一个 JAVA 可视化面向对象程序设计支撑系统的设计*)

The Design of a Visual Object-Oriente Programming Support System for JAVA

刘建宾[1,2]　郝克刚[1]

(西北大学计算机系　西安710069)[1]　(汕头大学工学院计算机系　汕头515063)[2]

Abstract　UML(Unify Modeling Language)is an international industrial standard in Object-Oriented software modeling, However it lacks static modeling mechanism for progress. To solve this problem, UML is extended with Abstract Logic Structure Diagram to refine the level of static modeling to the level of class method. Based on static class diagram in UML and Abstract Logic Structure Diagram, JVOOPSS, a Visual Object-Oriented Programming Support System for JAVA with concise, practical and easy-to-use characteristics, has been designed and implemented, which provides the static modeling capability of relations among classes and procedures of class methods, and supports forward transformation and smooth transition from visual modeling of Object-Oriented programs to coding implementation, and generates whole source codes of a JAVA program according to the design models, and realizes unification between visual modeling and coding.

Keywords　UML(Unify Modeling Language), Abstract logic structure diagrams, Java visual programming, Software tools

1　引言

由 Grady Booch、James Rumbaugh 和 Ivar Jacobson 共同创建的一种可视化说明、建造软件系统的工业标准语言——统一建模语言 UML(Unify Modeling Language)[1,2]是面向对象软件建模的国际标准，具有直观自然，表现力强的特点。对过程建模，尽管 UML 提供了活动图和顺序图，但它们均属动态模型，UML 缺乏对过程的静态建模方法，所以 UML 的静态建模机制并不完善。UML 活动图从动态观点来建立过程模型，而最终程序代码是静态的，因而活动图过程模型与代码两者的结构是不一致的。这种结构的不一致不仅会导致过程模型到程序代码的过渡不平滑，并且还会导致活动图模型与程序代码的一致性难以维护。另外从本质上讲，UML 活动图与传统的程序流程图非常相似，所以程序流程图的缺点在 UML 活动图中仍然存在。若用活动图描述算法，由于跳转不受限制，允许非结构化元素，增加了设计的随意性，容易导致非常复杂的结构和难以理解维护的非结构化代码，使过程设计复杂化。鉴于活动图本身存在着缺陷，现在的许多建模工具和系统并不予以支持。

抽象逻辑结构图(Abstract Logic Structure Diagrams)[3~5]，简称抽象逻辑图，是我们提出的一种简单实用、容易理解、结构良好的程序可视化描述语言。抽象逻辑图的概念、逻辑和实现三个层次的表示，支持程序的平滑设计过程和程序代码的生成，较好地解决了分析、设计与实现三个不同阶段过程静态描述的一致性问题。把抽象逻辑图和 UML 相结合，一方面能够发挥 UML 强大的建模能力，使创建的模型符合国际标准，另一方面又能利用抽象逻辑图可视化描述过程静态结构的特点，补充和完善 UML 静态建模机制对过程静态描述的缺乏与不足，形成既符合国际规范又有自己特色的面向对象建模技术，开发具有自主知识产权的面向对象建模工具产品，这是我们设计系统的基本出发点。

UML 提供了多种图形建模工具从不同的视角来刻画系统。在 UML 的所有建模工具中，类图是最有用和最易学习掌握的。在面向对象程序设计过程中引入抽象逻辑图，与 UML 类图结合使用，扩展了 UML 静态建模能力，填补了 UML 类图中类方法过程的可视化描述，从而构成类规格说明的完整静态描述，并可以此生成完整的目标程序语言的类代码，实现可视化建模和编码的有机统一。我们以 UML 类图建模理论和抽象逻辑图程序设计理论为基础，研制了一个实用性强、操作简单的 JAVA 可视化面向对象程序设计支撑系统 JVOOPSS。该系统支持从类间关系建模、类方法的过程建模到编码实现、代码生成的 JAVA 面向对象程序设计全过程，实现了整个程序设计过程的可视化支援。

