计算机科学2000Vol. 27№. 1

软件体系结构描述语言 ADL 及其研究进展*)

Advancement of Architecture Description Language(ADL)

孙志勇　刘宗田　袁兆山
(合肥工业大学计算机学院　合肥230022)

Abstract Software architectures shift the focus of developers from lines-of-code to coarser-grained architectural elements and their overall interconnection structure. Architecture description languages (ADLs) have been proposed as modeling notations to support architecture-based development. Although little consensus in the research community on what is an ADL, this paper attempts to provide a framework of ADL. It presents a definition and a classification framework for ADLs. It also presents the aspects of further work on ADL research.

Keywords Architectures, Architecture description languages (ADL)

1. 引言

Perry 与 Wolf[1]，Garlan 与 Shaw[2]的早期研究成果发表后，软件体系结构的研究引起了学术界极大兴趣。目前，软件体系结构已经成为软件工程研究中的热点，我国学者也开始了跟踪研究[17]，在文[3]中，Shaw 与 Garlan 指出，软件体系结构的研究活动大致分四类：

1. 提供新的"体系结构描述语言"(ADL)来论述体系结构特征。也就是说，ADL 的目的是为实际工作者提供更好的刻画体系结构的方法，使他们可以相互交流。

2. 体系结构专门知识的编撰。这方面工作涉及到对不同体系结构原理和类别进行分类与推理。这些原理与模式是工程师们在软件实践中发展起来的。

3. 论述专门领域框架。这可以获得特殊类型软件的体系结构框架，如航空控制系统、运动机器人、用户界面等。成功后，对这些框架的简单实例化就可以形成该领域的新产品。

4. 研究软件体系结构的形式化基础。开发出新的符号语言使体系结构设计实践更易于理解。

软件体系结构研究的直接目的是降低应用系统开发成本，增加关联紧密的产品族中不同成员之间共性的潜力[1,4]。基于公共体系结构方法的软件开发将其焦点从代码行转移到了大粒度的体系结构元素(软件组件与联结)及其整体互联结构。为支持基于体系结构的开发，需要有形式化建模符号、体系结构说明的分析与开发工具。体系结构描述语言(ADL)及其伴生的工具集就是为了满足这样的要求。

ADL 作为形式化地表示软件系统体系结构的工具呈现出强大的生命力。ADL 近来成为软件体系结构群体中的研究热点[6~8]。实际工作者将这些 ADL 及其提供的支持工具集应用于实践中，获得了成功[12-17]。在十几种 ADL 出现后，为了进一步把握 ADL 的内涵，为新的 ADL 研究提供方向，国外一些学者对现有的 ADL 进行了分类与比较研究[9-10]。同时，我们注意到，不同的 ADL 所支持的抽象能力及其提供的分析能力变化很大，学术界对 ADL 的定义尚未取得一致。本文将从 ADL 的基本共性出发，总结 ADL 的现状，分析进一步工作的方向。

2. ADL 的基本概念

总的来说，ADL 是这样一种语言，系统级的软件设计者可以利用它所提供的特性进行软件系统概念体系结构建模。ADL 提供了具体的语法与刻画体系结构的概念框架[11]。概念框架反映了领域的特性和/或体系结构风格。

2.1　对 ADL 的不同理解

*)本文属国家教育部博士点基金课题"面向 Web 的组件化 CASE 模型研究"。孙志勇　博士生，研究方向：软件体系结构、组件技术、需求工程等。刘宗田　研究员，博士生导师，研究方向：软件工程、人工智能、反编译技术等。袁兆山　教授，研究方向：软件体系结构、软件工程等。

正如引言中所述，当前在学术界对 ADL 的定义尚未取得一致，下面的理解与定义大多来源于自身对某种 ADL 语言的开发所得到的体会。尽管它们有以偏概全之嫌，但还是为我们了解 ADL 提供了有价值的信息。

1. 在文[14]中，Tracz 定义一个 ADL 包含4"C"：组件(components)，连接子(connectors)，配置(configurations)，约束(constraints)。

2. 根据其 UniCon 的经历，Shaw 与同事列出了 ADL 应该展示的属性如下[15]：1)组件建模的能力，带属性断言、接口、实现；2)连接子建模的能力，带协议、属性断言与实现；3)抽象与封装；4)类型与类型检查；5)适应分析工具的能力。

3. 在文[5]中，Luckham 与 Vera 在研究 Rapide 的基础上提出，ADL 应该满足以下要求：1)组件抽象；2)通信抽象；3)通信完整性，要求在结果实现中，只有在一个体系结构中相联结的组件可以通信；4)动态体系结构建模的能力；5)分层复合；6)相关性，或者是行为与体系结构间关联(映射)的能力；7)组件建模的能力，带属性断言、接口、实现；8)连接子建模的能力，带协议、属性断言与实现；9)抽象与封装；10)类型与类型检查；11)适应分析工具的能力。

