基于构件/构架的可复用串行通信开发方法的研究

Analyse, Design and Realization of Developing Reusable Serial Communication Component/Architecture

宁 伟

(泰山学院数学与计算机科学系 山东泰安271000)

Abstract This paper puts forward a kind of method of developing communication architecture of serial port which is in common use by applying the theory of component/architecture and software reuse to serial communication , and may help to increase the benefit and effect of tapping software. These series of methods could also be used in other domains of developing Component/Architecture for reference.

Keywords Component, Architecture, COM, Software reuse

近几年来,随着计算机技术的发展及人们对信息需求的提高,串行通信技术作为计算机与计算机或单片机间通信的桥梁得到了越来越多的应用,被广泛地应用在工业控制、商业通信、电力调度等诸多领域。通常所开发的串行通信程序,存在代码冗长、结构复杂、复用性差、编程技术不易于掌握、可靠性差等缺点,而且应用需求量大,程序开发周期长。能否解决好这些问题已成为开发通讯程序的程序开发商共同面临的挑战。本文通过对"构件-构架理论"和软件复用理论应用于串行通信的研究,提出了一种通用的可复用的串行口通讯构架的开发方法,从而提高串行口通信程序的可复用性和可靠性,缩短程序的开发周期。

1 软件复用及构件和构架

1.1 关于软件复用

通常情况下,应用软件系统的开发过程包含以下几个阶段:需求分析、设计、编码、测试和维护等。当每个应用系统的开发都从头开始时,其中必然存在大量的重复劳动,如:用户需求获取、需求分析、设计、编码、测试和文档都会有重复。

软件复用是在软件开发中避免重复劳动的解决方案,出发点是应用系统的开发不再采用一切从零的开发模式,而是以已有的工作为基础,充分利用过去应用系统开发中积累的知识和经验,从而将开发的重点集中于应用的特有构成成分。

通过软件复用。在应用系统开发中可以充分地利用已有的开发成果,消除了包括分析、设计、编码和测试等在内的许多重复劳动,从而提高了软件生产率,同时,通过复用高质量的已有成果,避免了重新开发可能引入的错误和不当,从而提高了软件的质量。

1.2 构件技术

构件是指应用系统中可以明确辨识的构成成分,而可复用构件是指具有相对独立的功能和可复用价值的构件。

可复用构件应具备以下属性:(1)有用性:必须提供有用的功能;(2)可用性:必须易于理解和使用;(3)质量:自身及其变形必须正确;(4)适应性:应该通过参数化等方式在不同的语境中进行配置;(5)可移植性:应能在不同的硬件平台和软件环境中的工作。

构件技术是支持软件复用的核心技术,是近几年来迅速发展并受到高度重视的一个学科分支。随着对软件复用理解的深入,构件概念已不再局限于源代码构件,而是延伸的需求、系统和软件的需求规约、系统和软件的构架、文档、测试计划、测试案例和数据,以及其他对开发活动有用的信息。这些信息都可以称为可复用软件构件。

1.3 软件构架

软件构架是对系统整体结构设计的刻画,包括全局组织与控制结构、构件间通讯、同步与数据访问的协议、设计元素间的功能分配、物理分布、设计元素集成、伸缩性及性能、设计选择等。

研究软件构架,对于进行高效的软件工程具有非常重要的意义:通过这一研究,有利于发展不同系统在较高级别上的共同特性;获得正确的构架,对于进行正确的系统设计非常关键;对于各种软件构架的深入了解,使得软件工程师可以根据一些原则在不同的软件构架间作出选择;从构架层次上表示系统,有利于系统较高级别性质的描述和分析。特别重要的是,在给予复用的软件开发中,为复用而开发的软件构架,可以作为一种大粒度的、抽象级别较高的软件构件进行复用,而且软件构架还为软件的组装提供了基础和上下文,对于成功的复用具有非常重要的意义。

软件构架研究和如何快速、可靠地从可复用构件构造系统的方法,着重于软件系统自身的整体结构和构件间的互连。其中主要包括:软件构架原理和风格,软件构架的描述和规约,特定领域软件构架,构件向软件构架的集成机制等。

