第3章 软件质量度量和配置管理

“当你能够测度你所说的，并将其用数字表达出来，你就对它有了一些了解；但当你不能测度，不能用数字表达它时，你对它的了解就很贫乏、很不令人满意。”

——Lord Kelvin

3.1 概述

在软件开发中，软件质量度量的根本目的是为了管理的需要，利用度量来改进软件过程。人们是无法管理不能度量的事物的。在软件开发的历史中，我们可以意识到，在20世纪60年代末期的大型软件所面临的软件危机反映了软件开发中管理的重要性。而对于管理层人员来说：没有对软件过程的可见度就无法管理；而没有对见到的事物有适当的度量或适当的准则去判断、评估和决策，也无法进行优秀的管理。软件工程的方法论主要在提供可见度方面下工夫。但仅仅是方法论的提高并不能使其成为工程学科。这就需要使用度量。度量是一种可用于决策的可比较的对象。度量已知的事物是为了进行跟踪和评估。对于未知的事物，度量则用于预测。本章将讨论软件度量的一些基本问题。但应认识到软件度量的成果是非常初步的，还需要大量工作才可能真正地做到实用化，但它的实用化成就将对软件的高质量和高速发展有不可估量的影响。

3.1.1 度量

如Lemmerich所言，测量在科学领域有悠久的历史。相对早在1889年就定义好了度量单位——米的长度测量，温度的度量复杂得多。Fahrenheit和Celsius分别在1714年和1742年提出了基于某固定点间隔递增等级的温度度量方法。Celsius将100℃和0℃之间分为100个等份。但问题是一直不能唯一确定50℃。而且长度的测量总是一个比例尺度，但是温度可以用间隔（摄氏/华氏温度表）或者比例尺度（开氏温度）来衡量。

虽然术语“Measure”（测量）、“Measurement”（测度）和“Metrics”（度量）经常被互换地使用，但注意到它们之间的细微差别是很重要的。因为“Measure”（测量）和“Measurement”（测度）既可以作为名词也可以作为动词，所以它们的定义可能会混淆。在软件工程领域中，“Measure”（测量）对一个产品过程的某个属性的范围、数量、维度、容量或大小提供了一个定量的指示。“Measurement”（测度）则是确定一个测量的行为。下面给出相关定义。

Measure：度量（名词），是根据一定的规则赋予软件过程或产品属性的数值或类别

(ISO/IEC 14598—1)。数值是对软件产品、软件过程的特征的量化计数的结果,类别是特征的定性表示。

Measure:**度量**(动词),按照度量过程中的过程定义,对软件过程或软件产品实施度量,表示实际的动作(ISO/IEC 14598—1)。

Measurement:**测度**,是按照一定的尺度用度量(名词)给软件实体属性赋值的过程(ISO/IEC 14598—1)。它强调对软件实体属性进行量化的过程性,是提取软件过程或软件产品属性的度量(名词)的过程。它所蕴涵的内容是度量的过程,度量过程可分为评估度量的过程和直接度量的过程,评估度量的过程是对计划实施度量的过程,直接度量的过程是在实施项目过程中收集数据和分析数据的过程。

Metric:**度量**,是已定义的测量方法和测量尺度(ISO/IEC 14598—1)。在很多场合与 Indicator 交叉出现,但其内涵大于 Indicator,Metric 概指软件环境中任何一个软件对象的属性的量化表现。

Indicator:**指示器**,或称为指标,是用于评价或预测其他度量的度量(ISO/IEC 14598—1)。指示器是一个或多个度量的综合,是对软件产品或软件过程的某一方面特征的反映。不同的度量目的,有不同的度量指示器选择。在具体的实施过程中,可操作的度量成千上万,应选择最能反映当时度量环境的指标作为度量指示器。

