大粒度可复用MIS组件与通用MIS引擎

罗 军 乔旭峰
(重庆大学计算机学院 重庆400044)

摘 要 根据复用粒度的大小,可将软件复用技术分为代码复用、功能复用和应用复用3个层次。解决现今MIS应用开发所面临的困境,不仅需要深刻理解MIS应用的本质,也需要利用软件复用这项非常有效的技术。本文基于信息群的理论,介绍了如何实现数据库结构的复用,在此基础上,我们成功构建了一个通用MIS引擎。利用通用MIS引擎,开发人员可以方便地定制出所需的MIS应用。这样得到的MIS系统是廉价的、高质量的、灵活的和易于部署的。

关键词 软件复用,MIS,信息群,软件组件,软件工程

Reusable MIS Component with Large Grain Size and General MIS Engine

LUO Jun QIAO Xu-Feng
(Department of Computer Science ,Chongqing Univ. Chongqing 400044)

Abstract According to the grain size of software reuse, the technology of software reuse can be divided into three levels: code reuse, function reuse and application reuse. Facing the predicament of developing MIS application, it is necessary to understand the essence of MIS application, at the same time, software reuse is also an effective technology to solve this problem. With the theory of information group and the skill of reusing database structure, the general MIS engine has been developed successfully. Utilizing the general MIS engine, the developers can customize out necessary MIS application conveniently which is low-priced, high quality, flexible and easy to deploy.

Keywords Software reuse,Management information system,Information group,Software component,Software engineering

1 引言

1.1 管理信息系统的困境

尽管近年来软件工程及编程技术等领域的进步突出,但构建高质量的信息系统在现今仍然非常困难。在构建大中型的应用系统中,需求的全面捕获是困难的;而有效地理解用户需求说明并得到良好的分析与设计,即使对经验丰富的系统分析人员和设计人员来说也是一个不小的挑战;且不说编码人员需要大量而紧张的劳动,系统即使交付后维护人员也经常是胆战心惊,害怕随时可能出现的Bug和恼人的需求变更。虽然在各行业全面进行信息化的今天,很多企业与组织迫切需求自己的管理信息系统,但面对上面诸多状况,他们经常会或多或少地有些犹豫不决。

管理信息系统辅助企业与组织完成日常性的任务,是企业与组织实现信息化的重要组成部分。但MIS开发的现状却不甚理想,这直接影响了整个社会的信息化进程。企业与组织希望信息管理系统作为其日常管理的工具,是廉价、灵活、容易部署的,但现今的MIS系统通常开发周期漫长、开发成本居高不下、难于部署、经不起现实运行环境的考验。

1.2 软件复用技术现状

软件复用技术是解决当前管理信息系统困境的方法之一。软件复用技术的目标是引入高质量的可复用产品以提高新软件的质量,避免重复的开发工作。复用的对象不仅是代码,复用也包含了对问题的正确认识、设计思路、技术方法以及成功的开发过程的复用。软件开发是一项设计性的活动,认识问题并加以分析是设计,编写易于复用的代码同样需要精巧的设计,能够对这些设计结果加以重复利用,可以大大提高软件开发的效率。软件组件作为相对独立的软件单元,常常是复用的单位。软件复用可分为三个层次:

代码复用 一方面,避免重复代码可以减少软件出错的几率,另一方面精巧的代码结构利于以后的变更与扩展。使用面向对象的编程方法比面向过程的方法更易做到以上两点,而"设计模式"则是基于面向对象技术进行代码层次复用与设计的经验之作。

功能复用 避免对经常性编程任务的重复开发。常用算法、常用功能、常用技术的实现等都是在编程开发中常常要面临的问题。用软件组件对此加以封装,则未来无需对相同的功能再进行开发。功能复用可以极大地提高开发的效率。

应用复用 在现实的应用中,存在大量的相似性需求。如:进销存管理、收费系统、财务系统,甚至千变万化的MIS系统等,它们本质上有着相同的需求要求。这样类似规模的系统如果可以适应不同的现实环境,则它也是可复用的。

