2 北京航空航天大学计算机学院软件开发环境国家重点实验室 北京 100191

摘 要 近年来,自主无人系统的开发成本随着硬件设备性能的提高而增加,如何高效、智能化地完成开发工作成为无人系统的热门研究领域。基于 AADL(Architecture Analysis and Design Language)的自主无人系统可成长框架从系统架构方式、基于配置项的系统工作模式以及原型系统等方面实现了无人系统(无人机、无人车等)软件适应性方案,以支撑资源、任务、环境变化时无人系统软件的成长和演化。搭建系统框架时采用基于模型驱动的思想,使用 AADL 模型基表示系统的中间组件,既保留了组件之间的继承关系,又便于开发者更直观地观察系统结构。系统模块化是实现其可成长性的基础,通过统一规范接口书写规则的方式,AADL 模型基将可替换算法封装在中间组件中,算法的迭代与进化映射出了系统的可持续演化过程。通过爬虫的方式建立一个不断扩展的系统组件库,组件库除了支持自适应扩展功能外,还支持自定义模型的基功能。系统框架的可成长特性除了表现在系统文件的内容可扩展外,还表现在系统配置方案的选择多样性。在不同的环境、任务、资源状况下,系统的最佳配置项方案可能不同,为了找出适应条件的无人系统配置项选项的最优解,采用进化算法的思想,使系统实现自主进化的过程。最后,利用代码自动生成技术,实现 AADL 模型到系统文件之间的转换。通过可成长软件管理平台的运行与测试,验证了自主无人系统可成长框架的可行性。

关键词:可成长系统;AADL 模型基;抽象语法树;代码生成
中图法分类号 TP311

Growth Framework of Autonomous Unmanned Systems Based on AADL

DING Rong[1] and YU Qian-hui[2]
1 Institute of Artificial Intelligence, Beihang University, Beijing 100191, China
2 State Key Laboratory of Software Development Environment, Computer Science Department, Beihang University, Beijing 100191, China

Abstract Recent years, the development cost of autonomous unmanned systems increases with the improvement of hardware equipment performance. How to efficiently and intelligently develop systems is a hot research field for unmanned systems. The growth framework of autonomous unmanned systems based on AADL has improved the software adaptability of unmanned systems (drones, unmanned vehicles, etc.) from the system architecture, the system working mode based on configuration items, and the prototype system. It realizes the growth and evolution of unmanned system software when resources, tasks, and environments change. The system framework is based on model-driven thinking, and the AADL(Architecture Analysis and Design Language) model base is used to represent the intermediate components of the system. It not only retains the inheritance relationship between components, but also facilitates developers to observe the system structure more intuitively. System modularization is the basis for the growth. Through a unified standardized interface, the AADL model base encapsulates replaceable algorithms in intermediate components, and the iteration and evolution of the algorithm maps the sustainable evolution process of the system. An ever-expanding library of system components is established by crawlers. In addition to supporting adaptive extension functions, the component library also supports custom model-based functions. The growth characteristic of the system framework is not only manifested in the expands of the content of the system files, but also manifested in the diversity of system configuration options. The optimal configuration item scheme of the system may change under different environments, tasks, and resource conditions. In order to find the optimal solution of the unmanned system configuration item options under adaptive conditions, the idea of evolutionary algorithm is adopted to make the system realize the process of autonomous evolution. Finally, the automatic code generation technology is used to realize the conversion from AADL model to system file. The feasibility of the growth framework of the autonomous unmanned system is verified through the operation and test of the growth software management platform.

