TBus:一种CASE环境体系结构风格研究与实现*)

郭 兵[1,2] 谢 俊[1] 赵平原[1] 沈 艳[2] 熊光泽[2]

(成都国腾通讯(集团)有限公司 成都610041)[1] (电子科技大学 成都610054)[2]

摘 要 CASE(Computer Aided Software Engineering 计算机辅助软件工程)环境作为一类复杂的系统软件,其体系结构至关重要。本文在工具总线(ToolBus)的基础上,从软件体系结构的角度,提出了一种基于工具总线的CASE环境体系结构风格TBus,对体系结构模型、系统行为进行了形式化描述,以及相应的工具结构模型、工具适配器结构模型、工具集成机制等方面进行了深入的研究与分析。建立TBus体系结构风格,对于促进分布式CASE环境和软件平台开发具有重要的指导意义。

关键词 CASE环境结构,工具集成,工具总线,软件体系结构

TBus:Research and Implementation on a Kind of CASE Environment Architectural Style

GUO Bing[1,2] XIE Jun[1] ZHAO Ping-Yuan[1] SHEN Yan[2] XIONG Guang-Ze[2]

(Chengdu GoldTel Communication(Group)Co.,LTD,Chengdu 610041,China)[1]

(Computer Science and Engineering College,University of Electronic Science & Technology of China,Chengdu 610054,China)[2]

Abstract CASE(Computer Aided Software Engineering)environment,which software architecture is very important, is a kind of complicated system software. From the viewpoint of software architecture,this paper presents TBus which is a kind of CASE environment architectural style based on ToolBus,while describes the architectural model and system's behavior in formal method,researches and analyzes corresponding tool structural model,tool adaptor structural model and tool integration mechanism,etc. Building TBus possesses important guiding significance to facilitate the development of distributed CASE environment and software platform.

Keywords Software architecture of CASE environment,Tool integration, ToolBus,Software architecture

1 引言

CASE环境作为一类复杂的系统软件,一般由一套工具集(Tool Suite)和工具集成框架(Tool Integration Framework)组成,能够给应用软件整个开发过程的大部分阶段或所有阶段提供计算机辅助支持,提高了软件的开发质量和开发效率。而如何开发和构造一个CASE环境,满足集成性、开放性、适用性、灵活性、分布性等要求则一直是个难题,目前还没有一个通用的实现方法,其中软件体系结构(Software Architecture)是一个关键因素[1]。

与操作系统的结构相类似,CASE环境结构可以分为三种,即单块式(Monolithic)结构、层次(layered)式结构和工具总线(ToolBus)结构。目前大多数CASE环境采用单块式或层次式结构实现,工具总线结构是近十年出现的新技术,其基本思想是工具总线只提供工具集成所需的最基本的公共服务,完成集成框架的数据集成和控制集成功能。工具由具有规范接口的工具构件(Component)组成,以"即插即用"的方式加入到工具总线上,同时,工具构件只能通过工具总线相互间接通信,从而减少了CASE环境的层次,大大简化了CASE环境的互连结构,提高了其开放性、适应性和灵活性[2~4]。

2 基于工具总线的CASE环境体系结构风格

2.1 工具总线概述

美国Maryland大学的James Purtilo和Richard Snodgrass于1994在一篇论文中首先提出"软件总线(Software Bus)"的概念,并实现了一种软件总线原型Polylith[5]。在此基础上,人们进一步提出OO软件总线(Object-Oriented Software Bus面向对象的软件总线,简称软总线)的概念,将此种软总线技术用于CASE环境的构造中,作为工具集成的"骨架",这种软总线称为工具软总线(Tool Software Bus),简称工具总线(ToolBus)[6]。

从软件体系结构的角度看,工具总线作为一种连接件(Connector),是用于工具集成的结构性构件,为分布、异构的工具构件间提供了通信、协作和便利的服务,实现了工具集成设施和工具逻辑处理功能的分离,将工具集成设施从以前的隐式方式改为显式方式,为支持基于Internet的协同工作奠定了重要的基础。

