中 国 科 学 (A 辑)
第25卷 第5期 SCIENCE IN CHINA (Series A) 1995年5月

面向对象的CASE环境青鸟II型系统的设计与实现*

杨芙清 邵维忠 梅 宏
(北京大学计算机科学技术系,北京 100871)

摘要 青鸟II型系统(JB2)是一个面向对象的软件工程环境. 其集成机制由对象管理系统、消息服务器、界面类库等部件构成,并且提供了一个具有永久对象处理能力的面向对象的编程语言. 介绍了JB2的总体方案、主要部件及其工具结构模型,并讨论该系统所采用的方法与技术. 最后,讨论了JB2环境的集成度与开放性,并介绍使两者得到兼顾与统一的方法与经验.

关键词 CASE环境 面向对象 永久对象 集成 开放性 工具结构模型

自NATO会议以来,20多年过去了,软件工程领域的研究已得到了长足的发展,在需求工程、形式规格说明、软件设计方法学及系统生成、测试和维护等方面成果纷呈,显示了软件工程学科强有力的生命力. 软件工程思想已为越来越多的人所接受,其优越性也已为实践所证明. 从80年代开始,计算机辅助软件工程(CASE)的研究及实践吸引了众多的研究者,其宗旨是利用计算机对软件工程提供技术支持,以提高开发人员的软件生产率并改善软件质量. 目前CASE已成为软件工程领域中最活跃的研究方向,特别是在软件生产自动化仍是难以企及的目标的情况下,CASE更是解决软件危机,提高软件生产率和改善软件质量的实际的手段和途径. CASE的研究经历了从单个工具到工具集合再到集成环境的过程. 所谓CASE环境包括了工具集成机制和一个工具集. 集成化的CASE环境更是体现了在控制、数据和界面三个方面的高度集成.

目前,国际上对CASE集成机制及相关工具的研究非常活跃,许多著名公司纷纷投资于集成化CASE环境的研制,已有不少商用产品出现,同时产生了一些CASE环境模型,如PCTE以及NIST/ECMA的参考模型等[1],[1],[2]. 在国内,CASE的研究开始于80年代初期并已取得不少成果. 在"七五"和"八五"期间,我国均把集成化软件工程环境的研制列为国家重点科技攻关项目. 在"七五"期间,我们和兄弟单位一起研制成功了集成化软件工程环境青鸟I型系统(JB1). 这是一个支持软件生命周期全过程的CASE环境[2,3]. 目前该项科研成果已经转化为产品,具有在不同的硬件及操作系统之上的版本,并已应用于管理信息系统、证券交易系统、地理信息系统及国产系统软件等多个领域的软件开发.

1994-09-10收稿,1994-12-15收修改稿
*国家"八五"重点科技攻关计划资助项目

1) Commission of the European Communities. PCTE functional specifications. Version 1.5 Nov 1988
2) NIST. A Reference Model for Computer Assisted Software Engineering Emironment Frameworks, Ver 1 Oe May. 1990

青鸟II型系统(JB2)的研制是我们在"八五"期间与国内20余所大学、科研机构和产业部门联合承担的国家重点科技攻关项目,其主要目标是:

(1)提供一个支持软件生命周期全过程的集成化软件开发环境.既作为一个通用的软件开发平台,又可通过剪裁和扩充而成为支持专用领域的软件开发平台.

(2)为支持面向对象的软件开发,提供对象管理系统、面向永久对象的编程语言以及OO工具系列.

(3)环境集成机制在数据集成、控制集成和界面集成三个方面达到软件工具的紧密集成,并且有较好的开放性.

(4)除提供面向系统的软件工具之外,还提供了一批面向领域的软件工具.

(5)设计目标和实现技术力求达到90年代的先进水平,并努力跟踪国际标准化组织的最新动向.

(6)以实用化作为根本目标.

目前JB2的开发工作已经完成,集成工作也已接近尾声.本文旨在介绍JB2的设计思想和主要实现技术.首先给出JB2的总体方案,然后详细介绍环境集成机制的各主要部分及采用的工具结构模型,讨论环境的集成性和开放性,最后提出将要进一步研究的几个方向.