Keywords Growable system, AADL model base, Abstract syntax tree, Code generation

到稿日期:2020-09-05 返修日期:2020-10-24 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家重点研究发展计划(2017YFB1001802)
This work was supported by the National Key Research and Development Program of China(2017YFB1001802).
通信作者:丁嵘(dingr@buaa.edu.cn)

1 引言

随着硬件设备的计算能力的提高，多样化的用户需求使软件系统的开发成本和软件复杂度随之提高。为了降低开发难度和缩短开发周期，本文从可成长软件的建模方式、基于配置项的可成长软件工作模式、原型系统等方面提出了无人系统（无人机、无人车等）软件适应性方案，以支撑资源、任务、环境变化时无人系统软件的成长和演化[1-8]。

自主无人系统可成长框架是一种系统开发工具，根据用户需求或者具体的任务资源条件，从库中提取相应功能的模块组件并生成完整的系统文件，其采用基于 AADL 模型驱动思想的开发方法，使开发者脱离基于代码层级的开发模式[2-3]，改善了以往由于系统结构信息冗杂而造成的开发难度较大的问题。

可持续演化的软件系统开发技术近年来发展迅猛，在智慧系统开发方面取得了一系列显著的研究成果。本文提出的可成长系统框架主要结合了两种系统自动开发技术：1）建立可不断扩展的系统文件库；2）在搭建系统的过程中，软件的配置项参数文件可实现自适应调整。

基于模型驱动思想设计的自主无人系统可成长框架秉承了“按需引用”的设计理念[4-5]，不同于传统软件工程自下而上的“按需制作”开发模式，基于 AADL 模型基的系统架构更注重软件系统独立的体系结构，因此，在实现系统的迭代与进化时，将系统的中间组件作为开发主体以便于程序在以模型基为架构底层的基础上实现自动化开发。

本文的主要贡献如下：

（1）提出了使用系统体系分析与设计语言 AADL 描述可成长系统架构，并将传统的代码层次开发模式转变为模型基层次开发模式，使开发者更易于完成系统开发工作，并且能够直观地展示用户需求与系统功能模块之间的映射关系。

（2）为实现系统架构的可成长性，采用了两种系统自适应进化技术相结合的方式：1）通过搜索开源系统文件并收录入库，来增加系统模块的复杂度；2）采用进化算法[6-8]，模拟配置项选项在不同的应用场景下的适应度，根据适应值和条件函数选出满足开发需求的最优参数配置方案。

（3）利用 OSATE[9] 工具（一个开源的 Eclipse 插件）以及 XML 解析器和代码生成器，可以实现 AADL 组件文件与系统程序文件之间的相互转化，结合参数文件可实现自主系统的开发工作，使开发者真正脱离程序语言层级的开发环境。

本文第 2 节对系统架构描述语言 AADL 及基于其模型基的系统框架进行了简单介绍；第 3 节着重讲述了自主无人系统可持续演化的工作模式；第 4 节介绍了通过可成长软件管理系统进行无人自主开发的测试工作以及应用结果；最后总结全文。

2 基于 AADL 模型基的系统框架

1991 年，Honeywell 实验室基于嵌入式系统提出了 MetaH 语言。2001 年，美国汽车电子协会 SAE 提出了航空架构描述语言 AADL。以上两种语言不足以解决现实中航空系统存在的一些问题。2004 年，SAE[10]、卡内基梅隆大学 CMU 和 Honeywell 实验室在 MetaH 和 AADL 的基础上提出了架构分析和设计语言 AADL，并基于 Eclipse 在 2004 年推出 Osate1.0 版本（Open Source AADL Tool Environment），之后在 2009 年推出 Osate2.0 版本。

AADL 是一种常用的基于模型驱动思想[11-12]的描述嵌入式实时系统软硬件的体系分析与设计语言，旨在用于与实时性能相关联（如时序、安全性、可调度性、容错性、安全性等）的规范、分析、自动集成和代码生成。本文通过 AADL 描述将软件系统建成模型，使用一套标准的 AADL 组件语言对客观世界中的软件系统的组件进行抽象，使模型在具有统一的接口规范和结构化表示的同时，还能够精确地表达程序的可分析语义、系统框架的定义和运行时的数据流。近年来，AADL 系统架构建模语言被广泛应用于以 ROS 机器人系统为代表的自主无人系统的设计和开发。

