软件体系结构描述语言研究现状分析*)

田丽从　张　莉　周伯生

(北京航空航天大学软件工程研究所　北京100083)

摘　要　软件体系结构描述语言ADL(Architecture Description Language)为软件体系结构的表示和分析提供了语言符号和支持工具。目前,已定义的ADL超过20种,新的ADL还在不断出现。然而,各种ADL并没有在实际项目开发中得到真正的推广。为了明确ADL的研究进展情况,分析了ADL的研究现状,讨论了ADL研究中存在的主要问题及解决思路。

关键词　体系结构描述语言(ADL),软件体系结构,统一建模语言(UML),设计过程,比较

Analysis of the Current Status of Architecture Description Languages

TIAN Li-Cong　ZHANG Li　ZHOU Bo-Sheng

(Software Engineering Institute, Beijing University of Aeronautics and Astronautics, Beijing 100083)

Abstract　Architecture Description Languages (ADLs) provide both notations and tools for modeling a software system's architecture. Now, there are already more than 20 ADLs defined, and new ADLs continue to come up. However, few of them are really adopted in real-world software development. In order to assess their progress and prospects, the current status of ADLs research is examined, major problems are analyzed, and possible solutions are discussed.

Keywords　Architecture Description Languages (ADLs), Software architecture, Unified Modeling Language (UML), Design process, Comparison

1　引言

随着软件体系结构在软件开发中作用的不断提高和近期来软件体系结构概念的明确提出[1,2],要求为其表示和分析开发专门支持工具和环境的需求也越来越迫切。软件体系结构描述语言ADL(Architecture Description Language)就是为满足这一需求而定义的一种语言符号。目前,已定义的ADL超过20种。在国外,具代表性的ADL包括Aesop[3],C2[4,5],Darwin[6], MetaH[7], Rapide[8], Unicon[9], Wright[10], ACME[11]等;国内包括XYZ/ADL[12],ABC/ADL[13],FRADL,A-ADL等。

但是,目前各种ADL在实际项目中的推广情况却并不理想。除了研究报告,在实际项目开发中软件体系结构描述仍然基本停留在非形式化的基础上,主要是框线图的形式;也有一些开发人员在使用面向对象建模语言,如UML[14],来表示软件体系结构的设计结果。但总体来说,各种描述方式所提供的软件体系结构描述能力与期望中的ADL相比仍然存在较大的差距。设计人员仍然期待能够有好的ADL及支持工具推出。为了明确ADL的研究进展情况,以便集中研究力量解决关键问题,促进ADL的推广和应用,本文分析了ADL的研究现状,讨论了ADL研究中存在的主要问题及解决思路。

本文第2节介绍ADL的基本概念;第3节讨论ADL的研究现状;第4节分析ADL研究中存在的主要问题及解决思路;最后给出结论。

2　ADL基本概念

虽然软件体系结构研究至今已经历了10多年的发展,但总体上仍不够成熟。人们对于软件体系结构的许多方面都还没有统一的认识,如软件体系结构的定义,软件体系结构在整个软件开发生命周期中的作用等。这种情况导致人们对ADL的定义和理解也难以统一。关于ADL目前没有明确的定义,通常人们将"用来表示和分析软件体系结构设计的形式化符号称为ADL[15]"。

一般认为,ADL的核心设计元素主要包括:构件(Component);连接件(Connector);和体系结构配置(Architecture Configuration)。其中,构件表示系统中主要的计算元素和数据存储,如客户端、服务器、数据库等;连接件定义了构件之间的交互关系,如过程调用、消息传递、事件广播等;体系结构配置描述了构件、连接件之间的拓扑关系。构件、连接件定义中的一个重要方面是对其外部特性的描述,即接口的定义。通常,ADL还采用一种形式化技术作为语义信息描述的理论基础,如π-calculus[6]、偏序事件集理论[8]、CSP[10]、XYZ/E[12]等。ADL对设计元素的定义方式以及采用的形式化理论,很大程度上决定了ADL所具备的描述能力和适用范围。

