基于软件体系结构的可视化软件开发平台*)

胡 华[1] 林昌东[2]

(杭州商学院计算机与信息工程系 杭州310012)[1] (浙江大学出版社 杭州310027)[2]

Platform for Visualization Software Based on Software Architecture

Hu Hua[1] LIN Chang-Dong[2]

(Dept. of Computer & Information Engineering, HangZhou Institute of Commerce, Hangzhou 310012)[1]

(Publishing Company of Zhejiang University, Hangzhou 310027)[2]

Abstract Platform for visualization software is a kind of large and complex software system, the work to design and evolve platform for visualization software is also a complex work. This paper proposes using software architecture to analyze and evolve a large visualization platform based plane data to a visualization platform with ability of processing both plane and volume. The result of this paper proves the reasonability of our method.

Keywords Software architecture, Visualization, Software platform, Data independent

1. 引言

随着软件工程研究和实践的进一步发展与深入,软件系统的设计开发工作者们已经提出并成功地实现了许多有效的面向大型软件系统的软件设计开发方案与方法[1~7]。尽管现有的许多软件设计开发方案与方法,能够有效地支持大型软件系统的实践并提高开发效率,但是,在一个成功的大型软件系统的整个生命周期中,系统维护和系统进化往往占据着比系统设计和系统开发阶段更为显著的工作比例,因此,在软件系统开发的理论和实践中,系统维护和系统进化的成功实施往往成为评价一项大型软件系统成功实现的主要标志。

可视化软件开发平台是一种功能和结构都十分复杂的大型软件支持系统,可视化软件开发平台的设计、开发与进化也是一项极其复杂的工作[13~17]。根据系统技术与环境的发展与变化对已有的成功软件可视化应用支持平台进行维护与进化是提高可视化软件开发与使用效率的关键。我们根据软件工程研究的最新结果,通过采用改进的软件体系结构的系统分析与设计方法,对大型可视化软件开发平台进行了进化分析与设计,在较好地保持系统的原有特性的基础之上,有效地将一个原本只支持面模型处理应用开发的大型可视化软件设计开发支持平台进化扩充成为一个能够同时支持面模型和体模型应用开发的综合模型支持系统。

2. 面向系统进化的软件体系结构

基于构件和连接件(CC--Component and Connector)的有向图是传统软件体系结构方法经常采用的一种描述和分析软件系统整体抽象组成结构的分析工具[7~12]。虽然采用传统的构件和连接件的软件体系结构方法在软件系统的设计和开发过程中能够有效帮助软件人员进行系统整体结构的分析与设计,但是,对于需要进行系统结构和系统功能语义进化与重用处理的系统进化而言,传统构件与连接件的描述处理机制难以满足系统体系结构进化时的语义处理需求。在这种情况下,我们根据软件系统进化中的系统语义特点,借鉴面向对象技术中的继承性技术,对传统软件体系结构中的构件和连接件机制进行了语义继承表达和分析处理上的扩充,经过语义继承改进的软件体系结构的扩充的构件与连接件(ECC-Extended Component and Connector)机制不仅保持了传统软件体系结构的系统整体性结构表达与分析的特点,而且能够有效地分析、表达与处理软件系统进化中的语义继承情况。

定义1(接口) 为软件系统中构件与连接件相连时构件部分的交互单元。接口可以用(ID, Type, State, Partner, Next)形式的元组表示。其中,ID 是接口在系统中的唯一标识;Type 的取值为 Sinput(单数据输入)、Soutput(单数据输出)、Minput(变长数据输入)和 Moutput(变长数据输出)之一;State 的取值为 Enable 和 Disable 之一;Partner 的取值为与其交互的连接件中的交互部件的 ID 或 NULL(空值);Next 的取值可以为 NULL 或属于同一构件的其它接口的 ID。