JVOOPSS 的体系结构如图1所示。

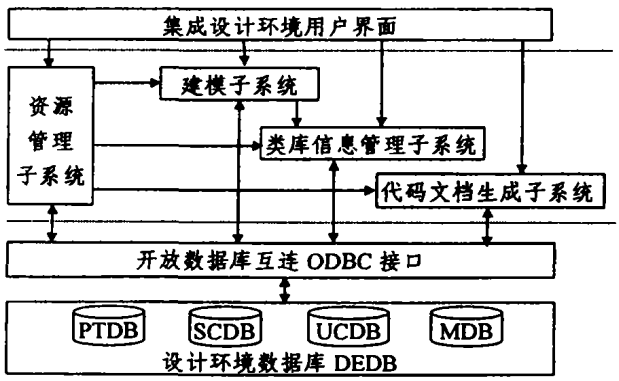

图1　JVOOPSS 体系机构

JVOOPSS 整个系统由集成设计环境用户界面、建模子系统、类库信息管理子系统、代码文档生成子系统、资源管理子系统、设计环境数据库共六个部分组成。整个系统分为控制

*)国家863项目、国家自然科学基金项目、汕头大学"211"工程项目。刘建宾　博士生，副教授，硕士生导师，主要从事软件方法与支撑环境的研究工作。郝克刚　教授，博导，主要研究方向为：组件技术，面向对象技术，电子商务等。

界面、功能模块与信息表示三个层次。

系统的最上层为集成设计环境用户界面，它是用户和系统控制中心进行交互的接口。通过该接口，用户可调用功能层中四个模块的所有功能，完成类图的绘制、类属性和方法的定义、包属性的定义、抽象逻辑图编辑、直至代码文档生成等各项程序设计任务。

系统的中间层即功能层由四个功能模块构成。建模子系统包括 UML 类图编辑器和抽象逻辑图编辑器，它们分别用于支持类间关系静态建模和类方法的过程静态建模。用户在类图编辑过程中可调用类库信息管理子系统中的类信息编辑器和包信息编辑器对类图中的类和包的描述信息进行定义和维护。类信息编辑器和包信息编辑器构成类库信息管理子系统，负责管理系统和用户设计的类和包的描述信息。代码文档生成子系统由代码生成器和文档生成器组成。代码生成器负责类图框架代码、类代码、方法代码和包代码的生成。文档生成器完成项目、类和包的信息描述文档生成，以及类图和类方法的三层抽象逻辑图的图形文档生成。资源管理子系统包括系统类库展示和项目管理。系统类库展示主要是为了方便用户查找和使用 JDK 1.3的系统包和类。项目管理主要实现项目信息管理和项目构成要素即类图、类、类属性、类方法及包的树形结构展示，实现对这些项目构成要素进行信息维护、图形编辑、代码生成和文档生成的工具调用和控制。

系统的最底层为信息表示层，即设计环境数据库。它是整个系统的基础和核心，所有工具均通过 ODBC 来存取环境数据库中的相关信息，实现设计资源的存储表示和共享。设计环境数据库中包含项目数据库 PTDB、系统类库 SCDB、用户类库 UCDB、模型数据库 MDB。

2 功能模块

2.1 建模子系统

建模子系统由支持 UML 类图建模和抽象逻辑图过程建模的两个图形编辑器组成。类图编辑器主要完成类框、包框及文档注释框等类图要素的绘制及基本操作，同时支持类间关系的表示。抽象逻辑图编辑器支持类方法的概念、逻辑、实现三层的过程建模与表示，提供三层表示的有效性和一致性支持。

2.1.1 UML 类图编辑器　在 UML 类图编辑器中我们将每一个类图要素与其操作封装为一个功能要素并作为一个控件来处理。从面向对象的角度来说，就是将抽象的要素与其操作作为类来进行设计。这样的设计对软件的重用与扩充较为有利。UML 类图编辑器提供类图要素编辑及类间关系编辑两方面的功能：

1)类图要素编辑。类图要素包括类框、包框、接口框、注释框。对各要素框图提供基本框架的绘制功能。对类要素还提供属性和方法的绘制功能。在各要素框图上提供的操作包括移动、增加、删除、复制、剪切及粘贴。在类图要素编辑过程中可随时调用信息管理子系统中的信息编辑器对各属性单进行编辑。