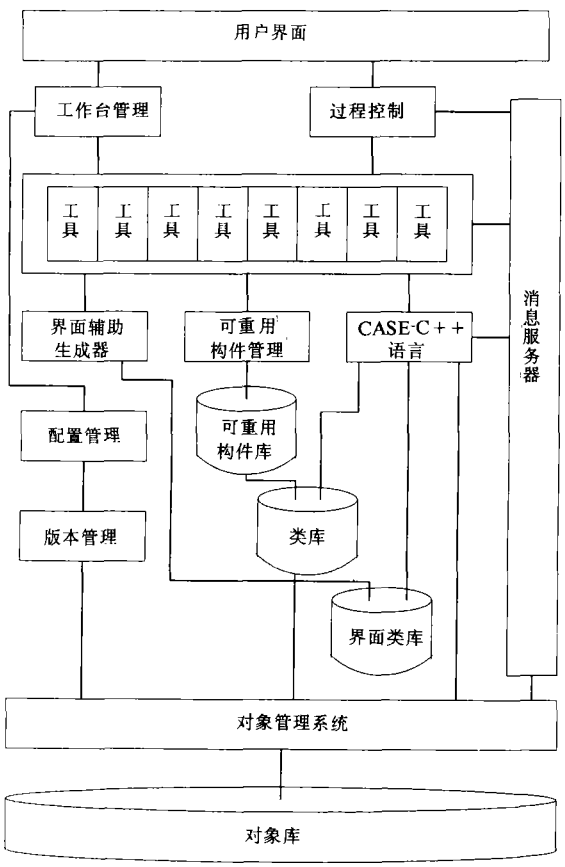

图1 JB2的总体结构

1 系统总体方案

JB2是一个面向对象的集成化软件工程开发环境,系统本身的开发也较全面地采用了面向对象的技术.在设计上,JB2参照国际上一些著名的环境模型,通过对它们的分析和评估,吸取其优点,提出了符合国情,基于我们自己预研成果之上的集成环境模型.

JB2集成机制体现了数据、控制和界面三方面的高度集成,其数据集成基于对象管理系统(OMS),控制集成基于消息服务器,界面集成基于OSF/Motif以及在此基础上开发的JB2界面类库和界面辅助生成器.同时,环境中提供了一个面向永久对象的编程语言CASE-C++作为工具的书写语言,保证了工具在环境中的紧密集成.JB2的集成机制形成了一个开放的工具插槽.工具的设计遵照统一的工具结构模型,并提倡以CASE-C++语言编写,从而很容易集成到环境中并达到较高的集成度.外来工具在按规范要求封装后也容易被环境

接纳．工具之间的通讯由消息服务器完成．JB2 的工具分为三大类：① 传统类工具，覆盖整个软件生命期，支持传统的软件开发方法；② OO 工具，包括从 OOA 到 OOD 到 OOP 的一系列工具，支持面向对象的软件开发方法；③ 应用类工具，面向特定的应用领域，如数据库应用生成工具、地理信息系统开发工具、实时系统辅助生成工具等．图 1 显示了 JB2 的整体结构．

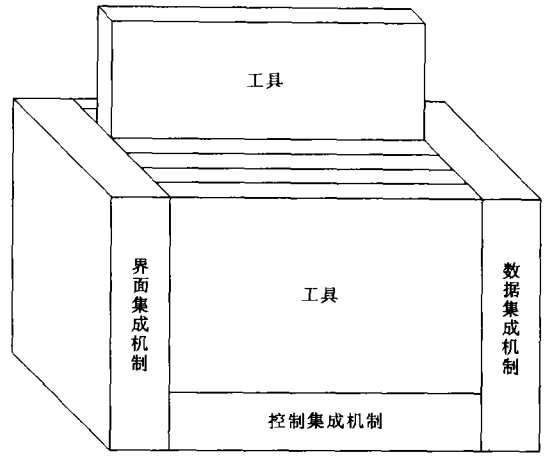

图 2　JB2 工具插槽

JB2 环境的开放的工具插槽保证了环境易于剪裁和扩充．按结构模型规范要求封装好的工具，实质上构成了一个“插件”，工具向环境的集成仅是将“插件”插入工具插槽中，工具从环境中删除只需将“插件”从插槽中“拔出”．JB2 集成机制所体现出的插槽形式为环境的扩充和剪裁提供了很大方便，也保证了集成和开放的有机统一．图 2 显示了工具“插件”和环境工具插槽间的基本关系：