某些"非ADL",如模块交互语言(MIL)、面向对象设计语言(如UML)、程序设计语言(如Ada)等都不同程度地具有某些软件体系结构层次的描述能力。目前,对于如何判断一种语言是不是ADL还没有公认的标准。本文讨论所基于的

*)基金项目:北京市科技新星计划资助项目(项目编号:H013610270112)。田丽从　博士生,主要研究领域为UML及其支持环境,软件体系结构。张　莉　教授,主要研究领域为过程工程,软件工程环境,软件体系结构。周伯生　教授,博导,主要从事过程工程、过程工程环境、软件工程研究。

ADL 都是在相关文献中获得普遍认可的。

此外，鉴于目前 ADL 定义还没有形成统一的标准，各 ADL 所能提供的支持能力与其支持工具密切相关的实际情况，本文将 ADL 支持工具也作为 ADL 的一部分加以讨论。

3 ADL 研究现状

总结 ADL 的相关研究内容，大致可概括为如下3个方面：ADL 定义及支持工具；ADL 的分析和比较；ADL 与标准建模技术的结合。下面针对这3个方面对 ADL 的研究进展情况进行介绍。

3.1 ADL 定义及支持工具

这是 ADL 各相关研究内容的核心，主要目的是为软件体系结构的表示和分析提供语言符号和支持工具。目前，已定义的 ADL 多于20种。各种 ADL 通常都带有相应的支持工具，综合起来，已经在一个较广的范围内为软件体系结构的创建、可视化显示、模型解析、分析、演化、模拟、代码生成、编译等诸多方面提供了支持，但每种 ADL 单独提供的能力却十分有限。表1对几种典型 ADL 各自的特点进行了总结。

表1 典型 ADL 特点比较

ADL	研究单位和代表人物	适用范围	主要设计元素/特点	具备的主要能力
Aesop	美国卡内基梅隆大学软件工程研究所，David Garlan 等	同一设计环境中多种软件体系结构风格的应用；特定风格软件体系结构设计环境的快速生成	定义了六种通用对象类型：构件、连接件、端口、角色、表示和绑定；以子类型方式对通用类型进行扩展可定义新类型；每个对象类型用一个 C++类表示，软件体系结构风格信息内嵌在 C++类的代码实现中	外部工具集成；语法制导的类型检查；代码编译；可执行系统的生成；循环、资源冲突、调度可行性检查
C2	美国加利福尼亚大学软件研究所，Richard N. Taylor 等	主要面向 C2风格软件体系结构的描述，适用于分布式异构环境中、基于消息的图形用户界面应用系统的设计	主要设计元素包括构件、连接件、以及它们之间的拓扑关系；构件间只能通过连接件相连，具有“受限的可见性”；异步通知消息和请求消息是构件间唯一的通信方式	体系结构演化；体系结构动态配置；多形式的类型检查；设计决策过程支持；可执行代码生成
MetaH	美国蜜井(Honeywell)公司	一种特定领域的 ADL，支持对可靠性和安全性要求较高的多处理器实时嵌入式系统的创建、分析和验证，主要适用于航空电子控制软件系统	语义基于形式化调度和数据流模型；提供预定义的构件和连接件类型，与领域密切相关；构件类型包括事件、端口、子程序、包、监视器和进程等；连接类型包括事件连接、端口连接、等价连接和存取连接等	软硬件绑定；实时调度；可靠性和安全性分析；代码自动生成、编译和链接
Unicon	美国卡内基梅隆大学软件工程研究所，Mary Shaw 等	一种通用类型的 ADL，支持多种常用构件和连接件类型的综合应用	主要设计元素为构件和连接件，构件和连接件类型以枚举方式预定义，并定义了构件类型和连接件类型之间的匹配规则；连接件机制内嵌到工具的实现当中；	类型检查；编译、链接和可执行系统的自动生成；外部工具集成；进程调度分析
Rapide	美国斯坦福大学，David C. Luckham 等	基于事件的、复杂、并发、分布式系统的体系结构描述	主要设计元素由接口(相当于构件)和连接规则组成，接口定义包括动作、服务、行为和约束；构件的计算和交互语义通过偏序事件集(posets)定义；构件的行为约束通过抽象状态和状态转移规则定义	软件体系结构模拟执行；模拟结果分析(约束检查器、posets 浏览器、模拟动画)；软件体系结构的动态配置和代码生成
Wright	美国卡内基梅隆大学软件工程研究所，Robert J. Allen 等	一种完全形式化的 ADL，将连接件语义进行了显示的形式化表示，支持用户将复杂的交互模式定义成新的连接件类型	主要设计元素包括构件、连接件和配置，构件定义包括端口和计算，连接件定义包括角色和胶水(Glue)；支持用户定义接口类型和风格；以 CSP 作为语义基础	用户自定义软件体系结构风格；风格约束检查；模型一致性、完整性检查；死锁分析
Darwin	英国科学、技术和医药皇家大学，Jeff Magee 和 Jeff Kramer 等	主要用来对基于消息传递的分布式系统进行描述	主要设计元素由构件组成，构件定义包括提供的服务和要求的服务；没有对连接件提供显式的描述，构件之间的交互通过绑定关系表示；以 π-calculus 为语义基础；	系统动态配置；代码自动生成和编译
ACME	美国卡内基梅隆大学软件工程研究所，Garlan 等	一种体系结构交换语言，为 ADL 及其工具之间信息的共享和交换提供了一个公共的集成架构	主要包括7个核心设计元素：构件、连接件、系统、端口、角色、表示和重映射	体系结构信息交换和共享
XYZ/ADL	中国科学院软件研究所，唐稚松等	一种通用类型的 ADL，具有较强的形式化理论基础；能够支持多种设计方法和多种软件体系结构风格的综合运用	主要设计元素包括构件(包括接口和计算)和连接件(接口和交互协议)，语义基于时序逻辑语言 XYZ/E，可对设计元素的静态和动态语义进行严格的形式化描述	支持软件体系结构的逐层细化、分析、验证和自动代码的生成；支持用户自定义软件体系结构风格
ABC/ADL	北京大学信息科学技术学院，梅宏等	一种通用类型的 ADL，以面向对象分析和设计方法为主导	主要概念包括构件、连接件和体系结构风格，并吸取了面向 Aspect 的开发思想	最突出的特点是支持软件体系结构描述向详细设计和实现的映射，并支持构件的组装

