建立可复用构件的研究

黎 娅
(重庆大学 重庆 400030)

摘 要 实践证明,软件复用技术能够为软件开发带来巨大的好处。本文阐述了可复用构件的特性,重点介绍了建立可复用构件相关技术,同时分析了当前软件复用技术中的问题。

关键词 软件复用,可复用构件

1 引言

随着计算机应用日益普及和深化,计算机软件的数量以惊人的速度急剧膨胀。由于计算机系统发展的早期时代所形成的一些错误概念和做法,许多大型软件几乎无法维护,造成了大量人力、物力的浪费。西方计算机科学家把软件开发和维护过程中遇到的一系列严重问题统称为"软件危机"。为了解决软件危机,更多的计算机科学家专注于研究计算机软件工程学,以寻求更有效的办法。

在软件工程学中,软件复用技术是值得大家关注的一个热点。复用应该作为一门独立的科学,是对象技术的一种必要的补充。软件复用(software reuse)是一种由预先构造好的、为复用目的而设计的软件构件来建立或者组装软件系统的过程。其实它的基本思想非常简单,即放弃传统的、一切从头开始的软件开发方法,由公共的可复用构件来组装新的系统,这些可复用构件包括对象(object class)、框架(framework)或者软件体系结构等。经研究和实践表明,复用技术能够带来巨大益处。它能提高生产率、缩短开发周期、降低软件开发和维护的费用、能生产更加标准化的软件,并且提高软件开发质量,增强软件系统的互操作性、减少软件开发的数量、使开发人员能更容易对不同性质的项目进行开发。美国得克萨斯州 Irving 的 GTE 电话运营公司认为,复用的潜力是非常巨大的,它们计划在三至五年的时间内,将其应用系统的 80%至 90%由库中可复用的对象构件组装而成。

2 可复用构件的特性

一个可复用构件应该是高质量的,它具有易于理解、可靠和可移植等特性,能够激发复用者的信心。另外,要弄清楚一个概念—可复用性:它是指一个软件构件在不经修改,或者稍作修改后,就能用于多个软件系统、版本或者实现的难易程度。在建立可复用构件时,我们应该把可复用性作为建立构件的一个目标,而不是通过后来对构件进行修改使它具备可复用性。

3 建立可复用构件的处理过程

3.1 对构件进行一般化处理,使它成为可复用的

抽象出共性而去掉差异之处的过程称为一般化(generalization)。一个一般化的构件利用了一组相似构件的共性而隐藏了差异之处。通过对若干个相似构件实例的比较,发现它们之间的相同点和不同点。为了复用一个一般化的构件,复用者必须在原有的"共性"基础上对其进行扩充,加入使构件能够满足某一特殊复用目的而需要的不同之处或者细节。

3.1.1 使用参数来使一个构件一般化

参数化方法(parameterization)是指可以为定制一个一般化的构件提供一个受控的方式。当构件被复用时,它用一组在允许范围内的取值替代参数,该参数是嵌入在构件中的,用来表示差异之处的"占位符"。参数化方法允许一个一般化的构件去解决不同的问题。常见的差异类型包括不同的操作环境,不同的性能要求,不同的算法和规则等。例如,一组相似构件之间,需要对这组相似构件中的每一个构件的要求进行研究,作为发现构件间差异之处一种方法。

3.1.2 应用一般化技术

一个构件只有经过一般化处理,才能使它在一个系统的多个场所或者在多个系统中得到复用。通常采用的技术手段包括:分离出对内部实现细节的依赖,例如对语言的环境依赖。将行为和功能进行抽象,脱离具体应用。扩展构件的特性。以抽象、简明和可表达的形式来说明构件能做什么的信息。与具体的构件信息相分离。从较高的抽象层次来描述构件,避免在多个构件间分解一个抽象的概念。保持构件的接口是抽象的。