2　环境集成机制的各个主要部分

2.1　对象管理系统

JB2 的对象管理系统(JB2/OMS)是 JB2 环境的核心，其作用是对软件开发过程中定义的对象类以及对象实例进行存储和管理．JB2/OMS 实现了永久对象的存储与管理，可并发地运行于网络中各个工作站之上；支持在多用户、多程序之间对象的共享和并发执行．它的并发控制机制保证了对象属性信息的一致性．JB2/OMS 对于环境的工具的开发者和最终用户提供了统一的面向对象技术支持，即，它管理和存储的对象，既包括构成工具本身的对象，也包括用户在使用工具时产生的结果对象，以及用户运用 CASE-C++ 语言开发的任何应用程序中的对象．JB2/OMS 分为以下几个主要部分：

2.1.1 类库　JB2 的类库保存并管理环境中长期使用的全部类定义以及类之间的继承关系．库中的类包括：由环境提供的一组预定义的类，各个工具开发者提供的工具成分类，一般用户在软件开发过程中不断提交的类．此外，JB2 为支持基于 OSF/Motif 的图形用户界面的开发，提供了一个丰富而高效的界面类库，它是整个 JB2 类库的一个相对独立的子库(详见 2.3)．类库的主要意义是支持类定义在不同用户、不同程序间的共享和重用．类库中的每个类都是一个可重用构件．在一个 CASE-C++ 程序中定义的类，如果要准备重用，或者愿提供给其它用户共享，则通过 export 语句提交到类库中长期保存，其它 CASE-C++ 源程序通过 import 语句中列出所需要的类便可共享类库中自己有权共享的类．所谓共享，包括以下两种方式：① 利用类库中的类，创建自己的对象；② 以类库的类作为父类，继承它的属性和服务，定义它的子类．从软件重用的意义上讲，除以上两种方式外，系统还支持把类库中的类复制到新开发的程序中，加以适当的修改而产生新的类定义．JB2 的类库提供了权限管理机制．每个类仅供其开发者授权的用户共享．类库完整地保留了各个类之间的继承关系，目前

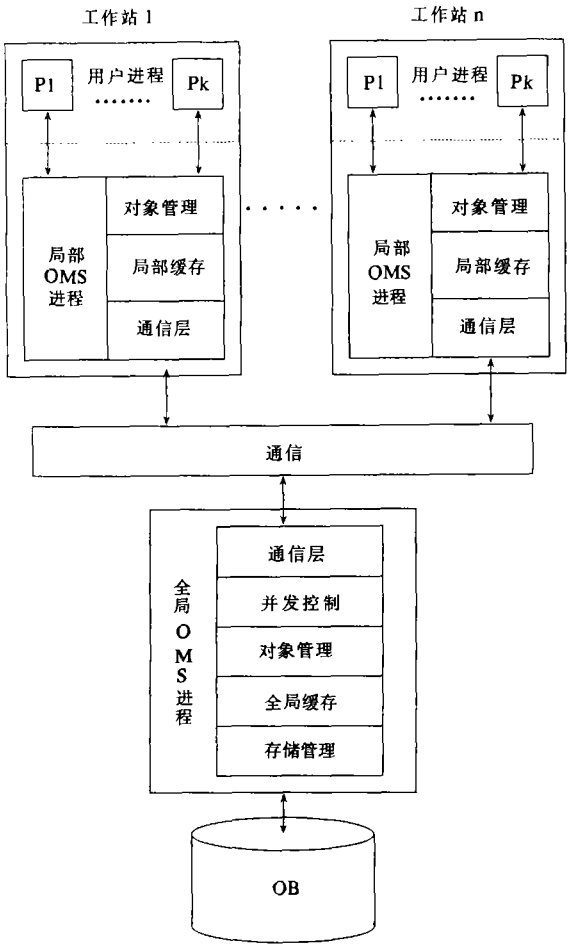

图 3 JB2/OMS 结构示意图

的版本仅支持单继承，所以，它是一个树形结构.

