**面向对象设计中软件指标与代码可重用性之间的关系构建**

摘要：就生产成本效率而言，设计模式在面向对象设计的软件度量和内部代码体系结构形式中的作用在软件工程中起着至关重要的作用。本文讨论了代码可重用性，它是IT生产中经常使用的节省成本的方法。在回顾了有关软件度量研究的现有文献之后，我们发现很少有研究证明倾向于代码可重用性。因此，我们开发了一个简单的分析模型，该模型使用三个软件项目（客户关系管理项目，供应链管理项目和企业关系管理项目）的案例研究来建立标准软件度量的设计组件与代码可重用性之间的关系。我们还使用随机的基于Markov模型验证了我们的提议，发现所提议的系统可以随着软件项目方法的不确定性水平的提高而提取代码可重用性最大化价值的重要信息。

1. **引言**

在当今世界，每个行业或服务部门都依赖于基于计算机的应用程序。为了提高性能并获得竞争优势，软件的质量已成为至关重要的因素。在世界许多地方，软件服务的开发和外包是一个快速发展的主要行业[1]。软件开发的过程通过术语“软件工程”来描述，该术语指的是在系统目标标准的上下文中使用系统性过程来执行软件的分析，设计，实施和测试以及维护。这样开发的软件必须可靠，可用，高效，可修改，可测试，可维护，可互操作，可移植且准确[2]。在软件开发过程中，面向对象的设计被认为是评估软件质量的重要特征之一[3]。无论组织的规模如何，任何组织中的软件开发都主要采用面向对象的设计方法。因此，面向对象的设计被视为标准，系统对象可以通过该标准具有特定的功能以及必要的特性。采用面向对象方法的原因在于，它能够可视化问题并在与对象相关的所有宏观和微观层面上获取解决方案，还可以确保更好的可靠性，适应性，灵活性和可重用性[4]。当前，软件工程师使用软件指标来评估某个软件项目的设计组件和必要资源。软件度量标准的优势在于，它允许通过更好的平台评估设计模式，并有助于以定量方式进行应用程序测试。通过这样的测试，可以证明软件的可靠性。 通常，当公司从客户那里收到新要求时，他们会先按照要求制定一个设计方案。被架构师确认的方案将被发送到产品制造部门。编码完成后，再将产品发送给客户。尽管重用早期客户端的代码以开发新客户端的应用程序是不道德的[5]，但是代码可重用性也处理了知识产权的安全性。生产团队若从头开始进行全新的开发，这不仅需要努力，而且需要大量的时间和金钱。现在，大多数大型组织都使用设计模式，这些设计模式可以重复使用而不会产生道德问题。设计重用的主要目的是帮助开发人员在新产品中使用它，这有助于减少重新开始的开发成本。但是，在重用设计时应格外小心，以使当前和将来的客户都能最佳地重用设计。采用设计重用还将有助于确保及时的生产和交付过程。设计重用并不意味着整个设计都被使用； 可能会在新的或将来的类似项目中使用当前设计的一定比例。因此，设计团队的重点应该是使设计不仅专注于现有客户，而且还应该能够为将来的客户提供最小比例的可重用设计。但实际上这并不容易，因为客户的未来需求是不可预测的。

第2节概述了软件指标。第3节重点介绍CK指标的关键点。第四部分讨论相关工作。接下来是第5节中的问题识别。第六部分讨论了提议的系统。第7节讨论研究方法。 第8节提供了提议系统的结果和结果分析，最后，第9节提供了结论和未来的工作。

1. **关于软件指标**

在这个IT革命时代，出现的软件开发在日常生活的每个阶段都至关重要。从学术到公共服务，医疗保健到银行，娱乐到体育，软件的使用以这样或那样的方式促进了这些领域的发展。促进软件开发进步的各种因素是其关键功能，例如灵活性，可移植性，设计可重用性，软件指标等。设计可重用性在软件开发中起着至关重要的作用，因为它不仅参与当前工作的开发，而且为当前以及新项目的未来需求奠定了蓝图。设计可重用性还有助于降低生产成本，减少所需的人力和开发时间，因为它提供了一个由设计组成的框架，可以在一段时间内重用。软件开发的另一个重要因素是软件度量，它定义了对过程或软件系统将具有某些属性进行度量的标准度。在软件工程中，可以使用不同的度量来度量不同的参数，例如使用过程度量来度量过程，使用项目度量来度量项目以及使用产品度量来度量产品。在我们的工作中，由于涉及到面向对象的设计，因此本文将重点介绍与面向对象的度量标准相关的一些内容。为了开发面向对象设计的度量标准，下面列出了七个不同的可测量的特质[6]。