在文[7,8]中作者提出了一种抽象的工具总线模型,对工具总线的功能抽象、内部结构、接口规范、实现方式等问题进行了全面而深入的研究,并在CORBA规范的基础上实现了一种工具总线原型LambdaBus。

2.2 体系结构的描述

在工具总线的基础上,我们接着提出了一种面向CASE环境领域的软件体系结构风格(Software Architectural Style),称之为基于工具总线的CASE环境体系结构风格,简称TBus体系结构风格。我们在研究、设计与开发TBus体系

*)本课题得到国防科工委"九五"预研项目基金资助(No. 15. 3. 1. 2)。郭 兵 博士,成都国腾通讯(集团)有限公司与电子科技大学联合培养企业博士后,主要研究方向为嵌入式软件开发平台、SOC协同设计和验证和中间件技术。熊光泽 博士生导师,主要研究方向为实时计算机系统及其应用。

结构风格时，设计目标首先是指导一类嵌入式软件开发平台的开发，然后可用于更宽的 CASE 环境领域，如通用的 IDE 环境、面向电子商务的开发平台等。因此，在设计工具构件类型和工具总线时，力图使之具有相当的灵活性，能够适应更宽的应用范围[9]。

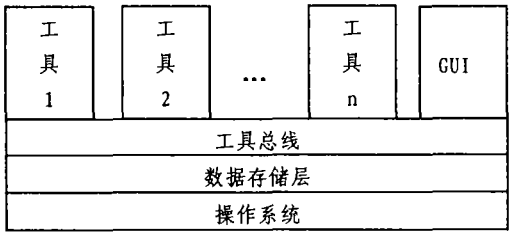

图1 基于工具总线的 CASE 环境结构模型

TBus 体系结构风格构成了一种基于工具总线的 CASE 环境结构模型(如图1所示)，包含工具构件、工具总线和约束(Constraints)三部分，即 TBus 体系结构风格＝{工具构件，工具总线，约束}，其中[10~12]：

(1)*工具构件* 主要完成工具的逻辑处理功能，通过工具总线与其它工具构件交互，为顺利实现 CASE 环境的集成，工具构件的接口必须符合工具总线的接口规范。

(2)*工具总线* 在 TBus 体系结构风格中起连接件的作用，是工具构件间相互通信的中介，主要功能是实现工具构件到 CASE 环境的集成，对工具构件实施统一和一致的管理，方便工具构件间的消息通信和数据交换。

(3)*约束* TBus 体系结构风格规定了工具构件通过工具总线的双向通信关系，工具构件与工具总线的主要拓扑规则和约束为：

·CASE 环境中的工具构件只有一个底部，工具总线则只有一个顶部。

·工具构件的底部应连接到工具总线的顶部，所有工具构件间的交互都必须通过工具总线，禁止工具构件间的直接通信。

·工具总线可以和任意数目的工具构件和工具总线进行连接。

·不能有循环。也就是说，一个工具构件不能接收它自己产生的消息。

对图1所示系统(简称 System_CASE)的 TBus 体系结构风格，采用 C2SADEL 语言描述如下[13]：

```
architecture System_CASE is {
  component_types {
    component Tool1_type is extern {Tool1.c2;}
    component Tool2_type is extern {Tool2.c2;}
    ...
    component Tooln_type is extern {Tooln.c2;}
    component GUI_type is extern {GUI.c2;}
  }
  connector_types {
    connector ToolBus_type is extern {ToolBus.c2;}
  }
  architectural_topology {
    component_instances {
      Tool1 : Tool1_type;
      Tool2 : Tool2_type;
      ...
      Tooln : Tooln_type;
      GUI : GUI_type;
    }
    connector_instances {
      ToolBus : ToolBus_type;
    }
    connections {
      connector ToolBus {
        top Tool1,Tool2,... ,Tooln,GUI;
      }
    }
  }
```