定义2(构件) 是软件系统中担当处理与计算的单元或实体的抽象。构件可用(ID, Dic, RefID, InPtr, OutPtr)形式的元组表示。其中,ID 是构件在系统中的唯一标识;Dic 为构件标识符字典;RefID 的取值可以为 NULL 或另一构件的 ID(此时表明当前构件是一个从其他构件继承来的进化构件);InPtr 为构件输入接口队列首接口的 ID;OutPtr 为构件输出接口队列首接口的 ID。

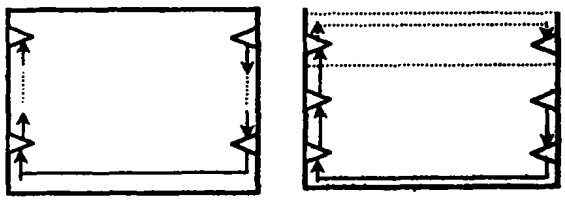

图1 构件示意图(右图中的虚线为构件的继承部分)

定义3(角色) 为软件系统中构件与连接件相连时连接件部分的交互单元,它可以用(ID, Type, State, Partner, Next)形式的元组表示。其中,ID 是角色在系统中的唯一标识;Type 的取值为 Sinput(单数据输入)、Soutput(单数据输出)、Minput(变长数据输入)和 Moutput(变长数据输出)之

*)本课题得到国家教育部出国留学人员启动基金,浙江省教育厅科技计划项目资助。

一;State 的取值为 Enable 和 Disable 之一;Partner 的取值为与其交互的构件中的接口的 ID 或 NULL(空值);Next 的取值可以为 NULL 或属于同一连接件的其它角色的 ID。

定义4(连接件) 是软件系统中担当构件间协调处理的协议单元或处理实体的抽象。连接件可用(ID,Dic,RefID,InPtr,OutPtr)形式的元组表示。其中,ID 是连接件在系统中的唯一标识;Dic 为连接件标识符字典;RefID 的取值可以为 NULL 或另一连接件的 ID(此时表明当前连接件是一个从其他连接件继承来的进化连接件);InPtr 为连接件输入接口队列首接口的 ID;OutPtr 为连接件输出接口队列首接口的 ID。

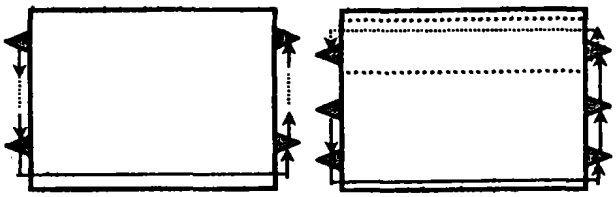

图2 连接件示意图(右图中的虚线为连接件的继承部分)

定义5 定义2和定义4中的继承是单重的半序继承。即,若实体 ID 从 REFID 继承而来,则 REFID≤ID 且 REFID 是 ID 的唯一直接前趋。

若用 S 表示实体集合,$y=next(v)$表示 y 是 v 的直接继承实体,$x=pri(v)$表示 x 是 v 的直接继承实体,则定义5可以用一阶谓词表示如下:

$\forall (x,y\in S)(x=next(y))\rightarrow((y\leqslant x)\ \&\&\ (y=pri(x)))\ \&\&(\forall (z\in S)(z\neq y)\rightarrow(z\neq pri(x)))$。

显然,定义5是检验构件或连接件继承和软件体系结构进化是否合理的一个重要规则。

3. 基于软件体系结构的系统进化

我们采用扩展的软件体系结构机制对可视化软件开发平台 Open Inventor™[15]进行了分析设计与进化扩充。Open Inventor 是 SGI™公司使用 C++语言开发的一个新一代面向对象的可视化应用开发支持平台。采用 Open Inventor 作为可视化应用的开发支持平台,应用程序员不仅可以快速建立一个能够动态生成和描述三维的可视化场景的可视化应用程序,而且该应用程序还具有从 VRML 的脚本化三维数据中生成三维的可视化场景的能力。尽管 Open Inventor 是一个功能十分强大的软件开发支持平台,但是由于它不具有体数据的描述处理能力,因此,在很多需要进行体数据处理的现代可视化应用中,Open Inventor 的使用受到了极大的限制,将 Open Inventor 进化成为一个有多种数据模型支持能力的可视化软件支持平台已成为 Open Inventor 进一步走向实用的关键。

