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。

——开尔文(Lord Kelvin)

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。

人们无法管理不能度量的事物，在软件开发的历史中可以意识到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。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，但是仅是方法论的提高，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、评估。对于未知的事物，度量则用于预测。软件度量的成果非常初步，还需要大量工作才可能真正地做到实用化，并对软件的高质量和高速发展产生不可估量的影响。

本章正文共分4节，3.1节介绍度量和软件度量，3.2节介绍软件质量度量，3.3节介绍软件过程度量，3.4节介绍软件配置管理。

3.1 度量和软件度量

3.1.1 度量

测量在科学领域有悠久的历史。

相对在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。华氏(Fahrenheit)和摄氏(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。摄氏将0～100度分为100等份，但问题是一直不能唯一确定50摄氏度。而且，长度的测量总是一个比例尺度，温度可能用间隔(摄氏/华氏温度表)或者比例尺度(开氏温度)来衡量，如图3-1所示。

虽然术语Measure(测量)、Measurement(测度)和Metric(度量)经常被互换，但注意到它们之间的细微差别是很重要的。因为Measure和Measurement既可作为名词也可作为动词，所以定义会混淆。在软件工程领域中，Measure对一个产品过程的某个属性的范围、数量、维度、容量、大小提供了一个定量的指示，Measurement则是确定一个测量的行为，下

°C	°F	°C	°F	°C	°F
70	158	140	284	210	410
80	176	150	302	220	428
90	194	160	320	230	446
100	212	170	338	240	464
110	230	180	356	250	482
120	248	190	374	260	500
130	266	200	392	270	518

摄氏温度和华氏温度转换

图 3-1　摄氏和华氏度量

面给出相关定义。

- Measure(名词)：根据一定的规则赋予软件过程或产品属性的数值或类别，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(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。
- Measurement：按照一定的尺度用度量(名词)给软件实体属性赋值的过程，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。蕴含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：已定义的测量方法和测量尺度，在很多场合与 Indicator 交叉出现，内涵大于 Indicator，Metric 指软件环境中任何一个软件对象的属性的量化表现。
- Indicator：指示器或称指标，是用于评价或预测其他度量的度量。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 Metric)一样，非常有名。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，采用软件度量(Software Measurement)。从文献上看，两个术语是同义词。在这里，量度(Metric)不作度量空间理解，而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，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、产品进行数据定义、收集、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、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;工具是统计、图表、数字、模型;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;另一部分认为软件无法通过度量分析。当前的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。

3.1.3 作用

可度量性是学科是否高度成熟的标志,度量使软件开发逐渐趋向专业、标准、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、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·梅隆大学软件工程研究所(SEI)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如下:通过软件度量增加理解;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;通过软件度量指导软件过程改善,主要是理解、评估、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、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、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) 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

软件度量的效用有以下几个方面。

- 理解:获取对项目、产品、过程、资源等要素的理解,选择和确定进行评估、预测、控制、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，合理地配置资源。

- 评估：对软件开发的项目、产品、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，并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之所在探讨提升软件项目、产品、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 的定义如下：一个系统、组件、过程符合特定需求的程度；一个系统、组件、过程符合客户或用户的要求或者期望的程度。

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来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、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，等等。所谓的软件质量要素指以下两个方面：

- 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。

对于特定的软件，首先判断什么是质量要素才能给出提高质量的具体措施，而不是一股脑想把所有的质量属性都做好，否则不仅做不好，还得不偿失。

如果某些质量属性并不能产生显著的经济效益，可以忽略，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，同时还要有效的软件管理，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”。软件过程是用来生产软件产品的一系列工具、方法、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、软件支持过程 3 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,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的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,但是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 3-2 给出软件技术的发展与复杂的软件需求随着时间的推移差距日益加大。这种差异成为制约软件质量的重要因素。

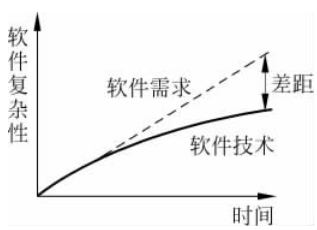

图 3-2 软件技术和软件需求的差距

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,代表了人们对软件质量特性的认识程度、理解程度,也代表了软件质量评价研究的进展状况。

当前软件质量模型有很多,比较常见的有以下几种模型,即 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,分别面向软件产品的运行、修正、转移,如图 3-3 所示。

- 正确性:程序满足需求规约和实现用户任务目标的程度。
- 可靠性:程序满足所需的精确度和完成预期功能的程度。
- 效率:程序完成其功能所需的计算资源和代码的度量。
- 完整性:对未授权人员访问软件或数据的可控制程度。
- 可用性:学习、操作、准备输入、解释程序输出所需的工作量。
- 可维护性:定位、修复程序中的错误所需的工作量。

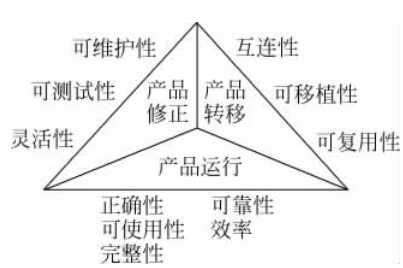

图 3-3 McCall 模型

- 灵活性：修改运行的程序所需的工作量。
- 可测试性：测试程序，确保完成所期望的功能所需的工作量。
- 可移植性：把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。
- 可复用性：程序可以在另外一个应用程序中复用的程度。
- 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，对于软件系统而言，除了有用性以外，开发过程必定是一个时间、金钱、能量的消耗过程。考虑到系统交付时使用的用户类型，Boehm 模型从几个维度来考虑软件的效用。