从代码复用到功能复用再到应用复用,复用的粒度不断加大,这是一个量变到质变的过程。从程序设计方法论,到日益成熟的组件、中间件市场,到大量的系统级和应用级的FrameWork,再到软件工程领域对过程控制的重视,软件复用获得了极大的技术支持;领域工程的出现,也显示了复用技术发展到了应用层次复用的阶段。

罗 军 副教授,主要研究方向为网络及数据库,大型MIS系统建模及设计,基于数据库的应用系统平台的架构,基于软件体系结构的重用技术。乔旭峰 在读硕士,主要研究方向为基于软件体系结构的重用技术,数据库,大型MIS系统建模及设计。

2 用复用技术解决 MIS 的问题

2.1 MIS 领域大粒度复用面临的主要问题

现今 MIS 开发面临的主要障碍在于需求提取的困难、对变更的不适应、应用环境的多样性及编程开发的困难。需求的提取通常不能一蹴而就，这为以后的变更埋下了种子，如果软件不是足够灵活，则很难适应变更。编写代码要求高素质的程序员，且通常要花费整个项目不少的时间，这使得开发 MIS 的代价很高。

构建可复用的 MIS 组件可以缓解以上问题。首先，它可以减少编程开发所需的成本；其次，可复用也意味着这些 MIS 组件必须足够灵活以适应不同的环境，这也为其提供了适应变更的潜在能力。或许我们永远也不能省却需求提出这一步骤，但可复用 MIS 组件可以使开发人员集中精力提取需求并加以分析，而对编码、维护甚至变更则无需太多的关注，因为可复用 MIS 组件可以大大减少编码工作、可以保证软件的质量、足够灵活可以适应频繁的变更。

无疑，可复用 MIS 组件的前景诱人，但构建这样的组件面临着种种障碍：

信息系统中数据模式的多样性 不同的信息系统有不同的数据模式。现在，构建 MIS 系统之前要首先分析并建立其数据模式，如果 MIS 组件不能适应不同的数据模式，就无法复用。

数据处理手段的多样性 数据处理本质上是新增、修改、删除、查询及其他们之间的组合，这样的多样性使得构建可复用的 MIS 组件困难重重。无法提供丰富的数据处理手段，就无法得到一个易于使用的 MIS 系统。

应用环境的多样性 现实应用环境多变，B/S 架构是一种通用的应用模式，但如何适应不同的 Web 服务器、不同的操作系统平台都是必须考虑的问题。平台相关的组件很难得到充分复用。

可见，现实应用所展现的多样性是我们构建大粒度可复用 MIS 组件的障碍。如果我们为这样的多样性所迷惑而不去发掘背后的相似性，构建可复用 MIS 组件是不可能的。

3 大粒度 MIS 组件复用的思路

实现应用层次的复用不仅是软件技术方面的问题，也是认识分析问题的方法论的问题。软件的复用性很大程度上取决于对可复用对象的认识深度或者说对可复用对象的抽象层次。抽象层次越高、与具体环境和特定细节越无关，则它被未来系统复用的可能性也越大。抽象层次越高，我们才能发现更具价值的可复用对象，才能得到更大粒度的可复用组件。

考虑编程中一个可以被反复调用的方法，可以为其指定不同的参数，则方法的表现或输出就会有所不同。这个方法要实现的功能是由(可变的)参数和(不变的)方法体共同所作用而实现。如果没有参数，则这个方法只能表现出单一的具体功能，但有了参数，则方法可以实现一类功能，其复用的潜力更大。

现实应用也一样，有其可变的一面和固定的一面。对 MIS 系统来说，其本质就是对信息的存储和操作。不同模式的信息，其处理可能是类似的；相同的信息，要求的处理方式可能是多样的。因此，可复用 MIS 组件的主要功能就是提供通用的存储解决方案和可以适应不同模式信息的丰富的处理手段，这是可复用 MIS 组件不变的部分；而其可变的部分就是信息的具体模式与各个信息类别具体需要哪种处理方式。

