生成器技术与软件重用

田振军* 夏宽理 喻 勇
复旦大学计算机科学系 (上海 200433)

摘 要 软件工程中一个重要的问题就是如何经济地构造一个大型的软件系统。特定领域软件生成器是解决这个问题的有效工具。领域模型解释了如何从构件库组装系统,而生成器正是对它的实现。该文较为详细地介绍了特定领域软件生成器,并且与其它的软件重用方法进行了比较。文章最后简要介绍了 GenVoca 生成器,并结合一个数据结构领域的实例以及一个销售 /仓库系统进行了进一步的说明。

关键词 构件 域 领域模型

Generator Technology and Software Reuse

Tian Zhenjun Xia Kuanli Yu Yong
(Department of Computer Science , Fudan University, Shanghai 200433)

Abstract: A key problem in software engineering is building complex software systems economically. Domain-specific software system generators are valid tools in solving this problem. Domain modals explain how systems can be assembled from component library, and generators are their realizations. This paper introduces in detail domain-specific software system generator, which is compared with other reuse methods. At last, this paper introduces GenVoca generator, and interprets it further through a concrete example of data structure domain and a sale/store system.

Keywords: Component, Ream, Domain modal

1 背景

大型软件系统的开发既困难又费时，通过软件重用可降低软件的开发费用，因而软件重用是软件工程界长期追求的目标。

函数库提供了一种重用方法,例如:C 函数库。但函数库并不能彻底解决软件重用的问题。如果库中的每个模块实现简单的功能，则构造一个大型的系统就需要成百上千的模块,选择,定制和组合这些模块也要耗费大量的时间。如果每个模块实现复杂的功能，则它可能仅适用于少数的应用领域。通过函数库中模块的重用提高软件开发效率存在这样一个障碍:获得的效率与模块的大小成正比,而其可重用性与模块的大小成反比。并且函数库的可扩展性较差。比如一个函数库包含 n 个模块,如果要加入一种新的特征,则新的函数库包含 2n 个模块:n 个结合了新的特征,n 个没有结合新特征。每加入一个新的特征,函数库会成倍增长,无论开发者和使用者都承受不了。一种更有效的方法是引入较大的结构单元,即子系统(也称为构件)。

在面向对象技术推出后，类以其自身独有的特性 (封装,继承和多态)极大地支持了软件重用。经过多年的实践,人们发现类作为重用单元太小了。一个类很少单独完成一个完整的功能,它往往与其它类一起提供完整的功能。这些相互交织在一起的,有密切关系的类组合成构件。

构件逐渐成为人们关注的对象。每个构件实现特定领域的某个特征,定义抽象的接口而隐藏其具体的实现。构件提高了系统的生产力是容易理解的：构件是重用而不是重写。构件由领域专家编写,因此它们的质量是可以保障的。构件的可靠性高于典型的软件模块，因为它们在不同的环境下测试过。使用构件可提高系统的性能。人们逐渐认识到一个复杂的软件系统应通过构件组装的方式来构造。构造过程可看作是从一系列可重用构件组装系统的过程。一些构件组合成复杂的子系统，而它又可作为进一步组装的组件。软件系统生成器提供了构造软件系统的有效途径。

2 软件系统生成器

特定领域软件构架的研究表明，软件系统生成器提供了构造大型复杂软件系统的一种较经济的很有前途的方法。这些生成器都是针对特定领域的,它们的构造模型(称之为领域模型) 表示如何通过组合可重用的预先生成的构件构造相似的软件系统。使用生成器技术,软件系统的开发既快捷又节省费用。生成器技术的使用必须满足两个前提条件:首先,生成的软件所在的领域必须是成熟的;其次,由于生成器的构造费用高，该领域的各种系统应该是同一系统的不同的变种，并且构造其中任何一个都要花费很高的代价,那么使用生成器技术才是比较经济的。

2.1 软件生成器的概念

在讨论软件生成器的概念之前有必要先介绍构件,域和领域模型等基本概念:

构件(component):具有抽象界面和具体实现的功能单元。其具体实现对外是不可见的,可见的只是界面。

域(ream):构件的集合。这些构件使用不同的方法去实现相同的界面。构件相互之间可互换。不同的构件继承了相同的界面,同时添加新的数据成员,函数成员以及类成员。

*作者简介:田振军,男,25 岁,硕士研究生,主攻方向为软件工程。

© 1995-2004 Tsinghua Tongfang Optical Disc Co., Ltd. All rights reserved.

