第 4 章 面向软件开发过程的软件配置管理系统

软件配置管理领域的现状是多种理论、多种模型与多种工具并存，没有哪一种工具能提供所有配置管理模型和概念的支持。总体来说，各种理论和模型都有如此共识：软件配置管理系统是团队软件开发的重要支撑体系，配置管理技术的关键是如何将配置管理与开发过程相结合。本章从软件开发过程的模式着手，讨论软件开发过程与软件配置管理的关系，剖析软件配置管理对软件开发过程的四大要素，即人员、任务、方法和产品的管理支持作用。基于软件配置管理和软件开发过程关系分析，以 ISO 和 CMMI 中对软件配置管理的要求，提炼出强力支持软件开发过程的软件配置管理系统的模型。

4.1　陷入泥潭中的项目——没有完善配置管理的软件开发

本节首先讲述一个没有完善配置管理的软件开发的真实案例，然后再分析这种缺乏完善的配置管理系统可能会对项目造成的影响。

1．案例

首先从一个真实的软件开发项目来看没有或不健全的配置管理系统会引发的问题。

某软件开发公司负责为某资金雄厚的电力公司开发一套电网计费系统，该电力公司要求系统分两期完成。开发公司在签订第一期的合同后即组建项目组开始软件开发。项目组决定采用瀑布开发模型，预计经过系统分析、设计、编码、集成、测试，最后把合格的产品交付给客户。该软件开发公司配备了简单的软件配置管理系统，能进行版本管理，但不能对整个开发过程实现系统化支持。

项目启动后，开发人员到客户现场进行了用户调研，搜集了系统的基本需求。但是，由于涉及的系统和用户角色较多，短时间内不能全面了解系统的需求。分析人员本着边做边看的心态，将基本的需求描述下来后交给设计人员，期望需求文档中的许多疑点在后续过程中将会得到补充和完善。设计人员针对基本的系统需求形成了设计文档，并交给编程人员进行编码。与此同时，分析和设计人员手中都有各自生成的文档，并在编码阶段进行过多次修改。没有人能够确定这些修改是否及时传达给了相关人员。编程人员加班加点地按照所分配模块的设计文档进行编码，保证模块编译通过，在自己设计的的试验环境中基本能运行后，就把代码检入到代码库中。全部模块编码完成后，本来预计两周的集成联调花费了两个月才完成。原因是系统输入数据的格式和模块接口在编码过程中发生了改变，许多模块需要大量改写，而部分开发人员在改写时用错了代码版本，引起了严重的代码执行错误。联调成功后，留给测试人员的时间已经不多了，测试人员只能草草地进行一些基本功能的测试，之后就由项目经理把第一期的产品按照合同规定的日期交付给客户。

可以想象这种过程所交付的产品的质量。由于使用中遇到的问题太多，产品交付后的几个月，项目组一直忙于应付用户上报的产品缺陷，而其中有些缺陷应当划归第二期系统的功能需求。为此，整个项目组卷入与客户的无穷无尽的就功能细节进行讨价还价的会议中，同时项目组内部也为各个缺陷的产生原因争论不休，相互推托。由于许多严重的缺陷始终得不到有效地解决，客户甚至扬言如果第一期工程中的问题在延期半年后仍然不能妥善解决，将不签订第二期的合同，并且该电力公司将不再委托该软件开发机构开发其他任何项目。在各种内外因素的压力下，不少开发人员感觉前途渺茫，工作效率低下，工作状态极差，陆续选择了离职。至此，整个项目陷入内外交困的泥潭中。

2．影响

这是一个很普通的软件开发项目，具有类似过程和结果的项目在现实中比比皆是，特别是在一些中小型软件开发机构中时有发生。其根本原因就是缺乏有力的工具和方法去管理软件开发过程，没有完善的软件配置管理系统支持。

具体来说，缺乏完善的配置管理系统可能会对项目造成如下影响。

1）不能准确地界定项目需求

项目的需求是经常变化的，但必须有一定的机制记录需求的变化，确定一个阶段的相对固定的需求。配置管理系统中的需求基线就是用以记录、界定项目需求的有效方法。没有完善的配置管理支持，不能建立项目的需求基线，开发人员就不能精确地把握项目需求，也不能跟踪需求的变化。

2）不能有效地管理源文件版本

没有与开发流程紧密结合的配置管理系统，很难实现完善的版本管理，经常会造成编程人员要修改代码时不知道该用哪一个版本，多个人同时修改时有些人的修改被覆盖了，一个文件改变后受影响的开发人员不能及时得到通知等版本管理问题。版本混乱通常是引起开发进度滞后、产品质量下降的重要原因。

3）多版本并行开发很困难

正式的产品通常都会后多个版本并行开发的情况。没有完善的配置管理系统，在进行多个产品版本、多个维护分支并行开发时，开发人员要手工地保持多份不同的副本，改正同一个问题，需要在多个副本上进行修改，或者在不同副本中复制修改后的文件，导致应有的版本差异消失。此外，在版本分支和合并时缺乏自动化支持，会带来许多重复劳动，或者引发严重的回归缺陷。

4）无法有效地管理和跟踪变更

没有良好的配置管理将无法对软件的变更进行有效的记录、跟踪和控制。使产品的修改成为编程人员制造的黑盒，项目管理者无法了解更改的原因和结果，更不能对变更进行引导与控制。

5）不能及时了解项目的进展状况

缺乏配置管理工具对开发过程的支持，部门主管无法确切得知项目的进展情况，即便是项目经理也不知道各个开发人员的具体工作，项目进展随意性很大。项目开发过程变成项目经理无法了解、无力理清、无从下手的一个黑箱。 所有的问题往往都会集中到项目里程碑时一起出现，其结果往往是容忍部分缺陷的存在或者拖延开发工期。

6）无法开展规范化的测试工作

缺乏必要的配置管理和变更控制机制，开发和测试人员不能顺利地进行分工和合作。测试人员很难确定测试的范围和目标，测试的代码版本也经常不确定。另外，由于开发人员对复杂的功能模块经常遮掩搪塞，导致测试工作往是走走过场，测试结果既无法考核又无法量化，根本不能作为产品质量保证的依据。