1. 复杂性：通过评估类之间的关联方式进行分析。
2. 耦合：这是面向对象的元素之间的物理连接。
3. 充分性：它定义了抽象应具有其所需功能的程度。
4. 内聚性：通过分析作为问题域或设计域部分的类的属性组来定义。
5. 原始性：用于指示原子操作级别的程度。
6. 相似性：用于在行为，结构，目的或功能方面定义类之间的相似性。
7. 易失性：用于定义面向对象设计中发生更改的可能性。

**2.1规格**

使用四个不同的视角，例如容量（volume），长度，数量(population)和功能。通过评估OO实体的总数（以类或操作的形式）来进行总体测量。容量的测量是在任何时间动态地实现的。功能性表示为由面向对象的应用程序提供给用户的值。 使用诸如继承深度之类的互连（interconnected）设计，可以测量树的长度(length)。 面向对象的设计指标/度量标准(metric)集中于适用于类和设计特征的测量（measurement）。 通过这些测量，设计人员可以在早期处理阶段访问软件，从而可以进行更改，以最大程度地减少复杂性并增强设计的持续能力[7]。

1. CK-度量/指标（metirc）

1994年，Chidamber和Kemerer引入了面向对象程序的标准软件指标(metric)。 CK度量（metric）在了解软件的设计方面和提高软件质量方面起着至关重要的作用。 CK指标的主要目的是向每个类级提供对软件程序的累积质量的详细测量。指标与每一个软件的小部分相关联，他还提供了软件质量的整体信息。 面向对象程序的六种基于类的度量指标，如下所示：

类的加权方法（WMC）

用于定义类的复杂度总和。 总体而言，它代表了类的复杂性，并且该措施可用于指示类的开发和维护工作。

类的响应（RFC）

此指标表示集合中增长方法的数量（number of growing methods within a set,），它可以响应于发送给执行特定任务的对象的消息

继承树深度（DIT）

这是常用指标之一； 它用于估计类层次结构中的深度范围。 它还用于评估可维护性和可重用性。

孩子数（NOC）

这是对在继承关系下与特定类关联的类的数量的测量。 一个有很多孩子的类意味着一个糟糕的类，并且设计不好。

对象类之间的耦合（CBO）

这定义了与其耦合的所有其他类集的数量。 CBO在确定测试和可重用性的复杂性方面是有优势的。

方法内聚性的缺失（LCOM）

它是相似度为零的方法数量与相似度为非零的方法数量之间的不同。 两种方法之间的相似之处是共同使用的特征数量。 LCOM中的零并不表示内聚能力，高值也不表示任何推断（inference）。 在LCOM中，很难定义单位并测量特质。 不建议使用LCOM进行准确的测量，因为它无法正确量化特质。

面向对象的度量指标已在世界不同地区的不同领域和编程语言中成功使用。 这些度量指标始终如一地说明了与质量因素的关系，如重用，缺陷，成本以及可维护性和可能超出规模的关系。 这些度量指标的评估是通过某些工具来实现的，例如metamill [8]，metric1.3.6 [9]，Analyst 4J [10]，OOmeter [11]和Dependency finder。

1. **文章的评论**

本节讨论现有文献，其中介绍了各种软件度量设计方法及其贡献。 多年来，已经开发出许多方法来解决使用度量标准来检测和纠正面向对象（OO）软件系统中的设计缺陷的问题。 此外，随着引入的软件度量标准越来越多，项目经理发现很难解释和理解度量标准分数。 由于面向对象的度量标准需要对面向对象的概念有很好的理解，而且，目前还没有单一的度量标准可以提供面向对象的软件系统的所有功能。 表1显示了有关使用指标的面向对象（OO）软件系统中设计缺陷的现有调查。 表1将突出显示关于各种问题和用于增强软件工程中软件指标性能的各种技术的列表讨论。