2.1.2 对象库 JB2/OMS 的对象库保存并管理 JB2 环境中的全部永久对象的实例信息. 这个对象库是 CASE-C++ 语言实现永久对象概念的基层支持. 正是由于这种支持，才使用户(包括工具开发者和一般用户)在程序中声明的永久对象能够超越程序运行时间而长期存在. 此外，它使程序员可以把永久对象看作是"无缝的"，而不必关心它在内外存之间的转换(详见 2.2). 对象库的每个对象有一个全系统唯一的"内部标识". 对象可以在网络中并发执行的各个程序或进程之间共享. 对象库本身是集中式的，由一个"全局 OMS 进程"响应来自各个工作站的请求. 每个工作站上，有一个"局部 OMS 进程"响应本工作站的各个进程的请求. 为了提高效率，相应地设立了一个全局缓冲区和每个工作站上的局部缓冲区(见图 3).

为了保证在并发条件下共享对象的数据完整性，OMS 设立了锁管理机制：每个对象被一个进程独占时加锁，要访问这个对象的其它进程在队列中等待，对象解锁时唤醒等待该对象的进程.

2.1.3 OMS 浏览器 OMS 浏览器是 OMS 的交互式浏览及维护界面(因此它也是整个 JB2 系统的图形用户界面的一个组成部分)，它允许用户浏览类库和对象库中的全部可见的类及对象以及它们之间的结构关系. 用户还可以通过它交互式地定义新类、修改类的权限或修改处于开发状态的类定义(每个类有"开发"和"完成"两种状态，仅开发状态下允许修改). 环境的特权用户除了可使用上述功能之外，还可以删除已经无用的对象实例和对象类.

2.2 CASE-C++ 语言

CASE-C++ 语言是 JB2 环境中的一个面向对象的编程语言(OOPL). 设计和实现这个语言的主要动机是为了提高在 CASE 环境下面向对象的软件开发工作的效率并达到软件工具在环境中的紧密集成，而现有 OOPL 在许多方面不能很理想地达到上述要求. 我们选择了目前应用比较广泛的 C++ 语言作为基础，并针对上述要求加以扩充，所以命名为 CASE-C++ 语言. CASE-C++ 对 C++ 完全兼容，它对 C++ 的扩充主要有以下几个方面：

2.2.1 支持永久对象 CASE-C++ 语言在 JB2 的对象管理系统(JB2/OMS)支持下提供了

永久对象(Persistent Object)的定义和处理能力．用户在程序中创建一个对象时，可以用保留字persistent声明它是永久的．这种对象的生命期可超越程序运行的时间而在OMS的对象库中长期保存．本程序或其它程序再次使用这个对象时，对象呈现它上一次使用时的状态．永久对象的描述及处理机制使用户可以把对象看作是"无缝的"——程序员不必关心每个对象是在内存还是在外存．对象在内、外存之间的转储是由CASE-C++语言编译系统在每个必要的时刻，自动地调用OMS的基本读写函数实现的．因此程序员不必借助文件系统或数据库来保存对象，从而解除了在程序中进行对象与这些存储管理系统之间的数据变换以及显式地使用"装入"、"存储"等语句所带来的负担．

2.2.2　对象之间关系表达与存储——链(Link)机制的引入　CASE-C++引入了链的概念和处理机制来表示对象之间的关系．一个链，从一个源对象出发，链接到一个目的对象．它作为对象的一个属性(实例变量)在源对象中定义，它的值则是目的对象的永久性标识．这一点是链和非永久性OOPL中"对象指针"概念的重要区别，对象指针的值只是目的对象的内存地址．因此不能长期保存；而链则可以随着永久对象的存储，使对象之间的关系也得到长期的保存．在CASE-C++中，链的另一个特点是它可以带有自己的链属性(链属性的数目和数据类型由程序员确定)，从而使链可以描述较复杂的关系信息．这一特点使语言对于对象之间关系的描述能力大为加强．链既可以作为对象内的实例变量，也可以是存在于对象之外的独立链变量．两种情况都可以通过CASE-C++的P-new函数动态地创建目的对象．前一种情况便于形成对象之间的结构关系，后一种情况可以表示一个结构的"源头"，例如一个树结构的根．链机制的引入使CASE-C++语言能够有力地表达OOA和OOD结果的语义[4,5]．它能很方便地表示并长期存储由不同类的对象和多重关系构成的复杂结构，特别是OOA模型和OOD模型中常见的整体-部分结构和实例连接，同时使用户可以很方便地进行对象的导航式访问．

