

设计模式在软件架构可靠性评估平台中的应用^①

Application of Design Patterns in Software Architecture Reliability Evaluation Platform

张 伟 陈未如 (沈阳化工学院 计算机科学与技术学院 辽宁 沈阳 110142)

摘 要: 软件架构设计对软件质量有很大影响。本文设计了软件架构可靠性评估平台, 可用于在软件设计阶段对基于软件架构的软件可靠性进行评估。在该平台的设计开发过程中使用了一些设计模式, 包括组合模式、抽象工厂模式、访问者模式、原型模式等, 从而优化了系统结构, 使系统具有很好的灵活性。

关键词: 设计模式 软件架构 可靠性评估平台

1 软件架构及架构评估平台

软件架构(Software Architecture, 简称 SA)是在 1969 年的 NATO 软件工程会议上首次提出的^[1], 直到 1995 年 SA 才被确立为软件工程一个研究方向。软件或计算系统的软件架构就是该系统的一个或多个结构, 它包括该软件系统的各个组件、这些组件的外部属性及相互关系^[2:6-8]。

一个好的软件肯定有一个好的架构, 一个不好的架构肯定不会产生一个好的软件^[2:31-34]。软件架构可靠性评估平台可以用于在软件的设计阶段对软件的架构进行可靠性评估。如果基于软件架构的软件可靠性达不到指定标准, 则需要考虑对软件架构进行调整或重新设计。

使用软件架构可靠性评估平台应能进行软件架构设计或表示, 并对基于设计架构的软件进行可靠性评估。

在软件架构可靠性评估平台设计开发过程中, 采用了多种设计模式, 包括组合模式、抽象工厂模式、访问者模式、原型模式等, 目的是保证平台架构质量优良, 以避免代码重复、低内聚、高耦合、类设计面向对象差、结构复用性差等问题。

2 设计模式

模式最早出现在建筑大师 Christopher Alexander 的关于城市规划和建筑设计的著作中, 他

曾在其著作中指出: “每一个模式描述了一个在我们周围不断重复发生的问题, 以及该问题的解决方案的核心。这样, 你就能一次又一次地使用该方案而不必做重复劳动^[3]。”

起源于建筑领域的模式提供了一系列建筑相关问题的解决方案, 从而使人们不必做重复劳动。同样在软件设计中, 也会遇到不断重复出现的问题, 将这些问题的解决方案归纳总结就形成了今天的设计模式。设计模式的诞生是计算机软件设计的里程碑, 它对软件行业的发展有至关重要的作用。

计算机科学中对设计模式的简单定义就是对于一类重复出现的问题的一种可重用的解决方案, 在软件工程中一个设计模式对应解决一类软件设计问题^[4]。设计模式是成功的软件架构设计经验总结, 是被实践证明的可复用的解决方案。设计模式是在软件设计发展到面向对象设计阶段提出来的, 它解决了软件开发中有关对象的创建、结构和行为等一系列问题。如果多个项目有相同的问题背景, 那么可以直接套用相应的设计模式加以解决。在软件设计中使用设计模式可以减少各个类之间的依赖和耦合, 增强结构复用性, 减少因变更所做的设计调整。设计模式分为三类: 创建型、结构型和行为型。创建型模式是创建对象而不是直接实例化对象, 这会使程序在判断给定情况下创建哪一个对象时更为灵活; 结构型模式可以将一组对象组合成更大的结构, 例如复杂的用户界面或报表数

^① 收稿时间: 2008-09-03

据等；行为型模式定义系统内对象间的通信，以及复杂程序中的流程控制[5]。

3 设计模式应用

3.1 软件架构可靠性评估平台结构

软件架构可靠性评估平台设计有很多类，包括基类 CAevBase，架构类 CAevArchi，组件类 CAevComponent，连接件类 CAevConnector，配置类 CAevConfigure，端口类 CAevPort，以及角色类 CAevRole 等,其中 CAevComponent、CAevConnector、CAevConfigure、CAevPort 和 CAevRole 都是 CAevBase 的派生类。各类之间的关系在组合模式、抽象工厂模式、访问者模式、原型模式中得到了很好体现。