1. **问题识别**

软件工程在成功交付以质量为导向的项目中发挥了重要作用。 上一节中讨论的现有文献讨论了各种用于增强软件指标原始性能的技术。 发现大多数技术使用定量的，经验的，基于树的技术。 一些独特的进化技术，例如 神经网络，模糊逻辑等也已被使用。 所有上述所有技术都有可能的优点和缺点。 在本节中，我们将对现有系统进行检查，讨论所发现的问题，如下所示：

**5.1缺乏基准研究**

除了90年代引入的CK度量外，我们还没有对软件度量进行过任何有力的讨论，尤其是面向对象的编程。 尽管互联网媒体中存在大量论文，但是很少有作者在他们的思想中引入任何新奇的东西。 一些想法是实施CK度量或在旧的参数方程式上引入新的数学公式，例如 DIT，WMC，NOC等。同样，我们还没有遇到其他研究人员发现或使用过的任何研究模型都对代码可重用性有很大帮助。

**5.2没有关于CK度量关系的研究**

所有介绍软件度量套件的论文100%都有自己的公式/表述。作者通常检查与CK度量相关的问题，并尝试引入新的软件度量套件。然而，在过去并没有重大的尝试去调查CK度量与代码可重用性之间的关系，但这（重用性）在任何IT行业或任何生产公司都是非常普遍的操作。

**5.3缺乏对代码可重用性的关注**

代码可重用性是任何IT产品中的普遍做法。 但是，它在全球研究界中受到的关注很少。 关于代码可重用性与软件度量之间可能的关系的研究较少探讨。 在个案研究的帮助下，本文以简单公式的形式解决了上述这些问题。 下一节将讨论所提出的系统，该系统可增强代码可重用性的性能，并使用户能够可视化软件指标与代码可重用性之间的足够统计信息。

1. **所提议的系统**

拟议的研究旨在开发一种分析框架，该框架可在具有代码可重用性的不同类型的CK-Metrics组件之间建立关系。 所提出的系统包括两种不同类型的实验原型，它们是：i）用于估计代码可重用性的建模，以及ii）用于评估设计指标对代码可重用性的影响的框架。过去进行的所有研究都极大地推动了这项提议的研究，这表明可以通过执行有效的代码可重用性和实现各种测量驱动的模块来实现软件质量的提高。 评估代码可重用性指数的建模（模型）考虑了以一个流程流，其中通过设计和代码构件以及领域知识和经验对CK度量数据进行了进一步的经验分析。可以从UML（统一建模语言）类图或等效的Java代码获取CK-Metrics数据。 有各种处理工具，例如Rational Rose等，可用于从代码工件中提取度量数据，并有望在CK-Metrics成分之间建立联系，例如WMC（每个类的加权方法），DIT（继承树的深度） ，NOC（孩子数），CBO（对象类之间的耦合），RFC（类的响应集），LCOM（缺乏内聚的方法）等。实证分析（经验分析）生成了有关属性对代码可重用性影响的数据，并建立了属于不同类型CK-Metrics成分的各种多功能评估方程之间的关系，并将它们与代码可重用性进行映射。 最后，使用GQM范例和来自设计指标的加权因子，实现了用于估计代码可重用性的软件框架。 通过框架进一步处理了代码可重用性模型，该框架使用静态类图和动态序列图计算不同类型的CK-Metrics成分。提出的模型还评估了与数据相关的经验分析，该分析还生成了各个指标对代码可重用性的影响，即代码相对于各个设计指标的线性组合。拟议的系统还将CK-Metrics数据作为输入并评估代码的可重用性指数，可以在软件开发生命周期中将其进一步映射到CK-Metrics成分中。现有的各种传统研究都基于使用经验原型的代码可重用性设计指标。 CK指标标准组件与代码可重用性之间的关系是基于遵循如下传统的框架而开发的：

1. 可以看出，具有较高的“继承深度”值的类将具有较高的代码可重用性概率
2. 结合力/内聚性(cohesion)低的类可带来更好的软件设计和代码可重用性。
3. 由WMC和NOC的较高值组成的类扩展了软件开发动态场景中的代码可重用性。
4. 具有较高CBO和RFC值的类增加了计算复杂性，并使代码的可重用性最大化。