2.2.3　支持类定义的重用和共享　CASE-C++语言提供了使类定义成为可重用构件或供其它用户共享的设施．在CASE-C++程序中定义的类，可以用export语句提交到类库，以供重用和共享；另一方面，程序中可以用import语句从类库中引入自己所需要(并且有权使用)的类．CASE-C++语言允许C++语言原有的类定义形式(以小写class为保留字)，同时增加了扩充的类定义形式(以大写的CLASS为保留字)．凡属以下几种情况必须使用扩充的类定义形式：① 类定义的内部使用了CASE-C++的扩充语法成分(例如链)；② 准备移出以提供重用和共享的类；③ 准备创建永久对象的类．

2.2.4　宽谱系的对象粒度　CASE-C++可以表示不同粒度的对象．小者可以是最简单的对象，大者可以是由若干数据、文件、嵌套对象，以及指向动态结构的链所组成的复杂对象．大部分工具的输入信息和用户利用工具所产生的结果在逻辑上都具有相当的复杂性，而且要求保留不同的版本(例如结构化分析工具可以用对象表示DFD图中的图元同时又把整个DFD图连同它的附加文档组织成一个大粒度的对象，并且要为用户保留不同的版本)．CASE-C++有足够的能力描述此类多层次、多粒度的复杂对象，而且可在JB2/OMS的支持下实现配置管理和版本管理．

2.3　用户界面、界面类库和界面辅助生成器

JB2的用户界面是统一采用OSF/Motif开发的，为了保证界面风格的一致性，首先制订了

《JB2界面设计规范》,从整个环境到各个工具的界面开发均需严格地遵守这个规范. 尽管用Motif开发用户界面已比直接使用X-window简练得多,但用户界面部分的编程工作量仍然相当可观. 为了从根本上提高编程效率,并从技术上保证界面风格的一致性,我们在OSF/Motif之上开发了一个JB2界面类库. 界面类库是JB2总类库的一个独立的子库,其中的类是针对各种常用的界面构件定义的界面对象类. 每个类具有两种描述形式,一种是符合C++语法的类定义,另一种是符合CASE-C++语法的类定义. 这些类都是用Motif编程实现的. 在JB2界面类库的支持下,C++或CASE-C++程序员要定义一个界面成分(例如窗口、菜单或Button等),只需调用相应的界面类的构造函数来创建一个具有指定属性值的对象. (此外,可用类中提供的其它函数来控制界面的变化,或删除界面对象). 这使得界面部分的编程工作大为简化(典型地,百余行的Motif程序可以简化到一行到几行C++或CASE-C++程序). 另一方面,它也使不熟悉Motif的程序员能够快速地胜任界面开发工作. 界面类库中预定义的界面类目前已比较全面地包括了一般用户常用的界面成分. 用户可以利用这些预定义的类通过派生或组合产生自己的新类并提交到类库.

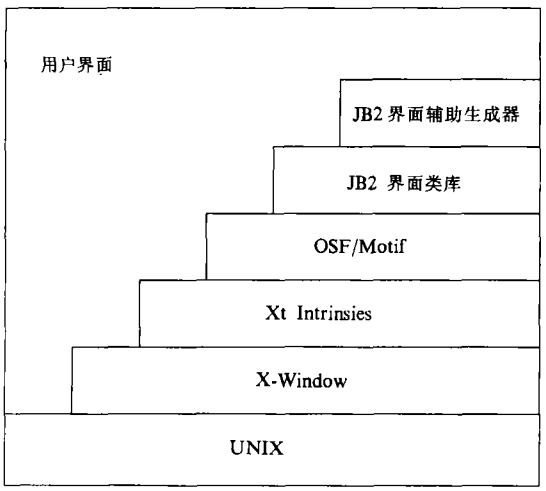

图4 JB2界面支持层次

界面辅助生成器是JB2环境中的一个交互式界面生成工具,它为用户的界面开发工作提供了可视化的支持. 用户可以在屏幕上直接构造自己所需要的界面成分. 开发结果是随时可见的. 最终辅助生成器将把屏幕上的开发结果转换为C,C++,CASE-C++或UIL程序源代码. 界面辅助生成器既可看成环境集成机制的一个组成部分,也可以作为一个独立的软件工具. JB2的界面类库和界面辅助生成器可以有效地保证界面设计的规范化和风格的一致性,并且明显地提高了界面的开发效率和质量. 从它们初步可用时起,JB2的界面开发人员就已开始在实际工作中使用,从而减轻了许多工作负担. 综上所述,JB2的界面开发支持层次如图4所示.