2)类间关系的编辑。分为类间关系的表示和类间关系的操作两部分功能。类间关系的表示是完成 UML 中类间关系语义的表示。在设计中我们严格遵守 UML 对关系表示的定义，同时定义了实现关系来表示类与类的对象间的实现关系，以此来建立类与其实例对象之间的关系。类图编辑器支持的类间关系有继承关系、关联关系、依赖关系、实现关系以及接口关系。类间关系的操作包括：类间关系的建立、类间关系的撤消。

2.1.2 抽象逻辑图编辑器　是一个全屏幕的语法制导的图形编辑器，实现类方法的过程建模，不仅提供了过程建模的各种编辑功能，而且提供了抽象逻辑图有效性与一致性的自动维护功能。整个编辑器分为结构树的编辑和节点编辑两部分。

对结构树的编辑提供了基本操作、编辑操作和语义操作。基本操作包括节点的添加、删除和修改。添加节点是指在树中插入一个指定节点类型的新节点，插入位置可以是某一存在节点的儿子、兄弟或兄长。删除节点即子树删除，是指删除树中一个已存在的节点和其下的所有子孙节点。修改节点是指改变树中一个已存在节点的节点类型或语义描述信息。编辑操作包括节点及子树的复制、剪切、粘贴，以及操作的撤消/重做功能。复制与粘贴功能的含义较为丰富，具体分为复制节点、复制子树，粘贴到当前、粘贴为儿子、粘贴为兄弟、粘贴为兄长等。撤消/重做功能则可以让用户撤消或重复某些操作。语义操作包括语义聚集和语义消解操作。语义聚集是指将一组为完成某种特定功能而逻辑上联系紧密的一组兄弟结点提取形成一个新的顺序结点，使这组兄弟节点成为新节点的孩节点，并用新节点取代这组兄弟节点在树中的原有位置。语义消解是指将结构树中没有必要存在的顺序节点删除，把删除节点的所有孩节点上提一级并取代位置。根据抽象逻辑图理论的划分，我们将抽象逻辑图的结构树分为概念、逻辑和实现三层视图。用户可以自由选择其中的一层或若干层视图进行查看，可以在任何一个视图上进行节点的添加、修改和删除，系统根据抽象逻辑图语法和三层视图间的映射与逆映射规则自动对其它层的结构和节点进行添加和调整，自动维护各层视图的有效性及不同视图间的一致性。

节点的编辑负责对三层视图节点的信息进行定义和维护。为了方便用户使用，我们把同一节点的三层视图映象即概念节点、逻辑节点、实现节点放在同一个节点编辑器中进行维护，提供了节点内容的修改、节点语义传送、概念和逻辑节点类型间的映射与逆映射、实现节点输入表达式语法检查等功能。节点内容的修改即对节点类型和节点语义的修改。节点语义传送实现三层节点语义的相互传送。概念和逻辑节点类型间的映射与逆映射功能是指如果用户修改了节点在某一视图上的类型，节点编辑器会根据节点映射与逆映射规则，使该节点在其它视图中的节点类型自动做相应改变，通过计算机的辅助，实现映射与逆映射过程的自动化。在实现节点上输入表达式时，节点编辑器根据 JAVA 语言语法自动检验表达式的合法性。

2.2 类库信息管理子系统

类库信息管理子系统以数据库的形式存放了系统类库、用户自定义类库的描述信息，统一进行管理。首先，它完成二个 JAVA 程序构件—类和包的信息表示、存取和存储管理工作；其次，它提供了方便的信息编辑工具来定义和修改类和包的描述信息，用户只需根据提示输入必要的信息，进行一些选择，就可定义或修改类和包的各种描述信息，添加、删除或修改类的属性和方法，添加、删除包中的类，而无须编写任何代码；再次，它对保证类和包描述信息的有效性、一致性起到重要作用。类库信息管理子系统通过类信息编辑器和包信息编辑器实现其功能。

2.2.1 类信息编辑器　类是 JAVA 语言中最基本也是

最重要的程序构件。类信息编辑器负责对出现在类图中的JAVA类和接口的描述信息进行管理。类的描述信息由类的常规信息、类的属性信息、类的方法信息以及类的注释说明组成。类的常规信息有类名、可见性、是否接口以及约束条件等。类的属性信息包括属性名、可见性、类型、默认值、约束性以及对该属性的一些注释说明。类的方法信息有方法名、返回值类型、可见性、约束条件、参数列表以及对该方法的说明文档。