2.3 系统行为的描述

遵循 TBus 体系结构风格的 CASE 环境中工具构件间的交互行为如图2所示。工具总线为一定数量的工具 $T_i(i=1,2,\cdots,n)$提供协作服务，组合成一个完整的系统。每个工具构件的内部行为或实现是不相关的，它们可以采用不同的编程语言实现，或根据不同的功能设计规范而产生，同时，每个工具构件应该能够维护其内部的状态。一般情况下，每个工具都包含一个适配器构件，将工具构件内部的数据格式调整为工具总线的数据格式和消息协议[6]。

工具总线中包含可变数量的线程 $S_i(i=1,2,\cdots,m)$，线程 S_i 的并发执行代表了整个 CASE 环境的预期行为。工具与线程间的关系，可能是一对一，也可能多个线程控制一个工具，或者一个线程控制多个工具。

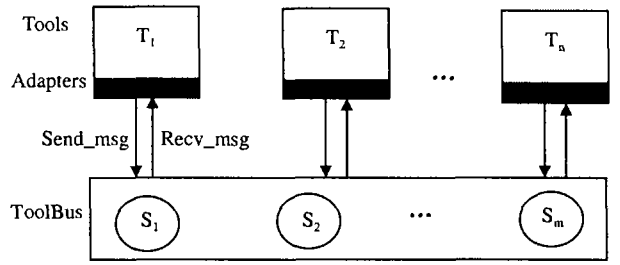

图2 工具构件间的交互行为

C2SADEL 语言侧重于 CASE 环境体系结构静态的描述，下面采用进程代数(Process Algebra)方法对于整个系统的动态行为进行描述，主要描述工具构件间的动态交互关系。CCS(Calculus of Communication System)和 CSP(Communicating Sequential Processes)是进程代数方法的代表，是描述通信和并发的演算系统，它们均以进程为计算单位，进程的基本动作是原子性动作。π 演算(π Calculus)是 R. Milner 对 CCS 的进一步改进，其基本计算实体只有名字和进程，进程之间的通信通过传递名字来完成。虽然工具和工具总线通常是一个多线程的程序，由于进程和线程具有很多相似性，可以简化为一个抽象的进程实体，因此，本文采用 π 演算方法，刻画工具构件间的通信过程[14]。

设：

$\xrightarrow[send_msg]{}$表示工具构件发出一条消息。

$\xrightarrow[recv_msg]{}$表示工具构件收到一条消息。

MPS(Message Passing Style)表示消息传递方式，分为同步和异步两种方式。

因此，CASE 环境的系统行为描述如下：

$App(s) \triangleq$

if t=synchronous: $\overline{Oc}(a).\bar{a}[\xrightarrow[send\_msg]{}].a[\xrightarrow[recv\_msg]{}].App(s);$

if t=asynchronous: $\overline{Oc}(a).\bar{a}[\xrightarrow[send\_msg]{}].App(s) \mid a[\xrightarrow[recv\_msg]{}].App(s);$

其中，$t \in MPS$.

$Adapter(s) \triangleq \overline{Oc}(\xrightarrow[send\_msg]{}) \mid \bar{a}(\xrightarrow[recv\_msg]{}) \mid Adapter(s);$

$$ToolBus(s)\triangleq Oc(a).a(\xrightarrow[send\_msg]{}).\bar{o}[\xrightarrow[seng\_msg]{}].o(\xrightarrow[recv\_msg]{}).\bar{a}([\xrightarrow[recv\_msg]{}]|Oc(a).a(\xrightarrow[send\_msg]{}).\bar{O}[\xrightarrow[send\_msg]{}]|o(\xrightarrow[recv\_msg]{}).\bar{a}[\xrightarrow[recv\_msg]{}]|Toolbus(s);$$

$$Tool(s)\triangleq(Oc,a)(App(s)|Adapter(s));$$

$$CASE \triangleq (o)(\Pi s\in case\ Tool(s)|\Pi s\in case\ ToolBus(s));$$

App(s)：工具构件发出或接收一系列的消息，用 s 表示一个工具构件，而 App(s)则是指工具构件的一次应用(指发出消息或接收消息)。t 表示消息传递方式。发送方工具构件沿通道 Oc 向 ToolBus 发出消息，接收方工具构件通过通道 a 从 ToolBus 接收消息。