通过上述比较可以发现，每种 ADL 分别侧重软件体系结构设计的某些方面，适于解决不同的问题。在软件体系结构研究初期，这种“百家争鸣”的情况为人们深入理解软件体系结构不同方面的特性，以及软件体系结构设计在开发过程中的作用有积极的意义。但是，随着软件体系结构技术的不断发展，尤其是经历了对基本概念的收集、整理阶段，逐渐向实际项目开发中推广和应用时，这种情况长时间维持下去，将不利于软件体系结构设计工具的研发和软件体系结构技术的推广。

Garlan 等试图通过在 ADL 间实现信息共享和交换的方式来对这一状况进行缓解。ACME 的定义和 ADL Toolkit[16] 的研制是这一目的的直接体现。ADL Toolkit 是以 ACME 为数据交换基础而定义的一个工具集成架构，其最终目标是实现对任何 ADL 工具的集成(而不管它运行在何种平台，采用何种实现方式)，从而保证 ADL 间信息的共享和交换，使各 ADL 能够在功能上实现互补。这一架构目前仍处于研发过程之中，在模型语法层次的交换方面已经取得了一些可喜的成果，而在语义方面的交换则仍需进一步的研究。

3.2 ADL 的分析和比较

对 ADL 的概念缺乏统一的认识是导致 ADL 种类繁多

的重要原因。为了进一步明确 ADL 的概念，确定 ADL 到底应该具备哪些特性，以便为新一代 ADL 的定义提供可参考的依据，一些研究人员分别通过不同的方法对各种 ADL 的共性和差别进行了比较。

Vestal[17]最早采用分析综合的方法，对四种 ADL (LILEANNA，MetaH，Rapid 和 QAD)各自所具备的特性进行了考察。Vestal 认为从语言学和结构概念上来说，这些 ADL 有很多相似之处，它们的主要区别在于对结构元素语义信息的描述的侧重点各不相同。Vestal 的这一结论启发了人们探索 ADL 本质特性的思路。