AADL 模型以中间组件为核心，由 AADL 建模语言描述的接口信息和组件实现声明共同定义了组件分类器，这些组件可以表示软件和硬件结构以及它们之间的交互。因此，在本文提到的自主无人系统文件库中，单个中间组件或多个组件的组合形式都可以作为其他模型中可重用的模型基。基于 AADL 模型基的系统架构方式，本文提供了在开发之前对系统设计进行分析的工具，并在整个系统生命周期中支持基于模型基的开发方法。

AADL 是一种由上下文无关的语法描述的声明性语言，其描述规范主要包括以下几类：软件元素（如进程、线程、数据等）、硬件元素（如处理器、内存、总线等）、系统和摘要。除此之外，用户可以通过引入自定义的组件类别和功能描述生成特定的结构类型。与面向对象的编程语言类似，AADL 提供组件继承性，支持该特定结构的重用和演化发展，这也为后续工作中以模型基为开发主体的可成长开发模式奠定了基础。

AADL 允许建立整个系统的实例模型，或者任一子系统的实例模型。根据主要子系统，或者根据带线程或不带线程的进程，建立部分系统模型。将系统的实例模型组合在一起对体系结构进行分析，软件的主要功能实现与 AADL 描述性组件模块相映射，由此提取出可重用的模型基用于系统开发。

目前，许多面向编程语言的建模工具[13-14]提供系统架构模型的集成和解析功能，根据映射规则使 AADL 模型描述语言转变为可执行的代码文件。本文使用 OSATE 提供了用于 AADL 开发的 IDE，除此之外，OSATE 还提供了中间模块可视化功能，通过拖拽图形等形式搭建的系统结构可以直接生成 AADL 描述文件，在 OSATE 的帮助下，搭建好的图形界面模型可以直接导出基于 AADL 语言表述的系统架构的代码。再结合转换规则，基于 OSATE 工具和 XML 解析器开发代码自动生成器，完成自动生成模块代码到自动转化成目标程序语言的整个过程。

3 可成长框架的工作模式

本文采用基于 AADL 模型基的方法搭建了一个复杂软件系统体系结构，该系统框架侧重于对系统结构的思考。由

于软件系统具有复杂性和需求的不确定性，单一的系统框架显然不适用于更多的应用场景，因此实现自主无人系统的可演化发展是突破该领域难点的核心问题。在使用 AADL 模型基架构的基础上，本文利用中间组件的可扩展性以及配置项选项的自适应性实现可成长软件系统的自动开发。

软件系统的迭代与进化是可成长框架的重要特征。本文提出的可成长系统框架主要在以下两方面体现了软件的可成长特性：1）在智能化系统开发与设计环节，通过不断积累的源文件库达到系统进化的目的；2）在系统运行过程中，根据资源、任务和环境的影响自适应调整系统配置，使系统具有良好的应用体验。

本文自动收集开源的系统文件，并将其修改为规范的 AADL 模型组件。定义携带软件功能信息的程序片段为模型基，模型基大多由多个底层通信组件组成，模型基库的存储需要考虑占用存储空间等问题，对于实现同种功能的系统组件而言，其差异性可能表现在少部分代码或采用的算法不同，如具有导航功能的系统组件可以采用多种路径规划算法。为了方便管理和节省空间，模型基在验证正确和无重复后，通常只将一种同类型的模型基入库。因此，爬取多种可替换的算法是扩充模型基的主要方法，本文采用代码自动改接口技术使算法传参的数量和数据类型与模型基内部接口的标准规范相统一，使一个软件系统可重用的模型基对应多种算法，从而在扩展模型基库的同时，尽可能减小模型基库的规模和复杂度。本文对中间组件的内部结构进行了深入研究，并采用替换内部算法、替换函数参数等方式实现组件模板化，库文件仅需收录可替换的算法文件和可重用的组件项目就可以扩大 AADL 组件库。