3.1.2 软件度量

软件度量或者说软件工程度量领域是一个在过去 30 多年研究非常活跃的软件工程领域。软件度量(Software Measurement)和软件量度(Software Metrics)一样非常有名。但目前学界还没有明确这两个术语的区别。参照测量理论的相关术语,我们采用**软件度量**(**Software Measurement**)。从文献上看,这两个术语是同义词。量度(metric)在这里不作度量空间理解,它理解为:度量是客观对象到数字对象的同态映射。同态映射包括所有关系和结构映射。换句话说,软件品质和软件度量成直对关系。这是度量和软件度量的根本理念。

软件度量是对软件开发项目、过程及其产品进行数据定义、收集以及分析的持续性定量化过程,目的在于对此加以理解、预测、评估、控制和改善。没有软件度量,就不能从软件开发的暗箱中跳出来。通过软件度量可以改进软件开发过程,促进项目成功,开发高质量的软件产品。度量取向是软件开发诸多事项的横断面,包括顾客满意度度量、质量度量、项目度量,以及品牌资产度量、知识产权价值度量等。度量取向要依靠事实、数据、原理、法则;其方法是测试、审核、调查;其工具是统计、图表、数字、模型;其标准是量化的指标。

软件度量研究主要分为两个阵营:一部分认为软件可以度量,一部分认为软件无法通过度量分析。无论如何,研究主流是关心软件的品质和认为软件需要定量化度量。目前有超过上千种软件度量方法被软件研究人员及从业人员提出,并且到今天有超过 5000 份论文出版发表。

3.1.3 软件度量的作用

可度量性是学科是否高度成熟的一大标志,度量使软件开发逐渐趋向专业、标准和科

学。尽管人们觉得软件度量比较难操作，且不愿意在度量上花费时间和精力，甚至对其持怀疑态度，但是这无法否认软件度量的作用。美国卡内基·梅隆大学(CMU)软件工程研究所在《软件度量指南》(*Software Measurement Guidebook*)中认为，软件度量在软件工程中的作用有三：

① 通过软件度量增加理解；

② 通过软件度量管理软件项目，主要是计划和估算、跟踪和确认；

③ 通过软件度量指导软件过程改善，主要是理解、评估和包装。软件度量对于不同的实施对象，具有不同的效用，表 3-1 是其详细说明。

表 3-1　软件度量的作用

角　　色	度 量 效 果
软件公司	(1) 改善产品质量； (2) 改善产品交付； (3) 提高生产能力； (4) 降低生产成本； (5) 建立项目估算的基线； (6) 了解使用新的软件工程方法和工具的效果和效率； (7) 提高顾客满意度； (8) 创造更多利润； (9) 构筑员工自豪感
项目经理	(1) 分析产品的错误和缺陷； (2) 评估现状； (3) 建立估算的基础； (4) 确定产品的复杂度； (5) 建立基线； (6) 从实际上确定最佳实践
软件开发人员	(1) 可建立更加明确的作业目标； (2) 可作为具体作业中的判断标准； (3) 便于有效把握自身的软件开发项目； (4) 便于在具体作业中实施渐进性软件开发改善活动

总而言之，软件度量的效用有如下几个方面。

- **理解**：获取对项目、产品、过程和资源等要素的理解，选择和确定进行评估、预测、控制和改进的基线。
- **预测**：通过理解项目、产品、过程、资源等各要素之间的关系建立模型，由已知推算未知，预测未来发展的趋势，以合理地配置资源。
- **评估**：对软件开发的项目、产品和过程的实际状况进行评估，使软件开发的标准和结果都得到切实的评价，确认各要素对软件开发的影响程度。
- **控制**：分析软件开发的实际和计划之间的偏差，发现问题点所在，并根据调整后的计划实施控制，确保软件开发良善发展。
- **改善**：根据量化信息和问题所在，探讨提升软件项目、产品和过程的有效方式，实现高质量、高效率的软件开发。