2.4 可重用构件库

JB2可重用构件库的设计目标是为环境提供一个支持多种重用级别的软件重用机制. 源程序一级的重用主要在JB2/OMS中的类库和JB2界面类库的基础上实现. 其它级别的重用还包括分析构件和设计构件的重用. 对象化的构件是库中最规范的可重用构件,其中包括在OOA阶段产生的分析类,OOD阶段产生的设计类和用OOPL定义的编程类;同时库中也将包括按传统方法开发并按构件库的描述规范所描述的非对象化构件. JB2构件库的组织允许表示可重用构件之间的多层关系,例如对象化构件之间的继承关系、组成同一系统的构件之间的组合关系、从分析构件到设计构件到编程构件之间的演化关系,以及应用领域分类关系等等. JB2可重用构件库以基于关键词的检索和交互式的提问细化和理解、定位作为重要的

检索策略、并在多层关系模型的支持下提高检索效率.

2.5 消息服务器

JB2 的消息服务器是在 UNIX 系统的 IPC/RPC 基础上开发的一个高层次的消息服务设施，它用于工具之间或工具与环境其它独立部件之间的通讯服务. 消息的发送和接收单位是工具(或环境部件). 消息是网络透明的，其语义由接收单位和开发单位之间约定，并按 JB2 的统一约定进行描述和登记. 消息服务器是提高环境的控制集成度的重要设施，它为相关工具之间的配合工作和切换提供了方便而有效的支持. 例如，设计工具在工作时可以发送消息要求分析工具显示相关的分析文档并进行修改，而操作员不必在这两个工具之间频繁地退出和进入. 用 JB2 消息服务器实现工具之间的通讯比直接使用 UNIX 系统的 IPC/RPC 要简练得多，例如程序员不必关心事件队列的定义，存取和查询等细节. 消息服务器也是 JB2 过程控制的基础，它为过程控制的实现提供了通讯服务.

2.6 过程控制

过程控制是提高软件质量及生产率的重要保证. 对过程模型及其描述语言以及过程驱动 CASE 环境的研究也是当前 CASE 研究的一个热点. 为此，JB2 提供了一个集成控制语言以满足过程控制的需要. 我们提出了一个面向对象的通用过程模型 OOSP，在该模型中，软件开发过程由一系列对象组成，对象间的交互构成了开发的活动. 相应地，提出一个面向对象的过程描述语言 OOPDL，使用该语言可以灵活地描述并控制开发管理者希望的软件过程，并能动态地修改软件过程以不断适应实际需要. OOPDL 的运行平台构建于消息服务器之上. JB2 还不能算是过程驱动的环境. 但是，使用 OOPDL 语言，开发者可以半自动地运行软件过程，更好地管理软件开发，提高生产率及软件质量.

2.7 配置管理、版本管理和工作台管理

配置管理、版本管理和工作台管理是当前真正受用户欢迎的 CASE 环境中都应具备的功能部件. 青鸟 II 型系统中的上述部件在概念上和功能上与青鸟 I 型系统基本相同[5]，所不同的是，青鸟 II 型系统因为有 OMS 的支持，所以为上述部件的实现带来了技术上的便利. 关于这一点，我们的经验和体会是：① 应该充分发挥 OMS 的作用，上层部件不再直接存放对象的物理实体，而是存放和管理它们的逻辑映像. ② 具有多个版本的对象，由版本管理部件管理其版本结构，但每个版本的实体被看作 OMS 中的一个独立的对象. 这种处理方法既避免了版本管理部件的物理存储负担，又减少了 OMS 设计的逻辑复杂性.

3 工具结构模型

工具与环境的接口问题是关系到环境的集成度和开放性的关键问题之一. 在没有任何约定的情况下开发出来的工具很难在环境中达到紧密的集成. 单纯依靠小范围内的特殊约定进行工具与集成机制的开发又很难使环境具有开放性. 解决这一问题的理想途径是开发者都遵守统一的标准. 然而目前做到这一步尚不现实. 首先是因为国际上尚未形成正式的环境及工具标准. 其次从形成标准到标准的普遍采用还需要更多的时间.