领域模型(domain modal):域的集合以及定义该领域中各种系统之间的组合规则。

软件系统生成器(software system generator,简称 SSG)是对领域模型的实现。它将目标系统的高层描述转换成实际的源代码。SSG 包括一个构件库和一个生成器。构件实现了特定领域的基本特征,而生成器用于组装这些构件。使用特定领域的构件,通过生成器可以很快地建立一个大型的系统。虽然不同的生成器产生的系统不同,但生成器本身是非常相似的。这些共同之处有利于简化未来生成器的开发。GenVoca 领域模型确定了当今一些有代表性的 SSG 中构件设计和组织的相似之处。GenVoca 模型的实现就是一种特定领域的 SSG。

2.2 SSG 与其它软件重用方法的比较

软件系统生成器与应用程序生成器之间有一定相似性。每个应用程序生成器都提供自己的编程语言,以及相应的解释器或编译器,例如:Lex、Yacc 以及所谓的第 4 代编程语言。应用程序生成器的优点是很明显的,它提高了软件开发效率。但是它的适用范围有限,而且不够灵活。相比之下,软件系统生成器继承了应用程序生成器的优点,并且具有其它的优点,例如:GenVoca 及 P++方便了特定领域生成器的开发;把每个特征封装成单个的构件,使得加入新特征既不需理解也不需修改其它的代码;由于不同的构件实现不同的特征,只需挑选适当的构件集合就可满足特定的要求。

生成器只是通过可重用库构造软件系统的各种方法中的一种。CORBA 也是对构件进行组合以生成应用程序。但其构件是独立设计的,并且来自不同的环境,相互间是独立的。相比之下,生成器所针对领域的构件由同一种语言生成,构件之间相互作用且可互换。

软件生成器是软件构架的一个子域。生成器将对构件组合的描述作为输入,输出的是优化过的源代码。两者不同之处是生成器组装的构件被设计成相互之间可通信且相互可交换的。而在软件构架中不存在这种限制。

3 GenVoca 生成器

SSG 将会成为软件开发的重要工具,一类重要的生成器被称为 GenVoca 生成器,其所实现的领域模型被称之为 GenVoca 模型。GenVoca 模型的产生,是通过研究不同领域的软件系统生成器中关于构件设计和组合的共同之处,并将其用形式化的方法描述出来。构件为基本的单元。同一系统中的不同构件都具有与该系统有关的标准界面,定义了其它的构件访问它的方式。构件提供的参数,用于定制构件本身的行为。如果构件参数中至少有一个与该构件属于同一个域,则称该构件为对称构件。系统为相互作用的构件的集合,本身可视为一个转换。为了支持 GenVoca 模型,P++语言对 C++进行扩展,以支持构件的封装,抽象,参数化和继承。该文通过两个不同领域的实例:数据结构和销售 /仓库系统详细介绍 GenVoca 生成器思想在实际中的应用。

3.1 模型的形式描述语言

为了形式化地描述构件、域及系统,GenVoca 模型提出一种简洁的类似文法的表示方法。通过这种方法,系统可用一个类型表达式表示,而软件的重用表现为两个或多个表达式共用一个构件。以小写字母表示构件,大写字母表示域。

t: T 表示构件 t 为域 T 中的一个构件。

t : { T } 表示构件 t 为域 T 中一个或多个构件的集合。

a [b :F , c : { G }]: T 表示域 T 中的构件 a 有两个参数:b 和 c。其中 b 为域 F 中的一个构件,c 为域 G 中构件的集合。

a [T]: T,b [T] :T a 和 b 都是对称构件,a [b]和 b [a]都是正确的组合方式。由此可见同一域的对称构件相互之间可按任意的顺序组合。

a [b [c], d [c]] 构件 b 和 d 有相同的构件参数 c,这个表达式表示出对构件 c 的重用。

sys = a [b [c]] 该等式表示系统 sys。sys 由构件 a 实现,构件 a 由构件 b 实现,而构件 b 由构件 c 实现。

3.2 GenVoca 模型的局限性

GenVoca 模型并不能应用到所有的软件系统中去,有客观也有主观的原因。GenVoca 模型假定同一域的构件具有相同的接口,因而相互之间可互换。而事实上,由于不同的开发者依据的标准不同,尽管开发出的构件实现了相同的功能,却具有完全不同的接口;有的构件并不符合 GenVoca 关于构件的定义;有的系统本身无法抽取出 GenVoca 模型。尽管如此,由于 GenVoca 模型独立于特定的领域,它可作为进一步研究的基础。