3.2 软件质量度量

3.2.1 软件质量和软件质量要素

对于软件质量，CMM 的定义是：

(1) 一个系统、组件或过程符合特定需求的程度。

(2) 一个系统、组件或过程符合客户或用户的要求或期望的程度。

在全面质量管理中，“质量”一词并不具有绝对意义上的“最好”的一般含义。质量是指“最适合于一定顾客的要求”。这些要求是产品的实际用途产品的售价。也就是可以满足软件的商业目标而不需要追求尽善尽美。重视软件质量是应该的，但是“质量越高越好”并不是普适的真理。只有极少数软件应该追求“零缺陷”，对绝大多数软件而言，商业目标决定了质量目标，而不该把质量目标凌驾于商业目标之上。

软件质量是许多质量属性的综合体现，各种质量属性反映了软件质量的方方面面。人们通过改善软件的各种质量属性，从而提高软件的整体质量，否则无从下手。软件的质量属性很多，如正确性、精确性、健壮性、可靠性、容错性、性能、易用性、安全性、可扩展性、可复用性、兼容性、可移植性、可测试性、可维护性、灵活性等。所谓的软件质量要素，是指：

- 从技术角度讲，对软件整体质量影响最大的那些质量属性才是质量要素。
- 从商业角度讲，客户最关心的、能成为卖点的质量属性才是质量要素。

对于一个特定的软件而言，首先判断什么是质量要素，才能给出提高质量的具体措施，而不是一股脑地想把所有的质量属性都做好，否则不仅做不好，还可能得不偿失。如果某些质量属性并不能产生显著的经济效益，可以忽略它们，把精力用在对经济效益贡献最大的质量要素上。软件质量保证也就是对于它的重要的质量要素的保证。

3.2.2 影响软件质量的因素

软件业通过多年的实践，总结出软件质量是人、过程和技术的函数，即 $Q=\{M,P,T\}$。其中，Q 表示软件质量，M 表示人，P 表示过程，T 表示技术。

首先是人的因素，即 M。软件开发是智力劳动，软件是人的脑力劳动、进行创造性思维的成果。只有积极进取的开发人员才能把这项工作做好，同时还要有有效的软件管理，所谓软件管理就是人员的管理，有效的软件管理能够更好地发挥人的作用，用户、分析员、设计员、程序员、测试员配合得当，是开发高质量软件的重要前提。

其次是过程因素，即 P。“过程”一词有许多定义，ISO 9000 将过程定义为“一组将输入转化为输出的相互关联或相互作用的活动”。软件过程是人们用来生产软件产品的一系列工具、方法和实践的集合。软件过程可以分为软件工程过程、软件管理过程和软件支持过程三大类。其中，工程过程指软件开发和生产的过程，如需求分析、设计、编码、测试等过程。管理过程指对软件开发和生产过程进行管理的过程，如项目策划过程、跟踪监控过程、质量保证过程、配置管理过程等。支持过程指对有效软件开发和生产进行支持的过程，如评审过程、培训过程、度量过程等。这些过程不是相互孤立、彼此隔离的，它们都关系到软件产品的

质量，必须将其科学、系统地组织成一个有效的运作体系，才能使组织所有与质量相关的活动有条不紊、高效地运行。

最后是技术因素，即 T。近年来软件技术虽然取得了不少进展，提出了许多新的开发方法，比如充分利用现成软件的复用技术、自动生成技术，也研制了一些有效的软件开发工具或软件开发环境，但在软件项目中采用的比率仍然很低。传统的手工艺开发方式仍然占据统治地位。软件的复杂性与软件技术发展不相适应的状况越来越明显，图 3-1 显示出软件技术的发展与复杂的软件需求，并且随着时间的推移，这个差距日益加大。这种差异也成为制约软件质量的重要因素。

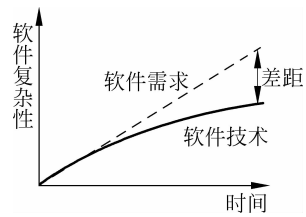

图 3-1　软件技术和软件需求的差距

3.2.3　质量保证模型

软件质量模型是软件质量评价的基础，软件质量模型代表了人们对软件质量特性的认识程度和理解程度，也代表了软件质量评价研究的进展状况。当前的软件质量模型有很多种，比较常见的有以下几种模型：McCall 模型，Boehm 模型，FURPS 模型，ISO 9126。