7）缺乏客观的产品交付依据

没有完善的配置管理系统，不可能实现从产品功能需求、设计到编码测试的可跟踪性，不能准判断产品开发团队的状况和产品的完成情况，没有客观的数据来确定产品是否可以交付。

8）对软件版本的发布缺乏有效的管理

没有完善的配置管理系统，往往会在产品发布时无法明确标注各个组件的版本，甚至用户提供错误的版本。对于特定客户出现的问题，无法重现其使用的版本，只能到用户的现场才能进行调试查错。

9）软件复用率低下

没有良好的支持代码组件化的配置管理系统，软件复用的效率将大打折扣。产品开发人员不能将产品的共用组件置于独立的配置管理单元加以控制，只能通过手工的方式维护可共享的组件。为了省去烦琐的手工共享方式，开发人员往往将本可以复用的组件代码混合到其他代码中，降低了代码复用比率。

10）不能有效地促进开发团队和开发过程的提高

没有完善的配置管理系统，不能准确、全面地收集和保存开发过程中的动态数据。软件开发的历史数据是反映软件开发队伍的能力的标尺。没有这个标尺，就无法对软件开发团队和开发过程的成熟度有准确的认识，也不能发现其中的优缺点，以便进行扬长避短的改进。

这些影响对不同的项目或项目的不同阶段表现的程度不一，但对于复杂的或长期的项目而言，它们往往是将项目拖入泥潭的主要因素。本章后续的几节将深层次地讨论配置管理为何对软件开发有如此巨大的影响，如何实现配置管理系统对开发过程的有力支持。

4.2　软件开发过程的基本要素与本质特性

本书前两章对软件开发过程和软件配置管理的基本内容进行了介绍。在进一步分析配置管理对软件开发过程的支持之前，先对软件开发过程进行一个较为深入的透视。

4.2.1 什么是软件开发过程

在《统一软件开发过程》一书中是这样描述软件开发过程的：软件项目的最终结果是一种产品，产品在其开发期间是由许多不同类型的人员建造的。指导项目人员工作的是软件开发过程，它是一种模板，阐明了完成项目所需的各个步骤[6]。

《统一软件开发过程》中确定软件开发的四要素（4P）为人员（People）、项目（Project）、产品（Product）和过程（Process）。人员是软件项目的主要推动者，包括架构师、设计师、开发人员、测试人员、管理人员，此外还包括用户、客户及其他的相关个体；项目是管理软件开发的组织元素，项目的成果是可以发布的产品；产品是在项目的生命周期内创建的制品，包括模型、源代码、可执行代码、文档等；过程是将用户需求转化为产品所需要的完整活动的集合。过程一般可以通过一种或一组工具自动完成。

人员、项目、产品、过程和工具的关系如图 4-1 所示。

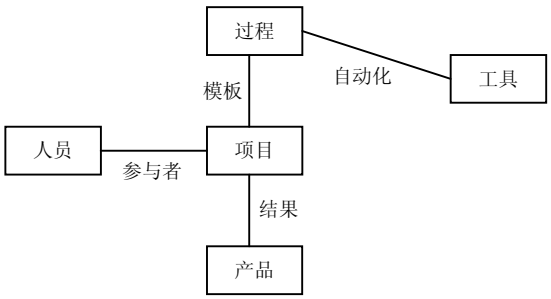

图 4-1　统一软件开发过程中的四要素

基于对统一过程和其他开发过程的研究归纳，有些研究人员提出了“软件过程模式”的概念。这一概念指明软件过程模式是“从成功或失败的软件开发实践中总结而成的，是软件过程生命周期、人员、方法、产品四大类要素相互关联的有机整体”。过程模式理论中确定的生命周期、人员、方法与产品是软件开发过程中的四大要素。这四大要素回答了软件开发过程中“谁”（人员）、为实现什么”（产品）、“如何”（方法）、“做什么”（生命周期）等问题[7]。

结合统一过程理论和过程模式理论，本书认为软件开发过程的决定性要素是：人员、任务、方法和产品。任务是软件开发项目的工作目标和内容；人员是任务的创建者和执行者；产品是任务完成后得到的成果；人员在创建和执行任务时都必须按照一定的方法进行。软件开发过程的本质就是项目参与“人员”（开发团队）采用一定的“方法”（阶段、过程）去完成一些“任务”（项目目标）交付特定的“产品”（开发结果）的一个生产性过程。在这整个的过程中，只有合理地运用或管理好这些要素，才能使软件产品的开发生产到达可控制、可跟踪和可预测的状态，使项目参与人员成为项目的主人，而不是项目的奴隶。

4.2.2　软件开发过程的四要素

为了更清楚地认识这些决定软件开发成败的关键因素，有必要对人员、任务、方法和产品这四要素进一步分解说明。

1．人员

软件开发过程中的人员主要包括软件开发活动的参与者和支持者。他们在整个项目开发过程中担当一定的角色，负责一定的任务或工序。项目中所有的人员按照一定的组织机构形成一个整体的开发团队。

按角色职责划分，这些人员通常分为机构管理者、项目管理者（项目经理）、小组负责人、设计师、程序员、测试人员、发布与客户联络员。人员中的角色和个体之间可以是多对多的关系。所有这些角色的个体效率和协作效果是决定项目成败的关键。这些人员是各种软件开发工具，包括配置管理工具的直接使用者。

2．任务

软件开发过程中的任务是软件开发过程中各种活动的起因。每个开发人员都是在特定任务的驱动下进行开发活动的。

任务是可以分解的，大的任务分解成小任务后可分配给不同的开发人员并发执行，任务的分解可以按开发小组、按开发阶段、按产品部件或按产品阶段等方式进行。软件开发项目本身也就是项目中最大的任务，由项目经理负责完成。不论哪一个层次的任务，都包含目标、范围、依赖关系、里程碑、花费、进度、完成质量等管理属性。

3．方法

软件开发过程中的方法包括开发阶段划分、各阶段采用的开发技术、开发工具的使用、开发人员沟通协作方式及其他开发管理手段。其中，开发阶段划分是软件开发过程中最主要的方法。

尽管不同开发机构、不同项目规模和不同开发模式对开发阶段的划分不同，但总体上应包括需求分析、系统设计、编码、测试、产品发布等阶段或活动。开发管理手段包括人员管理、需求管理、进度管理、质量管理、发布管理等。开发技术、开发工具和沟通方式等方法则因不同项目会有很大的差异。