4. 在文[10]中，将 ADL 划分为4个组成部分：组件、连接子、配置、支持工具。其中，组件、连接子、配置构成了 ADL 的体系结构描述特性，而支持工具则为软件体系结构设计者提供必要的帮助。

2.2 体系结构与 ADL 的定义

体系结构并没有标准定义，但学术界基本接受 Garlan 和 Shaw[1]所提出的定义：

[软件体系结构是一个设计层]，它在计算的算法与数据结构之上；设计与描述总体系统结构合并为一个新问题。结构问题包括总体组织与全局控制结构，通信协议、同步、数据存取、设计元素的功能分配、物理分布、设计元素复合、伸缩与性能，设计决定之间的选择。

从上小节中我们看到，学术界对 ADL 的理解还存在分歧。我们认为：ADL 是这样一种形式化语言，它在底层语义模型的支持下，为软件系统的概念体系结构建模提供了具体语法与概念框架。基于底层语义的工具为体系结构的表示、分析、进化、细化、设计过程等提供支持。其三个基本构成元素是：1)组件——计算或数据存储单元；2)连接子——用于组件间交互建模的体系结构构造块及其支配这些交互的规则；3)体系结构配置——描述体系结构的组件与连接子的连接图。

2.3 ADL 与其它语言的区别

并非所有用于软件设计的符号语言都可以认为是 ADL。根据上述对 ADL 所达成的基本共识，可以将下面这样的语言排除在 ADL 之外：高层设计符号语言、MIL、编程语言、面向对象(OO)的建模符号、形式化说明语言。

ADL 关注概念体系结构并显式地将连接子作为第一类实体，从而将模块接口语言(MIL)、编程语言、OO 建模符号等排除在 ADL 之外。MIL 描述的是使用已实现系统中模块间的关系并只支持连接的一种类型。编程语言描述系统的实现，其体系结构隐含在子程序定义与调用中。

ADL 通常要包含一形式化语义理论，用以刻画体系结构的底层框架，它影响了 ADL 对某些专门领域系统的建模适应性。形式化描述理论的例子有 Petri 网、状态图、偏序事件集合、通信序列过程(CSP)、基于模型的形式化(Z 语言)、代数形式化(如 Obj)以及公理形式化(如 Anna)。Z 只能对体系结构某些方面建模，如体系结构风格规则。偏序事件集合、CSP、Obj 以及 Anna 已被现有的 ADL 采用(分别为 Rapide、Wright、及 LILEANNA)。

ADL 与需求语言的区别在于后者描述的是问题空间，而前者则扎根于解空间中。ADL 与建模语言的区别在于后者对整体行为的关注要大于对部分的关注，而 ADL 集中在组件的表示上。

3. ADL 的构成要素

一般认为，体系结构描述的基本构成要素有：组件；连接子；体系结构配置。ADL 必须提供对这些构造块的显式说明手段。

3.1 组件

组件是一个计算单元或数据存储。也就是说，组件是计算与状态存在的场所[15]。在体系结构中，一个组件可能小到只有一个过程(如 MetaH 过程)或大到整个应用程序(如 C2、Rapide 中的分层组件，或 MetaH 中的宏)，它可以要求自己的数据与/或执行空间，也可以与其它组件共享这些空间。作为软件体系结构构造块的组件，其自身也包含了多种属性，如接口、类型、语义、约束、进化、非功能属性等。现有的 ADL 对这些属性的侧重有所不同。

接口是组件与外部世界的一组交互点。与 OO 类或 Ada 包说明相同，ADL 中的组件接口说明了组件提供的那些服务(消息、操作、变量)。为了能够充分地推断组件及包含它的体系结构，ADL 提供了能够说明组件需要(即要求体系结构中其它组件为之提供的服务)的工具。这样，接口就定义了组件能够提出的计算委托及其用途上的约束。组件类型是实现组件重用的手段。组件类型保证了组件能够在体系结构描述中多次实例化，并且每个实例可以对应于组件的不同实现。抽象组

件类型也可以参数化，进一步促进重用。现有的 ADL 都将组件类型与实例区分开来。

由于基于体系结构开发的系统大都是大型、长时间运行的系统，因而系统的进化能力显得格外重要，组件的进化能力是系统进化的基础。ADL 是通过组件的子类型及其特性的细化来支持进化过程的。目前，只有少数几种 ADL 部分地支持进化，对进化的支持程度通常依赖于所选择的实现(编程)语言。其它 ADL 将组件模型看作是静态的。ADL 语言大多是利用语言的子类型来实现对进化支持的：利用 OO 方法，从其它类型派生出它的接口类型，形成结构子类型(Rapide)。