Adapter(s)：工具适配器构件完成消息发送或接收时协议和数据的转换功能。

ToolBus(s)：每个发送方或接收方都连接到 ToolBus，其任务是并发地接收工具构件发出的消息，并将消息沿 o 通道输出到相应的工具构件。

2.4 支持的工具集成机制分析

TBus 体系结构风格能够"粘贴"和协调工具成为一个完整的 CASE 环境，满足 CASE 环境对集成性的要求，主要因为工具总线为工具提供了以下集成设施：

(1)*数据集成* 工具总线不但为工具间数据交换提供了一套数据管理和控制设施，而且尽可能提供统一的公共数据表示(如 XML)，并根据工具特定需求，为不同类型的数据提供一条数据总线进行数据交换。

(2)*控制集成* 工具间的所有控制集成都通过消息传递实现，不管工具在同一台机器上，还是在网络中的不同机器上。

(3)*表示集成* 由于人-机界面程序往往是多线程的，同时，用户界面和工具并不都是完全分离的，往往包含一些简单的工具和过程控制设施，因此，可将用户界面看作一类特殊的工具，工具间的界面控制和过程控制能够通过数据集成和控制集成的设施完成。但另外一部分表示集成机制，如用户界面的一致性开发和保证，需要提供相应的工具和构件库，这部分功能不包含在工具总线中。

由于 CASE 环境的体系结构已经确定，工具总线通过提供一条公共的通信通道，将工具集成在一起，工具内部是紧耦合的关系，工具之间是松耦合的关系。因此，开发统一规格的工具，提高工具的适应性是一个重要的问题。

3 工具的组成

3.1 工具结构模型

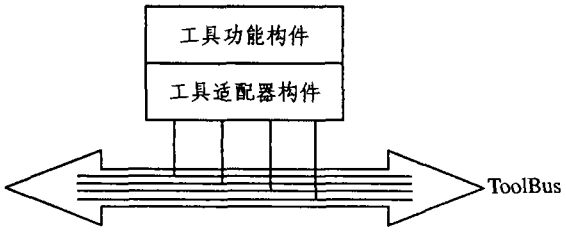

图3 工具结构模型

在图3所示模型中，一个工具可简化为由工具功能构件和工具适配器构件两个独立的工具构件组成，即工具构件＝{工具功能构件，工具适配器构件}。其中，工具功能构件完成工具自身的逻辑处理功能，工具适配器构件实现工具与工具总线规范的数据接口和控制接口。所有与工具总线接口有关的部分都集中体现在工具适配器构件中，这种设计将工具逻辑处理功能与工具集成设施相分离，从而隔离了 CASE 环境对工具功能构件设计的影响，使工具开发者的大部分工作可以独立于 CASE 环境。按照这一模型设计的工具，相当于 CASE 环境中的一个标准插件，工具适配器构件是数据集成和控制集成两方面的"插头"，工具总线则是与之对应的"插槽"，容易实现工具的"即插即用"功能[15]。

3.2 工具适配器结构模型

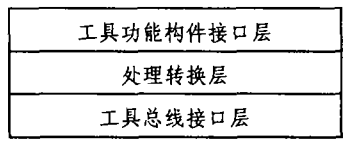

图4 工具适配器结构模型

CASE 环境采用基于工具总线结构的一个结果是，一般情况下，每个工具都将封装一层软件，这层软件扮演"工具适配器"的脚色。工具适配器构件位于工具功能构件和工具总线之间，处理工具功能构件和工具总线间交互不匹配问题，主要作用是：