4．产品

软件开发过程生成的产品可以是不同类型的制品、不同阶段的制品，如产品代码、设计文档、系统模型、界面原型、构件、测试计划、测试规程等。多数产品都是分阶段逐步

完善的，因此它们一般都有多个部件、多种完成状态、多个不同的版本。最终发布给客户的产品必须是固定的、可重构的。同一个项目可能会发布多个产品或同一产品的多个版本发布给最终用户。

4.2.3 软件开发过程的本质特性——复杂性

软件过程模式的四要素及其相互关系是项目计划、风险评估、人员管理、质量保证等项目管理实践的重要依据。软件开发项目中所有活动都是围绕这 4 个要素进行的。从更大的范围来看，软件开发过程的四要素及其本质过程也适用于其他领域的项目实施或产品开发的过程，区别在于不同领域中的每个要素所包含的内容不同，复杂程度不同。

软件开发过程的复杂性就是源于软件产品的复杂性、易变性的，从而使软件开发过程中的人员复杂化、方法复杂化、任务分工复杂化。传统产品的生产通常只需要把个各部件生产出来后，按照物理空间位置关系拼装在一起就形成了完整的产品，而且产品一经形成不会轻易被改变。与此相反，软件产品除了公认的易变性（即很容易被改变）之外，其部件划分和组装方式都是通过各种复杂的软件接口实现的，稍有出入就不能组装成产品，因此具有较大的不确定性。

如何使软件开发项目和产品的开发生产能像传统工业产品一样进行有效地分工分块、快捷的拼装合并，这是软件开发领域的一个持久的“结”，迄今为止没有完美的解决办法。在接下来的几节中将讨论软件配置管理系统如何通过对软件开发过程各个要素的有力支持，在很大程度上缓解软件产品的这个“分”与“合”的“结”给软件开发过程带来的负面影响。

4.3 软件配置管理与软件开发过程要素的关系

软件配置管理和软件开发过程是紧密联系的，这种联系体现在软件配置管理过程和相关功能分布在软件开发过程的各个环节，对软件开发过程中的各个要素都有重要的支持或调控作用。正如 Brad Appleton 在《软件配置管理模式》一书中所说的，对软件开发过程而言，软件配置管理是软件工件、特征、变更，以及团队成员之间的“黏合剂”，在从概念到交付等整个软件开发过程中，软件配置管理是将这些元素联合起来的纽带[5]。

下面详细分析软件配置管理与软件开发过程的四要素，即人员、任务、方法和产品的关系。通过这些分析，可以看到配置管理过程如何对各个要素进行管理与支持，如何将各个要素有效地关联起来，以实现成功的软件开发过程。正确地实施运用这些过程和关联才能避免前述软件项目中的问题，协调软件产品开发过程中有序的“分”与“合”的活动。

4.3.1　对人员的支持

软件开发过程中的人员包括项目参与人员和各种相关人员。尽管不同开发机构、不同开发模型对这些人员的组织方式和职责划分大不相同，但仍可从人员职责权限、参与的工作环节（工序）和工作内容范围等角度划分出人员的通用角色。不同角色对配置管理系统的功能需求不同。

1．从职责权限方面划分

从职责权限来看，软件开发项目中的人员大体可分为项目发起人（Sponsor）、项目管理人员、技术负责人员、开发小组负责人员和开发活动执行人员。

1）项目发起人

项目发起人是项目的支持者和监管者。其通常负责项目的财政支持、客户需求、外围联络和产品交付等关乎项目基本生存条件和项目价值体现的重要事务。项目发起人关心项目的可行性、开发团队的高效性、产品的可用性、发布的及时性。通过对配置管理系统所收集的开发过程和配置项相关数据汇总分析，可以客观地度量关于项目中人员、产品和进度状况，形成项目发起人所需要的数据报告，便于他们对项目进行总体监督调控。

2）项目管理人员

项目管理人员承担对项目开发实施和产品交付整个过程中开发人员和任务的管理和协调职责，是开发项目的总负责人。其负责项目内部具体开发任务的定义、分派、监督和调整。项目管理人员关心项目中任务的完成状况、人员的工作负荷和产品的完备程度。

配置管理系统中的变更管理过程为项目管理人员提供了任务定义和分配的支持；配置管理状态报告和配置审计活动有助于项目管理人员及时了解项目状况和产品演化情况；通过配置基线可以有效地定义和验证开发里程碑，以总体把握项目任务完成状况和产品完备程度。此外，通过对传统配置管理系统功能稍加扩充延伸，就可以有效地支持项目管理者对项目中人力资源进行管理调配。

3）项目技术负责人

项目技术负责人在项目开发技术层面承担指导和决策职责。这种角色在有些开发过程中称为系统架构师，在有些过程中称为程序经理[8]或开发经理[9]。项目技术负责人通常负责制订项目开发策略、产品功能和架构设计、工作量评估和工作结果检验等事务。

技术负责人是变更控制委员会（CCB）的重要成员，是决定变更请求的主要力量。配置管理系统中的变更管理机制是项目技术负责人推动开发进度、引导产品演化的主要途径。此外，他们还通过组建配置构件来定义产品结构；通过发布需求和设计基线来引导开发团队

的开发进程；通过评审配置报告（如代码修改行数统计）和配置审计结果来校正实施偏差。

十人以上的开发项目通常会将开发人员划分为多个开发小组，由每个小组的负责人协调管理小组的开发任务和活动，同时在技术上起一定的指导作用。小组负责人需要及时响应分配给该小组的任务，调整小组内部的任务分配，掌握小组人员的工作进度和完成质量。配置管理系统通过记录每个开发人员的变更请求和变更结果，能在一定程度上满足小组负责人的需要。通过扩充定制现有配置管理工具，还能够提供更便利的小组管理功能。

开发团队中的大多数成员的主要职责是负责执行分配到的具体的开发任务，如设计、编程、测试、编写用户文档等事项。他们是配置管理系统的主要使用者，在执行任务的日常工作中通常都需要与配置管理系统进行交互，例如检出旧版本、检入新版本、查找历史版本等操作。事实上，配置管理系统中的大部分配置项都是由这些开发人员生成演化的，这些配置项的版本记录在开发过程中经常需要查考引证。因此，配置管理系统为这些执行具体开发任务的开发人员提供了开发成果库和知识库的功能。同一个产品不同时期、不同地域的开发人员通过配置管理系统提供的配置项信息能够进行最客观地沟通、最直接地协作。