在选用不同功能模块组合集成一个软件系统时，部分功能模块的选择依据通过配置项参数来表示，当任务模式、任务环境等发生变化时，系统组成必然有所不同，例如：导航任务从稀疏地图变为稠密地图时，应选择与地图匹配的路径规划算法来提高导航的准确性。因此，除了实现软件系统的中间组件可成长性外，如何使系统架构的配置项参数选择方案具有可成长性也是本文的研究重点。

通过上述操作，AADL 组件和配置项文件共同搭建成一个基于 AADL 架构的可成长系统框架，通过代码解析器最终生成完整的项目文件。可成长系统的工作模式如图 1 所示。

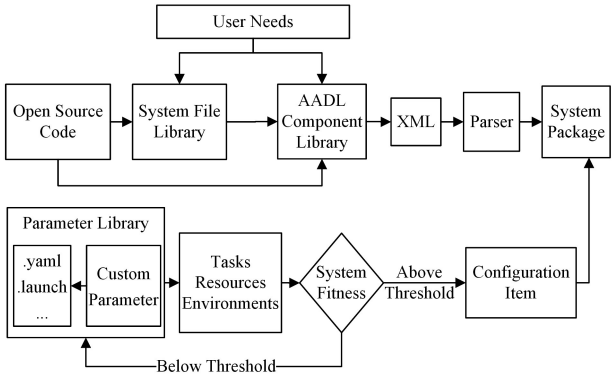

图 1　可成长系统的工作模式

Fig. 1　Working mode of growable system

3.1　利用爬虫扩展 AADL 组件

系统中间组件库主要包括 3 部分内容：原始 AADL 组件、新增 AADL 组件以及可替换的算法文件。原始 AADL 组件是在可成长框架搭建初期，通过实验分析，人工提取出可作为 AADL 模型基的系统功能模块。新增 AADL 组件既可以来源于外部搜索，又能够由开发者自定义功能组件。可替换的算法文件用于替换组件内部的代码片段，一个 AADL 组件对应多个算法文件，通过爬虫不断地搜索框架中可替换的算法，以达到可成长的目的。本文结合 AADL 已有的基础文件，与新增的功能模块共同构建系统模型。

爬虫有两种方式，第一种通过调用网页的接口，根据关键词爬取相关的代码文件。爬取后，将单个文件转换成抽象语法树，先与库中已有的文件进行对比，将相似度过高的文件判定为重复并删去；再将文件与不同类别的代码进行对比，根据不同类别的相似度对其分类。第二种通过解析网页的 HTML，根据关键词爬取整个项目文件。下载并解压后，将项目文件放入系统环境中编译，并测试是否可行，对项目中的每个文件进行操作，将其转换成抽象语法树之后，与库中项目的每个文件进行逐一对比，根据所有文件的加和相似度进行分类。

考虑到开源代码中的项目和算法文件更新不频繁等问题，采用周期性自动爬取的方式，每间隔一段时间，服务器自动爬取开源文件。为了进一步优化爬虫的效率和效果，记录未访问过的 URL 并将项目发布出去，在下一次爬取时跳过访问过的项目文件和单个算法文件，根据发布的项目进行去重复处理，判断其类别后上传到服务器实现入库操作。

3.2　利用抽象语法树处理代码

在计算机科学中，抽象语法树（Abstract Syntax Tree，AST）[15] 是源代码语法结构的一种抽象表示。它以树状的形式表现编程语言的语法结构，只需要关注词法分析和语法分析就可以从代码中生成 AST。目前，借助抽象语法树的分析方法被广泛应用于代码自动生成、代码转换以及代码纠错等领域。

抽象语法树不依赖于源语言的语法，在语法分析阶段，采用上下文无关的文法，将词法分析识别出的节点信息进行树状结构的等价转换，无论源语言采用怎样的文法规则，抽象语法树都会在语法分析时构造出相同的语法树，这样可以给编译器后端提供清晰、统一的接口，从而减少工作量，也使编译器易于维护。