3.2 连接子

连接子是用来建立组件间的交互以及支配这些交互规则的体系结构构造模块。与组件不同，连接子可以不与实现系统中的编译单元对应。它们可能以兼容消息路由设备实现(如 C2)，也可以以共享变量、表入口、缓冲区、对连接器的指令、动态数据结构、内嵌在代码中的过程调用序列、初始化参数、客户服务协议、管道、数据库、应用程序之间的 SQL 语句等等形式出现[15,16]。大多数 ADL 将连接子作为第一类实体，也有的 ADL(Rapide 与 MetaH)则不将连接子作为第一类实体。

连接子的接口是一组它与所连接组件之间的交互点。为了保证体系结构中的组件连接以及它们之间的通信正确，连接子应该导出所期待的服务作为它的接口。它能够推导出体系结构配置的形成情况。体系结构配置中要求组件端口与连接子角色的显式连接。

体系结构级的通信需要用复杂协议来表达。为了抽象这些协议并使之能够重用，ADL 应该将连接子构造为类型。构造连接子类型可以将作为用通信协议定义的类型系统化并独立于实现，或者作为内嵌的、基于它们的实现机制的枚举类型。

为完成对组件接口的有用分析、保证跨体系结构抽象层的细化一致性、强制互联与通信约束等，体系结构描述提供了连接子协议以及变换语法。为了确保执行计划的交互协议，建立起内部连接子依赖关系，强制用途边界，就必须说明连接子约束。ADL 可以通过强制风格不变性来实现约束(Aesop、C2、SADL)，或通过接受(Accept)属性限制给定角色中服务的(UniCon)，而在 Wright 中，则通过为每个角色说明接口协议来实现约束。

3.3 配置

体系结构配置或拓扑是描述体系结构的组件与连接子的连接图。体系结构配置提供信息来确定组件是否正确连接、接口是否匹配、连接子构成的通信是否正确，并说明实现要求行为的组合语义。

体系结构适合于描述大的、生存期长的系统。利用配置来支持系统的变化，使不同技术人员都能理解并熟悉系统。为帮助在一个较高的抽象层上理解系统(或系统族)，就需要对软件体系结构进行说明。为了使开发者与其有关人员之间的交流容易些，ADL 必须以简单的、可理解的语法来配置结构化(拓扑的)信息。理想的情况是从配置说明中澄清系统结构，即不需研究组件与连接子就能使构建系统的各种参与者理解系统。体系结构配置说明除文本形式外，有些 ADL 还提供了图形说明形式，文本描述与图形描述可以互换。多视图、多场景的体系结构说明方法在最新的研究中得到了明显的加强。

为了在不同细节层次上描述软件系统，ADL 将整个体系结构作为另一个较大系统的单个组件。也就是说，体系结构具有复合或等级复合的特性。另一方面，体系结构配置支持采用异构组件与连接子。这是因为软件体系结构的目的之一是促进大规模系统的开发，即倾向于使用已有的组件与不同粒度的连接子，这些组件与连接子的设计者、形式化模型、开发者、编程语言、操作系统、通信协议可能都不相同。另外一个事实是，大型的、长期运行的系统是在不断增长的。因而，ADL 必须支持可能增长的系统的说明与开发。大多数 ADL 提供了复合特性，所以，任意尺度的配置都可以相对简洁地在足够的抽象高度表示出来。

4. 进一步研究方向

现有的 ADL 中，开发的焦点似乎集中在增强语言的分析与系统生成上。体系结构毕竟只是达到目的的手段，开发者能够推断目的系统的信息比仅仅为体系结构的信息要珍贵得多。能够快速开发出系统或管理最终产品质量才是 ADL 的真正目的。ADL 经历了一个在数量上快速增长的时期，今后的研究工作重点将放在 ADL 共性研究与语言互换、与其它产品集成、提高支持工具的有效性等方面。

◇ADL 之间的信息互换 现有的 ADL 大多是与领域相关的，所以不利于对不同领域体系结构的说明。但这些针对不同领域的 ADL 在某些方面又大同小异，造成资源的冗余。其实，大多数 ADL 具有一系列的共同概念。如何用一种公共形式把各种语言综合起来，使得能够交换各种体系结构描述信息，将是今后 ADL 研究的重点之一。ACME 被称为体系结构互换语言，为集成 ADL 作了开创性的工作。