2．按开发工序和工作范围划分

以上是从职责权限或组织架构方面分析软件开发过程中的人员角色和软件配置管理对各种角色的支持作用。还可以从开发人员从事的开发工序和工作范围两个方面来讨论这种支持关系。

按工序划分，软件项目中的开发人员通常包括架构设计人员、子系统设计人员、编程人员、测试人员、发布与产品管理人员等角色。每种角色的工作内容、方式、结果，以及所关注的问题各不相同。常见的软件工程文献中对这些角色的区别和关联都有详尽的阐述。在此，关注的是软件配置管理系统能够支持各种开发工序中的角色。通过相应的配置管理工作空间，配置管理系统可协助不同开发角色方便地进行设计、开发，生成开发结果，并将各种结果妥善保存在配置库中。通过变更控制、基线管理和版本管理等机制，配置管理系统能使各个角色的开发结果在整个开发过程和开发团队中进行正确、有效地演化、延续和传播，从而支持不同角色之间在软件开发过程中的分工和合作关系。

软件开发团队中即使是相同职权、参与相同工序的开发人员，也可以再按工作内容、范围细分成不同的小组或个体。例如，每个用例的分析人员、每个构件的设计人员、每个模块的编程人员或每个功能的测试人员等。这些角色都可能是由相当独立的开发小组构成的。这些开发小组多数时间内是在各自的工作对象范围内进行设计、开发或测试工作，生成各自的工作结果，但这些工作结果最终必须要合并在一起才能形成整体的项目结果或完整的产品。另一方面，作为整体的软件产品，其演化过程中必然要涉及各个部件的修改，只有为每一部件指派固定的开发人员，才能最有效地修改该部件。配置管理系统提供了配置项标识、存储功能和相应的版本分支合并、配置项组合、作者标记、变更记录、访问控

制、基线管理等功能，为各个开发小组或个体合并开发成果、建立部件关联、确定修改责任等同级协作过程提供了重要的技术手段。

总之，软件配置管理通过对开发团队中不同职权、不同工序、不同工作内容的开发人员的支持，使软件开发团队中的所有角色能够有效地进行分工合作，以达到软件开发项目的目的。

4.3.2　对任务的支持

如前所述，软件开发过程中的所有活动都是因任务而起的，以完成特定的任务为目的。任务是可以细分和组合的，可以把开发过程中所有人员从事的所有工作划分为一个个相互关联的任务，软件开发项目中的所有工作项就是这些任务的集合。大多数软件开发团队按软件开发阶段、软件功能特性、软件代码模块或软件发布版本划分软件开发任务，并分配给具体的个人或小组。图 4-2 所示为常见的软件开发项目任务分解图[11]。其中分析、设计和开发任务中的各个子任务又可以按功能模块和具体任务再进行细分。

软件开发工作分解为各个任务后，就拥有了制订日程安排、编制预算和分配资源的基础信息。以此为基础，就可以为任务分配时间，分派给具体的开发人员，并在整个项目开发周期内对各项任务的完成状况进行跟踪、控制、调整、汇报。

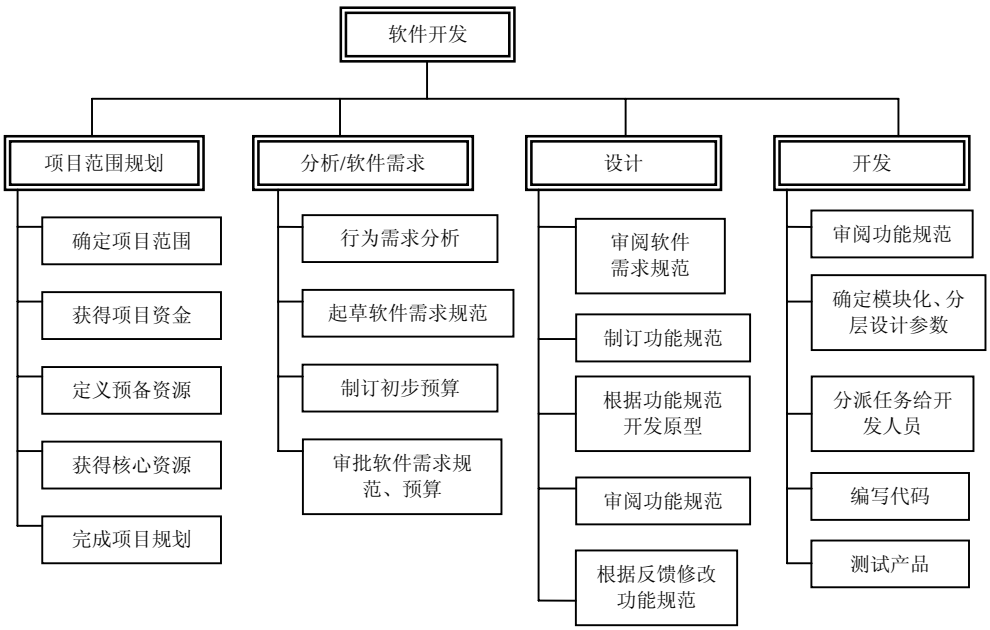

图 4-2　软件开发任务分解示意图

对大中型软件开发项目而言，所有的开发任务将会细分为一张极其复杂的图，加上任务间的组合、顺序和依赖关系，以及任务和人员间的多对多关联，因此必须有一定的开发

管理工具来支持这种繁杂的任务关系图。软件配置管理工具就是这些对开发任务进行支持管理的工具中最重要的一种。

软件配置和开发过程管理系统对任务的支持可以分为从基本到完善的 4 个级别，每个较高级别都是在较低级别的基础上的进一步完善。处于第一个级别的系统只支持任务的定义和分配，并不对任务的步骤、进展、依赖关系进行进一步的规定和支持，所由余下的工作由手工或其他系统完成；处于第二个级别的系统除了能够定义任务外，还能规定任务的状态转换，并且用某种机制来支持状态转换的规则，这样用户可以规定任务完成的步骤，控制任务的进展，但是任务之间是毫无联系的，需要人为地保证任务的依赖关系；处于第三个级别的系统在第二个级别的系统基础上增加了任务关系的支持，用户可以在系统中定义任意任务之间的关系和相关的规则，系统提供保证这些关系和规则的机制。因此，能够轻易地构建任务之间的关系网络，便于管理控制。对大型项目而言，开发团队需要对整个项目的任务有全面的把握，这就需要第四个级别的系统进行支持。处于第四个级别的系统能对任务和任务关系分类，并内置典型的任务关系支持，如父子关系、参照关系、模块关系，使任务关系系统地反映项目的状况，同时能从任务数据中提炼出反映开发过程质量和产品质量的度量数据（Metrics）。