随后，Clements[18]站在 ADL 使用者的角度，采用特征分析的方法，从"面向系统"、"面向语言"和"面向过程"三个方面以穷举的方式提出若干问题，根据每种 ADL 对这些问题的答案来确定该语言是否具备某种特性或能力。Clements 等人关于 ADL 的这种比较方法，并不利于发现哪些问题对于 ADL 是最本质的，但却可以详细地了解每种 ADL 已具备的特性。该研究结果还揭示出软件体系结构设计结果与其它开发活动的衔接问题在 ADL 研究中被普遍忽视。

Medvidovic 等人[19,20]在前面这些研究成果基础上，做了更为具体的分析，先后采用两种方法对 ACME，Aesop，C2，Darwin，MetaH，Rapide，SADL，Unicon，Weaves，Wright 等10种 ADL 的特性进行了比较。第一种方法对比了上述 ADL 对软件体系结构领域(Architectural Domains)的支持情况。软件体系结构领域表示 ADL 在基于软件体系结构的开发中应该提供的支持或具备的能力，包括：1)软件体系结构表示；2)对设计过程的支持；3)分析能力，包括静态分析和动态分析；4)对软件体系结构演化的支持能力，包括规格说明时期的演化和运行时期的演化；5)对软件体系结构细化的支持能力；6)对模型间追踪关系的支持能力；7)软件体系结构的模拟和运行能力。

第二种方法采用了基于特征的分类和比较方法。通过综合各种关于 ADL 的需求，提出一个 ADL 的比较框架(限于篇幅，比较框架的详细信息请参阅文[20])，定义了 ADL 中应包含的最基本的结构元素、各个元素应具有的特性，以及支持工具应具备的支持能力。这一比较框架较为全面地综合了现有 ADL 的各种特性，对于已有 ADL 的改进和新的 ADL 的定义都有很重要的参考价值。

各种关于 ADL 的比较研究成果不同程度地表明，当前 ADL 研究主要侧重于对软件体系结构分析、演化，以及代码自动生成能力的研究，对模型的细化、跟踪、尤其是对设计过程的支持是 ADL 研究中的弱点。

3.3 ADL 与 UML 的结合

可以说 UML 和 ADL 分别体现了软件体系结构描述问题的两种观点。一种观点认为，软件体系结构描述的主要目的是为了帮助系统相关人员理解系统，并便于他们之间的交流。因此，软件体系结构描述必须是简单、易理解的，而且最好是可视化的，并能支持设计人员从多个视图描述系统，描述符号不一定要具备严格的形式化语义，向开发人员提供一些简单的分析功能就足够了，这种观点的描述手段以 UML 为代表；另一种看法，则主张软件体系结构描述应具备严格的形式化语法和语义，应提供强有力的分析能力，这种观点直接导致了 ADL 的产生。

这两种观点各有利弊。如果能将这两种方法融合起来，就可相互弥补，促进软件体系结构技术的推广。在 UML2.0[26]被正式发布之前，Kruchten[21]，Garlan[22]，Hofmeister[23]，Robbins[24]，Medvidovic[25]等研究人员对 UML1.X 与 ADL 融合的可能性进行了探讨。研究结果表明，UML1.X 提供的建模符号从概念上来说与软件体系结构存在较大差异；但对其进行适当扩展后，却不失为一种比较实际可行的方法。最重要的是，利用 UML 进行软件体系结构建模可较容易地保证各阶段设计结果的紧密衔接。

最近，UML2.0[26]被正式发布，其中确实增强了对软件体系结构和基于构件的开发的支持能力，增加了诸如结构类、端口、连接件、绑定等体系结构概念，并对构件的语义也做了相应的修改。但是，根据 ADL 的研究现状来分析，在软件体系结构描述技术得到统一和真正规范化之前，UML 仍不可能替代 ADL 的地位，ADL 研究仍有广阔的发展空间。