3.1.3 将可复用构件分为两个部分来设计

把可复用构件分成固定部分和可变部分。若需要对一般化的构件进行大量修改，复用的效益就可能被抵消掉，也达不到复用的目的。因此，复用者仅允许通过修改可变部分来定制构件，固定部分只能被复用，而不能修改。面向对象的框架就是一个很好的例证，框架的固定部分使由抽象类和具体类组成的类层次结构，可变部分则是一些空缺方法，复用者使用时可以用具体的内容来填充。另外，可变部分的定制是受限制的。例如参数替换，事先已定义好合法取值范围。

3.2 对构件进行标准化操作，使它成为可复用的

通过标准化，软件复用会变得更加容易。建立标准化特性的可复用性构件，我们不仅为以后的复用创造了机会，它的复用性也会得到提高，并且有着更好的质量和通用性。一方面为构件在一定范围建立标准。例如调用、控制和结束功能、错误处理、用户接口等方面都需要标准化。另外一方面利用结构化的编程规则来标准化构件。利用结构化编程规则，设计有利于建立标准化的程序控制结构和程序模块格式。例如，单入口和单出口的模块。

3.3 对构件进行自动化操作，使它成为可复用的

增加可复用构件的大小，增大了它的复用影响。例如，复用一个完整的系统体系结构比仅复用一个底层的模块相比，将更节约成本。自动化的构件能够由机器来控制，因此能够使与建立、管理和复用构件有关的活动自动的进行。现在很多软件工具可以完成自动生成代码，检查和识别冗余，发现复用的机会等功能。大大减少人工操作，减少工作量。通过建立自动化的构件不仅减少了复用成本，并且有利于提高可复用构件的质量。

3.4 对构件的质量进行验证，使它成为可复用的

每个可复用构件都要经过验证的过程。验证的具体过程将取决于可复用构件的类型。验证信息可跟复用构件放在一起，作为它的文档的一部分。验证过程中，对构件质量的验证通常包括：数据检查：检查构件对外部数据的依赖程度。可移植性检查：证明构件能正确地跨平台运行。编程标准：检查构件是否遵循公司标准。使用历史：检查构件是否在一段时间内在一个或多个系统中使用。复用历史：检查构件是否成功地复用三次以上。复杂性：检查构件是否满足相关复杂性要求。审查检查：构件是否经过了正式的评审。

3.5 为构件建立复用文档

为了能够复用一个构件，需要理解、比较、选择、修改和集成该构件的信息。在现实中，复用的一个主要的障碍在于对构件的理解程度。因此建立复用构件需要完整的复用文档。一般来说需要六种类型的文档：详细信息、分类信息、说明性信息、质量/验证信息、复用信息、详细的文档。我们需要那些描述构件能够做些什么，它的属性，什么情况下能够被复用等等相关信息。文档可以采用叙述性和/或图等各种形式，并且能够对它进行自动化的处理，从而便于对其快速的定位、管理和修改。另外需要注意的是文档资料的规范化，否则会妨碍复用者对文档信息的理解。

4 克服建立可复用构件的困难

4.1 技术因素

构件与应用系统之间存在很大的差异。很难做到开发者开发的构件，在另外一个环境中正好合适，或者很少的修改就可以完全使用。因此，可复用构件本身的开发存在高投入和高风险。没有可供复用的构件，没有软件复用库。或者是拥有大量的可复用构件，要从中挑一个合适的构件，也不是一件轻而易举的事情。目前这方面的研究成果和实践经验都不够成分。

4.2 管理因素

在软件生产的管理中，存在各种不利于软件复用技术发展的现象。一些与复用的目标不协调的制作与政策。管理者不信任复用所带来的商务价值。企业文化和价值观念阻碍复用的实施。在项目开发的开始，没有以向着造就可复用构件的方向努力，妨碍了复用水平的提高和复用规模的扩大。开发人员不愿意改变当前的工作方式。要想从软件复用技术中获得巨大利益，管理上则需要更加重视和提高。