总功效分解成可移植性、有效性、可维护性。其中，有效性细分为可靠性、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，惠普提出了一套考虑软件质量的因素，简称为 FURPS，即功能性（Functionality）、可用性（Usability）、可靠性（Reliability）、性能（Performance）、支持度（Supportability）。质量因素从早期工作中得出的 5 个主要因素定义了以下评估方式。

- 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、整体系统的安全性来评估。
- 可用性：通过考虑人的因素、整体美学、一致性、文档来评估。
- 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。
- 性能：通过处理速度、响应时间、资源消耗、吞吐量、效率来评估。
- 支持度：包括扩展程序的能力可扩展性、可适应性、服务性 3 个属性，代表了更一般的概念——可维护性、可测试性、兼容度、可配置性，以及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。

- 功能性：适合性、准确性、互操作性、依从性、安全性。
- 可靠性：成熟性、容错性、可恢复性。
- 可用性：可理解性、易学性、可操作性。

- 效率：时间特性、资源特性。
- 可维护性：可分析性、可改变性、稳定性、可测试性。
- 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。在本质上,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。

在把一个项目作为一个整体考虑时,DRE 按以下方式定义：

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

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。在现实中,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)。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 也能够用来在项目中评估一个小组发现错误的能力,在这些错误传递到下一个框架活动或软件工程任务之前(例如需求分析任务产生了分析模型)可以复审该模型以发现、改正错误。在分析模型的复审中,未被发现的错误会传递给设计任务(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。

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 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、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,本身也是一个系统的过程。与任何系统的过程一样,软件过程度量包括确定需求、制订计划、执行、结果分析等一系列完整的步骤。

通常,软件过程度量包括以下活动：选择和定义度量、制订度量计划、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,等等,如图 3-4 所示。

1. 软件过程度量的目标

软件过程度量的目标是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

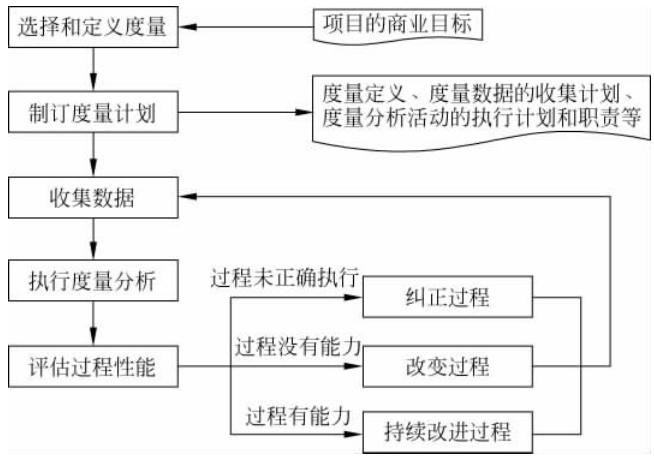

图 3-4　软件过程度量的过程

程进行控制、评价、改善。最终，软件过程度量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 3 个，即工作产品、软件项目、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。同行评审记录虽然在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订软件开发计划(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。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在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，尤其是在质量度量方面，一定要对原有标准进行剪裁。

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)和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,但也有联系。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,产品的好坏体现了过程的好坏。

3.3.2 常见问题

1. 度量得太多、太频繁

软件过程有成百的方面可以度量,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,制订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. 度量得太少、太迟

某些度量计划只定义了很少的度量,以至于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什么关系,那么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多种理解。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,在制订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。同时应公开一些度量结果供所有利益相关者查阅,以便能

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，应综合分析原因，否则有可能得到错误解释。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大于利，而打算回到原来的开发方式，但是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制订度量计划来保证计划的合理性。

7. 自动化工具欠缺

自动化工具欠缺也就是缺少数据的收集、组织、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，同时使数据的分析、反馈更加容易和准确。另外可减少人为的错误，使管理者更加相信数据的有效性。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　基于目标的方法

基于目标的软件过程度量方法(Goal-Question-Metric，GQM)是于 1988 年美国马里兰大学的 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定义方法，用来告诉组织/机构要采集哪些数据。

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。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，使得测量由“被动引发到过程中”变成“主动地渗透到过程中”，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，因而得到了广泛认可。

由图 3-5 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。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，如果同时服务于多个目标，则有可能因为观测角度不同得到不同的结果。

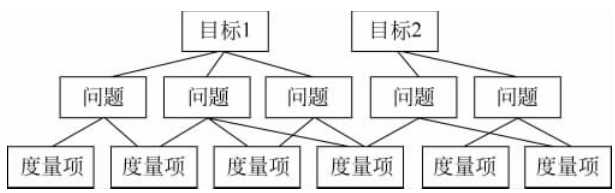

图 3-5　GQM 模型

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。

如图 3-6 所示,一个目标主要受下面几个因素控制。

- Issues(侧重点):度量对象的质量重点。
- Viewpoint(立场):信息使用者。
- Object(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。

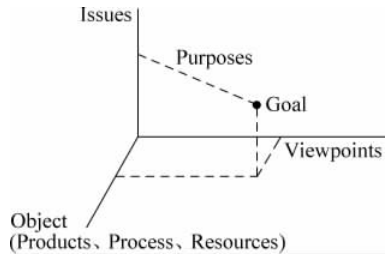

图 3-6 GQM 的目标定义

因此,目标开发需要了解以下 3 个方面的信息。

- 分析组织的经营战略和方针:通过分析可以挖掘出实现组织目标的目的和侧重点。
- 分析组织的过程和产品定义:只有这样才能准确地锁定实现目标要度量的对象。
- 分析软件组织的架构模型:可以锁定关注目标的特定角色。

当然,没必要将组织中的所有元素都牵涉进来。确保这一点必须在选择之前进行细致的相关分析。根据以往经验,建议先从解决实际问题着手,以后随着组织过程的逐步成熟再对度量目标进行改进。

对于确定的每一个目标(比如目标是从过程改进人员的角度分析同行评审过程,目的是提高同行评审过程的能力),能够以量化的方式导出一些相关的问题。

通常,获得问题可以从以下方面考虑:

(1) 对于特定目标陈述中的对象应该抓住哪些可以量化的特征?例如:

- 什么是当前同行评审的效率?
- 实际同行评审过程是按照文档化的流程执行的吗?
- 同行评审发现缺陷的数量与评审对象规模、评审小组人数有关系吗?