多数的配置管理系统通过变更控制和基线管理过程可以实现前两个级别的任务支持。更完善的配置管理系统具有强大的任务管理支持，能够支持任务的关联和系统化管理，实现第三和第四个级别系统的功能。

4.3.3 对方法的支持

如前所述，软件开发项目通常由许多的开发人员、不同的角色参与，完成许多繁杂的任务，最终生成复杂的软件产品。为了顺利完成开发过程，方法和工具至关重要。

从宏观上讲，软件开发过程中采用的首要方法就是阶段划分法。ISO/IEC15504 标准中将软件系统开发过程分为 6 个子过程：开发系统需求并设计、开发软件需求、开发软件设计、设计实现软件、集成并测试软件和集成并测试系统。如果纯粹从软件项目开发的角度来看，开发过程包括开发软件需求、开发软件设计、设计实现软件、集成并测试软件，这就是软件开发过程中进行阶段划分的基本标准。

前面介绍的各种软件开发模型，不管是传统的瀑布模型、螺旋模型，还是新生代的迭代模型和敏捷模型等，在实际运用时还是围绕着需求分析、设计、编码实现、测试这些核心活动划分开发阶段和任务的，只不过在每个阶段的名称、周期、迭代的方式、衔接方式和中间产品交付方式等方面有所差异。

通过基线管理，在各个开发阶段结束时发布里程碑基线，配置管理过程能够有效地支持软件开发阶段的划分。

如图 4-3 所示，在软件开发过程的每个主要阶段结束时，该阶段的开发任务基本完成了，相应的开发结果必定生成并检入到配置库中，此时生成的产品基线就能代表项目在相应里程碑的成果，同时宣告一个开发阶段基本结束。例如，需求分析阶段结束时生成的需求基线代表需求文档、用例基本定型，需求分析工作告一段落，设计任务即将开始。不过，注意上述语句中的“基本”二字——基线并不意味绝对不变，即使某一阶段的基线生成了，还可在后续阶段对基线化的成果进行变更，依据变更管理过程对前一阶段的工作进行完善。不断地进行完善，修正初步成型的产品，就是软件开发的复杂性的主要表现。为了尽量避免不可预测的或不必要的返工，有些软件开发模型，如螺旋模型和敏捷模型采用迭代式开发阶段和基线发布过程，即进行多次由需求分析到测试的开发过程，每次只实现特定的功能，产品基线也相应地分为阶段性基线和功能性基线。功能性基线是用于对特定功能进行测试的基线，其代表一次迭代基本结束，相应功能基本实现。

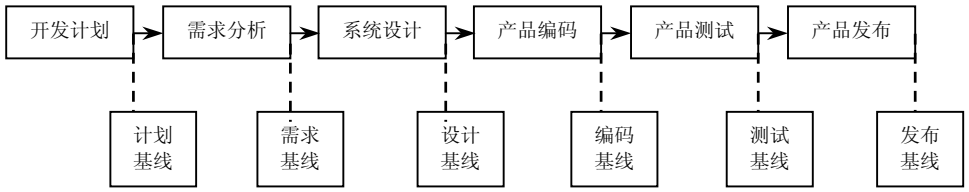

图 4-3　不同开发里程碑的基线

可见，不论开发阶段是否迭代，基线都体现不同开发阶段实现的不同子目标，生成不同形式、不同状态的产品，直到最后一个阶段或最后一次迭代完成时生成的发布基线就代表最终产品。

从微观来看，软件开发过程中需要有一定的方法来支持开发人员的分工协作、信息共享、任务安排、交付管理、进度管理、质量控制等日常开发细节，以保证各个开发角色在整个开发过程中紧密合作，有条不紊地完成开发任务。这种方法通常以一种或多种开发工具体现，即利用相关的工具为项目的所有开发人员搭建一个集成、统一的开发平台。不同角色的开发人员运用开发平台的不同功能完成相应的开发任务，由开发平台自动进行任务结果收集、产品集成和数据统计等功能。

配置管理系统是软件开发平台的核心和基础，它能为各种软件开发工具提供和保存开发数据，为不同用户提供独立的开发空间，使不同阶段、不同角色的开发人员在整个软件开发过程中能各自独立地设计、开发、测试，同时又能将最终的产品集成到配置库中。好的配置管理系统能与其他软件开发工具无缝地集成。这些工具包括需求管理工具、设计建模工具、集成开发环境（IDE）、集成构造工具、测试工具、项目管理工具、质量管理工具，以及缺陷跟踪工具等。配置管理工具是这些工具的枢纽，为这些工具收集和存储开发过程数据和各开发阶段的成果，由这些外围的工具为开发人员提供操作界面和数据表示。

基于配置管理系统和其他开发工具关联而成的开发平台能够方便地实现开发人员的协

作互动、数据共享、问题反馈、进度报告等日常开发工作需求。先进的配置管理工具还能记录整个开发过程中每个功能点从需求到最终产品的各个环节中的变更对应关系，以直观、形象的方式展示产品的演化历程。此外，对配置管理系统所收集的开发过程数据，例如代码改动次数、修改行数、修改人员、修改时间、任务信息、模块信息等进行挖掘、统计、分析，能生成客观的反映软件开发过程质量和产品质量的量化数据。在后一节中将会详细讨论这些量化数据在项目管理和开发过程改进中的作用。