图3是我们对 Open Inventor 进行系统分析和体系结构抽象后得到的一个系统体系结构图(在图中,为了简单起见,我们将构件和连接件中的接口与角色等细节略去,将构件的图示化简为矩形框,连接件的图示化简为直线)。在图中,构件"场景数据"代表以 VRML 文本形式存储的.IV 场景描述数据文件;构件"场景"代表由可视化应用生成和处理的三维场景树(在 Open Inventor 中由 Separator,Group,Node,Field 等组成);构件"系统状态"(在 Open Inventor 中由 State,Element,DB 等组成)记录和处理系统中对场景树的变换处理和渲染情况;构件"管理控制"(在 Open Inventor 中由 Engine,List,Action 等组成)在协调系统状态的记录的同时,还管理与触发场景的生成与渲染;构件"渲染处理"(在 Open Inventor 中由 Window,RenderArea,GLRender,Render 等组成)通过对场景树的周游遍历,按场景树节点的要求对场景进行显示处理。最后对于连接件,除构件"场景数据"与构件"场景树"之间的连接件由 Input 通过系统调用的文件 I/O 实现外,其他构件之间的连接件通过操作系统和 C++提供的事件处理与过程调用机制实现。

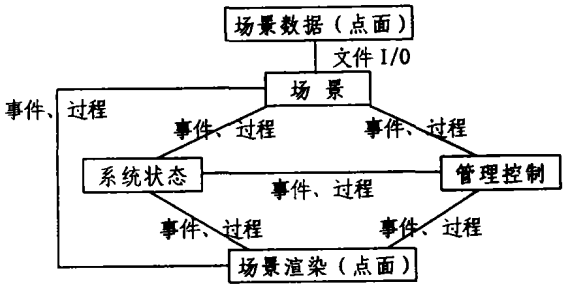

图3 可视化软件开发平台的软件体系结构图

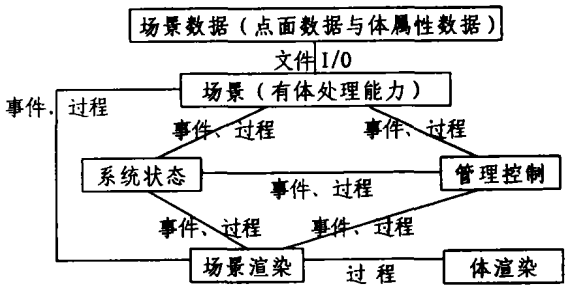

图4 进化扩充后的可视化软件开发平台的软件体系结构图

根据图3的分析结果和体数据模型的处理需要,我们对 Open Inventor 进行了如图4所示的系统进化。首先,要使得 Open Inventor 具有体数据的场景描述与生成能力,我们必须对"场景数据"构件(.IV 文件)进行了包容能力的进化与扩充,进化与扩充后的"场景数据"构件能够通过 Input 连接件生成具有体处理能力的场景树;其次,除了场景描述与生成以外,我们还对"场景渲染"构件进行了体数据模型能力的进化与扩充,使之能够同时渲染处理点面实体和体实体对象。在图4中,从原来的"点面渲染"构件进化而来的"综合处理构件"通过采用"过程调用"的方式使用实际的体渲染构件进行体实体的渲染处理。

最后,通过对图3和图4进行分析比较,我们不难发现,如果忽略构件功能进化的细节并将图4中体渲染和场景渲染构件合并,则图4在抽象的拓扑结构上与图3是相同的,这个现象说明了我们的系统进化没有破坏原来系统的体系整体性,系统的进化处理是合理协调的。