(2) 结合模型中的侧重点,这些特征应该怎么来描述?例如:

- 同行评审的效率与基线的偏差是多少?
- 同行评审的效率正在提高吗?

(3) 结合模型中的侧重点,应该如何评价度量对象的这些特征?例如:

- 每人时发现的缺陷数量明显提高了吗?
- 项目经理能够明显觉察到评审效率的提高吗?

一旦选好了应该回答的问题,并且确定其可以量化,便可以选择或构造一些数量指标。在选择数据项时至少要考虑以下方面。

- 现有数据的有效性:尽量利用现有数据,如果实在没有相关数据积累或者现有数据的可靠性太差,也要少选、精选需要进行采集的数据项。总之,应该最大限度地利用现有数据。
- 度量对象的稳定性:对于成熟、稳定的度量对象多应用客观度量,对于不成熟、不稳定的对象可以结合主观判断和评价来获得数据。
- GQM 建模的渐进性:GQM 建模是一个持续改进的过程,所选择的度量项不仅可以评价度量的对象,也反映了模型本身的可靠性和质量。

实施以上步骤，最后得到的 GQM 样例如表 3-3 所示。

表 3-3　GQM 分解样例

GQM		
目标	用途	控制、改进
	对象	同行评审过程
	侧重点	能力
	需求方	过程改进人员
	环境	符合 CMMI4 要求的研发规范
问题 1		什么是 PR 的过程能力
度量项	PR 排错能力	同行评审过程缺陷密度的均值和控制限
问题 2		如何判断一次同行评审的有效性
度量项	有效性项目经理评价缺陷密度值状态	项目经理对评审结论的评价 缺陷密度值落在控制限之外：Y 缺陷密度值落在控制限之内：N
问题 3		项目经理对评审对象质量提高的评价
问题 4		……

3.4　软件配置管理

配置管理的概念源于美国空军，为了规范设备的设计与制造，美国空军于 1962 年制定并发布了第一个配置管理的标准——AFSCM375-1，CM During the Development & Acquisition Phases。

软件配置管理概念的提出则在 20 世纪 60 年代末 70 年代初。当时美国加州大学圣巴巴拉分校(University of California，Santa Barbara，UCSB)的 Leon Presser 教授在承担美国海军的航空发动机研制期间撰写了一篇名为“Change and Configuration Control”的论文，提出控制变更和配置的概念，这篇论文同时也是管理该项目(进行过近 1400 万次修改)的一个经验总结。

Leon Presser 在 1975 年成立了一家名为 SoftTool 的公司，开发了配置管理工具(Change and Configuration Control，CCC)，这是最早的配置管理工具之一。

随着软件工程的发展，软件配置管理越来越成熟，从最初的仅仅实现版本控制发展到现在的提供工作空间管理、并行开发支持、过程管理、权限控制、变更管理等一系列全面的管理能力，形成了一个完整的理论体系。同时，在软件配置管理的工具方面出现了大批的产品，如最著名的 ClearCase、开源产品 CVS、入门级工具 Microsoft VSS、新秀 Hansky Firefly。

软件配置管理(Software Configuration Management，SCM)是一种标识、组织、控制修改的技术。软件配置管理应用于整个软件工程过程。我们知道，在软件建立时变更是不可避免的，变更加剧了项目中软件开发者之间的混乱。SCM 活动的目标就是为了标识变更、控制变更、确保变更正确实现，并向其他有关人员报告变更。从某种角度讲，SCM 是一种标识、组织、控制修改的技术，目的是使错误降为最小，并最有效地提高生产效率。

软件配置管理作为 CMM 2 级的关键域(Key Practice Area，KPA)，在整个软件的开发

活动中占有很重要的位置。软件配置管理是贯穿于整个软件过程的保护性活动,被设计用来:

- 标识变化;
- 控制变化;
- 保证变化被适当地实现;
- 向其他可能有兴趣的人员报告变化。

所以,必须为软件配置管理活动设计一个能够融合于现有的软件开发流程的管理过程,甚至直接以软件配置管理过程为框架来再造组织的软件开发流程。

3.4.1 目标

国外已经有30多年的软件配置管理历史,而国内的发展却是在21世纪的事。国内的软件配置管理已经取得了迅速发展,并得到了包括BAT(百度、阿里巴巴、腾讯)3家在内的软件公司的普遍认可。软件配置管理是在贯穿整个软件生命周期中建立和维护项目产品的完整性,基本目标如下:

- 软件配置管理的各项工作是有计划进行的。
- 被选择的项目产品得到识别、控制,并且可以被相关人员获取。
- 已识别出的项目产品的更改得到控制。
- 使相关组别和个人及时了解软件基准的状态和内容。

为了达到上述目标要贯彻执行以下方针:

- 技术部门经理和具体项目主管确定配置管理的工作过程。
- 施行软件配置管理的职责应被明确分配,相关人员得到软件配置管理方面的培训。
- 技术部门经理和具体项目主管应该明确在相关项目中所担负的软件配置管理方面的责任。
- 软件配置管理工作应该享有足够的资金支持,需要在客户、技术部门经理、具体项目主管之间协商。
- 软件配置管理应该实施于对外交付的软件产品,以及那些被选定的在项目中使用的支持类工具等。
- 软件配置的整体性在整个项目生命周期中得到控制。
- 软件质量保证人员应该定期审核各类软件基准以及软件配置管理工作。
- 使软件基准的状态和内容能够及时通知给相关组别和个人。