2 串行口通讯领域分析

串行口通讯的接收功能一般包括如图1所示的几个过程。

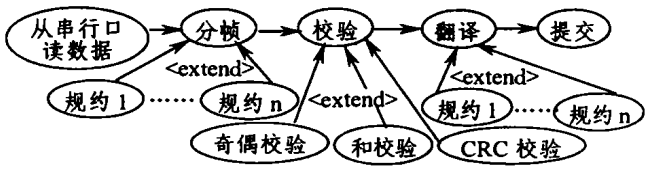

图1 串行口通讯的接收器 use case 图

串行口通讯的发送功能一般包括如下图2所示的几个过程。

宁 伟 博士生,副教授,主要从事计算机应用技术和测量数据处理方面的研究。

图2 串行口通讯的发送器 use case 图

由上图可见，在串行口通讯的每一步都存在变化点。如果根据这一些变化点，开发一系列的构件，这些构件的组合将是几十种，即相当于几十种串行口通讯程序。这种以几倍的代价换取几十倍的收益的复用方法是很有意义的。

具体到使用哪种复用方法，应根据具体情况而定。分帧可使用参数的方法，只要确定了帧头和帧长就可分帧了。对于校验可采取参数的方法，可根据不同的参数确定不同的代码段或子函数，但为了更灵活也可采用继承的方法。翻译及转换打包是最关键也是最难解决的问题。一种方法是对每一种规约都建立一个类，这种方法针对性强且可靠性高，但复用性差，因为每引入一个新的规约就必须编写一个新类。另一种方法是建立一个规约翻译器，它可根据用户输入的规约表和具体规约意义表自动地将二进制代码翻译为可读数据，存入数据库或数据文件中。尽管这种方法在原理上是可行的，但实现起来还有一定的困难，特别是有时规约和规约之间存在很大差别(如有的规约是固定帧长的，而有的规约是可变帧长的)，无法在原理上统一起来。作为这两种方法的折衷，可通过分析规约的实质将其分为几类，对每个类建立一个规约翻译器。这样做解决了针对性和通用性之间的矛盾，但开发周期仍是一个问题。

串行口的通讯方式分为循环发送接收和呼唤发送接受。循环发送接收是指发送方不停地发送数据，接受方不停地接受数据。呼唤发送接受是指接收方在需要接收数据的时候发送一帧请求帧并准备接受数据，接收方在接到请求帧后发送数据帧。这就决定了串行口通讯程序的行为模式必须有四种形式：循环发送、循环接受、呼唤发送和呼唤接受。通讯程序的 usecase 如图3所示。

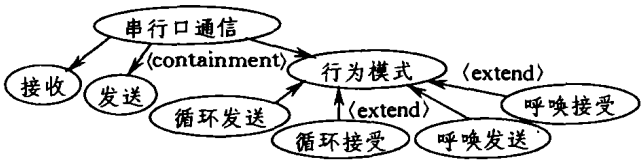

图3 串行口通讯的 use case 图

有时一台计算机不止使用一个串口通讯，而是同时处理 n 个串口的通讯问题。这就需要考虑 n 个串口同时通讯的策略问题。通常多串口通讯有两种方式：轮询式和线程式。轮询式实现方法简单，但不适应于对实时性要求较高的情况。线程式实现复杂，但实时性能良好。因此，多串口通讯的 usecase 如图4所示。

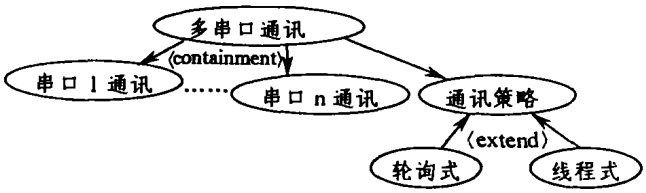

图4 多串行口通讯的 use case 图

值得一提的是，无论在串口通讯还是在多串口通讯的构架中，作者并没有按照传统的面向对象的方法，将不同的行为设计为子类，然后再在每个子类中包含相应的构件，而是按照设计模式的 Command 行为模式将不同的行为封装在类中，然后将其包含在构架中。这样做简化了结构，提高了复用性。

3 构架、构件设计

本文不打算对各构成构件的实现做详细探讨，仅给出不同层次的构架结构图和相应的程序设计模块。

3.1 串口接收构架(如图5)

```
procedure execute
begin
readfromseri(port); //从串口读出数据并放入属性 seridata 中，port 为串口号；
departframe(); //根据帧定义表 frametable 将 seridata 中的数据分为若干帧并记入属性 framedata[]数组中。
For framedata 中的每一项 do
If check^.checkframe then //如果帧检验合格，
translator^.translate; //将 framedata[]中的二进制数据转换为数据表 data[]中；
apply; //提交到数据库中；
End;
```