4 ADL 研究中存在的主要问题

ADL 要在实际项目开发中得到推广和应用，关键是缩短体系结构理论与实际开发活动的距离。从这个角度出发，我们将 ADL 研究中存在的问题归纳为如下几个方面：

4.1 种类繁多，令应用人员无所适从

每种 ADL 各自关注软件体系结构的某个方面，且各有千秋，这种情况分散了 ADL 的研究力量，也使应用人员在选择 ADL 时无所适从。

对于这一问题，归纳起来大致有以下几种解决思路：①定义一种"统一的(Unified)"ADL；②ADL 工具集成；③用标准建模技术(如 UML)替代 ADL；④维持现状。

第①种思路：吸取现有 ADL 中的各种研究成果，重新定义一种"统一的"ADL，使该 ADL 具备其它所有 ADL 的能力。这种方案目前看来不太现实，不仅因为定义一种适合于各类系统的描述语言本身是非常困难的事情，还因为目前研究领域对 ADL 应该具备哪些能力并没有完全统一的认识。当人们对软件体系结构理解足够深入之后，这种方式也许是一种比较理想的方案。正如 UML 的出现促进了面向对象建模技术的发展和推广，这种方式或许也可作为 ADL 发展的理想目标。对于这一目标的实现，发现和定义语言的核心(Kernel)，并使其具备良好的扩展能力是至关重要的。

第②种思路：Garlan 等人正在为这一目标而努力，这种方式在语法层次的交换已经取得了可喜的成果，但语义层次的交换却较困难。

第③种思路：利用 UML 进行软件体系结构建模存在的优势和不足在前面已经进行了讨论。目前来看，在软件体系结构描述尚难统一和规范化的情况下，用 UML 来描述软件体系结构不失为一种实际可行的做法。

第④种思路：维持现有局面，直到有一种 ADL 胜出。在以上三种方案(或其它方案)取得令人信服的成果之前，这种情况是必然要存在的。

此外，还有其它一些思路，比如定义面向特定领域的 ADL 等。对比各种思路，可以看出，在目前这种状况下，很难找到一种单一的解决方案。实际上，这一现象本身并不会严重阻碍 ADL 的发展，最为关键的问题是各 ADL 及其支持工具应加强自身的实用化程度。Garlan[11]指出，对于实践者来说，最重要的是能够有一个切实可用的工具使其在合适的抽象层次上记录系统的结构，具备严格而形式化的语义则是第二位的。

4.2 忽视需求到软件体系结构设计元素的映射

软件体系结构通常被认为是需求到系统设计元素的首次映射，是需求与设计之间的桥梁[15]。Shaw 和 Garlan[27]指出“除了定义系统的组成和拓扑结构，软件体系结构还表明系统需求和系统结构元素的对应关系”。但一直以来，需求描述技术和软件体系结构描述技术总体上是独立发展的。在如何实现需求规格说明到软件体系结构设计的过渡，如何将需求说明与软件体系结构的描述和构造较好地结合起来方面的技术很少[28]。各种 ADL 对软件体系结构设计的支持通常只考虑软件体系结构描述及其之后的开发活动（比如分析、演化、维护、实现等），多数 ADL 支持软件体系结构模型向可执行代码的生成，但对需求到软件体系结构，或软件体系结构到需求的关系的支持却几乎是空白。这有悖于借助软件体系结构来弥补需求与设计之间的鸿沟的初衷[28]。

这个问题近两年来开始引起一些研究人员的关注。2001和2003年的软件工程国际会议设立了“需求到软件体系结构”的讨论专题（STRAW'01，STRAW'03）[29]对需求与软件体系结构的关系进行了专门的探讨，但实质性进展还较为少见。