提出的模型还引入了代码可重用性的框架，该框架已使用深入的经验分析和数据建模进行了评估。 经验模型考虑了两种不同类型的中高级项目，其中进行了实验分析，考虑了大量的类来调查设计模式的度量标准的代码可重用性。与每个项目相关的类根据不同的度量值进行配置和分组，以避免知识产权问题。拟议的研究开发了一个有效且计算效率高的框架。 拟议研究的贡献包括：i）确保对异构对象软件模块的代码可重用性进行估计； ii）计算加权多项式方程的线性组合； iii）在CK-Metrics组件与代码可重用性之间建立有效关系 。对与经验模型相关的性能指标进行了评估，以确保所提出系统的有效性。下一部分将讨论有关研究方法，该方法详细阐述了设计质量指标与代码可重用性之间的关系。

**7.研究方法**

所拟议的系统是借助分析建模方法作为研究方法的标准而设计的。该系统的设计基于代码复用的概念，其使用了ck -metric构件。图1突出显示了代码可重用性的建模，其中从UML中提取了CK度量数据。可以使用多种形式的与Java相关的案例来获取CK度量。 在这方面，可以使用各种与工业相关的工具，例如Metamill，Metric 1.3.6，rational rose 等来获取CK-metrics的组件。但是，可以手动使用类图来估计类的数量。 可以借助静态类评估各种参数，例如 DIT，WMC和NOC，而各种形式的顺序图可用于评估其他指标，例如RFC和CBO。然而，很重要的一点是不能从设计模式(如UML)中直接评估LCOM。但是，出于相同的原因，可以使用基于行业的复杂自动化工具。

插入“代码重用性原理示意图”

为了评估与设计相关的问题（即代码可重用性），我们考虑使用Java开发的示例项目，其中包含大量类。 拟议的系统旨在了解CK度量的各个组件如何影响代码的可重用性。我们开发了一个简单的功能，以便在指标和代码可重用性之间建立关系。 我们使用加权系数的概念以及线性方法来评估CK度量对代码可重用性的可能影响分析。我们还使用GQM(goal-questioon-metric)的概念作为研究方法的核心部分，该方法允许基于操作级和定量级制定代码可重用性的概念级别。

# 7.1假设（hypothesis）提出

正如前面所讨论的，所提议的研究是为了了解CK指标的成分部分与代码可重用性之间的潜在关系，因此，为此构建了一个适当的假设。该研究对各种基于Java的ERP（企业资源计划）和SCM（供应链管理）相关软件项目进行了分析评估，并遵循原假设进行了构建。

Ho1：每个类中的DIT值适中，可以保留更好的代码可重用性。

Ho2：最大化RFC值，代码设计的复杂性和可重用性降低。

Ho3：代码重用性随类中存在的NOC值的增加而降低。

Ho4:假设中CBO值的增加对代码可重用性影响不大。

Ho5：随着每个类别中WMC值的增加，代码的可重用性下降

**7.2开发分析模型**

拟议系统的开发是使用分析建模方法来检验假设的。 该系统还应用简单的数学估计技术来研究CK度量的组件与代码可重用性之间的可能关系。 该系统考虑了用Java开发的ERP和SCM相关软件项目的案例研究。 ERP软件项目考虑的总数类为220。其中大约有180个基类。 同样，SCM软件项目共有大约570个类，可维护的类约为380个。 虽然，我们的技术可以包括更多的软件项目，但是我们选择考虑仅使用ERP和SCM软件项目。CK度量的成分（组件）有多个值，这些值可以更适当地安排或构造。提出的系统对代码的可重用性进行了简单的分析建模：

插入数学公式

上面的等式（1）表示代码可重用性的数学表示，其中α被认为是每个CK度量（指标）

中可用的类的总数，而β是CK度量中相当于通过结合代码可重用性而新设计的类的数量的组成成分。 这意味着Cr值越高，代码可重用性的程度越高。手动进行α和β参数的评估，并通过电子表格记录下来。然而，在结果分析中所考虑的值被近似化以得到对比结果，用于研究软件工程中CK度量的组件对代码可重用性的影响。最后，针对不确定性的存在，对分析模型进行了验证。

**8.结果分析**

本研究采用SPSS[35]工具进行分析。我们同时进行数值分析和图形分析来评估结果的有效性。表1显示了ERP和SCM软件项目的案例研究的数值结果，其中使用前一节中说明的简单公式1，密切观察和计算了必要的CK度量组件，以确保代码的可靠性。下面是对拟议系统图形结果的讨论。