为了使抽象语法树构造出具有统一规范的语法结构，在对不同语言进行分析时，将抽象语法树构成的中间代码作为输出供后端处理，这导致抽象语法树并不会完全贴合源代码中的书写规则，抽象语法树会删除仅在解析时才有意义但在逻辑上无用的元素，例如源代码嵌套括号或某些关键字都被隐含在树的结构中，并不会以节点的方式呈现。因此，很多编译器采用独立的抽象语法树构造方法，为前端、后端建立一个清晰的接口。

本文使用 ANTLR（Another Tool for Language Recognition）[16] 工具作为抽象语法树的解析器。ANTLR 是一种开源语言分析工具，它提供了一个框架，可以通过包含 Java，

C++，或 C#的语法描述来构造语言识别器、编译器和解析器。ANTLR 解析器主要包括词法分析器、语法分析器和树分析器。ANTLR 插件可以根据输入自动生成抽象语法树，遍历抽象语法树并进行一些操作，如增删叶节点、提取节点信息等。除此之外，它允许自定义识别字符流的词法规则和用于解释标识符流的语法分析规则，这为程序语言和 AADL 描述语言之间的代码转换提供了技术支持，不仅如此，ANTLR 还可以提供抽象语法树可视化的功能，直观地展示源代码的语法结构。

3.2.1 代码分类与去重复

通过爬取获得的整个工程或项目文件，需要进行预处理才能将其封装为 AADL 组件模块。事实上，大部分爬虫获得的项目文件都会被筛选舍弃，项目文件除了无法在系统环境中编译运行外，还存在文件重复等问题。因此，为了节约存储空间，预处理主要是对相似代码进行去重复的操作。

对于单个算法文件，使用 ANTLR 工具将其转化为抽象语法树，将其与库中已有文件的抽象语法树进行对比，采用二进制编码的方式对抽象语法结构树进行编码。这样做的好处在于基于代码结构对比相似度，根据相似度数值分析结果，减小了由变量名称和函数名称等自定义产生的字符串带来的影响。

对于整个工程文件，将工程包里的每个可执行代码都转化成抽象语法树，与已收录文件的抽象语法树进行逐一对比的方式，计算抽象语法树的相似度加和。一般情况下，整个工程文件的重复率极低，因此计算相似度多用于对代码进行分类。

为了管理库文件，除了删除相似度过高的文件外，还需要进行代码分类操作，将爬虫获取的代码进行分类，用于架构不同的系统功能模块。由于系统框架中可替换的算法有限，且存在原始 AADL 组件成分，因此我们主要采用了监督分类的方式对代码进行区分，为代码相似度设定阈值，超过阈值即将其判定为不具有先验知识的类属，并舍弃此类文件。例如在机器人导航系统中，根据代码相似度将算法主要区分成路径规划算法和 SLAM 算法。

3.2.2 代码自动改接口技术

为了将开源的代码和工程包转化为服务于可成长软件系统框架的 AADL 组件，我们对代码接口，也就是函数名、参数和返回值进行了分析，采用将接口参数修改为统一格式的方法，将其改写成符合 AADL 建模语言书写规则的命名方式。

为了完成自动提取接口信息的工作，需要程序自动定位接口参数。在最初的工作中，我们首先尝试了词法分析，通过直接修改的方式统一程序与系统框架之间的参数接口命名。此过程的难点在于如何忽略自定义的变量名对程序分析的影响。因此，我们再次利用抽象语法树对程序结构进行分析，抽象语法树的优势在于不依赖源程序语言的语法规则，通过找到并提取相应的函数名称、输入输出、返回值和源码路径等信息，可以直接整合成 AADL 组件。代码文件基于抽象语法树的处理过程如图 2 所示。