2.2.2 包信息编辑器 包是一种松散的类的集合。包的引入能更好地管理类并提高运行效率。包信息编辑器负责对出现在类图中的JAVA包的描述信息进行管理。包的描述信息较为简单，由包名、包的组成类列表以及包的注释说明组成。在定义包的类成员时，我们规定只能挑选已出现在类图中的类作为包的成员。这样做利于程序开发过程的规范化，只有在完成类定义的基础上，才能定义包。同时有利于保持类图信息和类库信息的一致性。

2.3 代码文档生成子系统

软件是代码与文档的集合。代码与文档是整个支撑系统产生的最终结果。生成子系统的功能是读取建模子系统和信息管理子系统所产生的设计信息资源，把设计环境数据库DEDB中存储的设计信息内部表示形式变换为JAVA源程序代码，并产生相应的规格化文档。代码生成器负责代码的生成，文档的产生则由文档生成器完成。

2.3.1 代码生成器 以建模子系统和信息管理子系统产生的设计模型和信息为基础，根据类图模型、类的定义、类方法的抽象逻辑图过程模型、包的定义以及JAVA语言语法自动生成JAVA源程序代码，并且自动完成格式的缩进、添加注释等工作。生成的代码可以随时查看、复制和保存。代码生成器可产生类图模型的框架代码、类方法的过程代码、完整的类代码，完整的包代码。类图模型的框架代码是类图中所有类的框架代码集合。每个类的框架代码包括类属性代码及类间关系表示代码。代码生成器首先根据类图到JAVA程序框架代码的映射规则完成类间关系部分的代码生成，然后再根据类属性信息表生成所有类属性的代码，这样就得到类的框架代码。类方法的过程代码根据抽象逻辑图的实现层模型直接生成得到。将类所有方法的过程代码填充到类框架代码中就可得到完整的类代码。将组成包的所有类代码合并就可得到完整的包代码。

2.3.2 文档生成器 其工作方式和功能与代码生成器相类似，只是它所生成的结果不同。文档生成器可生成项目文档、类文档和包文档。项目文档有项目概要表、类图、项目类成员列表、项目包成员列表。类文档包括类概要说明表、类属性列表、类方法列表、类方法的三层抽象逻辑图。包文档有类概要说明表和包成员类列表。对所有文档都支持存档与打印输出功能。类图文档能以多种图形格式输出。抽象逻辑图文档以文本形式输出。为了方便用户，在生成抽象逻辑图文档时还加入对间隔、连线粗细、字体的一些控制，用户可以进行一定的调整后再保存或打印输出，还可以选择另存或直接送至剪切板。

2.4 资源管理子系统

设计资源由系统类库资源和用户的设计资源构成。系统类库中存放了JDK1.3的包和类的信息。用户设计资源包括用户设计的类图、类与包的信息描述、类方法的三层抽象逻辑图，以及生成的各种代码和文档。用户设计资源是按项目来组织的，因此资源管理器可划分为系统类库的展示和用户项目管理两部分。在资源管理器中，用户可以方便地实现对系统类库内容的查找以及对用户项目构成要素的操作。资源管理器通过与建模子系统、类库信息管理子系统、代码文档生成子系统中的各种工具之间的操作互动实现设计资源的管理。

在资源管理子系统中引入系统类库，是为方便用户查找系统包、每个系统包中的类、每个系统类中的属性与方法。系统类库的展示是抽取系统类库的内容并以树形结构展示出来。系统类库的展示分为目录的展开和系统类属性与方法的展示。目录的展开完成系统包和类的展示。系统类属性与方法的展示是对每一个系统类抽取其属性、操作及可见性描述，并在类一级展示给用户。对类中属性与方法的可见性以图标的方式表现。