对现实应用高度抽象的结果可以作为设计可复用组件的依据；抽象程度越高，则将来设计的软件组件的适用性将更广，但同时也意味着用软件技术来表现这样的抽象结果将遇到更大的挑战。我们不得不在抽象程度与现实技术的限制中间寻求平衡，以找到可行的设计方案。例如，对于不同模式的信息去寻求通用的存储方案可能难以实现，则需要我们降低抽象的程度，或许可以将信息进行分类(当然是有限的分类个数)，为各个信息类分别寻求通用存储方案。

因此，对问题进行合理抽象，寻找其固定项和可变项，是设计大粒度的可复用软件组件的一条可行的思路，正如下面的等式所形象表示的那样：

应用功能＝可复用软件组件＋可变项定义

“应用功能”是我们试图解决的一类问题，对其固定项，用软件技术加以实现并封装成软件组件。这样的组件能够理解“可变项定义”并进行解释，不同的“可变项定义”能得到不同的解释结果，这个解释结果就是具体所需的功能。只要遵循定义可变项的规则(这个规则依赖可复用软件组件的具体实现)给出不同的“可变项定义”，将驱动可复用组件向用户展现多样化的、符合用户需求的应用功能。

随着“应用功能”规模的增大，其对应的可复用软件组件的粒度也随之增大。这样复杂的组件可能是由众多功能较为单一的组件所组成。在此思想指导下，我们实现了多个大粒度的可复用 MIS 组件，并由它们组成了一个通用 MIS 引擎。

4 通用 MIS 引擎的设计

4.1 数据库结构的复用

绝大部分的 MIS 系统构建在关系数据库之上，我们的通用 MIS 引擎也采用关系数据库作为数据存储的手段。因此，如果不能做到数据库物理结构的通用性，则开发通用 MIS 引擎只能是一句空话。因此，设计可复用的数据库结构，是开发可复用 MIS 组件的基础，也是关键。

实现数据库结构的复用有两种思路。一种思路是根据具体需求要求的数据模式，在系统初始化时自动生成物理表。但这种实现方式不利于以后对数据结构的更改，特别是当有一定量的业务数据已经进入数据库后，对数据结构的更改甚至会引起数据的丢失。另一方面，不同结构的数据存储在不同的物理表中，也不利我们利用信息群的理论统一处理不同结构的信息。

另一种方式是并不在数据库中建立与需求一致的物理表，而是通过有效地组织数据字典，将不同结构的信息数据最终存储在有规律的若干物理表中。如图1所示：

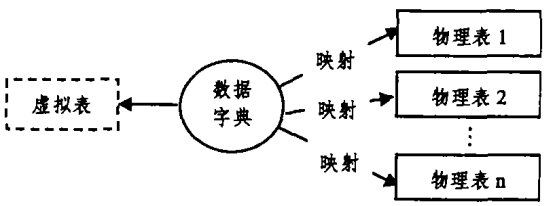

图1 信息群示意图

我们称需求所要求的数据模式为虚拟表(在数据库并不存在对应结构的物理表)，而通过数据字典将其映射到数据库中特定的某张物理表中。数据字典需要描述每个虚拟表映射到哪个具体的物理表上，以及虚拟表的各个字段映射到物理

表的哪些字段上。有了数据字典里的这些信息，就可以把对虚拟表的请求与处理转化为对物理表1到 n 的请求。

物理表1到 n 就像一个盛放数据的水池，可以将任何结构的虚拟物理表映射到这个水池中来。数目不确定的虚拟表映射到确定数目的几张物理表中，可以为我们统一处理不同结构的信息带来方便，在面对不同的 MIS 应用时，也不用再去建立众多的物理表（将其映射到物理表1到 n 中即可）。