4.3.4 对产品的支持

软件开发项目的产品通常以不同类型的计算机文件呈现。可从这些文件的类型、状态和整个产品的结构、功能、状态等方面来探讨配置管理对软件产品的支持。

如前所述，软件产品是经多个阶段由许多角色合作形成的，每一阶段都会形成特定的文件，代表相应阶段的产品模型。不同阶段的产品文件有不同的文件类型，例如多媒体文档、图表、脚本、格式化数据、程序等。这些不同类型的文件一般由不同的工具生成和维护，具有不同的内容显示、比较、合并特性。配置管理系统通常能够识别不同类型的文件，将文件的类型属性连同其他属性一起保存在配置库中，在进行配置项操作时自动调用相应文件类型的工具进行文件内容处理，不需要用户手工选择打开特定的工具，这简化了文件的版本管理操作。此外，不同阶段的产品文件在开发过程中都会经历私有的、共享的、集成的（集成到产品基线中）等状态。配置管理系统通过良好的版本控制，能方便地提供文件及版本的状态转换，结合文件类型处理，为开发人员在整个开发过程中提供便捷的文件操作方式。

从产品结构上看，软件项目交付的产品都是以一定的结构模块体现的。配置管理系统通过配置项和子系统的集合通常能提供与产品构件一致的配置构件，并能有效地控制构件的变更、维护构件的演化历史，即将紧密关联的配置项集中为一个配置组件或模块，在配置库中独立管理控制。优秀的配置管理系统还能支持多层次构件，通过引用已有的构件配置定义更大的构件，直至定义整个产品的结构。基于构件的配置定义和引用过程体现了整个产品的组装过程，保证了配置项的一致性，提供了对产品构建过程的支持。产品开发人员通过基线操作就可以得到各个组合构件和整个产品的正确版本。产品构建人员通过基线操作就能够在任何时刻方便地重构不同阶段、不同部分的产品。

典型的软件产品开发策略是对已发布的版本的维护工作与新功能的开发工作同步进行，但是工作在不同版本的代码上，甚至由不同的团队开发。因此，从产品的发布状态来看，同一产品的多个已发布的版本和即将发布的版本经常会处于并行开发的状态。对已有版本的维护支持主要是修复其中发现的缺陷、开发相应的补丁程序，以保证现有用户的正常使用；新版本的开发主要是开发新的功能，或对新平台、新技术的支持。为了避免并行开发中的版本冲突，必须充分发挥软件配置管理的功能，制订完善的配置管理策略。支持

多版本并行开发的策略是软件配置管理策略中最为复杂的部分，也是软件配置管理最大的价值所在。只有做好基于产品的配置管理，对并行开发加以协调和控制，管理好版本分支，才能灵活地处理好并行开发任务之间的产品版本的顺序关系。产品版本之间的顺序关系与项目任务之间的依赖关系是相互影响的，从某种意义上说这种顺序关系会决定项目或团队的成败。只有定义并实现了合理产品版本关系才能使项目的每个开发任务基于最恰当的产品版本，才能最快速地交付新的版本。

产品版本除了时间上的并行开发之外，在空间上也会有并行开发的情况，即为不同的客户定制产品的不同版本，以满足不同用户的细节需求。这种并行开发中尽管产品的主体架构是相同的，许多代码是共享的，但仍有不少模块或不少程序必须采用不同的版本。产品的每个功能需求都可能有一些细节差异，相应地必须要由不同的代码或同一代码的不同版本实现，这就形成了产品需求和产品代码之间的矩阵关系。这种功能需求和产品代码之间的矩阵关系只有通过完善的配置管理系统的版本分支和基线引用功能才能很好地实现。

由此可见，配置管理系统通过对软件开发过程中的人员、任务、方法、产品等方面的支持管理，使软件开发过程中各个要素能够有序地关联起来，有力地支持了整个开发过程中的分解与合并、分工与合作的流程，支持了软件开发过程的有序化。当然，许多现有配置管理工具只能以简单的方式进行支持，对多数工具而言，需要进一步开发定制以提供更完善的过程支持、团队协作支持、任务管理支持和质量管理支持等功能。

4.4　软件配置管理驱动软件开发过程的量化和改进

在软件开发领域，度量是根据一定的规则对软件过程或产品属性赋予量化值的软件开发过程管理方法[34]。能够对软件开发过程进行有效的度量是达到 CMMI 四级成熟度的重要条件。尽管许多人认为软件开发是纯创造性的技术，甚至认为编程技术是一种艺术，就如同文学创作一样。但是有经验的软件从业人员大多认同软件开发是一种工程化活动，需要许多人参与，需要有客观的规范、标准，进行数学化的度量和基于数据分析的科学管理。优良的软件配置管理工具和相应的过程管理系统就能为软件开发项目提供基于数据分析的科学管理，进而促进开发过程的改进。

4.4.1　度量在软件开发过程中的作用

软件开发过程度量是软件过程管理的重要内容，是进行软件过程控制和软件过程改进的数据基础，是客观描述软件开发过程内外因素的有效手段。度量本身并不能增强或消弱软件过程的自适用能力，但它能揭示出软件开发项目和产品的清晰、明确的状态，为软件开发管理提供决定性的信息。客观、真实的软件开发度量数据是软件项目投资者的决策依

据，是软件项目管理者的重要管理手段，也是项目执行者的驱动力量。拥有客观的度量数据，软件项目管理者就可以制订出更精确的项目计划，并准确地掌握项目的当前状况，从而合理地调配项目资源，做出正确的决定。

但现实情况是，软件开发项目常常因预算超支和范围变更而声名狼藉，有不少软件开发项目始终停留在 90%完成状态，最终不了了之。正如有些专家指出的，导致软件项目失败的第一个可能的原因是缺乏计划，尤其是没能准确判断工作量和完成项目所需的时间；第二个可能的原因是执行不力，有太多的时候推动软件项目的是编程人员，而不是管理小组、目标和需求。要解决好项目的计划和管理上的问题就必须能对项目中的任务、要素进行度量，无法度量则无从管理。

前面例举的陷入泥潭的软件开发项目失败的主要原因就是管理不善。管理不善的根源是没有完善的开发过程和配置管理系统支持，不可能从系统中提取客观的数据，由此不可能如实地反映项目的状况和产品的质量，也不可能揭示开发过程中的不足和开发人员的效率，无从发现问题，更无从解决问题。

4.4.2 如何建立有效的度量体系

软件开发项目中的度量体系通常采用“目标-问题-度量（Goal-Question-Metric，GQM）”方法建立起来。这种方法的 3 个步骤如下。

（1）确定度量要达到的目标。“目标”是度量的动因。这个目标虽然不是软件项目或组织的商业目标，但应该与商业目标保持一致，是部分商业目标的具体化，辅助商业目标的实现。通常的目标是提高软件开发项目的效率和产品质量，或克服软件开发项目中的影响产品商业价值实现的重大问题。