◇ADL 信息与其它生命周期产品的集成 体系结构描述是整个软件生命周期的一个环节，ADL 如何能够承上启下将是十分重要的研究课题。一方面，要研究体系结构描述向其它文档转移，另一方面，要研究如

何利用需求分析成果来直接生成系统的体系结构说明。

◇**加强对进化的研究** 目前的 ADL 有些可以实现组件与连接子的进化，但这样的进化能力是有限的。其一，是因为这样的进化大多是通过子类型来实现的。其二，系统级的进化能力才是最终目的。系统级的进化必然涉及到系统族问题，对这方面的研究也将促进领域工程的发展。

◇**支持工具** 尽管现有的 ADL 都提供了支持工具集，但将这些 ADL 与工具应用于实际系统开发中的成功范例还很有限。支持工具的可用性与有效性较差，严重地阻碍了这些 ADL 的广泛应用。支持工具须在多视图、细化与回溯、跨体系结构层一致性检查等方面进一步加强。

参 考 文 献

1 Perry D E, Wolf A L. Foundations for the Study of Software Architectures ACM SIGSOFT Software Engineering Notes, 1992(Oct.): 40～52

2 Garlan D, Shaw M. An Introduction to Software Architecture: [CMU/SEI-93-TR-33]. Pittsburggh, Pa, SEI, CMU, Dec1993

3 Shaw M, Garlan D. Software Architecture-Perspectives on An Emerging Discipline. Prentice Hall, 1997

4 Garlan D, Shaw M. An Introduction to Software Architecture: Advances in Software Engineering and Knowledge Engineering, volume I. World Scientific Publishing, 1993

5 Luckham D C, Vera J An Event-Based Architecture Definition Language. IEEE Trans. on SE. 1995(Sep.): 717～734

6 Garlan D, et al Summary of the Dagstuhl Workshop on Software Architecture, February 1995. Reprinted in ACM Software

7 Garlan D. In: Proc. of the First Intl. Workshop on Architectures for Software Systems, Seattle, WA, April 1995

8 Wolf A L. In: Proc. of the Second Intl. Software Architecture Workshop (ISAW-2), San Francisco, CA, October 1996

9 Clements P C. A Survey of Architecture Description Languages. In: Proc. of the Eighth Intl. Workshop on Software Specification and Design, Paderborn, Germany, March 1996

10 Medvidovic N. A Classification and Comparison Framework for Software Architecture Description Languages, February 1996

11 Garlan D, et al. ACME: An Architecture Interchange Language Submitted for publication, 1996

12 Allen R. HLA: A Standards Effort as Architectural Style. In: Same to [8]

13 Luckham D C, et al. Specification and Analysis of System Architecture Using Rapide. IEEE Trans. on Software Engineering, 1995(April) 336～355

14 *Wolf A L. Succeedings of the Second International Software Architecture Workshop (ISAW-2).* ACM SIGSOFT Software Engineering Notes, 1997(Jan): 42～56

15 Shaw M, et al. Abstractions for Software Architecture and Tools to Support Them. Same to [13]

16 Garlan D, et al. ACME: An Architectural Interconnection Language; [Technical Report, *CMU-CS-95-219*]. Carnegie Mellon University, November 1995

17 王昕，金淳兆. 软件体系结构的研究与发展现状 计算机科学，1998，25(3)

(上接第53页)

参 考 文 献

1 Maruzzi S. The Microsoft Windows 95 Developer's Guide. Ziff-Davis, 1996

2 Roetzheim W Programming Windows with Borland C++ 4.5. Ziff-Davis, 1994

3 Schildt H. C++: The Complete Reference (Second Edition). McGraw-Hill, 1995

4 Bertalanffy L V. General System Theory. George Braziller, Inc., New York, 1973

5 Haken H. Synergetic Computers and Cognition—— A Top-Down Approach to Neural Nets. Springer-Verlag Berlin Heidelberg, 1991

6 Swan T. Foundations of Delphi Development for Windows 95. IDG Books Worldwide, Inc., 1995

7 汪应洛，主编. 系统工程理论、方法与应用. 高等教育出版社，1998

8 尼科里斯 B，普利戈金 I. 探索复杂性. 罗久里，陈奎宁译. 四川教育出版社，1976

9 国家计划委员会、建设部发布. 建设项目经济评价方法与参数. 中国计划出版社，1995

10 吉田耕作. 泛函分析. 人民教育出版社，1981. 7

11 肯尼思 法内科尔. 分形几何——数学基础及其应用. 东北大学出版社，1993. 5

12 Balakrishnan. P V Hall N. G A Maximun Procedure for the Optimal Insertion Timing of Executions. European Jouranl of Operational Research, 1995, 85: 368～382