用户项目管理包括项目构成要素展示和项目构成要素操作。项目构成要素展示是将用户设计的项目、类图、类、类属性、类方法、包、包中类等项目构成要素以及它们之间的构成关系以树型结构表现出来。项目管理器对项目中的各种构成要素提供了文件操作、信息维护操作、图形建模操作以及代码和文档的生成操作。文件操作有打开、关闭、保存、另存为、刷新等。信息维护操作包括增加、删除、修改。这些操作能否被执行与项目构成要素相关。对项目要素允许执行文件操作、信息维护操作以及文档生成操作。对类图可执行图形建模、代码生成和文档生成操作。对类要素和包要素可执行信息修改、代码生成和文档生成操作。对类属性要素可执行信息维护操作。对类方法要素可执行信息维护、图形建模、代码生成和文档生成操作。对包中类要素允许增加、删除操作。考虑到类图建模信息和类库信息的一致性，我们规定不允许在项目管理器中对类要素和包要素执行增加和删除操作，而只能在类图中进行可视化的增加和删除。这些操作中，除了项目管理需要的文件操作和项目信息维护操作是在资源管理器中实现外，其它的操作是通过调用建模子系统、类库信息管理子系统以及代码生成子系统中的工具来实现。

3 设计环境数据库

设计环境数据库实现设计资源的存储表示和信息共享。设计环境数据库中存放了项目信息、系统类库信息、用户类库信息、类图模型和抽象逻辑图建模信息。

3.1 项目数据库 PTDB

项目数据库由一张项目信息表组成。表中记录了项目名、版本号、版权、开发者、开发单位、项目存储路径、项目说明等信息。

3.2 系统类库 SCDB

系统类库用于保存JAVA语言自带的系统类的信息。库中有三张表：保存包信息的包信息表；保存类及接口信息的类信息表；保存类方法及属性信息的类成员信息表。包信息表包括序列码和包名字段。类信息表有序列码、类名、包名、基类名、访问控制符、修饰符、接口标志字段。类成员信息表有方法属性名、值类型、类名、访问控制符、修饰符、初始值、参数、类别字段。

3.3 用户类库 UCDB

用户类库用于保存用户设计的JAVA类和包的有关描述信息。库中有四张表：包信息表由包名、代码和注释字段组成。类表有类名、包名、访问控制符、修饰符、接口类别标志、代码和注释字段。类属性表由属性名、类名、值类型、访问控制符、修饰符、初始值和注释字段组成。类属性表由方法标识、方

法名、类名、返回值类型、访问控制符、修饰符、参数、代码和注释字段组成。

3.4 模型数据库 MDB

模型数据库存放类图模型和抽象逻辑图过程模型的建模数据。类图模型由类图表、类图要素表和要素关系表来描述。抽象逻辑图过程模型由节点信息表描述。类图表由类图说明和类图的框架代码两个字段组成。类图要素表由要素标识、要素名、位置坐标、要素类型四个字段组成。要素关系表由关系标识、关系类型、起始要素标识、结束要素标识、方向标识五个字段组成。抽象逻辑图节点信息表由节点标识、类名、方法标识、概念类型、概念语义、逻辑类型、逻辑语义、表达式和备注字段组成。

4 用户界面

我们从规范性、易操作性、健状性方面的要求设计了如图2所示的集成开发环境用户界面。集成开发环境的用户界面包括菜单栏、快捷工具栏、资源管理工作区、建模要素符号栏、图形编辑工作区和状态栏六个部分。