需求到软件体系结构的映射是一个难以解决的问题，因为需求描述与软件体系结构描述在形式上存在的差异较大。这个问题需要需求工程和软件体系结构研究领域的共同努力。

4.3 对软件体系结构设计过程的支持能力极为有限

软件体系结构设计是设计师根据需求制定一系列设计决策，然后通过选择和定义不同的结构形式将这些决策表达出来的过程。在这一过程中，制定设计决策所需的各种信息，并不是在一开始就能够完全显露出来，而是随着设计过程的不断深入逐渐明确起来的。比如，设计初期，所能决定的可能仅仅是一个粗略的系统框架，各个构件的详细接口信息和连接情况很难在一开始就十分明确，这时也许约束较少的“框-线图”更适合表达的需要；随着设计的深入，构件的行为、接口信息，以及交互关系等逐渐明朗，这时则更需要较为严格的描述形式。这就要求 ADL 在设计空间的约束上最好能够提供一个由弱到强的过渡过程。这一能力使开发人员能够在不同的阶段分别关注不同的问题，从而对资源做出合理的分配，而不必等到软件体系结构描述全部完成之后再一次性对所有问题进行分析。

此外，在软件体系结构的设计决策过程中，体系结构设计师需要考虑很多问题，借助和利用很多设计技术、方法和工具，如软件体系结构风格、设计模式等。建模工具中应该为这些知识的存储、重用、检索等提供便利。

在已有的 ADL 中，C2的设计环境 Argo[5]从人类认知（Cognitive）的角度考虑了对设计决策过程的支持，通过“自动设计批评家（Automated Design Critics）”对设计过程进行监视，在必要时，以“to do”列表的方式向设计者提供必要的设计知识。批评家监视的设计内容涉及语法语义正确性、完整性、一致性、图符表达、可选方案、设计参数优化等诸多方面。

Darwin 的建模工具 Software Architecture's Assistant 在这方面也有一定的考虑。在易用性方面，Darwin 考虑了自顶向下和自底向上的系统构建方式、模型的管理和导航特性、自动布图等；在分析方面，考虑了模型不同部分间信息的一致性检查和设计结果的验证；此外，还考虑了其它有关项目管理的信息，如项目状态、设计决策、非功能需求、版本控制等信息的提取等。

但其它工具对设计决策过程的考虑则非常有限。ADL 工具按照体系结构设计师的建模思路对设计过程提供实际的支持是 ADL 走向实用化的关键因素之一。因此，ADL 研究应对这一方面给予足够的重视。

4.4 ADL 支持环境基础架构的研究

Garlan 和 Clements[3,18]等指出 ADL 支持环境基础架构研究的重要性。目前已经存在很多适用于某种特定风格的系统开发环境，比如基于数据流的、面向对象的、黑板系统、控制系统等，这些系统往往是各自独立的，分别为某种特定类型的应用程序的创建提供强有力的支持。但是，在单个系统的开发中，经常会需要综合利用多种风格的知识，如基于数据流的系统中的某个构件内部可能需要层次风格的结构实现，层次结构中某一层的内部可能采用面向对象的方式，或者是与其它风格的混合结构。目前能够支持多种风格综合应用，尤其是支持自定义特定风格设计环境的 ADL 还很少。而且，每个环境的建造通常都是从头做起，花费了相当高的代价。

构建设计环境基础架构将有利于该问题的解决。通过基础架构可以为软件体系结构设计环境的快速生成提供一些基础的模块，如数据库管理、工具集成框架、结构编辑器生成器等。当需要建造某种风格的设计环境时，只要根据该风格的定义，利用基础架构中的元素，即可快速生成一个新的软件体系结构设计环境，并且与已有的设计环境可以紧密地集成在统一的框架内，如图1所示。

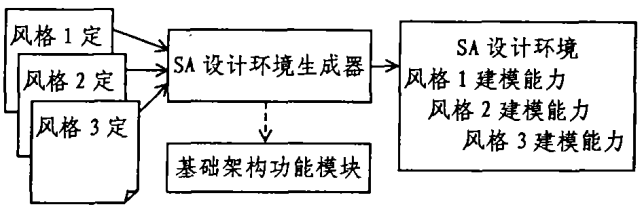

图1 基于基础架构的软件体系结构设计环境的建造

Aesop 在这一方面进行了开创性的研究。但 Aesop 实现中仍然存在诸多问题亟待解决，如对体系结构元素语义信息的描述几乎完全依靠外部工具来管理；对风格约束信息的表示嵌入在实现代码当中，只能隐式提供；新设计环境的生成过程非常繁琐等。