1. McCall 模型

McCall 提出软件质量模型，把软件质量进行基于 11 个特性之上，如图 3-2 所示，而这 11 个特性分别面向软件产品的运行、修正和转移。

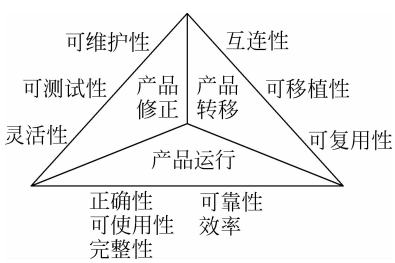

图 3-2　McCall 模型

（1）正确性：一个程序满足它的需求规约和实现用户任务目标的程度。

（2）可靠性：一个程序满足其所需的精确度，完成它的预期功能的程度。

（3）效率：一个程序完成其功能所需的计算资源和代码的度量。

（4）完整性：对未授权人员访问软件或数据的可控制程度。

（5）可使用性：学习、操作、准备输入和解释程序输出所需的工作量。

（6）可维护性：定位和修复程序中一个错误所需的工作量。

（7）灵活性：修改一个运行的程序所需的工作量。

（8）可测试性：测试一个程序以确保它完成所期望的功能所需的工作量。

（9）可移植性：把一个程序从一个硬件或软件系统环境移植到另一个环境所需的工作量。

（10）可复用性：一个程序可以在另外一个应用程序中复用的程度。

（11）互连性：连接一个系统和另一个系统所需的工作量。

2. Boehm 模型

Boehm 模型着手于软件总体的功效，也就是说，对于一个软件系统而言，除了有用性以

外，它的开发过程必定是一个时间、金钱和能量的消耗过程。考虑到系统交付时使用它的用户类型，Boehm 模型从几个维度来考虑软件的效用。总功效可以被分解成可移植性、有效性、可维护性。其中，有效性可以细分为可靠性、效率、运行工程；可维护性可以细分为测试性、可理解性、可修改性。具体如表 3-2 所示。

表 3-2 Boehm 模型

系统功效	可移植性	
	有效性	可靠性，效率，运行工程
	可维护性	测试性，可理解性，可修改性

3. FURPS 模型

McCall 和他的同事提出的质量要素代表了被提出的众多软件质量“检查表”之一，Hewlett-Packard 提出了一套考虑软件质量的因素，简称为 FURPS——功能性(Functionality)，可用性(Usability)，可靠性(Reliability)，性能(Performence)和支持度(Supportablity)。质量因素是从早期工作中得出的，5 个主要因素每一个都定义了如下评估方式。

(1) 功能性：通过评价特征集和程序的能力、交付的函数的通用性和整体系统的安全性来评估。

(2) 可用性：通过考虑人的因素、整体美学、一致性和文档来评估。

(3) 可靠性：通过度量错误的频率和严重程度、输出结果的准确度、平均失效间隔时间、从失效恢复的能力、程序的可预测性等来评估。

(4) 性能：通过测度处理速度、响应时间、资源消耗、吞吐量和效率来评估。

(5) 支持度：包括扩展程序的能力可扩展性、可适应性和服务性。这三个属性代表了一个更一般的概念——可维护性，以及可测试性、兼容度、可配置性组织和控制软件配置的元素的能力、一个系统可以被安装的容易程度、问题可以被局部化的容易程度。