4 一个简单的数据结构领域模型

数据结构领域是学习生成器的理想领域,因为大多数人都理解其算法。典型的数据结构包括:二叉树、列表、数组等。该领域模型依赖两个域 ds 和 mem,见表 1。ds 中的构件是对多种数据结构的实现和存储,其中的 bintree、dlist、index 和 compress 为对称构件。mem 中的构件管理内存空间。

下面通过两个实例说明构件的组合。

exp1 = bintree [compress [heap [transient]]]

exp2 = compress [bintree [heap [transient]]]

表 1

域	构件	说明
ds	bintree [ds]	二叉树
	dlist [ds]	双向连表
	index [ds, ds]	关键字索引
	compress [ds]	元素压缩
	heap [mem]	堆存储
	sequential [mem]	顺序存储
mem	transient	临时内存
	persistent	永久内存

bintree 构件将数据元素连接到二叉树上,compress 构件对元素进行压缩,heap 构件把堆地址分配给每个元素,transient 构件把元素存储在临时内存中。把 exp1 从外到内解释就意味着:exp1 定义了一个数据结构,它把每个元素连接到二叉树上,该二叉树的每个节点经压缩后存储在临时内存的堆中。exp1 与 exp2 不同。exp2 中个元素先经过压缩再连接到二叉树上。由于 bintree 和 compress 都为对称构件,因而两者的顺序可交换,形成不同的系统。由此可见对称构件的灵活性。

© 1995-2004 Tsinghua Tongfang Optical Disc Co., Ltd. All rights reserved.

5 一个仓库 /销售系统的实例

该仓库 /销售系统是一个比较简单的数据库应用系统。该系统较完整地体现了 GenVoca 生成器的思想。它分为销售管理、仓库管理和客户管理三大部分。该系统是基于 Oracle 的 Client /Server 结构。该文作者从事该系统的开发,因此对该领域有较为深入的了解。该系统中各个构件间的相互关系用图 1 表示。其中,方框表示构件,边表示构件间的联系,椭圆表示数据库中的表。

5.1 用 GenVoca 的描述语言描述该系统的模型

该领域模型包括 System、Manage、Transaction 和 Operation 4 个域,见表 2。其中,域 System 中的构件 Sys 根据用户不同的权限决定其应具有的管理功能。域 Manage 中不同的构件实现不同的管理功能,SaleManage、StoreManage 为对称构件。域 Transaction 中不同构件完成不同管理范围内的层次较高的各种操作 ,包括插入、更新和删除,每种操作具有原子性。域 Operation 中不同的构件针对不同表,完成层次较低的各种操作,包括插入、更新和删除。Cur 表存储当前期间各种材料的相关数据。Io 存储各种单据,包括:入库单、出库单、退货单等。SubIo 表存储 Io 表中各种单据的细目。Client 表存储与客户有关的信息。

用 H 代表该系统,其表示如下:

```
X = ClientManage [ ClientTran [ ClientOpr] ]
Y = StoreManage [ StoreTran [ CurOpr , IoOpr , SubIoOpr ], X ]
Z = SaleManage [ SaleTran [ CurOpr , IoOpr , SubIoOpr ], X , Y ]
H = Sys [ X , Y , Z ]
```

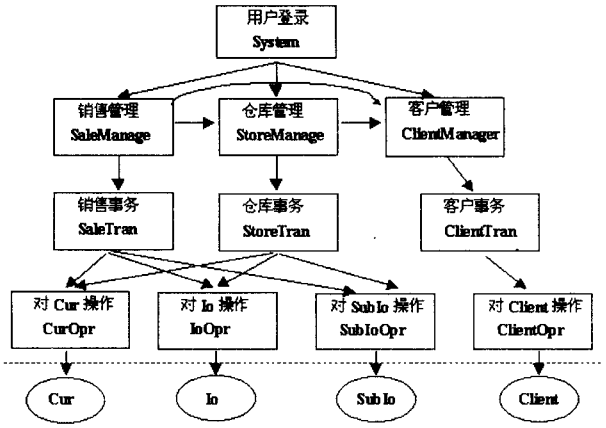

图 1

表 2

域	构件
System	Sys[a,b,c:Manage]
Manage	SaleManage[a:Transaction,b,c:Manage] StoreManage[a:Transaction,b:Manage] ClientManage[a:Transaction]
Transaction	SaleTran[a,b,c:Operation] StoreTran[a,b,c:Operation] ClientTran[a:Operation]
Operation	CurOpr IoOpr SubIoOpr ClientOpr