物理表1到 n 必须有足够多的具有不同数据类型的属性列，这样才可能容纳不同结构的虚拟表。这会造成一定的冗余，但确实是值得的。我们根据主键列的最大个数来决定 n 的大小。具有特定主码个数的虚拟表映射到具有同样主码个数的物理表上。一般来说，$n=8$ 应该可以满足绝大多数的应用。这样，不至于使数据库管理系统对主键的约束机制失效。对一个具体的 MIS 应用来说，n 是一个确定的数字。n 的大小表明了一个 MIS 应用在数据模式上的复杂度。

通过这样一个技巧，可以实现数据库物理结构的复用，这对构建通用 MIS 引擎至关重要。

4.2 信息群

在一个 MIS 系统中，所维护的信息之间是有联系的，这些信息就组成了信息群。如果一个信息在系统中与其他任何信息都没有关联，就构成了信息孤岛。信息孤岛的存在是不合理的，现实中也很少会出现信息孤岛。信息之间的联系体现在以下几个方面：

一个信息依赖于另一个信息。（关系数据库中的参照性约束就是这种情况。）

信息项之间具有计算关系，某个信息项的值发生变化，会影响另外一个信息项的值发生变化。

信息间的依赖关系是传递的，A 信息依赖 B 信息，而 B 信息依赖 C 信息，则 A 也依赖 C。

如果两个信息都依赖与同一个信息，则这两个信息间是兄弟关系。

如图2是一个信息群的示意图：

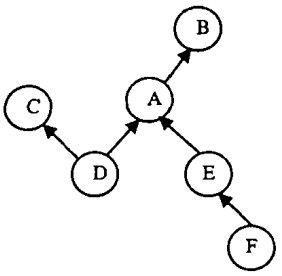

图2 数据库结构的复用

有关联的信息之间存在一条路径，顺着这条路径，可以依据其中一个信息而搜索到另外一个信息。信息群中存在的路径也反映了人类寻找、处理信息的方式。如上图，如果用户想处理 A 信息，则很可能希望先找到 B，缩小、明确搜寻范围后找到 A；处理完 A 之后，用户会习惯于马上处理与 A 相关的其他信息，如处理 D、E 或者 B。对应到现实中的例子，则可能是一个教师意图处理学生的信息，则需要通过先找到学生所在班级，定位到某个学生才能进行处理。处理完学生的信息后，教师可能会马上想处理这个学生的成绩信息、简历信息等。善于利用信息群中信息间的关联，可以为用户提供通用的信息处理方式。

通用 MIS 引擎正是从信息群理论中受益颇多。用户对数据的权限，也可从信息群中寻找到依据。例如，某个用户如果只对 A 中的某个数据拥有权限，则会限制其对 D 和 E、F 的访问权限。再一次拿学生作为例子，一个学生用户被限制只能访问系统中学生信息表中关于自己的那条数据，则也意味着这个学生用户只能访问依赖于这条学生信息的成绩信息以及简历信息。正是基于这样的规则，我们设计了通用 MIS 引擎中的信息内容权限机制。

信息群应该能够自动地、自发地维护其内部的约束，就像关系数据库管理系统能够自动维护实体完整性、参照完整性以及自定义完整性约束一样。

在信息群的基础上，我们将对群内信息的处理分为业务、查询、工作流、报表、复制五种。业务处理对单个信息简单的查询、增删改操作；查询处理用户对信息检索的要求；工作流则可以使信息进入流程化的处理；报表实现对信息的统计与分析的功能；复制则实现了信息在不同系统之间的信息共享。

4.3 MIS 组件

按照在上面提出的思路，则一个实际 MIS 应用可以按照如下等式得到：

MIS 应用系统＝通用 MIS 引擎＋配置项数据

配置项数据是“可变参数”，是驱动通用 MIS 引擎展现现实世界 MIS 应用多样性的动力。通用 MIS 引擎有多个 MIS 组件组合而成（如下图），它可以对配置项数据进行解释以得到所需功能。配置项数据则描述了信息群的结构，规定系统有哪些业务、查询、工作流、报表以及哪些信息需要进行复制。