FURPS 质量因素和上述描述的属性可以用来为软件过程中的每个活动建立质量度量。

4. ISO 9126

ISO/IEC 9126 软件质量模型包括 6 个质量特性和 21 个质量子特性。

(1) 功能性：适合性、准确性、互操作性、依从性、安全性。

(2) 可靠性：成熟性、容错性、可恢复性。

(3) 可用性：可理解性、易学性、可操作性。

(4) 效率：时间特性、资源特性。

(5) 可维护性：可分析性、可改变性、稳定性、可测试性。

(6) 可移植性：适应性、可安装性、一致性、可替换性。

3.2.4 缺陷排除效率

缺陷排除效率(Defect Removal Efficiency，DRE)在项目级和过程级都能提供有益的质

量度量。本质上，DRE 是对质量保证及控制活动的过滤能力的一个测量，这些活动贯穿于整个过程框架活动。

当把一个项目作为一个整体来考虑时，DRE 按如下方式定义：

$$\mathrm{DRE}=E/(E+D)$$

其中，E＝软件交付给最终用户之前所发现的错误数；D＝软件交付之后所发现的缺陷数。

最理想的 DRE 值是 1，即软件中没有发现缺陷。现实中，D 会大于 0，但随着 E 值的增加，DRE 的值仍能接近 1。事实上，随着 E 的增加，D 的最终值可能会降低（错误在变成缺陷之前已经被过滤了）。如果 DRE 作为一个度量，提供关于质量控制和保证活动的过滤能力的衡量指标，则 DRE 鼓励软件项目组采用先进技术，以便在交付之前发现尽可能多的错误。

DRE 也能够在错误传递到下一个框架活动或软件工程任务之前，用来在项目中评估一个小组发现错误的能力。例如，需求分析任务产生了一个分析模型，可以复审该模型以发现和改正错误。在对分析模型的复审中未被发现的错误会传递给设计任务（在这里它们有可能被发现，也有可能没被发现）。在这种情况下，定义 DRE 为：

$$\mathrm{DRE}_i=E_i/(E_i+E_{i+1})$$

其中，E_i＝在软件工程活动 i 中所发现的错误数；E_{i+1}＝在软件工程活动 $i+1$ 中所发现的错误数，这些错误来源于软件工程活动 i 中未能发现的错误。

一个软件项目组（或单个软件工程师）的质量目标是使 DRE_i 接近 1。即错误应该在传递到下一个活动之前被过滤掉。

3.3　软件过程度量

3.3.1　软件过程度量概念

软件过程度量是对软件过程进行度量的定义、方法、活动和结果的集合。软件过程度量不是单一的活动而是一组活动的集合，它本身也是一个系统的过程。与任何系统的过程一样，它包括确定需求、制定计划、执行和结果分析等一系列完整的步骤。软件过程度量通常包括如下的活动：选择和定义度量、制定度量计划、收集数据、执行度量分析、评估过程性能、根据评估结果采取相应措施等，如图 3-3 所示。

1. 软件过程度量的目标

软件过程度量的目标是为了对软件过程的行为进行目标管理，并在度量的基础上对软件过程进行控制、评价和改善。软件过程度量最终为项目管理和软件过程管理服务。

2. 软件过程度量的对象

软件过程度量就其对象而言，主要包括三个：工作产品、软件项目和过程。工作产品指软件项目执行过程中产生的交付的和不交付的过程产品，如用户手册、同行评审记录。同行评审记录虽然一般情况下不交付给顾客，但属于工作产品。软件项目的度量主要集中在项目质量、成本、进度等方面。过程度量主要从组织的角度考虑。软件开发组织定义统一的组

织标准软件过程(Organization's Set of Standard Process,OSSP),项目根据具体情况对组织标准软件过程进行剪裁,形成项目定义软件过程,根据项目定义软件过程制定软件开发计划(Software Development Plan,SDP)。过程度量主要指对项目定义软件过程和组织标准软件过程的度量。

在软件开发组织中,对工作产品的度量主要包括规模、质量两个方面。目前比较流行的工作产品的规模度量主要包括软件系统规模度量和软件产品文档规模度量。软件系统规模度量的方法主要有 LOC、FPA,软件产品文档规模度量方面还没有成型的方法。对工作产品质量的度量已经有相应的国际标准。虽然规模度量和质量度量国外都有比较成型的方法或标准,但在应用中,一定要根据实际情况,参考标准或相应方法制定本组织的标准,尤其是质量度量方面,一定要对原有标准进行剪裁。