5.2 构件间的相互作用

不考虑并发性时,构件之间通过对方提供的操作服务相互访问。考虑到并发性时,该系统使用称为 Cursor 的机制记录并发信息和传递数据。所有与 Cursor 有关的操作也作为构件接口的一部分。Cursor 具有的功能见表 3。其中,C 为指向 Cursor 的指针,Q 为查询条 件,L 为要返回的字段名表,a 为相应构件。

以 SaleManage 与 SaleTran 间的通信为例,查询一条满足条件 Q 返回字段名的集合为 L 的操作过程如下:

(1)C:=a.CreateCursor(Q,L)

(2)通过循环调用 a.Next(C)

(3)a.CloseCursor(C)

表 3

操作名称	功能
C := CreateCursor(Q,L)	生成 Cursor
Next(C)	指向下一条数据
Prior(C)	指向上一条数据
Insert(C)	插入一条数据
Delete(C)	删除当前的数据
Update(C)	更新当前的数据
CloseCursor(C)	关闭 Cursor

5.3 构件与分层模型间的关系

构件被视为一种转换,从抽象界面到具体实现的转换。构件中的抽象界面包括各种类型及函数的声明。转换就是把抽象的函数映射为具体的算法,抽象的类型映射为具体的表示。每一个构件实现了层与层之间的转换。

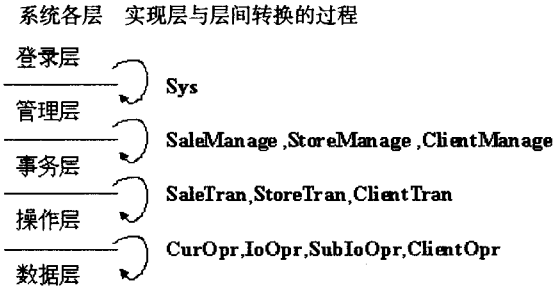

图 2

该系统具有明显的层次结构,可划分为 5 层:数据层、操作层、事务层、管理层和登录层。而上文列举的不同域中的构件正是层与层间转换的实现。高层为低层提供抽象的接口,而低层为高层提供具体的实现接口。

6 结束语

生成器技术要求编程者和软件设计者不是从行代码,也不是从面向对象的类,而是从构架级别来考虑软件。生成器技术是实现软件重用的有效方法,无论在效率上还是性能上都具有其它软件重用方法所没有的优越性,在几分钟之内就可构造一个大型的软件系统,并且可维护易扩展。通过生成器技术可实现软件生产的自动化。GenVoca 模型及 P++语言为今后其它领域生成器的构造提供了方便。

(定稿日期:1998 年 7 月)

(下转 66 页)

© 1995-2004 Tsinghua Tongfang Optical Disc Co., Ltd. All rights reserved.

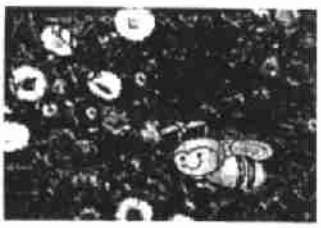

图 1　图像的合成

图 2　透明位图

图 3　源图案

图 1 是由风景画和小蜜蜂合成的图像，如果用 BitBlt 函数直接通过光栅操作 SRCCOPY 将图 3 小蜜蜂图案复制到风景画上，风景画上则会被切去一个长方形块来显示图 3，如果让小蜜蜂在风景画上飞来飞去，则风景画被连续的长方形块所破坏。针对这些问题，首先解决如何让小蜜蜂后面的风景画显示出来，即形成图 1，用 StrechBlt 函数通过下面两个步骤来实现。

第 1 步：用 StrechBlt 函数的 And 光栅操作 SRCAND 将一个纯色位图（黑和白）即图 2 重影射到一个彩色背景，引用调色板登录项 &H00（黑色）的纯色位图中的像素将迫使背景中的彩色像素变成黑色，而引用调色板登录项 &HFF（白色）的像素将使背景中的彩色像素保留原色，这样在彩色背景中就有一个小蜜蜂的黑色轮廓。