3.4.2 角色职责

对于任何一个管理流程,保证该流程正常运转的前提条件是要有明确的角色、职责、权限的定义。特别是在引入了软件配置管理的工具之后,理想的状态就是组织内的所有人员按照不同角色的要求,根据系统赋予的权限执行相应的动作。因此,在软件配置管理过程中主要涉及下列角色和分工。

1. 项目经理(Project Manager,PM)

项目经理是整个软件研发活动的负责人,根据软件配置控制委员会的建议批准配置管

理的各项活动，并控制它们的进程，具体职责如下：

- 制定和修改项目的组织结构和配置管理策略；
- 批准、发布配置管理计划；
- 决定项目起始基线和开发里程碑；
- 接受并审阅配置控制委员会的报告。

2. 配置控制委员会(Configuration Control Board，CCB)

配置控制委员会负责指导和控制配置管理的各项具体活动的进行，为项目经理的决策提供建议，具体职责如下：

- 定制开发子系统；
- 定制访问控制；
- 制定常用策略；
- 建立、更改基线的设置，审核变更申请；
- 根据配置管理员的报告决定相应的对策。

3. 配置管理员(Configuration Management Officer，CMO)

配置管理员根据配置管理计划执行各项管理任务，定期向 CCB 提交报告，并列席 CCB 的例会，具体职责如下：

- 软件配置管理工具的日常管理与维护；
- 提交配置管理计划；
- 各配置项的管理与维护；
- 执行版本控制和变更控制方案；
- 完成配置审计并提交报告；
- 对开发人员进行相关的培训；
- 识别软件开发过程中存在的问题，并拟就解决方案。

4. 系统集成员(System Integration Officer，SIO)

系统集成员负责生成和管理项目的内部和外部发布版本，具体职责如下：

- 集成修改；
- 构建系统；
- 完成对版本的日常维护；
- 建立外部发布版本。

5. 开发人员(Developer，DEV)

开发人员的职责就是根据组织内确定的软件配置管理计划和相关规定，按照软件配置管理工具的使用模型来完成开发任务。

3.4.3 过程描述

软件研发项目可以划分为 3 个阶段，即计划阶段、开发阶段、维护阶段。

然而,从软件配置管理的角度来看,后两个阶段所涉及的活动是一致的,所以合二为一,成为“项目开发和维护”阶段。

1. 项目计划阶段

在项目设立之初 PM 首先需要制订整个项目研发计划,之后软件配置管理的活动就可以展开了,因为如果不在项目开始之初制订软件配置管理计划,那么软件配置管理的许多关键活动就无法及时、有效地进行,直接后果就是造成了项目开发状况的混乱,并注定软件配置管理活动成为一种“救火”的行为。所以,及时制订一份软件配置管理计划在一定程度上是项目成功的重要保证。

在软件配置管理计划的制订过程中,主要流程如下:

(1) CCB 根据项目的开发计划确定各里程碑和开发策略;

(2) CMO 根据 CCB 的规划制订详细的配置管理计划,交 CCB 审核;

(3) CCB 通过配置管理计划后交项目经理批准,发布实施。

2. 项目开发和维护阶段

这一阶段是项目研发的主要阶段,软件配置管理活动主要分以下 3 个层面:

(1) 由 CMO 完成的管理和维护工作;

(2) 由 SIO 和 DEV 具体执行软件配置管理策略;

(3) 变更流程。

这 3 个层面既彼此独立又互相联系,是一个有机的整体。在软件配置管理过程中,核心流程如下:

(1) CCB 设定研发活动的初始基线;

(2) CMO 根据软件配置管理规划设立配置库和工作空间,为执行软件配置管理就绪做好准备;

(3) 开发人员按照统一的软件配置管理策略,根据获得的授权资源进行项目的研发工作;

(4) SIO 按照项目的进度集成组内开发人员的工作成果并构建系统,推进版本的演进;

(5) CCB 根据项目的进展情况审核各种变更请求,并适时地划定新的基线,保证开发和维护工作的有序进行。

该流程循环往复,直到项目结束。当然,除上述核心过程之外还涉及其他相关的活动和操作流程,下面按不同的角色分工。

(1) 各开发人员按照项目经理发布的开发策略或模型进行工作;

(2) SIO 负责将各分项目的工作成果归并至集成分支,供测试或发布;

(3) SIO 可向 CCB 提出设立基线的要求,经批准后由 CMO 执行;

(4) CMO 定期向项目经理和 CCB 提交审计报告,并在 CCB 例会中报告项目在软件过程中可能存在的问题和改进方案;

(5) 在基线生效后,一切对基线和基线之前的开发成果的变更必须经 CCB 的批准;