配置项数据可以被更改，即使在系统投入运行后，仍然可以改变配置项。而当配置项数据发生变动，通用 MIS 引擎会使用最新的配置进行解释，系统功能随之就发生了变化。但在更改信息的结构或信息间的关联时要小心，因为这可能造成对信息的处理（业务、查询等）也必须进行相应的改动。因此，架构在引擎之上的 MIS 应用可以以非常低的代价随需应变。

任何结构的信息对引擎来说其处理都是一样的，因此，在信息群的基础上，实现业务、查询、报表等这些 MIS 组件是非常方便的，还可以方便地监控任何时候用户对任何信息所作的操作，在此基础提供了审计、日志的功能。

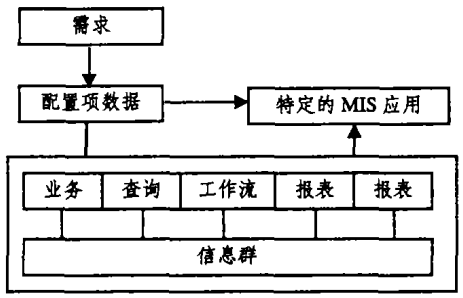

图3 通过 MIS 引擎与 MTS 组件

在最终用户眼中所出现的丰富多样的功能，是由配置项中所定义的丰富的信息以及不同的业务、工作流、报表所驱动而产生的。所以，我们也称我们的通用 MIS 引擎为“数据驱动的 MIS 平台”。对开发人员来说，通用 MIS 引擎是一个方便地开发 MIS 应用的平台；而在用户眼中，配备了配置项的通用 MIS 引擎所产生的 MIS 应用不会与其他的 MIS 应用有什么不同。

总结 构建大粒度的可复用 MIS 组件对于开发 MIS 应用来说意义重大，可以极大地提高开发 MIS 应用的效率。我

（下转第227页）

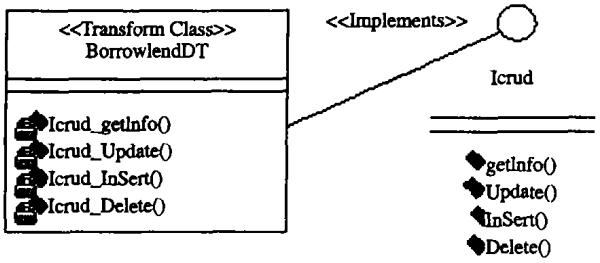

图8 "BorrowlendDT"类

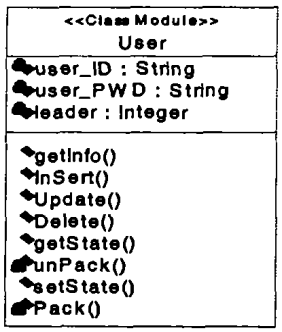

图9 "User"类

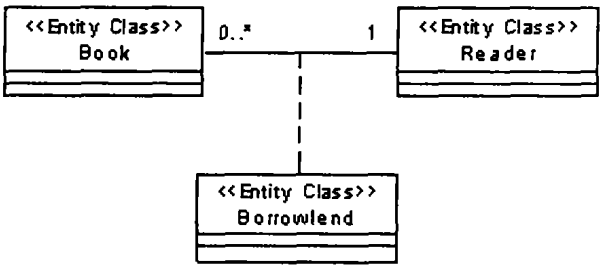

图10 关联类"Borrowlend"

因为 Rose 在生成关联类的 VB 代码方面存在缺点，所以从这个模型得不到我们想要的 VB 代码。我们需要的代码，或者说应该生成的代码如下：

```
'Borrowlend Class
        Public myReader as Reader
        private mcolbook as Collection
```

所以，不管是在代码生成前修改模型还是代码生成后进行逆向工程，我们都应得到如图11的模型。Mcolbook 与 myReader 作为角色名，"+"表示 Public，"－"表示 Private。