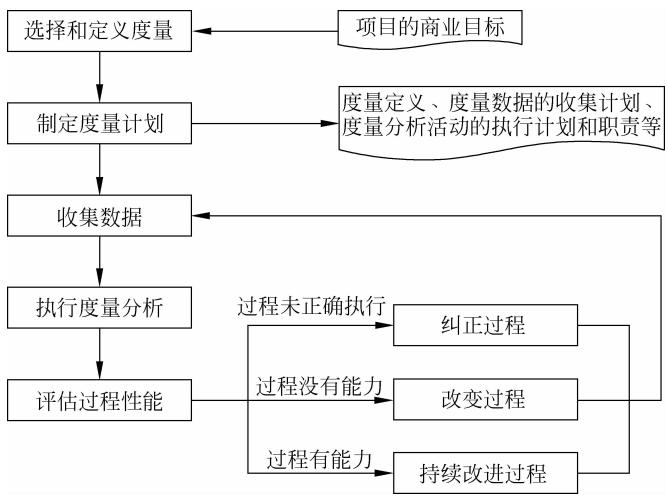

图 3-3 软件过程度量的过程

3. 软件过程度量的方法

对软件过程的度量方法是过程性方法,软件过程行为是事件行为,对过程的度量也具有过程性,从制定度量目标到收集数据再到数据分析表示出了典型的度量阶段。并且随着软件过程的进化,度量过程也随之进化发展,度量过程中的各个阶段所用到的技术、方法是动态更新的。

软件过程度量的方法主要包括常用的采集方法(在不同项目阶段针对不同类型内容的数据采集),常用的数据分析方法(多种数据表示方法和分析方法)。

4. 软件过程度量的结果

产品度量结果通常是软件产品的复杂度模型和可靠性模型等。对过程度量的结果是模型(例如花费模型,质量模型等)、关系(例如人员投入与质量的关系,进度与质量关系等)和由过程量化特征组成的过程基线。

软件产品度量和软件过程度量虽然存在不同,它们之间也有联系。产品度量内容可以是过程度量内容的一部分,因为对产品的度量结果是对产品的评价,而产品又是过程的结

果，产品的好坏从一个方面体现了过程的好坏。

3.3.2　软件过程度量常见问题

1. 度量太多、太频繁

软件过程有成百的方面可以度量，因而在实施度量时很容易选择了过多的数据项进行度量。过多的度量要求会使管理人员和参与人员困惑，并产生抵触情绪。而且由于度量必然带来成本支出，太多的度量要求会造成度量计划的成本太高，与其收益不匹配。因此，制定度量计划应该从较小的度量集合开始，实施取得成功后再逐步扩展。

2. 度量太少、太迟

某些度量计划只定义了很少的度量，以至于不能提供足够的信息来理解当前的情况并做出正确的决定。这同样会造成度量收益与其成本不匹配的情况，使投资人或管理者认为该度量计划没有意义，因而中止它的实施。同时，如果只对工作的少量方面进行度量，还可能造成现实的负面影响。人们有时会根据度量要求改变其行为。这样会带来难以预期的副作用。

3. 度量了不正确的事物或属性

如果度量的数据项与业务的关键成功决策没有什么关系，那就意味着度量了错误的东西。如果管理人员不能适时地获得改善项目管理或人员管理所需的数据，那么就应该重新评估度量集的定义。

4. 度量的定义不精确

模糊的或有歧义的度量定义会导致对它们的多理解。某些人会把软件多余的特性当作是缺陷，而另一些人却不会。如果没有一致的理解，一段时间之后的度量结果可能会表现出奇怪的特征，因为每个人都可能是以不同的方式进行数据收集和报告。因此，制定度量计划时必须精确定义每个需要度量的数据项以及由这些数据项组合计算得到的度量。