目前比较可行的办法是寻找一个对环境适应性较强、开发者容易实施的工具结构模型. 为此，我们在青鸟 II 型系统研制工作的初期，首先提出了如图 5 所示的工具模型. 青鸟 II 型系统中二十多个工具的开发单位使用这个模型的实践表明，这个模型较好地解决了工具与环

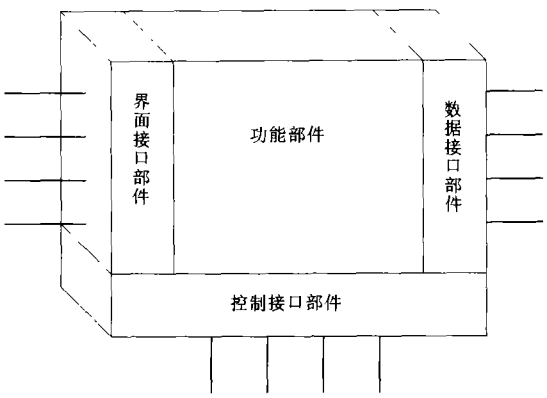

图5 JB2工具结构模型

境的接口问题,并使环境达到了较高的集成度和较强的开放性.

在这个模型中,一个工具由四个独立部件构成,即:功能部件、数据接口部件、控制接口部件和界面接口部件. 其中,功能部件是完成工具自身的功能所需的软件成分. 其它三个部件分别实现工具与环境的数据接口、控制接口和界面接口. 所有与环境接口有关的设计决策都集中体现在这三个接口部件中,从而隔离了环境对功能部件设计的影响,使工具开发者的大部分工作(有关功能部件的工作)可以独立于环境(这一点很受工具开发者欢迎). 按这一模型设计的工具,相当于环境中的一个标准插件,三个接口部件是数据、控制和界面三方面的"插头",环境的数据集成、控制集成和界面集成机制则是与之对应的"插槽".

这一模型对工具和环境集成机制具有双重的意义. 对工具而言,由于功能部件的独立性,当需要进入一个新的环境时,只需要更换或修改它的接口部件,这使得工具对不同的环境具有较强的可移植性. 对环境集成机制而言,一旦确定了"插槽"的规格,其设计则不受具体工具的影响,从而可按照统一的工具结构模式去追求较高的集成度. 集成工作也大为简化,因为不必再针对每个具体的工具去解决接口问题. 此外,按照这个工具结构模型对"外来工具"进行改造或"封装",使之具有符合环境要求的接口部件,如同给一个非标准的插件装上标准的插头,可以在很大程度上提高环境的开放性.

4 集成性与开放性

在集成化软件工程环境中,集成体现在多个方面. 例如,在文献[3]和[6]中提到的方法、概念、技术、工具、平台、过程、数据、控制、界面等方面的集成. 其中,数据集成、控制集成和界面集成是衡量环境集成级别(集成度)的主要因素,也是其它方面集成的基础. Wasserman[8]提出,以数据集成、控制集成以及界面集成(presentation integration)为坐标轴构成一个三维坐标系(图6),以每个轴上的刻度来衡量软件工程环境的集成度. 用这个坐标系衡量JB2的集成度,可以看到:在数据集成方面,主要的集成机制是JB2/OMS(包括对象库和类库),此外,有配置管理、版本管理和工作台管理. 在控制集成方面,主要集成机制是在UNIX消息服务基础上开发的JB2消息服务器. 在界面集成方面,以OSF/Motif为基本支持,在此基础上开发了JB2的界面类库和界面辅助生成器.

高集成度是集成化软件工程环境所追求的主要目标. JB2环境对数据、界面及控制三方面的集成均提供了较好的支持,从而使JB2达到了较好的集成度指标,保证了工具在环境中的紧密集成.