(6) CCB 定期举行例会,根据成员所掌握的情况、CMO 的报告和开发人员的请求对配置管理计划做出修改,并向项目经理负责。

配置管理的工作流程如图 3-7 所示。

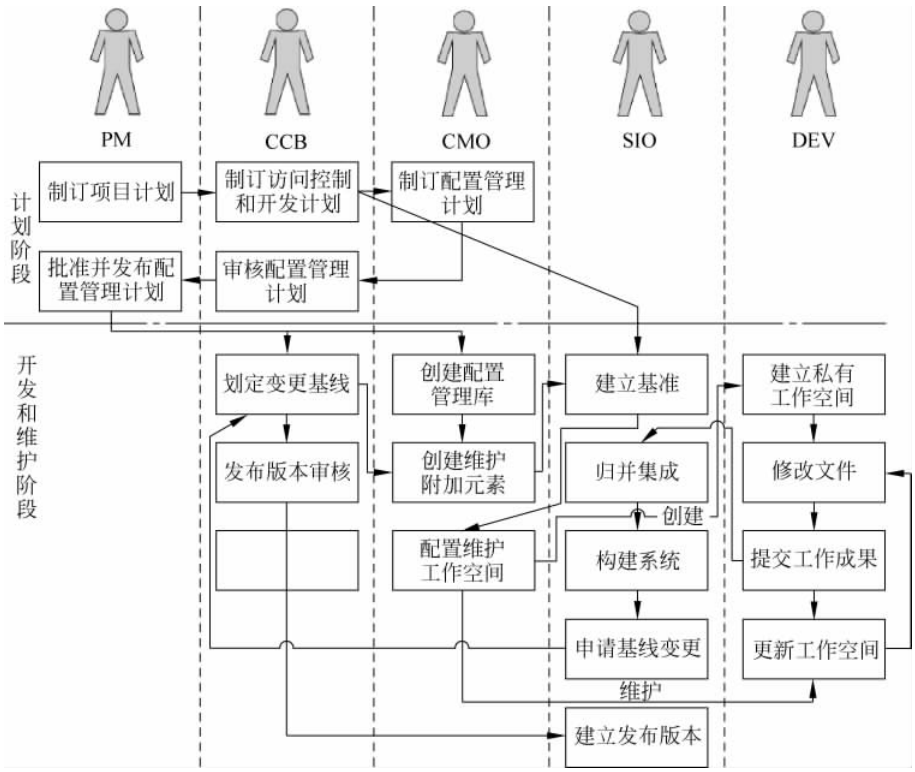

图 3-7　软件配置管理基本流程

3.4.4　关键活动

软件配置管理是软件质量保证的重要一环，其主要责任是控制变化。

然而，SCM 也负责个体 SCI 和软件的各种版本的标识、软件配置的审计（保证已被适当地开发），以及配置中所有变化的报告。任何关于 SCM 的讨论均涉及一系列复杂的问题：

- 一个组织如何标识和管理程序（及其文档）的很多现存版本才能使变化可以高效地进行？
- 一个组织如何在软件发布给客户之前和之后控制变化？
- 谁负责批准变化，并给变化确定优先级？
- 如何保证变化已经恰当地进行？
- 采用什么机制告知其他人员已经实行的变化？

下面针对这些问题给出 SCM 任务的定义。

1. 配置项识别

软件过程的输出信息可以分为 3 个主要类别：

- 计算机程序（源代码和可执行程序）；
- 描述计算机程序的文档（针对技术开发者和用户）；

• 数据(包含在程序内部或外部)。

这些项包含了在软件过程中产生的所有信息,总称为软件配置项。

由此可见,配置项的识别是配置管理活动的基础,也是制订配置管理计划的重要内容。

软件配置项分类软件的开发过程是一个不断变化的过程,为了在不严重阻碍合理变化的情况下控制变化,软件配置管理引入了"基线(Base Line)"这一概念。

IEEE 对基线的定义是"已经正式通过复审和批准的某规约或产品,因此可作为进一步开发的基础,并且只能通过正式的变化控制过程改变"。

根据定义,在软件的开发流程中把所有需要加以控制的配置项分为基线配置项和非基线配置项两类。基线配置项可能包括所有的设计文档和源程序等;非基线配置项可能包括项目的各类计划和报告等。

配置项的标识、控制是指所有配置项都应按照相关规定统一编号,按照相应的模板生成,并在文档中的规定章节(部分)记录对象的标识信息。在引入软件配置管理工具进行管理后,这些配置项都应以一定的目录结构保存在配置库中。所有配置项的操作权限应由 CMO 严格管理,基本原则是基线配置项向软件开发人员开放读取的权限;非基线配置项向 PM、CCB 及相关人员开放。

2. 工作空间管理

在引入软件配置管理工具之后,所有开发人员都会被要求把工作成果存放到由软件配置管理工具所管理的配置库中,或是直接工作在软件配置管理工具提供的环境之下。一般来说,比较理想的情况是把整个配置库视为一个统一的工作空间,然后再根据需要划分为个人(私有)、团队(集成)、全组(公共)3 类工作空间(分支),从而更好地支持将来可能出现的并行开发的需求。

每个开发人员按照任务的要求在不同的开发阶段工作在不同的工作空间上。例如,对于私有开发空间而言,开发人员根据任务分工获得对相应配置项的操作许可之后即在自己的私有开发分支上工作,所有工作成果体现为在该配置项的私有分支上的版本的推进,除该开发人员以外,其他人员均无权操作该私有空间中的元素;而集成分支对应的是开发团队的公共空间,该开发团队拥有对该集成分支的读/写权限,其他成员只有只读权限,管理工作由 SIO 负责;至于公共工作空间,则是用于统一存放各个开发团队的阶段性工作成果,提供全组统一的标准版本,并作为整个组织的 Knowledge Base。

由于选用的软件配置管理工具不同,在对工作空间的配置和维护的实现上有比较大的差异,对于 CMO 来说,这些工作是重要职责,必须根据各开发阶段的实际情况来配置工作空间,并定制相应的版本选取规则来保证开发活动的正常运作。在变更发生时,应及时做好基线的推进。

3. 版本控制

版本控制是软件配置管理的核心功能。

所有置于配置库中的元素都应自动予以版本的标识,并保证版本命名的唯一性。版本在生成过程中依照设定的使用模型自动分支、演进。除了系统自动记录的版本信息以外,为了配合软件开发流程的各个阶段,还需要定义、收集一些元数据(Metadata)来记录版本的辅

助信息和规范开发流程，并为今后对软件过程的度量做好准备。当然，如果选用的工具支持，这些辅助数据将能直接统计出过程数据，从而方便软件过程改进（Software Process Improvement，SPI）活动的进行。