4. 系统实现

我们根据上一节的系统体系分析结果,对 Open Inventor 进行了实际的进化处理。

首先,在场景数据标记处理方面,我们对标准的.IV 文件进行了进化扩充,使之能够在文件结构没有本质性改动的情况下具有点面模型数据和体数据的综合包容能力。以下是一个扩充了数据包容能力的.IV 文件片段(片段的最后一部分是一个椅子的简单体数据结点):

```
Separator {
    DEF leftTransform Rotation {
        rotation 1.0 0.0 0.0 0.0
```

```
    }
    PointLight {
        color 0.0 0.0 0.0
        location 5.0 5.0 5.0
        intensity 0.8
    }
    Volume {
        depth 69 width 69 height 129 fname "e:\volume.data\
        chair2.67x127x67.raw"
    }
}
```

其次，在场景生成处理部分，我们采用如下的两个扩充算法，使得可视化应用能够不加区分地生成包含了混合模型数据的场景树。

算法1

输入：包含了混合模型数据的.IV 文件(场景标记文件)

输出：混合模型数据的场景树

①初始并生成一个 Input 部件对象 in；
②生成类型为分隔组的一个场景树结点 root；
③以 root 为根，in 为读入构件，反复执行算法二直到生成场景树；

算法2

输入：分隔组结点 root 和读入构件 in

输出：root 下的混合模型数据的场景树

```
①if (读入文件结束或当前层结束) 返回，else 执行 ②；
②Classname=Node::read(in,classname,root);
③Switch (Classname) {
    Case Separator, Group:
        R=Node::createInstance(in,Classname); root←
        R; 递归调用本算法;
    Case 一般点面结点(包含光源、相机和空间变换等):
        R = Node :: createInstance ( in, Classname );
        R.readInstance(in); root.addchild(R); 跳转到
        ①;
    Case 体结点:
        R = Node :: createInstance ( in, Classname );
        R.readattribute(in); root.addchild(R); 跳转到
        ①;
};
```

第三，对于场景的渲染准备处理，我们采用如下的算法，使得可视化应用能够将混合模型数据的场景树与一个场景渲染窗口关联并遍历该场景树。

算法3

输入：混合模型数据的场景树

输出：显示和处理输入场景树的窗口

①调用 Win::init()生成一个可视化应用将要使用的窗口 curWin;
②创建一个 Renderarea 实体 currender 并将 curWin 与 curRender 关联;
③调用 currender->setScence()将应用场景 curSence 与 curRender 关联;
④调用 Win::show()将应用窗口 curWin 显示出来;
⑤调用 currender->render()遍历场景树 curSence;
⑥算法结束。

最后，对于场景的遍历渲染处理，我们采用如下的扩充算法，使得可视化应用能够不加区分地遍历和渲染包含了混合模型数据的场景树。

算法4

输入：场景树的一个结点 node

输出：渲染该结点开始的场景树

```
① Switch(type(node)) {
    Case Separator, Group:
        对本结点的所有孩子递归调用本算法;
    Case 一般点面结点(包含光源、相机和空间变换等):
        普通结点渲染;
    Case 体结点:
        根据体结点的属性数据 ftype 调用不同的体结点渲
        染算法(体数据在此时实际读出);
};
```