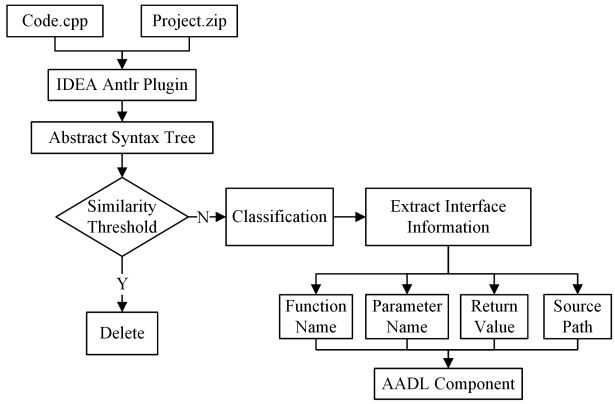

图 2 从爬虫到 AADL 组件的流程

Fig. 2 Flow from crawler to AADL component

上文提到了一个 AADL 模型基可以适用于多个相似算法文件，为了进一步优化系统框架，考虑在搭建 AADL 组件时，使可替换的内部算法文件共用一套接口模板。因此，在处理可替换的单个算法文件时，只需找出并修改接口参数的变量名称，从而避免重复提取接口信息的操作，以进一步节省开发时间。

3.3 基于配置项的可成长工作模式

本文提出的基于 AADL 模型基的可成长实现框架，同样是基于配置项参数的。本文针对无人系统软件任务、资源、环境自适应的实际应用需求，选择不同功能模块组合形成一个集成无人机系统，部分功能模块的选择依据通过配置项参数来表示。当任务模式、任务环境等发生变化时，系统组成必然有所不同，在这一过程中，我们使用不同的 yaml 参数文件来表示不同的算法选项。

一个完整的系统框架由参数文件、launch 文件启动节点以及库文件中的功能模块 3 部分组成，最终生成一个基于 AADL 架构的工程文件供用户使用。前文主要讨论了通过扩展功能模块库的方式实现系统框架的可成长性，接下来将讨论基于配置项选项的可成长工作模式。

针对无人机系统项目中的可变参数，用户通过提取参数服务器中的参数并生成 yaml 文件(无需重新编译)，或者提取源码中的所有参数并处理后写回源码(需要重新编译)的方式，获取配置项参数。提取参数后对参数进行分类，类型包括整型、布尔类型、字符串类型等。除此之外，用户通过输入配置项选项筛选库中已有的参数文件，甚至可以将自定义配置文件与参数文件相结合，用于生成项目所需要的参数文件。

受到进化算法(或称“演化算法”)的启发，我们将配置项参数的选择问题转化为基因编程问题，通过优胜劣汰的方式，筛选出适应度更高的配置项方案。首先将配置项参数作为遗传基因表达式，即遗传算子，将这些遗传算子组合起来形成一个参数文件，即个体。其次根据配置项文件的格式，对不同类型的参数进行随机初始化，生成多份 yaml 文件或多份源码，构成初始种群。然后采用二进制的编码方式对各个参数数值进行编码，该二进制编码串即可视为生物模型中的 DNA 碱基对，通过交叉和变异二进制字符串生成新的个体。二进制字符串相比十进制字符串具有更好的分割特性，使参数数值

表达式具有更高的计算精度。通过仿真测试，得到每个个体（参数集合）的适应度，将适应度排序，选取更优的个体集合作为下一代遗传进化的父本种群，重复上述步骤，直到适应度函数收敛，意味着在该仿真环境下，系统实现了自动生成适用于给定测试样例的参数配置近似最优解（由于资源分配的限制，可能存在局部最优的情况）的过程[17]。在真机实验中，使用多个机器人设备进行测试，选择适应性较好的机器人参数进行遗传和进化操作，让机器人在可变的环境下能够改变自身参数，更好地适应环境与任务的要求。基于遗传算法的配置项选择流程如图 3 所示。