第 2 步：用 StrechBlt 函数的 Or 光栅操作 SRCPAINT 将图 3 源图案映射到上述有小蜜蜂黑色轮廓的彩色背景中，引用调色板登录项 &H00（黑色）与背景中的彩色像素结合，背景像素决定组合的位图颜色，彩色背景中小蜜蜂的黑色轮廓与图 3 中的彩色小蜜蜂结合，图 3 中的小蜜蜂的彩色像素决定组合的位图颜色，即任何彩色值与 &H00（黑色）相或时，都得到原值。

设彩色背景的图案名称为 picdest，图 2 透明位图名称为 maskpic，图 3 源图案名称为 beepic，在原图像的 X，Y 处结合，其代码为：

```
dummy=Stretch(picdest.hDC, X, Y, beepic.ScaleWidth, beepic.ScaleHeight, maskpic.hDC, 0, 0, beepic.ScaleWidth, beepic.ScaleHeight, SRCAND)  'maskpic 映射到 picdest
dummy=Stretch(picdest.hDC, X, Y, beepic.ScaleWidth, beepic.ScaleHeight, beepic.hDC, 0, 0, beepic.ScaleWidth, beepic.ScaleHeight, SRCPAINT)  ' deepic 映射到 picdest
```

执行上述两个步骤，可以合成图 1 的图像，即通过小蜜蜂可看到后面的彩色背景。

如果想让小蜜蜂在彩色背景上动起来，飞来飞去，可另设临时位图，名称为 temppic，用 BitBlt 函数的光栅操作 SRCCOPY 不断将 temppic 图案拷贝到移动前小蜜蜂所占据的彩色背景图案上，并通过同样的方法不断地将小蜜蜂所占据的彩色背景图案拷贝给 temppic，再执行上述两个步骤，即可达到目的。设小蜜蜂在彩色背景图案移动前的位置为 Oldtop、Oldleft，移动后位置为 Currenttop、Currentleft，其代码为：

```
dummy =Bitblt (picdest.hDC, Oldleft, Oldtop, beepic.ScaleWidth, beepic.ScaleHeight, temppic.hDC, 0, 0, SRCCOPY)  ' 小蜜蜂移动前，temppic 映射到 picdest
dummy =Bitblt (temppic.hDC, 0, 0, beepic.ScaleWidth, beepic.ScaleHeight, picdest.hDC, Currentleft, Currenttop, SRCCOPY)  ' 小蜜蜂移动后，picdest 映射到 temppic
dummy=Stretch (picdest.hDC, Currentleft, Currenttop, beepic.ScaleWidth, beepic.ScaleHeight, maskpic.hDC, 0, 0, beepic.ScaleWidth, beepic.ScaleHeight, SRCAND)  'maskpic 映射到 picdest
dummy=Stretch (picdest.hDC, Currentleft, Currenttop, beepic.ScaleWidth, beepic.ScaleHeight, beepic.hDC, 0, 0, beepic.ScaleWidth, beepic.ScaleHeight, SRCPAINT)  'deepic 映射到 picdest
Oldleft= Currentleft;Oldtop= Currenttop;
```

通过循环过程，不断改变 Currentleft，Currenttop 参数，调用上面的函数，即可实现小蜜蜂在彩色背景上飞来飞去的效果。

这只是实现光栅操作位图的基本方法，如果想让小蜜蜂完成其它动作，需要多加几幅源图案，并相应多加几幅透明位图，透明位图使源图案可以在 Photoshop 或画笔中进行制作。（定稿日期：1998 年 11 月）

参考文献

1. 林丽，白剑波等．精通 Visual Basic for Windows．人民邮电出版社，1995
2. 余小游，吴东．用 Delphi 实现图像的显示特技．计算机应用，1998；(2)
3. 刘海，石东青等译．图形大师技巧．电子工业出版社，1995

（上接 48 页）

参考文献

1.The Design and Implementation of Hierarchical Software Systems With Reusable Components.ACM，1992；1(4)
2.The GenVoca Model of Software-System Generators.IEEE software engineering,1994
3.Software Development Using Domain-Specific Software Architectures.ACM，1995；20(5)

© 1995-2004 Tsinghua Tongfang Optical Disc Co., Ltd. All rights reserved.