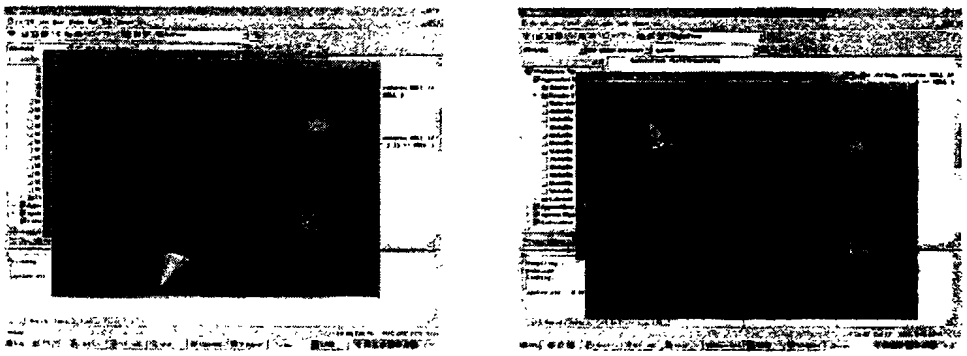

图5　在进化扩充后的可视化平台上开发的可视化应用之一(椅子为体数据，其余为面模型)

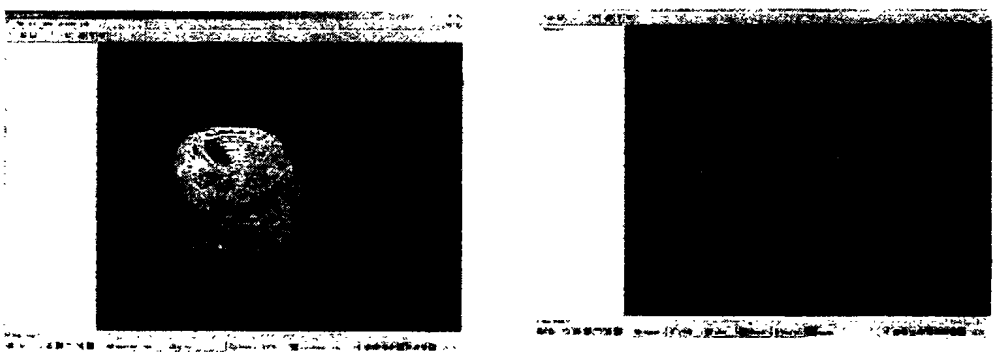

图6　在进化扩充后的可视化平台上开发的可视化应用之二(体数据)

图5和图6是我们采用扩充进化后的开发平台构建的两个可视化应用的运行结果。在图5显示的场景中，中间的椅子是一个体数据实体，其他的对象为简单的点面模型实体；而在图6的场景之中，有关的体数据都来源于同一个体数据源，通过.IV 文件的包装与标记，在场景中形成了不同的对象实体。此外，由于图5中的第一个应用的显示界面主要建立在 Open GL 的基础之上，因此，该应用可以不加修改地运行于 Mi-

(下转封四)

(上接第 115 页)

crosoft Windows 98/NT 平台和 X-Windows 平台下;而图 6 中的第二个应用的显示界面主要建立在 Microsoft Win 32 与 Open GL 的基础之上,因此,第二个应用主要运行于 Microsoft Window 98/NT 平台下。但是,这两个应用使用的场景数据是与平台应用无关的 .IV 标记脚本数据描述文件,因此,这两个应用的场景管理部分,是与操作系统平台独立的。对于更多的系统运行应用结果,有兴趣的读者可以通过访问 http://softlab.hzic.edu.cn/english/vplatform.asp 获得。

最后,为了保证进化处理后的可视化软件开发平台的操作系统兼容性并具有自主的开发版权,我们进化扩充的软件平台基础是 Open Inventor 的一个共享版本 Apprentice。

5. 相关工作

Volvis[16]是 Ari Kaufman 领导的可视化工作小组使用模块化技术在 C 语言基础上开发实现的一个体模型数据的处理与应用的支持环境,Volvis 目前只支持直接来源于数据文件的单一的体模型数据处理。Open Inventor[17]是 SGI™公司主导使用 C++语言开发的新一代面向对象的可视化应用开发支持平台。Open Inventor 的一个主要缺陷是不支持体模型数据的处理与应用。AVS/Express[14]是 AVL 公司设计开发的多维高级可视化系统,该系统不仅具有处理文本和二进制等图形数据的一般能力,而且还具有对传统关系数据库进行存储和查询的能力。IDL[15]是 RSI 公司设计开发的基于交互式数据语言的可视化系统,该系统采用矩阵和面向对象处理技术来支持通用文本和图像数据。与上述的工作相比较,本文的工作具有如下的特点与创新:

1)采用了扩充的软件体系结构的系统设计进化方法,使得系统进化扩充的结果能够对原有的工作基础实现有效的兼容;

2)通过语义继承方式实现的对脚本化数据标记文件(.IV)的体模型数据包容能力的进化扩充,有效地保持了原有系统的核心代码与具体数据处理的分离特性;

3)通过语义继承方式实现的开发平台体模型的进化扩充与处理,使得基于开发平台的可视化应用能够不加区分地处理体模型和点面模型数据的综合数据场景。

结论 有关的应用和实践表明,我们提出的采用软件体系结构的方法来实现可视化软件开发平台的系统分析与进化处理不仅能够有效地实现系统处理功能的扩展与进化,而且对于原有系统的体系结构也能够有效地保持与兼容。最后,本文提出的系统分析进化方法,可以进一步推广应用到其他的大型软件系统的分析与进化处理应用中。

参考文献

1 Mills H D. Management of software engineering. Part I: principles of software engineering. IBM System Journal, 1999, 38(2)

2 Shu N C. Visual programming: perspectives and approaches. IBM System Journal, 1999, 38(2)

3 Alexander C. The Timeless Way of Building. Oxford University Press, New York, 1979

4 周之英. 现代软件工程(下):新技术篇. 科学出版社, 2000

5 冯玉琳, 赵保华. 软件工程----方法、工具、实践(第二版). 中国科技大学出版社, 1992

6 王立福, 张世琨, 朱冰. 软件工程----技术、方法与环境. 北京大学出版社, 1997

7 Garlan D, Shaw M. An troduction to software architecture: [CMU software Engineering Institute Technical Report]. Jan. 1994

8 Garlan D, Perry D E. Introduction to special issue on software architecture. IEEE trans. On SE, 1995, 21(4)

9 朱冰, 梅宏. 软件体系结构的基础研究. 计算机科学, 1994. 4

10 车敦仁, 周立柱, 蒋维杜. 软件体系结构、应用平台及框架仓库技术. 计算机研究与发展, 1996. 7

11 胡华, 高济, 何志均. 基于软件体系结构的软件设计及构造. 计算机科学, 1999, 8

12 胡华, 高济, 何志均. 开放式软件体系结构的软件描述语言设计. 小型微型计算机系统, 2000, 2

13 黄志澄. 给数据以形象、给信息以智能----数据可视化技术及其应用展望. http://www.visualsky.com/viztxt.htm

14 陈佳胤, 曾远金. AVS/Express----世界领先的可视化产品. http://www.jig.com.cn/yingyong/index.htm

15 李静. IDL----科学数据可视化的理想工具. http://www.jig.com.cn/yingyong/keshi/2001-2/2-1.htm

16 Volvis Group. Volvis. http://www.cs.sunysb.edu/~vislab/start_volvis.htm

17 SGI. Open Inventor. http://www.sgi.com/software/inventor

计算机科学

(1974 年 1 月创刊)
第 30 卷第 4 期 (月刊)
2003 年 4 月 25 日出版

ISSN 1002-137X
CN50—1075/TP

定价: 20.00 元 国外定价: 5 美元
邮发代号: 78—68
发行范围: 国内外公开

主管单位: 国家科学技术部
主办单位: 国家科技部西南信息中心
编辑出版: 《计算机科学》杂志社
重庆市渝中区胜利路 132 号 邮政编码: 400013
电话: (023) 63500828 E-mail: jsjkx@swic.ac.cn
社 长: 牟炳林
主 编: 朱宗元
印刷者: 重庆科情印务有限公司
总发行处: 重庆市邮政局
订购处: 全国各地邮政局
国外总发行: 中国国际图书贸易总公司 (北京399 信箱)
国外代号: 6210—MO