开放性是标志软件工程环境水平高低的另一个主要指标. 但是高度的集成往往会影响环境的开放性;广泛的开放又往往会使环境难以达到高度集成. 在JB2的研制实践中,我们采

用以下途径，使两者达到有机的统一．

(1)集成机制的设计，采用代表国内外主流方向的方法论和技术支持．所谓"主流方向"不是指"最新"或"当前用得最多"，而是从发展趋势上看，最有把握被广泛接受的方法与技术．这样的方法与技术，一方面可使环境达到较高的集成度；另一方面容易被工具开发者接受，并且随着它们的推广普及，环境的开放性将越来越加强．

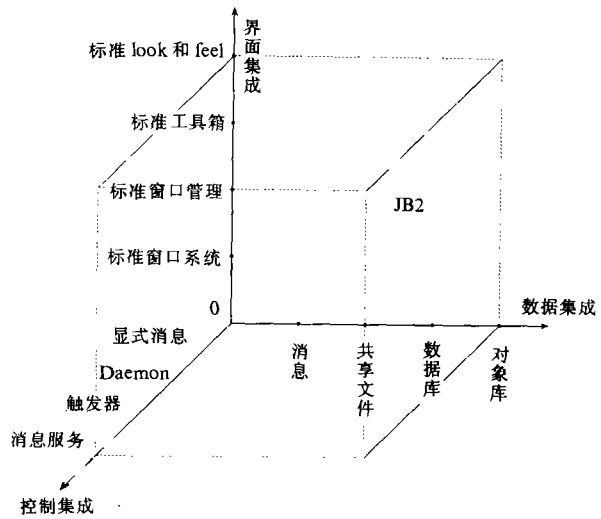

图6　环境的集成度

(2)集成机制提供统一的工具接口，即在数据、控制和界面三个方面，提供对各种工具一致的"插槽"．工具被看作一个个的"插件"，需按照统一的工具模型来设计．这种工具结构模型应该具有较强的适应性．

(3)集成机制对工具开发的技术支持应该有兼容性．例如，JB2为工具的开发和集成提供了JB2/OMS,CASE-C++,JB2消息服务器，及界面类库、界面辅助生成器等较高级的技术支持，同时，它对使用非OO方法及语言开发的工具和直接使用UNIX的IPC/RPC及Motif的工具，也能够接纳．当工具采用较低的技术支持时，其损失只是不能充分利用环境的功能，而不是无法集成到环境中．当然，这种兼容应该是有限度的，比如，不能支持工具采用风格完全不同的另一种窗口系统．

5　结束语

本文介绍了集成化软件工程环境青鸟II型系统的总体方案、集成机制的主要部件及工具结构模型，讨论了环境对面向对象技术的支持和集成与开放问题．由于这是一个规模很大的项目，不可能在一篇文章中对它的每一部分以及它所涉及的所有理论与技术都做详尽的介绍，对此，将陆续地撰写一批文章加以介绍和讨论．在本文之前，曾发表过几篇文章[7~9]，由于种种原因，个别的术语和实现细节与本文有所差异，凡此情况皆以本文为准．今后，我们将在以下几个方面继续开展工作：

(1)对象管理系统将在两个方面进行研究和开发．一是通用化，即支持不同的面向对象的编程语言，二是在实现上深入到操作系统内部，以求更高的效率．

(2)进一步扩充CASE-C++语言，特别是使它能描述并发的任务．

(3)对本文提出的环境集成机制和工具结构模型，在大量实践的基础上进行总结提高，争取能对软件工程标准的形成起到应有的作用．

参　考　文　献

1　Gallo F. Overview of PCTE and PCTE. ACM SIGPLAN Notices, 1989, 24(2)

2 杨芙清,方 裕,唐世渭等. 一个集成化的软件工程支撑环境. 软件学报, 1991, 2(2): 1～8
3 杨芙清,方 裕,唐世渭等. 软件工程核心支撑环境BETA-85. 中国科学, A辑, 1988, (5): 530～538
4 Coad P, Yourdon E. Object-Oriented Analysis. New Jersey: Prentice Hall, 1991
5 Coad P, Youndon E. Object-Oriented Design. New Jersey: Prentice Hall, 1991
6 Wasserman A I. Tool Integration in Software Engineering Environments. Berlin: Springer-Verlag, 1990. 137～149
7 Yang Fuqing, Shao Weizhong, Li Wei. A CASE environment of Jade Bird2. Chinese Journal of Electronics, 1993, 2(2): 9～13
8 Yang Fuqing, Huang Riri, Shao Weizhong. The design and implementation of a class manager in a object management system. 20th Euromicro Conference, Liverpool, 1994
9 杨芙清,邵维忠,柳军飞. 永久对象存储技术研究. 电子学报, 1994, 22(8): 1～8