5. 收集了数据却没有利用

度量得来的数据应该得到充分的利用，尽可能增大度量的收益。项目经理和管理层应该把度量结果与开发团队共享，让大家都能理解度量的好处，以及度量得到的信息如何帮助管理者进行决策和采取正确措施。应选择公开一些度量结果供所有利益攸关者查阅，以便他们能分享成功和处理不足之处。

6. 错误地解释度量数据

度量得到的趋势可以为人们揭示很多问题，但是不能单独地看待每一个信号，应综合分析它们的原因，否则有可能得到错误的解释。例如，在采取了质量改进措施之后，如果发现缺陷密度增加了，分析人员可能会认为这些质量改进措施弊大于利，而打算回到原来的开发方式，但这种情况的实际原因可能是因为改进的测试技术提高了缺陷的发现率。因此，在进

行度量结果分析时，需要跟踪一个相对较长时期的数据，不应对某个数据点的值反应过度。分析异常情况的原因应该综合考虑所有可能的原因，进行正确的判断。为了避免陷入这些困境，需要采用一些行之有效的方法来制定度量计划，保证计划的合理性。

7. 自动化工具欠缺

缺少数据的收集、组织和分析的自动化工具。自动化工具的使用可减少数据收集的成本，同时使得数据的分析和反馈更为容易和准确。另外可减少人为的错误，使得管理者更为相信数据的有效性。事实上，度量和分析从技术上讲并不是很复杂，建立一个有效的度量程序的最大挑战与各种公式、统计技术和复杂的分析技术都无关。困难主要在于如何决定哪些度量是对组织有意义的，采用什么流程收集和使用这些度量数据最有效。

3.3.3 基于目标的软件过程度量方法

GQM 是 **Goal-Question-Metric** 的缩写，它是 1988 年由马里兰大学的 Victor R. Basili 提出的一种面向目标的、自上而下由目标逐步细化到度量的度量定义方法，用来告诉组织/机构要采集哪些数据就足够用了。其隐含的一个假设是每一个组织、项目都有一系列目标要实现。要实现每一个目标，均要回答一系列问题才能知道目标有没有实现。而对提出的每个问题，都可以找到一个完整、可以量化的满意解答。它使得测量由“被动引发到过程中”变成“主动地渗透到过程中”，它由目标细化到度量的逐步求精的方法具备很强的灵活性和可操作性，因而得到了广泛的认可。

由图 3-4 可见，GQM 模型是一种层次状结构，最上层是一个目标，对该目标细化就得到几个问题，构成问题层。这几个问题将关注的方面分解为几个部分。对于每一个问题，可以进一步细化为几个度量项。不同的问题可能共享相同的度量项，不同的目标也可能涉及相同的问题。另外，度量值可能是主观的，也可能是客观的。在测量同一个度量项时，如果它同时服务于多个目标，则有可能因为观测角度不同而得到不同的结果。

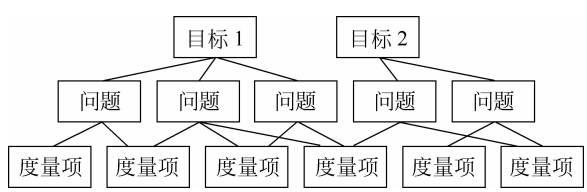

图 3-4 GQM 模型

GQM 建模，可应用于公司、部门或者仅仅一个项目上。然而，任何层次上的建模，都需要事先分析度量对象和上下环境的相关信息。问题的定义要尽可能详细地来描述要实现的目标。建立目标的过程对 GQM 建模起着关键作用。需要遵循一定的方法和步骤。由图 3-5 可以发现，一个目标主要受以下几个因素的控制。

- ISSUES(侧重点)：度量对象的质量重点。
- VIEWPOINTS(立场)：信息使用者。
- OBJECTS(对象)：要度量的对象。
- PURPOSES(目的)：一般是理解、控制和改进要度量的对象。