各个模式都有各自的适用范围。当你想表示对象的部分-整体层次结构时，可以使用组合模式，组合模式的优点是方便增加新类型的组件，缺点是很难限制组合中的组件[6:107-109]。当一个系统要由多个产品系列中的一个来配置时，可以使用抽象工厂模式，抽象工厂模式的优点是客户和具体产品的实现分离，易于交换产品系列，缺点是难以扩展抽象工厂以生产新种类的产品[6:58-60]。当定义对象结构的类很少改变，但需要在此结构上定义新的操作时，可以使用访问者模式。访问者模式的优点是元素类和操作类是分离的，当增加新的操作时不必修改具体的元素类，缺点是当每增加一个元素类时，需要改动操作类及其子类[6:219-222]。当一个类的实例只能有几个不同状态组合中的一种时，可以使用原型模式，原型模式的优点是增加和减少产品类不会影响整个类结构，缺点是每个原型子类都必须实现一个克隆方法[6:78-80]。

3.2 组合模式

组合模式的意图是将对象组合成树形结构以表示“部分-整体”的层次结构,该模式使得用户对单个对象和组合对象的使用具有一致性[7]。组合模式是一种结构型模式，它可以表示成一种树型结构，组合模式中的类可以是叶子结点也可以是一个树型结构。组合模式应用于一个复杂类由多个类组合而成的结构中。组合模式原理如图 1(a)所示。软件架构是由组件、连接件和配置组成，组件是架构中的功能单元，连接件是功能单元之间进行交互的通道，配置用来说明组件和连接件的连接关系[8]。软件架构不仅是由组件、连

接件和配置组成的，并且架构又可以看作是一个特殊的组件，这正适合使用组合模式设计类结构。组合模式应用如图 1(b)所示，其中 CAevArchi 类相当于 Composite 类，CAevBase 类相当于 Component 类。

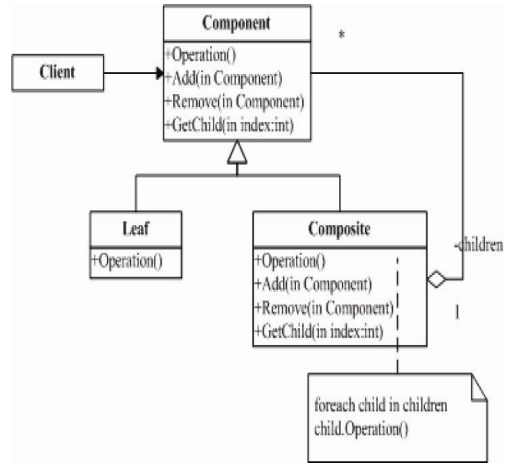


图 1(a) 组合模式原理图

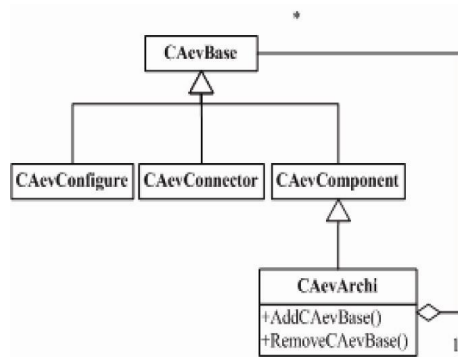


图 1(b) 组合模式应用图

3.3 抽象工厂模式

抽象工厂模式的意图是提供一个创建一系列相关或相互依赖对象的接口,而无需指定它们具体的类[6:57-58]。抽象工厂模式是一种创建型模式，它的作用主要是根据客户指定的参数在工厂类方法中创建相应类的实例，即客户需要的产品。对于客户而言，创建产品的过程是不透明的。抽象工厂模式原理如图 2(a)所示。该模式涉及工厂基类，产品基类和各个具体实现类。在软件架构可靠性评估平台中 LibFactory 是抽象工厂基类，ComLibFactory 和 ConLibFactory 是具体子类，用于作具体实现以生产相应的产品，例如生成组件或连接件实例，该实例对象需要转换为基类对象以便统一使用。抽象工厂模式应用如图 2(b)所示。