对这一问题研究的关键是解决诸如软件体系结构风格表示，新风格及其支持环境的集成，用户对风格语义信息的描述等问题。

4.5 其它

除了上述几个方面，ADL 在对软件体系结构细化过程的支持，以及与其它开发阶段（如详细设计）的衔接方面也存在较大的不足。这两个问题在文[19]和[18]中有详细的讨论。

结束语 本文对 ADL 及其支持工具的研究现状及存在的问题进行了分析。目前，我们正在针对其中某方面问题展开研究，在后续的文章中，会陆续介绍我们的一些解决方案。总体来说，我们认为如何设法缩短体系结构理论与开发实践之间的距离是 ADL 研究实现突破的关键。

参 考 文 献

1 Perry D E, Wolf A L. Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes, 1992, 17 (4): 40～52

2 Garlan D, Shaw M. An introduction to software architecture. In: Ambriola V, Tortora G. Advances in Software Engineering and Knowledge Engineering, Volume 1. New Jersey: World Scientific

Publishing Co.,1993
3 Garlan D,Allen R,Ockerbloom J. Exploiting style in architectural design environments. ACM SIGSOFT Software Engineering Notes,1994,19(5): 175～188
4 Medvidovic N,Rosenblum D S,Taylor R N. A language and environment for architecture-based software development and evolution. In:Proc. of the 21st Intl. Conf. on Software Engineering (ICSE '99),1999. 44～53
5 Robbins J E,Redmiles D F. Software architecture design from the perspective of human cognitive needs. In:Proc. of the California Software Symposium (CSS'96),Los Angeles,CA,1996
6 Magee J,Kramer J. Dynamic structure in software architectures. In:Proc. of ACM SIGSOFT'96: Fourth Symposium on the Foundations of Software Engineering (FSE4), San Francisco, CA, 1996. 3～14
7 Honeywell. MetaH language and tools. http://www.htc.honeywell.com/projects/ dssa_tools/dssa_tools_mh.html
8 Luckham D C. Rapide: a language and toolset for simulation of distributed systems by partial ordering of events. DIMACS Partial Order Methods Workshop IV,Princeton University,1996
9 Shaw M,DeLine R,Klein D V,et al. Abstractions for software architecture and tools to support them. IEEE Trans. on Software Engineering,1995,21(4):314～335
10 Allen R. A formal approach to software architecture:[Technical Report,CMU-CS-97-144]. Carnegie Mellon University,1997
11 Garlan D,Monroe R,Wile D. ACME: an architecture description interchange language. In:Proc. of CASCON '97,Nov. 1997
12 朱雪阳,唐稚松. 基于时序逻辑的软件体系结构描述语言 XYZ/ADL. 软件学报,2003,14(4):713～720
13 梅宏,陈锋,冯耀东,杨杰. ABC:基于体系结构、面向构件的软件开发方法. 软件学报,2003,14(4):721～732
14 Jacobson I,Booch G,Rumbaugh J. The unified software development process. MA: Addison-Wesley,1999
15 Garlan D. Software architecture. Wiley Encyclopedia of software engineering,Marciniak J (Ed.),John Wiley & Sons,Ltd. 2001
16 Towards an ADL ToolKit. http://www-2.cs.cmu.edu/～acme/adltk/
17 Vestal S. A cursory overview and comparison of four architecture description languages:[Technical Report]. Honeywell Technology Center, 1993. http://www.htc.honeywell.com/projects/dssa/ftp/papers/four_adl.ps
18 Clements P C. A survey of architecture description languages. Eighth International Workshop on Software Specification and Design,Germany,Mar. 1996
19 Medvidovic N,Rosenblum D S. Domains of concern in software architectures and architecture description languages. In:Proc. of the 1997 USENIX Conf. on Domain-Specific Languages, Santa Barbara,California,1997
20 Medvidovic N, Taylor R N. A classification and comparison framework for software architecture description languages. IEEE Trans. on Software Engineering,2000,26(1):70～93
21 Kruchten P B. The 4+1 view model of architecture. IEEE Software,1995,28(11):42～50
22 Garlan D,Kompanek A. Reconciling the needs of architecture description with object-modeling notations. In:Proc. of the Third Intl. Conf. on the Unified Modeling Language - 《UML》2000, York,UK,Oct. 2000
23 Hofmeister C,Nord R L,Soni D. Describing software architecture with UML. In:Proc. of Working IFIP Conf. on Software Architecture,1999. 145～160
24 Robbins J E,Medvidovic N,Redmiles D F,et al. Integrating architecture description languages with a standard design method. In: Proc. of the 20th Intl. Conf. on Software Engineering,1998. 209～218
25 Medvidovic N,Rosenblum D S,Redmiles D F,et al. Modeling software architectures in the Unified Modeling Language. ACM Trans. on Software Engineering and Methodology (TOSEM), 2002,11(1): 2～57
26 OMG. UML2.0 superstructure specification. http://www.omg.org/cgi-bin/doc?ptc/2003-08-02 /03-08-02.pdf
27 Shaw M,Garlan D. Software architecture: perspectives on an emerging discipline. Prentice Hall,1996
28 Shekaran C,Garlan D,Jackson M,et al. The role of software architecture in requirements engineering. In: Proc. of the First Intl. Conf. on Requirements Engineering,Apr. 1994. 239～245
29 International Workshop. From Software Requirements to Architectures(STRAW'2003). http://se.uwaterloo.ca/～straw03/