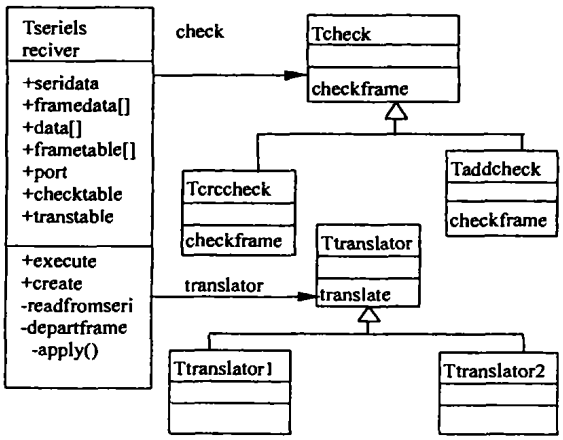

图5 串口接收类图

```
Procedure create(checknum,transnum)
Begin
 在校验登记表 checktable 中找到与 checknum 对应的类记为：Tcheck1；
 在翻译登记表 transetable 中找到与 tansnum 对应的类记为 Ttransel；
 check1:= Tcheck1.create; //产生 check1对象；
 transel:= Ttransel.create; //产生 transel对象；
 check:=@check1; //将 check1对象指针赋予 check 属性；
 transe:=@transel; //将 transel对象指针赋予 transe 属性；
End;
```

3.2 串口发送构架(如图6)

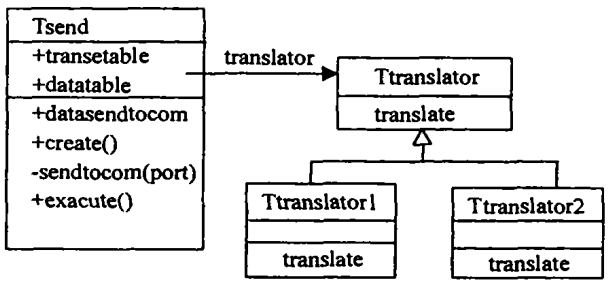

图6 串口发送类图

```
procedure exacute;
begin
 translator.tranlate;//调用转换器将存与 datatable 数组中的数据转化为数据帧，并将其存入数组 datasendtocom 中；
 sendtocom(port); //将数据帧发送到第 port 号串口；
end;
 procedure create(transenum)
```

```
begin
    在翻译登记表 transetable 中找到与 tansnum 对应的类,记为 Ttransel;
transel:= Ttransel.create; //产生 transel对象;
transe:=@transel; //将 transel对象指针赋予 transe 属性;
end;
```

3.3 串行口通讯构架(如图7)

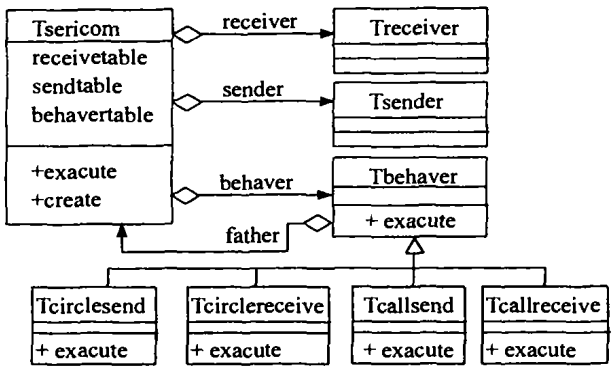

图7 串口通讯类图

```
Procedure create(receivenum,sendnum,behavernum,checknum,intransnum,outtransnum)
Begin
  根据 receivenum 在 receivetable 表中确定其对应的类记为:Treceivel;
  根据 sendnum 在 send table 表中确定其对应的类记为:Tsendl;
  根据 behavenum 在 behavetable 表中确定其对应的类记为:Tbehavel;
    receivel:=Treceivel.create; //产生 receivel对象;
    sendl:=Tsendl.create (intransnum,checknum); //产生 sendl 对象;
    behavel:= Tbehavel.create (intransnum,checknum); //产生 behavel对象;
    receiver:=@receivel; //将 receivel的对象指针赋予 receiver;
    sender:=@sendl; //将 sendl的对象指针赋予 sender;
    behaver:=@behaverl; //将 behaverl的对象指针赋予 behaver;
    Behaver^.father:=self; //将 behaverl的 father 属性置为本对象的指针;
End;
Procedure exacute
Begin
behaver^.exacute; //调用 behaver 所指对象的方法;
End;
```