**8.1 DIT（**继承树的深度**）对代码重用性的影响**

如图2所示，结果表明DIT的值对代码可重用性有了显着改善。仔细研究DIT的数值可以发现，随着参数α和β值的增加趋势，代码可重用性得到了显着提高。越来越高的DIT级别无疑将代表计算的一点点复杂性； 但是，使代码更可重用绝对是有效的。 还表明了，对于大规模的面向对象的设计，很有可能类别的数量急剧增加，也会增加DIT值。但是，使用拟议的系统，使用适中的DIT值实际上可以在代码可重用性方面保留更好的趋势。 因此，认为“每个类别中的DIT值适中，可以保持更好的代码可重用性”的原假设被接受并成立。

**8.2RFC**（类的响应集）**对代码重用性的影响**

图3显示了一些有趣的代码重用趋势。虽然Cr趋势是增加的，但是在RFC度量的初值中趋势不够平滑。此处探讨的基本趋势是，随着α和β的增加，代码的可重用性因子得到了增强。然而，它也带来了度量值(1-7)主要范围内的复杂性，这意味着尽管代码可重用性增加了，但它也带来了设计上的显著复杂性。出于分析目的，我们使用两种形式的数据来生成图形化结果。第一个数据是合成数据，可以在表2的RFC行中看到，其中参数α和β保持递增和递减顺序的组合。 结果表明代码可重用性提高，然而，这与对于复杂的ERP和SCM项目，α和β的值通常按升序排列的情况则不同。 因此，我们使用对α和β的递增顺序来绘制图形，以研究对代码可重用性的可能影响。

图3显示了综合数据的结果，其中参数保持在随机状态，这在复杂的软件项目中不太可能发生。 尽管结果表明代码的可重用性有所提高，但是结果可能存在潜在的前提。因此，我们使用来自ERP和SCM项目的原始数据，其中RFC参数（α和β）以递增顺序构造。 结果表明，代码可重用性显着下降。 因此，图4中陈述的结果符合以下假设：“为实现RFC的最大结果，代码设计的复杂性和可重用性降低了”。

**8.3 NOC**（孩子数）**对代码可重用性的影响**

图5突出显示了NOC对代码可重用性的影响。 结果表明，使用表2中提到的数值，代码的可重用性得到了显着改善。仔细查看这些数值，可以发现参数（α和β）处于最大化顺序的相同趋势。因此，总的来说，可以说NOC的增加导致了代码重用性的提高。 其背后的主要原因是NOC值越大，代表的基类数量就越多，这会导致代码的可重用性。 因此，“增加类中存在的NOC的值以提高代码中的可重用性”的原假设被接受。

**8.4 CBO**（对象类之间的耦合）**对代码可重用性的影响**

据说CBO值较高对软件工程不利，因为过多的耦合会使设计评估变得非常困难和复杂。 但是，使用面向对象的软件项目可能具有比预期更高的CBO值。因此，即使软件项目确实具有较高的CBO值，所提出的系统也可以大大提高代码的可重用性。图6显示了对于任何形式的面向对象代码，随着CBO值的增加，获得更多代码可重用性是可行的。 因此，假设“ CBO值增加对代码可重用性影响不大”的假设被接受。

**8.4 WMC**（类的加权方法）**对代码可重用性的影响**

从理论上讲，可以预期WMC值的增加可能会导致设计复杂。 但是，当我们使用提出的技术进行评估时，我们发现在某些初始回合中，对代码可重用性没有太多不利影响。仔细观察图7所示的结果，可以看到尽管代码重用性由于类中使用的各种方法而增加，但同时发现性能随着WMC的参数（α和β）的进一步提高而下降。 因此，在本系统中，“当WMC在每个类中的值增加时，代码重用性下降”的原假设被发现是可接受的。

仔细研究该研究的累积结果将发现，对于某些CK度量参数，代码可重用性增加，而对于某些其他CK度量参数，代码可重用性降低。 我们的观察需要更多的处理和分析时间来进行测试。 我们的工作方向是增强重要方法之间的内聚性，并确保潜在对象之间的耦合最小。此条件通过确保类之间的较高内聚性，确保了保留更好的代码可重用性。 如果耦合强度大于我们发现的情况，那么这种情况就完全不支持代码可重用性。 此外，还探讨了在ERP项目中包含更多方法会导致设计类产生更多的计算复杂性，从而导致劣等的设计模式。 超过150条消息的内部回调和通信导致代码可重用性下降。