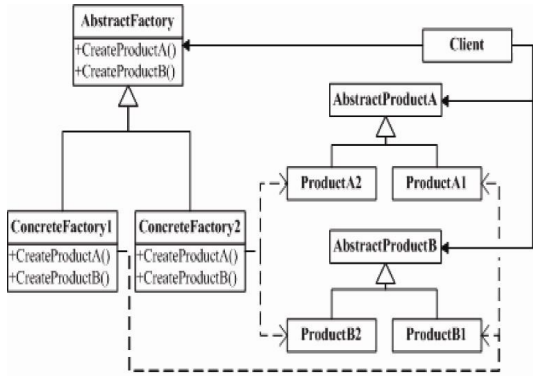


图 2(a) 抽象工厂模式原理图

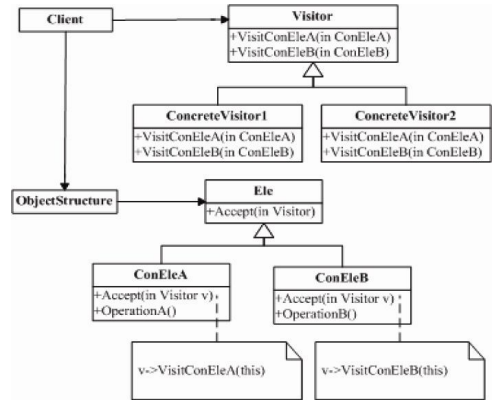


图 3(a) 访问者模式原理图

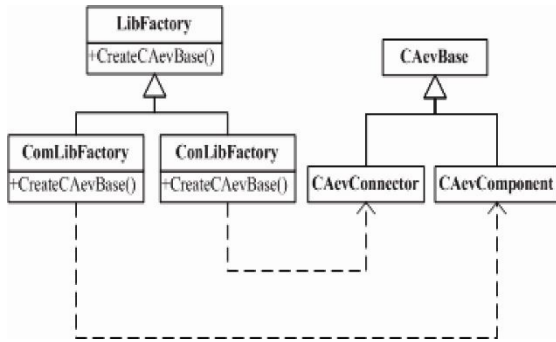


图 2(b) 抽象工厂模式应用图

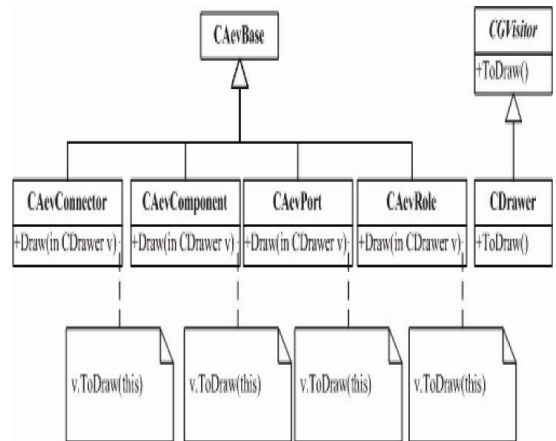


图 3(b) 访问者模式应用图

3.4 访问者模式

访问者模式的意图是表示一个作用于某对象结构中的各元素的操作，它使你可以在不改变各元素的类的前提下定义作用于这些元素的新操作[6:218-219]。访问者模式是一种行为型模式，它定义了一系列对某些对象或对象结构中的元素的操作，因某些操作具有一些共性，所以把这些操作同对象分离，操作被提取出来单独设计成类，当操作发生变化需要修改时不影响类结构。访问者模式原理如图 3(a)所示。软件架构可靠性评估平台中组件、连接件、端口、角色都有 Draw 方法，并且如果在每一个类中都设计一个 Draw 方法，那么有一些代码重复，例如定义画笔、钢笔和字体，释放画笔和钢笔等操作。使用访问者模式，需要定义一个外部类，它可以对其他数据进行访问。由外部类派生一个或多个对相应对象的操作类，这些操作类对相应对象进行操作，对象只能访问和调用自己的操作。该平台 CGVisitor 是外部类，CDrawer 是组件、连接件、端口、角色等对象的操作类。CAevComponent、CAevConnector、CAevPort 和 CAevRole 这些类是相应对象类。访问者模式应用如图 3(b)所示。