对于配置库中的各个基线控制项，应该根据其基线的位置和状态来设置相应的访问权限。一般来说，基线版本之前的各个版本都应处于被锁定的状态，如果需要进行变更，应按照变更控制的流程进行操作。

4. 变更控制

在对 SCI 的描述中引入了基线的概念。从 IEEE 对基线的定义中可以发现，基线是和变更控制紧密相连的。也就是说，在对各 SCI 做出了识别，并且利用工具进行了版本管理之后，如何保证在复杂多变的开发过程中真正处于受控的状态，并在任何情况下都能迅速地恢复到任一历史状态，就成为了软件配置管理的另一重要任务。因此，变更控制就是通过结合人的规程和自动化工具提供一个变化控制的机制。

前面部分已经把 SCI 分为基线配置项和非基线配置项两大类，所以这里涉及的变更控制的对象主要指配置库中的各基线配置项。变更管理的一般流程如下：

- （获得）提出变更请求；
- 由 CCB 审核并决定是否批准；
- （被接受）修改请求分配人员为提取 SCI，进行修改；
- 复审变化；
- 提交修改后的 SCI；
- 建立测试基线并测试；
- 重建软件的适当版本；
- 复审（审计）所有 SCI 的变化；
- 发布新版本。

在这样的流程中，CMO 通过软件配置管理工具进行访问控制和同步控制，而这两种控制是建立在前文所描述的版本控制和分支策略的基础上的。

5. 状态报告

配置状态报告就是根据配置项操作数据库中的记录向管理者报告软件开发活动的进展情况。

这样的报告应该是定期进行，并尽量通过计算机辅助软件工程（Computer Aided Software Engineering，CASE）工具自动生成，用数据库中的客观数据来真实地反映各配置项的情况。

配置状态报告应根据报告着重反映当前基线配置项的状态，作为对开发进度报告的参照。同时，也能从中根据开发人员对配置项的操作记录对开发团队的工作关系做一定的分析。

配置状态报告包括下列内容：

- 配置库结构和相关说明；
- 开发起始基线的构成；

- 当前基线位置及状态；
- 各基线配置项集成分支的情况；
- 各私有开发分支类型的分布情况；
- 关键元素的版本演进记录；
- 其他应予报告的事项。

6. 配置审计

配置审计的主要作用是作为变更控制的补充手段来确保某一变更需求已被切实实现。在某些情况下，配置审计被作为正式的技术复审的一部分，但当软件配置管理是一个正式的活动时，该活动由 SQA 人员单独执行。

总之，软件配置管理的对象是软件研发活动中的全部开发资产。所有一切都应作为配置项纳入管理计划，统一进行维护、集成。因此，软件配置管理的主要任务归结为以下几条：

- 制订项目的配置计划；
- 对配置项进行标识；
- 对配置项进行版本控制；
- 对配置项进行变更控制；
- 定期进行配置审计；
- 向相关人员报告配置的状态。

3.4.5 VSS 的使用

目前，配置管理工具可以分为 3 个级别。

- 第一级别即简单的版本控制工具，是入门级的工具，例如 Concurrent Version System (CVS)、Visual Source Safe(VSS)；
- 第二级别即项目级配置管理工具，适合管理中小型的项目，例如 PVCS、MKS；
- 第三级别即企业级配置管理工具，具有强大的过程管理功能，例如 CCC Harvest、ClearCase。

自 20 世纪 80 年代后期研制并完善了“增量存储算法”，后配置管理工具的春天便开始了，目前国内常用的配置管理工具有 VSS、CVS 和 ClearCase。VSS 是微软公司推出的一款支持团队协同开发的配置管理工具，因为其短小精悍，又继承了微软集成销售的一贯作风——用户可以用相对于免费的价格得到，用户量是第一位。VSS 简单、易用，人们在使用配置管理工具的时候 80％的时间只是用 Add、Check in、Check out 等几个功能。VSS 的主要局限性是只支持 Windows，不支持异构环境下的配置管理，另外对 Internet 的支持不够完善。

VSS 是使用服务器、本地机的概念来进行操作的，所有需要操作的文件都存在服务器版本文件和本地机版本文件，无论用户的 VSS 的架构是服务器客户机形式还是个人单机版形式，机制都是这样。用户所用的修改都是在本地机上完成的，修改完成后再上传服务器。单机版也是这样操作。

服务器版本文件是一个绝对受配置管理软件限制的文件，用户只能通过 VSS 的规定的权限和操作方法修改，因为它并不是一个人的，而是大家的。本地文件是一个基本不受限制

的文件，用户可以像操作本地文件一样操作它。

VSS 由 Visual SourceSafe 6.0 Admin、Microsoft Visual SourceSafe 6.0、Analyze VSS DB、Analyze & Fix VSS DB 几个部分组成。Analyze VSS DB、Analyze & Fix VSS DB 两个工具不是很常用，前者用于检查 SourceSafe 数据库文件的完整性，后者主要修正 SourceSafe 数据库文件存在的错误。Visual SourceSafe 6.0 Admin 的功能类似于 Windows 2000 的用户管理器，软件配置管理人员用来分配用户和设定相应的权限。管理员的管理操作一般都集中在 Visual SourceSafe 6.0 Admin 中，系统中只有一个系统管理员——Admin 可以登录到此程序中进行管理工作，一般在刚安装的系统中此用户的密码默认为空。而且，系统为 Admin 这个用户保留的一切权利不可更改。数据库的创建操作必须在服务器上执行，因为通过客户端创建数据库操作只是在客户端的机器上创建数据库，这个数据库往往只能单机使用。图 3-8 所示为 Visual SourceSafe 的使用。