对于类 Tcirclesend,方法 exacute 如下设计:

```
procedure exacute
begin
  father^.sender^.exacute; //调用其 farther 属性所指对象的 sender 属性所指对象的 exacute 操作;
end;
```

对于类 circlereceive,方法 exacute 如下设计:

```
procedure exacute
begin
  father^.receiver^.exacute; //调用其 farther 属性所指对象的 receiver 属性所指对象的 exacute 操作;
end;
```

对于类 callsend 中的 exacute 如下设计:

```
procedure exacute
begin
  father^.receiver^.exacute; //调用其 farther 属性所指对象的 receiver 属性所指对象的 exacute 操作;
  if father^.receiver^.data∈发送命令 then father^.send^.exacute;
    //若收到发送命令,则发送一帧数据;
end;
```

对于 callreceive 中的 exacute 如下设计:

```
procedure exacute
begin
  father^.sender^.datasendtocom=发送命令; //送出请求发送帧;
  father^.sender^.exacute; //调用本对象的 father 属性所指的对象的 sender 属性的 exacute 操作;
  do
    father^.receiver^.exacute; //准备接受一帧数据;
  until father^.receiver^.data<>null; //直到接收完毕一帧数据;
end;
```

关于多串口通讯构架结构图及实现模块可仿照上述给出,这里因篇幅所限,不再具体讨论。

4 构件、构架的组装

构件构架的组装分为插合和粘合两种情况。插合是指构件构架的组装不再需要编程,而是通过参数配置的方法,由构件构架内部的机制,实现根据用户的需要自动组合相应的控件。粘合是指通过程序语言将不同的构件构架组合起来。前面所提到的设计方法,能够实现在串口通讯功能内部的插合,至于串口外的程序的数据通讯,可通过数据库或 DDE 实现。因此,本设计方案完全可实现自动插合。

构件构架组合的原则是先将构件组合为小粒度的构架,再由小构架组合为大粒度的构架。本设计方案正是根据这一原则,由大构架提出组合要求,分配给小构架完成。小构架再将任务分配给更小的构架或构件完成。

结论 通过对上面的串行口通讯的分析,可以总结出开发基于构件的可复用程序的开发方法:

(1)全面细致地进行领域分析,区分出哪些行为是变化的,哪些行为是不变的,既找出变化点。

(2)要根据各个变化点性质的不同,确定使用哪种复用方法。

(3)结合前面的分析,确定构件及构架的结构及其组装方式,选择合适的设计模式。

(4)在设计时先设计构件,再设计小粒度的构架,最后设计大粒度的构架。当然也可以先设计整体构架,再将其逐步细化。

参 考 文 献

1 徐正权. 软件复用方法与技术[M]. 武汉:华中理工大学出版社,1998
2 王文福,张世琨,朱冰. 软件工程[M]. 北京:北京大学出版社,1997
3 杨芙清. 软件复用及相关技术[J]. 计算机科学,1999(5)
4 李景峰,刘西洋,陈平. 一种可重用构件模型——类属构件[J]. 计算机科学,1999(9)
5 Ivar Jacobson. Software Reuse [M]. ACM Press, 1997

(上接第151页)

可视化程度较低,程序调试起来显得不是那么得心应手。

参 考 文 献

1 宁建民 主编. WEB 数据库开发. 大恒电子出版社,2000
2 阮家栋,施美雅 著译. WEB 数据库技术. 科学出版社,2002.
3 杨浩 编译. JAVA SCRIPT 入门与提高. 清华大学出版社,2000
4 王燕,崔雨柏 译. ORACLE WEB 应用培训教程. 机械工业出版社,1998
5 黄樱,刘绍中. ORACLE WAS 的安全机制. 计算机应用,1998
6 傅豫波,李云静编译. ORACLE PL/SQL 程序设计. 机械工业出版社,1999