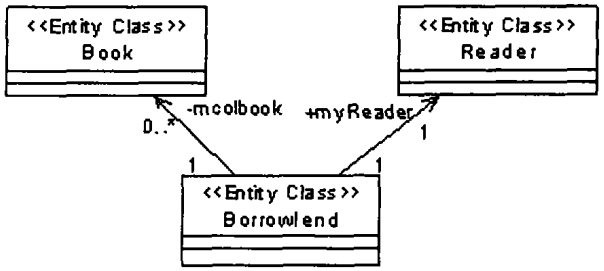

图11 修改后的模型

由实现过程可知代码的主要资源是类和组件图。在整个实现的过程中，层次关系非常清晰，体现了由软件体系结构建立模型的优势。当代码实现后，我们应该将应用程序代码逆向工程到可视化模块中，实际上，正向工程、编码、逆向工程——这是个重复的过程。

结束语 本文以一个图书管理系统开发为背景，探讨了基于 UML 和软件体系结构的建模与实现问题。由建模过程可以看出 UML 在体系结构建模方面存在很好的优势，它提供了一个统一的交流平台，而实现过程则充分体现了 VB 的面向对象特性。软件体系结构与面向对象开发方法相结合，不但从整体上正确描述了该图书管理系统的体系结构并且可以充分利用 UML 面向对象的优势，最终使得项目成功完成。

但是，作为一种通用的语言，UML 对软件体系结构的可构造性建模能力较弱，缺乏形式化语义，对体系结构的描述只能到达非形式化的层次。因此，下一步工作将探讨 UML 和体系结构描述语言 ADL 相结合来描述软件体系结构问题，ADL 形式化语义的精确性正好可以弥补 UML 非形式化的一些不足，二者的有机结合，不仅可更好地描述体系结构模型，还支持下一步的求精和验证工作[6]。

参 考 文 献

1 周莹新，艾波. 软件体系结构建模研究. 软件学报，1998，9(11)：867～872
2 唐稚松，等. 时序逻辑程序设计与软件工程，北京：科学出版社，2002
3 刘超，张莉. 可视化面向对象建模技术——标准建模语言 UML 教程. 北京：北京航空航天大学出版社，2000
4 张龙祥. UML 与系统分析设计. 北京：人民邮电出版社，2002
5 Paulr R. Developing Applications with Visual Basic and UML. Addison Wesley，1999
6 戎玫，张广泉. 软件体系结构求精方法研究. 计算机科学，2003，30(4)：108～110

（上接第216页）

们开发的通用 MIS 引擎可以用来定制不同行业的管理信息系统，在实际的应用中，取得了很好的效果。对我们的通用 MIS 引擎来说，配置项数据就是对具体 MIS 应用进行分析设计的结果，因此，一个 MIS 应用可按以下方式得到：

MIS 应用系统＝通用 MIS 引擎＋MIS 应用的分析设计结果

在这个过程中，无需任何编码。可以说，这是未来开发 MIS 应用的趋势：把握 MIS 应用的本质，构建大粒度可复用 MIS 组件，让 MIS 开发人员从编码工作中解脱出来，将更多精力集中在需求的提取与分析上。

参 考 文 献

1 Persse J R. Implementing the Capability Maturity Model[M]. Wiley，2001
2 Singhai A，Sane A，Campbell R H. Quarterware for Middleware [C]. In：Proc. of the 18th IEEE Intl. Conf. on Distributed Computing System，Amsterdam，May 1998. 192～201
3 Burton B A，Aragon R W，Baily S A，et al. The reusable software library[J]. IEEE SOFTWARE，July 1987. 25～33
4 张海潘. 著. 软件工程导论[M]. 清华大学出版社，2003
5 张畅，徐冬溶，潘云鹤. 于多源类比的 MIS 表格生成[J]. 计算机研究与发展，1998
6 杨肖鸾，等. 面向敏捷制造的虚拟企业管理信息系统框架研究[R]. 2001. 40～49
7 李相枢，基于软件体系分结构的 MIS 系统数据采集构件[J]. 计算机科学，2002，4
8 Kirland M. 基于组件的应用程序设计[M]. 北京：博彦科技发展有限有限公司译. 北京大学出版社，1999