(1)在工具功能构件的内部数据格式和工具总线的公共数据表示之间进行转换。

(2)负责工具功能构件和工具总线间通信协议的调整。

(3)负责向工具功能构件传递消息和接收执行结果，具有一定的工具功能构件管理功能。

工具适配器的主要结构如图4所示，其中工具功能构件接口层主要负责工具功能构件的消息接收、发送以及数据格式的识别，处理转换层主要负责数据格式的转换和通信协议的调整，工具总线接口层主要负责工具总线的消息接收、发送以及数据格式的识别。针对不同类型(包括功能、实现语言等)的工具功能构件，工具适配器构件的具体处理内容和实现也各不相同。

4 一个 TBus 体系结构实例 LambdaBridge

在 TBus 体系结构风格基础上，我们建立了一个 TBus 体系结构实例，该实例是一个嵌入式软件开发平台原型系统 LambdaBridge(如图5所示)，验证了工具总线和 TBus 体系结构风格的有效性。LambdaBridge 主要完成嵌入式软件编码阶段的任务，包括项目管理、版本控制、源程序编辑、交叉编译、交叉调试、优化、覆盖率测试、辅助文档生成等功能，其主要设计思想是在 CORBA 规范的基础上，实现了一种工具总线 LambdaBus，作为开发平台的核心结构，并抽象出嵌入式软件开发需要的一些公共服务(如与目标机的通信、版本控制以及许可证管理等)，建立一个嵌入式软件开发平台框架，便于各种工具构件(包含第三方商业上现成的工具构件和客户自建的工具构件)的集成，灵活创建开发平台，满足不同领域开发者的多种需求。

LambdaManager 是 LanbdaBridge 的集成用户界面，是所有开发工具单一的"存取点"，包含编辑器、项目管理等常用工具以及过程控制设施，因此，LambdaManager 作为一个特

(下转第161页)

务集的可调度性及其它实时性能问题，从而提高了设计和开发的效率。由于本文在RTETFusion模型中做出了所考虑的额外开销都为常量的假设，使得本融合机制只能适用于实时任务集比较小的嵌入式实时系统。修正本文的假设，改进RTETFusion模型，使得这个融合机制能够对一个大的实时系统进行实时性能分析是进一步的研究工作。在下一步的另一个研究工作将是根据环境的要求和相应的实时调度数学理论对RTETMeasuring方法和RTETFusion模型进行调整，使得融合模型RTPAFusion能够分析其它调度环境的实时性能。

参考文献

1 Waltz E, Llinas J. 多传感器数据融合. 赵宗贵等译. 北京：机械电子部第28研究所，1993

2 White F. A model for data fusion. In: SPIE Conf. on Sensor Fusion, Orlando, FL, April, 1988

3 Stewart D B. Measuring execution time and real-time performance. In: Proc. of 2001 Embedded Systems Conf. San Francisco, CA, April 2001

4 Katcher D, Arakawa H, Strosnider J. Engineering and analysis of fixed priority schedulers. IEEE Transactions on Software Engineering, 1993, 19(9):920～934

5 Secka A. Automatic debugging of a real-time system using analysis and prediction of various scheduling algorithm implementations: [M. S. Thesis]. Dept. of Electrical and Computer Engineering, University of Maryland, College Park, MD, Nov. 2000

6 Lehoczky J, Sha L, Ding Y. The rate monotonic scheduling algorithm: exact characterization and average case behavior. In: Proc. 10th IEEE Real-Time Systems Symposium, Santa Monica, CA, Dec. 1989. 166～171

7 Liu C L, Layland J W. Scheduling algorithms for multiprogramming in a hard real time environment. Journal of the Association for Computing Machinery, 1973, 20(1):44～61

（上接第156页）

殊的工具通过LambdaBus与其它工具相连。LambdaManager主要采用TCL/TK语言实现，因为TCL/TK语言具有可移植性、客户定制性等优点，同时，CORBA规范也已制定了IDL与TCL/TK语言的映射，为LambdaManager基于CORBA的构件化实现提供了可能。