图 3-8　Visual SourceSafe 的使用

注意，由于 VSS 是通过 Windows 的网络共享来完成服务器端受控版本文件的共享，因此 VSS 服务端的数据库必须建立在服务器的一个完全共享的目录之中，否则客户端将无法获得数据库中的文件。下面介绍数据库的备份与恢复，如果要备份数据库的一个项目，选择 Tools→Archive Projects 命令，弹出对话框，根据提示一步步进行备份，最后会形成一个扩展名为 *.ssa 的备份档案文件。如果要从档案文件中恢复 VSS 数据库中的文件数据，选择 Tools→Restore Projects 命令，根据提示一步步完成数据恢复工作。其中，在恢复过程中可以选择恢复为原有工程，也可以改变恢复成其他工程目录。

下面介绍 VSS 的主要使用方法。

1. 添加项目

用户可以在根结点下添加项目，方法是选择 File→Add File 命令，出现 Add File 对话框后选中相关文件，单击 Add 按钮。用户可以通过 File→Create Project 命令在根目录下创建一个项目后在此项目结点下添加文件。添加完文件后，所添加源文件的属性自动变为只读，并在所添加文件的文件夹下生成一个 vssver 文件，以后对文件的操作就基本上与原文件没有关系了。

2. 浏览 Source Safe Server 中的文件

在 Visual SourceSafe Explore 中双击要打开的文件会弹出一个对话框,直接单击 OK 按钮即可。这时 SourceSafe Explore 会将文件复制一份到本地机的临时文件夹中(临时文件夹的路径通过 Tools→Options 设置),因原文件已经变为只读,所以临时文件也是只读属性,而且文件名会通过系统自动更改。

3. 设置工作文件夹

SourceSafe 的文件夹需要在本地计算机上指定一个“Working Folder”。当 Check Out 时,相应文件会下载到这个本地工作文件夹中。在本地的文件夹中修改文件,然后把修改后的文件 Check In 返回到服务器的 SourceSafe 中。用户可以使用 Set Working Folder 命令建立 Source Safe 文件夹和本地 Working Folder 的对应关系。方法是在 SourceSafe 的文件目录树中选中要建立对应关系的文件夹,再右击选择 Set Working Folder。

4. 下载最新版本文件到本地机

使用 Get Latest Version 命令可以将一个文件、一组文件或整个文件夹的最新版本从 SourceSafe 中复制到本地计算机中,并用只读的形式保存起来。方法如下:

在左侧的文件树中选择相应的文件夹,右击选择 Get Latest Version 命令,这时会弹出一个对话框,其中包括 3 个复选框,当 3 个复选框都不选中时,只将 SourceSafe 文件夹根目录下的文件复制到本地计算机,如同 DOS 中的 COPY 命令;当 Recursive 被选中时,会将 source safe 文件夹下的所有文件夹及文件都复制到本地计算机,如同 DOS 中的 DISKCOPY。如果 Make Writable 被选中,复制到本地的文件是可写的。

如果单击 Advance 按钮,就会出现更多的选择项。Set File 中的 4 个选项如下:

- Current 为复制操作发生时的当前时间;
- Modification 为文件最近一次修改的时间;
- Check In 为文件最后一次 Check In 时的时间;
- Default 同 Current。

Replace Writable 中的 4 个选项的作用是当本地机有一个和要下载的文件同名时,且本地机的文件是可写的同名文件,设置系统如何执行复制:

- Ask 系统提示是否覆盖本地的同名文件;
- Replace 自动覆盖本地的同名文件;
- Skip 不覆盖本地的同名文件;
- Merge 将两个文件合并。

大家一定要养成先 Get Latest Version 的习惯,否则如果别人更新了代码,VC 会提示存在版本差异并问是否覆盖、整合、保留等,如果选错了,就会把别人的代码 Cancel 掉,所以大家一定要小心。

5. 下载文件到本地

当修改一个文件时,首先要把文件从 SourceSafe 中复制到 Working Folder 中,并且以

可写的形式保存，这一系列动作的命令就是 Check Out。具体方法是选择要下载到本地机的文件，右击选择 Check Out，这时会弹出一个对话框。

默认情况下 don't get local copy，是不选中的，意义如下：如果不选保持默认状态，当本地的同名文件是只读时，系统首先用 SourceSafe 的文件更新本地的文件，本地的文件变为可写。当本地的文件是可写时，则会出现另一提示框，其中的选项 Leave this file 为本地文件保留当前状态，SourceSafe 中的文件也保留当前状态，这样有可能两个文件不一致；选项 Replace your local file with this version from source safe 为用 SourceSafe 中的文件更新本地的文件。如果选择 don't get local copy 选项，则不把 SourceSafe 的文件复制到本地。

在文件 Check 成功后，可以看到文件上有红色标记，这时的本地文件是可写的，就可以修改文件了。但上面的选项让人心乱，为了操作更简便，推荐一种 Check Out 方法：

- 当本地的文件比 SourceSafe 中的文件内容新时，选择 don't get local copy 选项，然后 Check In 使本地机与服务器内容同步；
- 当 SourceSafe 中的文件比本地机的文件内容新时，则在 SourceSafe 中选择此文件，使用 Get Latest Version 命令，然后按照默认选项进行 Check Out；
- 当两者内容相同时，按照默认选项操作。

注意，SourceSafe 中使用了文件锁的概念，当一个文件被别人 Check Out 时，其他人不能 Check Out 此文件；如果文件锁是无效的，可以查看 Visual SourceSafe 6.0 Admin-tools-general-allow multiple chechouts 选项是否被选中，当 Check Out 修改文件完毕后一定要 Check In，以保证 SourceSafe 中的文件最新。