(上接第96页)

·针对复杂电子商务交易模式,如何设计有效的原子性支付协议

·如何实现支付协议中匿名性和原子性的相容

·电子拍卖交易模式中的原子性研究

·证券交易和期货交易中的原子性研究

·电子慈善事务中的原子性研究

·利用原子性实现电子支付系统的容侵

·电子支付协议原子性的形式化分析

结论 原子性是电子支付协议中应考虑的一个重要特性,国内外已进行一些重要的探讨和研究,提出了一些解决思路和方法,但仍有待人们进行深一步研究。

本文介绍了电子支付的原子性概念,分析了一些重要电子支付协议的原子性,就原子性电子支付协议实现策略进行了探讨,提出了一种新的原子性实现方法并用于构造一个原子性电子合同签署协议,阐述了电子支付原子性研究的最新发展和亟待解决的问题。

参考文献

1 Chaum D. Blind Signatures for Untraceable Payments. In: Proc. Crypto'82, LNCS,Springer-Verlag,1983. 199～203
2 Tygar J D. Atomicity in Electronic Commerce. In: Proc. of the 15th Annual ACM Symposium on Principles of Distributed-Computing, May 1996. 8～26
3 Heintze N,Tygar J D,Wing J,Wong H C. Model Checking Electroinc Commerce Protocols. In:Proc. of the 2nd USENIX Workshop on Electronic Commerce, Nov,1996. 147～164
4 Cox B,Tygar J D,Sirbu M. NetBill Security and Transaction Protocol. In: Proc. of the First USENIX Workshop on Electronic Commerce, July 1995. 77～88
5 Camp L J,Harkavy M,Tygar J D,Yee B. Anonymous Atomic Transactions. In:Proc. of the 2nd USENIX Workshop on Electronic Commerce, 1996. 123～133
6 Wang G,Das A. Models and Protocol Structures for Software Agent Based Complex E-Commerce Transactions. Journal of Electronic Commerce 2001 , LNCS 2115, Berlin. Springer-Verlag, 2001. 121～131
7 Schuldt H,Popovici A,Schek H. Execution Gurantees in Electronic Commerce Payments. In:Proc. of the 8th USENIX Workshop on Transaction and Database Dynamics, LNCS 1773,Berlin. Springer-Verlag,2000. 193～202
8 Adi K,Debbabi M,Mejri M. A New Logic for Electronic Commerce Protocols. AMAST 2000, LNCS, 2000, 1816: 499～513
9 Su J,Tygar J D. Building Blocks for Atomicity in Electronic Commerce. In: Proc. of the 6th USENIX Security Symposium, July 1996. 97～104