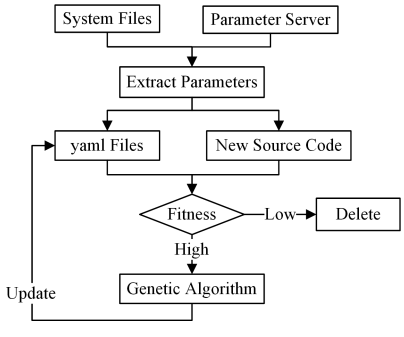

图 3　基于遗传算法的配置项选择过程

Fig. 3　Configuration item selection process based on genetic algorithm

3.4　根据 AADL 架构生成系统文件

AADL 中间组件、配置项参数以及 launch 文件启动节点共同组成了一个基于 AADL 架构的系统项目。为了完成系统开发的完整流程，开发者最终将获得一个由高级程序语言编写的项目工程包。因此，本节主要阐述从基于 AADL 的自主无人系统框架到代码生成的流程。

AADL 是一个正在发展的语言，其插件和辅助工具语言的转换主要借助其他语言的优势，来弥补 AADL 语言表达能力上的不足。通常情况下，基于 OSATE 开发的 AADL 模型采用两种方式进行转换：一类是在 OSATE 工具的基础上扩展 AADL 开发工具的功能；另一类是通过 XML 或其他文本描述语言，与 OSATE 做无缝的连接，例如 Cheddar[18] 可以实现系统的实时调度分析，ADeS[19] 可以对 AADL 模型进行验证，pix2code[20] 进行代码的生成等。

XML 简单且易于在任何应用程序中进行读写操作，这使其成为数据交换的唯一公共语言，高级语言程序通过转换成 XML 语言的方式可以更容易地与 Windows，Mac OS，Linux，以及其他平台下产生的信息结合。很多软件、框架都会采取 XML 文件作为配置文件。尽管 XML 可扩展标记语言是主流的通用语言，可以与多种高级程序语言交互，但是由于 XML 文件庞大且格式复杂，服务器端和客户端都需要花费大量代码来解析，需要占用较多的人工资源和时间。

本文根据 OSATE 集成的 XML 转换工具，将设计好的 AADL 文件转换为 XML 文件。由于 AADL 生成的 XML 格式特殊，需要特定的解析器进行解析，因此将 XML 文件导入自主编写的 XML 解析器中，该解析器可以根据提供的 XML 文件自动生成对应的无人机系统节点，最终生成可在自主无人系统可成长框架下编译运行的工程文件。

4　系统测试与后台管理

本文基于 B/S 架构搭建了可成长软件管理原型系统，挑选 30 名测试者以 ROS 系统为例进行了软件的持续迭代开发尝试。

可成长软件管理原型系统主要整合了现有的 AADL 组件库、AADL 基础框架以及代码转换器。用户借助 OSATE 工具或图形化界面设计并生成基于 AADL 的系统框架，将设计好的 AADL 文件通过 OSATE 工具转换为 XML 文件，借助可成长软件管理平台将提供的 XML 文件自动生成对应的 ROS 节点项目。

可成长软件管理平台拥有独立的测试和评分系统，对于上传到网站的所有项目，都支持查看、测试和下载功能，可以帮助开发人员更好地查阅和利用评分更高的开源工程项目。除此之外，后台管理功能为网站的管理者提供各项数据可视化的增删改查，以及测试任务的监控。

建立一个系统管理平台的意义是实现系统组件库和基础框架的信息共享，更重要的是有利于系统开发的可持续化成长。我们提供了空白的系统框架，供用户集成自定义的源码文件和配置项选项，平台通过收录这些用户自定义的系统框架，实现系统文件库的进一步扩展。

结束语　本文提出基于 AADL 架构与配置项选项的自主无人系统可成长框架，旨在提高软件系统的开发效率、缩短开发周期等。为了使无人系统的开发工作朝着智能化方向发展，本文采用 AADL 模型语言描述系统架构，搭建了一个基于组件库的可成长系统框架，通过不断地爬取算法和工程文件、更新组件库以及采用遗传算法的思想配置参数文件，使无人自主系统框架具有可成长性。

可成长软件管理原型系统搭建完成后，我们进行了 ROS 系统的持续迭代开发尝试，结果表明，本文提出的可成长框架使系统开发工作更加简单。未来，将探索更多的系统自适应成长的方法，并将其应用到实际的开发环境中；同时，将对其他版本、其他类型的无人自主系统开发进行更多的尝试，使可成长系统框架适用于更广泛的应用领域。

参 考 文 献

[1] ORTIZ J F，ALONSO D，ROSIQUE F，et al. A component-based meta-model and framework in the model driven toolchain C-Forge[C]// International Conference on Simulation，Modeling，and Programming for Autonomous Robots. Springer，2014：340-351.