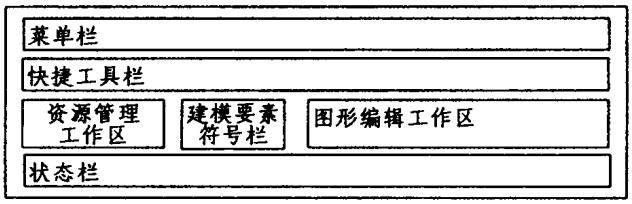

图2 集成开发环境用户界面

菜单栏设置了文件、编辑、视图、工具及帮助五个一级菜单。文件菜单有新建、打开存储、存储为、导出类型、打印及退出。编辑菜单包括拷贝、剪切、粘贴、删除、选择、刷新。在视图菜单下可对集成界面的显示内容和方式进行动态配置。工具菜单包括信息编辑、图形建模、代码生成、文档生成和选项。选项内容为路径选择、代码生成选择、项目选择的内容。在快捷工具栏上放置了一些基本操作的工具图标，包括新建、打开、保存、打印、复制、剪切、粘贴、刷新、代码、文档。在资源管理工作区中采用树形结构对系统类库和用户类库中的设计资源进行展示。在建模要素符号栏中放入类图或抽象逻辑图的建模要素，它根据不同的编辑内容自动调整建模符号。在图形编辑工作区中可对类图和抽象逻辑图进行编辑，并根据不同的编辑内容自动调入相应的图形编辑工具。对类图建模，载入类图编辑器；对类方法的过程建模则载入抽象逻辑图编辑器。抽象逻辑图编辑器将编辑内容分为概念、逻辑和实现三层视图，每一层视图都以树形方式组织，可同时对三层视图或单独对三层视图中的任一层进行编辑。编辑方式可在视图菜单中进行配置。系统提供的刷新操作可保证资源管理工作区的显示内容与图形编辑工作区的类图模型保持一致。

为方便使用，对资源管理工作区和图形编辑工作区中的所有编辑元素都提供右键快捷菜单，并且实现了选择对象与可执行操作的动态关联，对不可用的操作选项自动进行屏蔽。

结束语 我们以 UML 类图和抽象逻辑结构图为基础设计的 JVOOPSS 系统支持从类图建模、类方法的过程建模到完整 JAVA 程序代码生成的正向工程，实现了 UML 类图与抽象逻辑结构图的有机结合，实现了可视化建模与编码的统一，可用于 JAVA 应用程序和 JAVA 小程序的可视化开发。我们的系统与其它同类系统相比具有以下特点：

1)用抽象逻辑图填补了 UML 模型中缺乏的过程静态建模部分，将静态建模的级别细化到类方法一级，使系统的静态建模机制较为完整，因而系统能够根据设计模型生成完整的 JAVA 程序源代码。其它系统大多只能根据类图模型生成类的框架代码，对类的方法代码主要靠设计者使用传统方法进行设计与编码。

2)系统支持从类体系结构设计到类方法详细设计的 JAVA 面向对象可视化程序设计全过程，支持从类图建模、类的定义、类方法过程建模到 JAVA 程序源代码生成的平滑过渡。

3)系统以支持 JAVA 可视化程序设计为目标，不涉及复杂的动态建模部分，因而简洁实用，操作简单。

我们已成功应用 JVOOPSS 开发出能在网上运行的 JAVA 游戏程序，证实了系统的可用性。实践表明该系统对提高 JAVA 程序设计的效率，明显改善程序的可理解性和可维护性是有效的。

参考文献

1 Rumbaugh J, Jacobson I, Booch G. The Unified Modeling Language Reference Manual. Addison-Wesley, 1999
2 Booch G, Rumbaugh J, Jacobson I. The Unified Modeling Language User Guide. Addison-Wesley, 1999
3 刘建宾，龚世生. 抽象逻辑结构图及其应用. 计算机科学，1996，23(6)：83～86
4 刘建宾. JAVA 过程蓝图. 计算机科学，2000，27(7)：87～91
5 刘建宾，郝克刚，龚世生. 抽象概念结构图到 JAVA 过程蓝图的平滑过渡及一致性. 计算机科学，2001，28(8)

(上接第49页)

结论 NIST 之所以选择了 Rijndael 算法，是由于 Rijndael 是集高安全性、高性能、高效率、易实现及伸缩性强于一身的[4]。特别地，Rijndael 能通过广泛的计算环境，不管是用反馈模式还是非反馈模式，使得硬件和软件性能达到最优。密钥建立高效且灵活。Rijndael 很低的存储要求，使之能在有空间限制的环境中执行，Rijndael 能很好地抵抗各种攻击。另外，密钥和块长都设计得很灵活，同时算法在轮数选择方面能提供一定的灵活性，最后，Rijndael 轮运算对指令级并行很具潜力。

但是，加密的逆运算有一些限制。在智能卡上解密过程相对加密过程难于执行，需要更多的代码和循环。对于软件，加密和解密需要不同的代码和查找表。对于用硬件来实施加密和解密，解密只能部分地重用加密过程用到的电路。

参考文献

1 Daemen J, Rijmen V. The Rijndael Block Cipher Version 2. 03/09/99
2 American NIST, Draft of FIFS-AES. 02/28/01
3 冯登国，裴定一. 密码学导引. 科学出版社，1999
4 http://www.nist.gov/aes