从本节讨论的结果可以看出，有很多CK度量因素直接影响代码的可重用性过程。在研究新的代码集期间，至关重要的是，设计的新组件（成分）和类必须必须在预期的结果之内，这样才能声明代码是可重用的。因此，我们提出的技术可以提供一个可用于测量CK度量与代码可重用性之间关系的框架。 用于新代码集的设计必须具有允许的代码可重用性限制，设计人员可以在正式验证过程中轻松设置该限制。所提议的系统对于ERP和SCM以外的各种形式的软件项目具有高度可扩展性。

**8.6结果评价**

为了验证提出的系统，我们开发了一个简单的评估模型。 此验证模型的主要主题是了解3种类型的软件项目中的不确定性对代码可重用性的影响，如图8所示。该评估模型是在Matlab上设计的，该模型从客户关系中获取了软件度量套件的经验值 管理项目，供应链管理项目和企业关系管理项目。使用引入的代码可重用性方程式，系统执行计算。代码可重用性的结果将受到各种程度的不确定性影响。我们将不确定性定义为各种隐藏参数，例如能力差距，需求波动性，对遵守软件开发寿命的不确定，项目延误等，这是任何人都无法控制的。 我们想了解，在任何项目管理团队中存在隐藏或不可预见的情况下，在这种情况下代码可重用性的性能如何？因此，我们通过使用马尔可夫模型[36]在这方面应用数学方法。 应用马尔可夫建模，可以将所有实时不确定性映射到一个经验参数，并将其应用于拟议的代码可重用性模型以了解其行为。

图9使用分散图显示了基于面向对象设计的三种不同软件项目的比较分析。 结果显示了ERP项目的主导地位，发现代码的可重用性随着不确定性的增加而大大增加。 但是，此结果的不利方面是，这种主导结果只有在0.01-0.04水平的不确定度值时才可见。比ERP项目表现更好的是，发现SCM项目稀疏但代码可重用性价值不断提高。 仔细观察散点图可以发现，代码的可重用性从0.01-0.08的不确定性值显着提高。但是，由于包含了最大数量的类和方法，DIT和WMC的值会增加，从而导致设计复杂。 这就是为什么可以以稀疏方式和较慢的速度评估SCM的代码可重用性的主要原因，但与ERP项目相比，它具有更好的准确性。 我们还使用中型CRM项目验证了评估模型。通常，CRM项目的设计复杂度会成倍增加。 但是，使用所提出的方程式，我们验证了我们的假设，并发现尽管项目体系结构涉及大量设计复杂性，但CRM项目的代码可重用性仍得到了显着提高。

**9.总结**

目前，我们已经得出了最标准的软件指标与代码可重用性之间的关系。 我们已经在面向对象设计的三个复杂软件项目中对其进行了测试，发现即使不确定性非常大，我们的模型也可以显着计算各种程度的代码可重用性。 使用马尔可夫建模的数学和随机方法，我们证明了我们的模型可以在不确定性增加的情况下提取更多代码可重用性数据。设计模式在软件工程中起着重要作用。 随着客户需求的增加，IT行业和软件项目开发商越来越多地寻求咨询以最小化生产成本。 在各种各样的成本削减过程中，代码的可重用性是最突出的一个，它需要非常熟练的技术架构来做出决策。 代码的可重用性涉及两个具有挑战性的方面，即：i）确定要保留的代码部分相同，以及ii）确定需要从头开始设计代码的哪一部分。在第一个具有挑战性的方面，开发人员可以根据客户的需求轻松决定需要保留代码的哪一部分。 但是，困难的部分是做出与需要从基础构建的新代码相关的决策。通常，根据经验丰富的体系结构，需要编程的新代码集的设计方式应使将来的客户端具有一定级别的代码可重用性，这是不可预测的。 在这种情况下，不切实际的设计将完全丧失生产能力，并且可能无法满足新项目的可重用性因素。 因此，我们未来的研究方向将集中于估计复杂软件项目的代码可重用性水平。 我们预计，我们的设计理念将极大地鼓励和激励利益相关者将其视为迄今为止最具成本效益的工具。