浅谈计算机软件重构技术

　　软件重构是指在不改变软件的功能和外部可见性的情况下，对软件进行的改造，以下是小编搜集整理的一篇探究软件重构的论文范文，欢迎阅读查看。

****摘要：****本文阐述了软件重构技术的现状，研究意义和发展前景。

****关键词：软件重构 原型开发 简化 复用 测试****

****1、软件重构的定义****

　　软件维护最初最贴切的形容就是“打补丁”，所有的维护工作采用的都是一种消极但快速的方法。如果发现系统有问题，维护人员就快速找到这个问题，然后修改源代码，解决这个问题。如果要添加一个新的功能，那就找到一个相似的模块，修改代码以后放进去。这样反复多次之后，系统变得越来越臃肿，越来越难于理解，维护也越来越艰难、越来代价越高。人们期望解决这一问题的根本办法，于是软件重构技术应运而生。通过这种技术，使得编码更加谨慎合理，编码时如果非常小心，则代码会具有良好的形状和味道，在修改添加代码前，通过查看代码的形状和味道来决定是否进行更新。如果代码不容易被理解，则必须重构代码以使得维护变得容易，再进行相应的扩展和更新。

　　软件重构是指在不改变软件的功能和外部可见性的情况下，为了改善软件的结构，提高清晰性、可扩展性和可重用性而对软件进行的改造。简而言之，重构就是改进已经写好的软件的设计。也就是说，改进后的软件其原有的功能和行为都不改变，改变的仅仅是程序内部的编码。重构是在源代码级别上进行的，目的是使原有的代码在重构后具有良好的程序结构。

****2、软件重构的意义****

　　计算机技术的发展带来了全社会的变革，人们的生活对计算机的依赖越来越强，所使用的软件也越来越复杂，对开发软件所使用的技术也要求越来越高。面向对象开发方法等有利于软件利用的方法和技术得到了广泛的运用。但同时也显示出一些弊端。为了开发出使用方便，利于维护的软件系统，软件开发人员希望把初始系统设计得尽可能的完美、灵活、具有良好的可扩展性。这不仅要求软件开发人员具有较高的软件设计能力，而且，再完美的设计也不可能完全适应未来的需求变化。解决这一问题的方法是软件工程中的原型开发方法，在原型开发方法中，系统开发人员首先快速开发出一个系统原型提供给用户使用，通过用户在使用过程中的不断反馈来对系统进行修改直到最后开发出系统的成熟版本。软件扩展是在软件重构的基础上完成的。在需要扩展时，使用软件重构方法和重构工具，采取提取抽象数据类型、提取接口等技术对代码结构进行改造，以利于软件的维护、功能的增加等活动。

　　对于已经投入使用的系统，由于需求、业务、法规等因素的变化，也会因此而改变，需要在原有的系统上进行修改和扩充。在维护过程中可能会遇到设计风格不一致、过时的编程语言、系统文档过时，不完整和难于理解的问题。这时候，为也延长旧系统的生命周期，降低系统维护成本。就可以采用以原有系统进行重构的方法对软件进行重构，使其在改善程序结构的基础上适应需求的变化。

　　软件重构也能为程序员的编码带来好处，首先，可以简化设计，重构允许系统设计刚好满足现有需求，不必过多考虑未来的需求，降低了系统设计的复杂程度，当有新的需求时，可以利用重构技术，快速进行设计扩展。其次，可以增强程序的可读性，便于理解他人的编程目的，利于快速开发软件系统。再者，软件重构可以使程序的结构更加清晰，减少编程过程中出现的问题，并且，由于重构是建立在严格的测试之上的，所以能够快速发现问题。将复杂的软件测试分解为递进式的简单测试，降低了测试的复杂度。最重要的是重构可以使得代码的可理解性大大提高，原因在于，重构技术支持更小的类、更短的方法、更少的局部变量、更小的系统耦合，重构可以让程序员养成良好的编程习惯。

****3、软件重构的研究现状****

　　软件重构的研究紧紧围绕以下几个方面展开，目前已经取得了一些成果。

　　(1)基本重构方法和复合重构方法的研究，如Opdyke针对C++语言定义的一组基本重构方法，在其中定义的每个基本重构方法中，由前提条件和执行动作组成。在进行重构时以每个基本方法为重构的最小单位。通过执行动作对程序的修改步骤进行描述，前提条件是用来保证程序行为的不变。将这些基本重构方法组合起来形成复合重构，就可以用来解决复杂的'重构问题。Roberts等人受Opdyke研究的启发，对基本重构方法进行了改进，增加了后置条件约束，讨论了复合结构中基本重构之间的依赖性问题，提出了计算复合重构前置条件的计算方法，并开发了针对Smalltalk的重构辅助工具。

　　(2)不良程序结构的探查和整理。如冗余代码探查和消除、类层次的整理、横切关注的探查和封装等。其中冗余代码的探查和消除主要使用字符串比较的方法，类层次的整理主要使用“渐进增加类”算法实现，而横切叛党的探查和封装主要使用面向方面范型进行研究和实践。

　　(3)程序理解方法和工具。要进行软件维护和软件重构必须在理解程序结构的基础上施行。除了针对程序代码中的不良程序结构进行探查，还需要围绕程序的设计模型进行分析。如Pauw等人开发的Ovation系统，使用了执行模式视图在不同的抽象层次上对程序的执行进行可视化，并对生成的模型进行一些操作。Storey等人开发Rigi逆向工程环境则使用有向图表示软件构件和它们之间的关系。

　　(4)面向典型设计的重构方法。如Tokuda和Batory探讨了采用手工方式对程序应用设计模式的重构的方法，他们通过实验指出：一个典型的系统在使用重构和引入设计模式后，可以明显地增加软件质量。Tonella和Potrich提出了从C++代码中自动抽取UML交互图的方法。这个基于静态的程序分析技术的方法，基本可以描述在任何输入下程序的结果。将对象交互图的抽取过程分两步：第一步，从源码中推断出程序所创建的对象和程序中变量;第二步，把方法的每一个调用分解成交互过程中包含的发送类和接收类。

　　(5)软件重构辅助工具。如Opdyke开发的重构辅助工具C++重构工具，Roberts等人开发的Smalltalk Refactory Browser等。这些工具虽然没能成为实用的工具，但却是后续研发工作的基础。