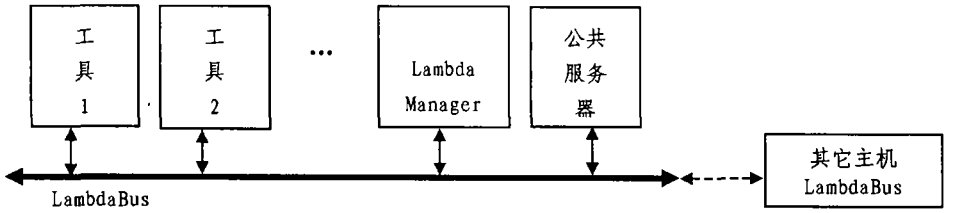

图5 LambdaBridge的总体结构

总结 由于工具总线的功能抽象集合和结构缺乏一个公认、合理的规范，以及工具通过工具总线的通信机制和策略，方便了CASE环境开发，却使得系统执行效率不高。因此，TBus体系结构还需要对松耦合与紧耦合、开放性与集成性等性能问题开展进一步的研究和实践，更好地满足系统的需求。

工具总线不但是一个软件系统，而且反映了一套系统的设计思想和软件体系结构。CASE环境采用了“工具总线＋工具构件”的体系结构[16]，一方面简化了CASE环境的结构，利于新的CASE环境的构造和已有CASE环境的重用；另一方面采用基于体系结构的构件化方法开发CASE环境，包括建模、分析、实现和演化，只要工具构件接口符合体系结构要求，工具构件的开发完全是并行的，对于CASE环境的快速开发带来了极大的便利。

参考文献

1 Shaw M. The Coming-of-Age of Software Architecture Research. In: Proc 23th Intl. Conf. on Software Engineering, Jan. 2001. 657～664

2 Liedtke J. On microkernel construction. In: Proc. of the 15th ACM symposium on operating system principles, 1995. 237～250

3 Wallnau K C, Feiler P H. Tool Integration and environment Architecture: [Technical report of SEI]. Carnegie Mellon University, 1994

4 杨芙清，邵维忠，宗志东，朱冰. 过程驱动的软件工程环境. 电子学报，1998，26(8)：24～27

5 Purtilo J M. The Polylith Software Bus. ACM Transactions on Programming Languages and Systems, 1994, 16(1):151～174

6 Bergstra J A, Klint P. The ToolBus - a component interconnection architecture. P9408, Programming Research Group, University of Amsterdam, Amsterdam, 1994

7 郭兵，沈艳，谢峻，赵平原，熊光泽. 工具总线模型研究. 已被《计算机应用》录用，2003

8 郭兵，沈艳，谢峻，赵平原，熊光泽. 工具总线：CASE环境的一种新结构. 已被《系统工程与电子技术》录用，2003

9 李保建，曾广周，林宗楷. 一种基于TriBus的软件集成框架. 计算机研究与发展，1999，36(9)

10 Medvidovic N, Khare R, Guntersdorfer M. An architecture-Centered Approach to Software Environment Integration. In: Proc. of the Ground System Architectures Workshop (GSAW 99), El Segundo, CA, March, 1999

11 Oreizy P, Medvidovic N, Taylor R N, Rosenblum D S. Software Architecture and Component Technologies: Bridging the Gap. In: Proc. of the Workshop on Compositional Software Architectures, Monterey, CA, Jan. 1998

12 Müller-Planitz C. THE CWAVE 2000 VISUAL AGENT WORKBENCH: [PhD dissertation]. Department of Computer Science, University of Utah, Aug. 2000

13 Medvidovic N, Rosenblum D S, Taylor R N. A Language and Environment for Architecture-Based Software Development and Evolution. ACM, ICSE'99, Los Angles, CA, pp. 44～53

14 Milner R, Parrow J, Walker D. A calculus of mobile process II. Information and Computation, 1992. 41～77

15 杨芙清，邵维忠，梅宏. 面向对象的CASE环境青鸟II型系统的设计与实现. 中国科学(A辑)，1995，25(5)：533～542

16 Johnson R E. Framework = (Components + Patterns). Communica-tion of ACM, Oct. 1999. 39～42