记住，Check Out 时是使代码对自己可写，对别人只读，仅 Check Out 自己需要修改的部分，否则工作的时候同组成员只能休息了。

6. 上传文件到服务器

用户必须利用 Check In 命令保证 Source Safe 本地的文件同步，Check In 与 Check Out 成对出现，作用是用本地的文件更新 SourceSafe 中被 Check Out 的文件。

具体操作是在 SourceSafe 中选中处于 Check Out 状态的文件，右击选择 Check In，此时会出现一个对话框，在默认状态下两个复选框处于非选中状态。

- 选中 Keep checked out 选项，可以在 Check In 后自动地再次 Check Out，等于是省略了下一步 Check Out 操作；
- 选中 Remove local copy 选项，可以在 Check In 的同时删除本地机上 Working Folder 中的同名文件。

一般按照默认选项就可以了。Check In 成功后，SourceSafe 和本地的文件是完全相同的，本地的文件变成了只读文件。当要再次修改文件时，再执行 Check Out 操作，此时本地机的文件属性自动变为可写状态。大家一定要记住，Check Out 后要 Check In，否则导致的后果如同写完了文件不保存一样。

大家要保证文档正确、可编译后再 Check In，否则会使其他人也无法通过编译，整个工程没法调试。

7. Undo Check Out 操作

当一个文件被 Check Out 后,如果想要撤销这项操作,可以使用 Undo Check Out 命令,操作步骤为选中处于 Check Out 状态的文件,右击后选择 Undo Check Out。当 SourceSafe 中的文件和本地的文件完全相同时不出现提示信息,文件恢复为普通状态;当 SourceSafe 中的文件和本地的这个文件不完全相同时出现提示对话框,其中包括 3 个选项。

- Replace 选项被选中后会出现系统询问是否覆盖的信息,如果单击 Yes,则用 SourceSafe 上的文件的最后一个版本覆盖本地机上的文件,如果选择 No,则保留本地计算机上文件的内容,SourceSafe 上的文件是上次 Check In 后的内容,此时两个文件可能出现不同;
- Leave 选项保留当前计算机上的内容,SourceSafe 上的文件是上次 Check In 后的内容,两个文件可能出现不同;
- Delete 选项删除本地计算机上的这个文件,选择一个选项,单击 OK 按钮后,文件回到普通状态。

8. Edit 操作

Edit 命令是一个组合命令,是先 Check Out 再修改的命令的组合。大家应当注意,执行 Edit 命令后修改了文件,但是 SourceSafe 中的文件并没有同步修改,还是要 Check In 完成本地文件与 SourceSafe 上文件的同步。

9. 查看文件的历史内容

选中此文件,右击选择 Show History,出现一个对话框,单击 OK 按钮,弹出一个窗体,可以看到文件的历史内容。对于文件的所有版本,如果要查看某个版本,可以单击 View 按钮。如果想下载某个版本,可以单击 Get 按钮。

10. 关于 SourceSafe 的权限

在默认情况下,项目安全管理以简单模式来运行,即用户对工程的操作的权限只有两种,一种是只读权限,一种是读/写权限。如果要启用高级模式,可以在 Visual SourceSafe 6.0 Admin-tools-project security-enable project security 中将此项选中。

SourceSafe 的权限分为下面 5 级。

- 无权限级:看不到文件。
- Read 级:自能浏览文件,可以使用 Get Latest Version 命令。
- Check In/Check Out 级:可以更新文件,但不能对文件进行删除。
- Delete 级:可以删除文件,但通过某些命令这些文件还能恢复。
- Destroy 级:可以彻底地删除文件,删除之后无法恢复。

为用户设定权限的工作一般由软件配置管理员在 Visual SourceSafe 6.0 Admin 中完成。权限管理就是管理用户和工程目录之间的操作权限的关系。因此有两种管理方式,一种是以工程目录为主线来管理权限,一种是以用户为主线来管理权限。若以目录为主线管理用户权限,则选择 Tools→Right By Project 命令,弹出对话框来管理项目的用户访问

权限。

如果以用户为主线来管理权限，则应先在主界面下方的用户列表中选中一个用户，再选择 Rights Assignments For User 命令，弹出对话框，在对话框下方列出了该用户对数据库各项目目录的访问权限，如果访问某个项目，在列表上没有列出，则说明该项目的权限是继承上级目录的访问权限。只要选中一个目录，就可以编辑该用户对该项目目录的访问权限。

权限复制就是将一个用户的权限直接复制给另外一个用户，管理员可以通过选择 Copy User Right 命令来实现。

11. 关于 Password 的更改

Password 一般是由软件配置管理员分配的，如果需要修改密码，可以选择 Tools→Change Password 命令修改。需要说明的一点是，当 SourceSafe 密码和 Windows 密码相同时，启动 SourceSafe 不会出现提示输入密码的对话框。这是微软的一贯作风，在 SQL Server 数据库管理系统下也能找到这个影子，因为微软认为 Windows 的密码应该比其他软件的密码级别高，既然能用相同的用户名和密码进入 Windows，那么也就有权使用相同的用户名进入其他的软件。

3.5 小结

测量使得管理者和开发者能够改善软件过程；辅助软件项目的计划、跟踪、控制；评估产生的产品(软件)的质量。对过程、项目、产品的特定属性的测量被用于计算软件度量。分析这些度量可产生指导管理及技术行为的指标。对于一个特定的软件，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。

过程度量使得一个组织能够从战略级洞悉一个软件过程的功效。项目度量是战术的，使得项目管理者能够以实时的方式改进项目的工作流程及技术方法。软件配置管理覆盖了整个软件的开发过程，因此是改进软件过程、提高过程能力成熟度的理想的切入点。

思考题

1. 简述影响软件质量的因素。
2. 简要描述几种常见的质量保证模型。
3. 简述软件过程度量的目标、对象、方法和结果。
4. 简要描述软件配置管理过程。
5. 安装并学习使用 Visual SourceSafe。