（2）标识现存的主要问题。“问题”是对实现度量目标的途径和难度进行分解。这一步骤需要标识在现有开发实践中要达到既定目标可能遇到的问题和难点。对同一目标可能有很多问题，每个问题都会从特定的角度反映目标，进而直接影响度量数据的采集。

（3）确定度量指标。“度量”是针对问题设计的用于量化软件开发过程中的实体或活动的一系列算法，包括多种度量指标。针对不同的问题有不同的度量指标，即使相同的问题也可以有不同的指标，需要结合项目的具体情况选定可行的度量指标。

尽管不同项目或组织建立的度量体系会选用千差万别的度量指标，还是可以给出一些较为通用的指标供大家参考。

最基本的 6 个软件项目度量指标是功能点、工时、成本、工期、缺陷数和代码行数。功能点指标用于量化软件的功能规模，按国际功能点用户组织（The International Function Point Users Group，IFPUG）提供的标准框架，可提高功能点计算的精确性；工时反映了完

成单个软件活动或任务所需要的小时数，汇总项目中所用活动或任务的工时即得到项目的总工时；成本指标反映软件项目直接和相关的资金支出，包括购买软硬件费用和开发人员报酬及相关花费；工期是指项目从开始到完成所用的天数；缺陷数反映软件的质量，也可用于追踪开发团队解决问题的能力和响应速度；代码行数反映软件项目的复杂程度，但它和编程语言相关，有些可视化编程语言的代码行数需要特殊计算。

这些指标之间可以相互组合演算以衍生出其他具有复合属性的度量指标，例如每个功能点成本、千行代码缺陷数等。此外，基于这 6 个指标，加上时间、组织结构、产品结构、任务类型、缺陷特性等因素就可以形成其他反映软件开发项目效率和质量的不同方面的许许多多的度量指标。

上述“目标－问题－度量”方法主要侧重于项目计划和度量体系设计阶段时考虑的因素，这也是实施软件开发过程度量的主要难度所在。在度量体系建立后，在项目日常开发过程中仍需要进行相关的度量活动，包括后续的度量数据的收集和基于度量数据进行过程分析、改进，最终实现项目或组织目标。

4.4.3　软件配置管理为度量提供客观数据

度量体系和度量指标确立后，如何在软件开发过程中获取这些度量数据呢？如果没有相应管理系统的支持，完全依靠手工调查表方式去收集开发度量数据的难度相当大，而且数据的客观性和准确性也得不到保障。支持过程管理的配置管理系统对软件开发过程度量起至关重要的作用。

正如前面提到的，优良的软件开发过程和配置系统收集了大量的开发过程中的动态数据（In-process Metrics），这些数据是度量指标的主要数据来源。即使是基本的软件配置管理系统，其版本控制、变更控制和基线管理功能也能提供不少与工期、缺陷数和代码行相关的度量数据，例如已实施的变更数、各基线变更数、各版本代码改动行数、每周新生缺陷数等。因为所有与变更（包括新功能和缺陷）、代码、版本和基线相关的数据都在软件开发过程中自动存储到配置库中，只需要按一定的条件和格式定期从配置库中导出相关数据并加以计算即可，不需要人工收集各个样本数据。

具有过程管理的软件配置管理系统在此基础上提供更全面的任务、产品、人员等方面的管理功能，因此也能自动收集更全面的数据用于过程度量。例如，任务及任务相关数据可用于功能点、工时、成本、缺陷等相关度量数据的计算，产品及其相关数据可用于代码行、工期等相关度量数据的计算，人员及其相关数据可用于成本、工期、效率等相关度量数据的计算。由前面的内容可了解到，具有过程管理功能的配置管理系统通常记录了任务、产品和人员方面的几十种数据项、上百个数据属性。它们的组合演算能形成反映软件开发过程的各个方面的度量数据。由于这些原始数据是在日常开发过程中由系统点点滴滴自动

地收集而成的，因此具有较强的客观性和准确性。拥有这些数据，软件开发项目很容易地就能实现定期自动生成所需度量数据，从而得到开发过程各个环节的量化信息。

4.4.4 基于度量的过程改进

有效地改进软件开发过程是软件企业以过程为中心进行软件工业化生产的必经之路。这也是 CMMI 和其他软件过程评价模型中一贯强调的需要持续进行过程改进的原因。

从广义上讲，一切软件过程的改进都是以度量为基础的。软件项目建立了度量体系，并从相应管理系统中生成了度量数据后，就可客观地反映项目当前所处的状态，以及与度量目标的差距。这种差距就为软件过程改进提供了准确的切入点，进而可用于分析过程缺陷，设计改进措施。

具体来说，基于度量的软件过程改进首先要建立软件过程的量化基线，即在项目的某个里程碑或特定时间段提取度量体系中确定的量化数据作为过程改进的基准。过程基线是实施过程改进的起点，也是评价过程改进的标准。

事实上，过程度量信息大大细化了过程中实体的可表示粒度，使软件过程基线不再仅仅是达到某个成熟度层次或某个关键过程域，而是在过程基线中给出的在产品质量、进度、成本等因素上的数据表现。基于度量的软件过程改进使软件开发过程不断地实现量变到质变的升华。此外，也可通过比较过程改进的进度数据和成本数据来评价过程改进的性能，找出最佳改进途径。

经过量化基线、数据比较、分析后，要改进软件开发过程最终必须实施过程变更。过程变更通常体现在对现有的过程、技术、方法、工具进行调整或采用新的方法或工具。无论何种变化，大多需要开发过程和配置管理工具的支持，甚至完全在过程管理或配置管理工具中实施相应的变化，或者加强过程的自动化，或者强化对人工环节的检查控制，如在系统中强制代码评审、强化提交代码时进行缺陷原因分析等机制。

总之，支持开发过程的配置管理系统可通过为软件开发项目提供量化的客观数据来促进软件开发团队改进过程，并通过系统化的机制帮助实现过程的改进。基于度量的过程改进使软件开发过程不断实现由量变到质变的飞跃，使开发团队实现由无序的艺术性开发转向有条不紊的工程化开发的转型。