3.5 原型模式

原型模式的意图是用原型实例指定创建对象的种类，并且通过拷贝这些原型创建新的对象[6:77-78]。原型模式是一种创建型模式，它是和抽象工厂模式配合使用的，它解决了抽象工厂模式不能解决的创建子类过多的问题，并且它还解决了抽象类与具体类相依赖的问题。原型模式原理如图 4(a)所示。该模式将抽象类与具体类的依赖关系转化为抽象类与抽象类的依赖关系，这样减少类与实现类之间的固定依赖，当依赖关系扩展时便于更改。该模式确定了原型类，并指定创建对象的种类，通过原型实例克隆生成新对象。这里原型类是 CAevArchiStyle 类即架构风格类，该类生成的对象可能由一个或几个 CAevComponent 子类和 CAevConnector 子类组成，通过传递 CAevBase 类型参数来生成 CAevArchiStyle 对象，通过 CAevBase 对象克隆出 CAevArchiStyle 类的包含对象。原型模式应用如图 4(b)所示。

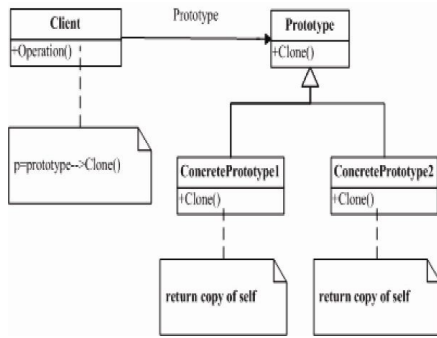


图 4(a) 原型模式原理图

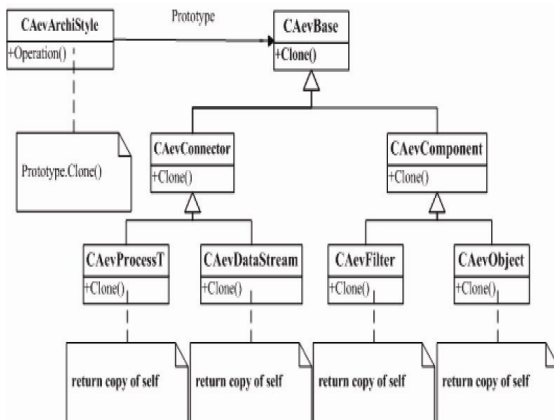


图 4(b) 原型模式应用图

4 结论

在软件架构可靠性评估平台中通过应用组合模式、抽象工厂模式、访问者模式和原型模式等设计模

式使架构易于扩展，并提高了平台的适应性，使松紧耦合适度，最大程度满足高内聚、低耦合，同时优化了系统结构，使系统开发更灵活，具有很好的灵活性。

参考文献

- 1 Kruchten P, Obbink H, Stafford J. The Past, Present, and future of software architecture. IEEE Software, 2006,23(2) :22 – 30.
- 2 Bass L, Clements P, Kazman R.孙学涛,杜学绘,刘冬萍译.软件构架实践.北京:清华大学出版社, 2002.
- 3 Alexander C, Ishikawa S, Silverstein M, Jacobson M, Fiksdahl-King I, Angel S. A Pattern Language. New York: Oxford University Press, 1977.
- 4 Gamma E, Helm R, Johnson R, et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Publishing Company, 1995.
- 5 Cooper JW. 张志华,刘云鹏,等译. C#设计模式.北京:电子工业出版社, 2003:2 – 5.
- 6 Gamma E, Helm R, Johnson R, Vlissides J.李英军,马晓星,蔡敏,刘建中等译.设计模式:可复用面向对象软件的基础.北京:机械工业出版社, 2006.
- 7 聂颖.设计模式在图形处理软件中的应用.计算机应用, 2004,24(S2):189 – 191.
- 8 冯冲,江贺,冯静芳.软件体系结构理论与实践.北京:人民邮电出版社, 2004:9 – 14.