[2] TAO Y. AADL-Based Model Verification and Code Generation Technology[D]. Chengdu：University of Electronic Science and Technology of China，2013.

[3] BARDARO G，MATTEUCCI M. Using AADL to Model and Develop ROS-Based Robotic Application[C]// IEEE International Conference on Robotic Computing. IEEE，2017.

[4] BARDARO G，SEMPREBON A，MATTEUCCI M. A use case in model-based robot development using AADL and ROS[C]//

the 1st International Workshop. 2018:9-16.
[5] YONG T, SHENG-LIN G, LIANG M A, et al. Code Automatic Generation and Integration Technology of AADL Model[J]. Computer Engineering, 2009, 35(8):59-61.
[6] BECKER K, GOTTSCHLICH J. AI Programmer: Autonomously Creating Software Programs Using Genetic Algorithms[J]. arXiv:1709.05703, 2017.
[7] GOUES C L, NGUYEN T V, FORREST S, et al. GenProg: A Generic Method for Automatic Software Repair [J]. IEEE Transactions on Software Engineering, 2012, 38(1):54-72.
[8] JIANG J, XIONG Y, ZHANG H, et al. Shaping Program Repair Space with Existing Patches and Similar Code[C]// The 27th International Symposium on Software Testing and Analysis. 2018:298-309.
[9] "Osate"[OL]. http://osate.org/.
[10] FEILER P H, LEWIS B A, VESTAL S. The sae architecture analysis & design language (aadl) a standard for engineering performance critical systems[C]// 2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control. IEEE, 2006:1206-1211.
[11] SCHLEGEL C, STECK A, LOTZ A. Model-driven software development in robotics[M]// Communication Patterns as Key for a Robotics Component Model. Introduction to Modern Robotics, 2011:119-150.
[12] HUGUES J, ZALILA B, PAUTET L, et al. From the prototype to the final embedded system using the Ocarina AADL tool suite [C]// ACM Transactions on Embedded Computing Systems (TECS). 2008.
[13] DHOUIB S, KCHIR S, STINCKWICH S, et al. Robotml, a domain-specifific language to design, simulate and deploy robotic applications[C]// International Conference on Simulation, Modeling, and Programming for Autonomous Robots. Springer, 2012:149-160.
[14] TAHA S, RADERMACHER A, GERARD S, et al. Marte: Uml-based hardware design from modelling to simulation[C]// FDL, 2007. 2007:274-279.
[15] HORWITZ S. Identifying the semantic and textual differences between two versions of a program[C]// Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI). 1990:234-245.
[16] ANTLR[OL]. http://www/antlr.org/.
[17] MORI M, TSENG C. A genetic algorithm for multi-mode resource constrained project scheduling problem[J]. Europpean Journal of Operational Reswarch, 2003, 144(2):38-365.
[18] SINGHOFF F, LEGRAND J, NANA L, et al. Cheddar: A flexible real time scheduling framework[J]. ACM Ada Letters, 2004, 24(4):1-8.
[19] Axlog. ADeS, a simulator for AADL[OL]. http://www.axlog.fr/aadl/ades_en.html
[20] BELTRAMELLI T. pix2code: Generating code from a graphical user interface screenshot[C]// Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems. ACM, 2018:3.

DING Rong, born in 1975, Ph.D, associate professor, is a member of CCF. His main research interests include software engineering, autonomous unmanned systems and natural language understanding.