4.5 ISO 和 CMMI 中的软件配置管理

通过前面几节的讨论，了解到了软件配置管理和软件开发过程紧密相关，并能为开发过程提供有力的支持。事实上，在各种与软件开发过程相关的策略、规范和标准中都会对

软件配置管理提出一定的要求。下面来看看作为规范软件开发过程的最重要的标准文献，ISO 9000 质量体系和 CMU-SEI 的 CMMI 中对软件配置管理的要求。

4.5.1　ISO 9000 中对配置管理的要求

ISO 9001 是ISO 9000 质量管理体系中具体规定一个组织在提供产品或服务的过程中需要达到的质量要求，包括产品设计、实现、采购、检测等全方面的质量控制要求。ISO 9001 在 1987 发布最初版后，还发布了修订的 1994 版、2000 版和最新的 2008 版。此外，由于 ISO 9001 是一个通用标识，并不特别针对软件行业，为了适应软件行业的应用，ISO 还针对 ISO 9001:1994 版发布了 ISO 9000-3，专用于软件开发行业实施 ISO 质量管理的指导。

1．ISO 9001:1994 中隐含的配置管理要求

在 ISO 9001:1994 版本中没有定义配置管理的概念，但可以在各个条款中找到隐含的配置管理要求，如下。

- 4.4 节中“应该标识、记录、维护、评审和批准设计和开发变更”条款隐含了变更控制的要求。
- 4.8 节中“如果需要的话，应该对产品进行标识，以使其具有可跟踪性” 条款隐含了配置项标识和版本控制的要求。
- 4.12 节中“应该维护产品的评审和测试状态记录”和 4.13 节中“应该控制不一致产品”条款隐含了配置项审计方面的要求。

尽管 ISO 9000:2000 发布之后 ISO 9000-3 就失效了，由于 ISO 9000-3 是专门针对软件行业的要求，其对配置管理进行了明确的表述，对软件开发实践有重要的参考作用。ISO 9000-3 对配置管理的表述为：配置管理提供了标识、控制和跟踪每个软件配置项版本的一种机制。在大多数情况下，必须维护和控制仍然在使用中的早期版本。配置管理系统应该做到如下几点。

- 唯一地标识每个软件配置项的版本。
- 标识共同构成一个完整产品的特定版本的各个软件配置项版本。
- 标识开发中的，或者已交付和安装的软件产品的联编状态。
- 控制由多人同时更新的软件配置项。
- 按需协调多个相关产品在一处或者多处的更新。
- 自始至终标识和跟踪变更请求和变更结果。

2．ISO 9001:2000 中隐含的配置管理要求

在后续的 ISO 9001:2000 中同样没有明确定义配置管理，但也隐含了大量的配置管理要求，如下。

- 在 4.2.3 和 4.2.4 节中要求把相关文档和质量记录纳入配置管理中。
- 在 7.3.7 节中要求标识、记录、评审并批准设计和开发变更，并要维护评审等活动的记录（延续 1994 版中的要求）。
- 在 7.5.3 节中要求必要时标识产品，使其具有可跟踪性，并能进行状态报告。

3．ISO 9001:2008 中隐含的配置管理要求

在最新发布的 ISO 9001:2008 版本中也是以隐含的方式表述了配置管理的要求。相关的条款如下。

- 在“7.3.6 设计和开发确认”一节中，要求“应依据所策划的安排对设计和开发进行确认。只要可行，确认应在产品交付或实施之前完成。确认结果及任何必要的措施的记录应予保持”。这隐含了建立产品需求和设计基线的要求。
- 在“7.3.7 设计和开发更改的控制”一节中，要求“应识别设计和开发的更改，并保持记录。在适当时，应对设计和开发的更改进行评审、验证和确认，并在实施前得到批准。更改评审结果及任何必要的措施的记录应予保持”。这隐含了进行变更控制的要求。
- 在“7.5.3 标识和可追溯性”一节中，要求“适当时，组织应在产品实现的全过程中使用适宜的方法识别产品。在有可追溯性的要求场合，组织应控制产品的唯一标识，并保持记录”。这一条款隐含了对产品和产品配置项进行标识，并保证产品配置的可追溯性的要求。
- 在“8.4 数据分析”一节中，要求“组织应确定、收集和分析适当的数据，以证实质量管理体系的适应性和有效性，并评价质量管理体系持续改进的有效性。这应包括监视和测量的结果，以及其他有关来源的数据”。这隐含了对开发过程、配置管理数据的分析应用和基于数据的量化管理的要求。

可见，ISO 9000 质量体系的一系列版本中涵盖了软件配置管理中的配置项标识、变更控制、版本控制、基线管理、变更和配置项的可追踪性、配置项审计、状态报告、基于配置管理的度量分析等主要的软件配置管理需求。

4.5.2 CMMI 中对配置管理的要求

CMMI（Capability Maturity Model Integration，能力成熟度模型集成）是美国卡内基梅

隆大学软件工程研究所（CMU-SEI）开发的一个用于评价软件开发组织过程成熟度的行业标准。由于 CMMI 是专门针对计算机行业制订的标识，因此它对软件开发过程中的各个方面都有相当详细、精确的规范性描述。

在 CMMI 中对软件组织的成熟度定义了 5 个等级，每个软件组织都处于 5 个等级中的一个。每个成熟度等级是由一组过程域组成的，每个过程域包含特定目标（Specific Goal）和通用目标（Generic Goal）。每个特定目标均含有一个或几个特定实践（Specific Practice ）。每个通用目标均包含若干个通用实践（Generic Practice），而每个通用实践又可以按其特性划分为执行约定、执行能力、指导实施和验证实施等。下面来看看作为 CMMI 第二级中的重要过程域的“配置管理”过程域的要求。

1．配置管理的目的

通过配置标识、配置控制、配置状态统计和配置审计来建立和维护工作产品的完整性。这一总体目标又分为特定目标和通用目标。特定目标包括：为指定的工作产品建立基线；跟踪和控制处于配置管理下的工作产品的变更；建立和维护基线的完整性。通用目标是：使配置管理过程作为一种被管理的制度化过程，以实现配置管理的过程的特定目标。通用目标具体包括过程的制订、资源配置、责任划分、人员培训、过程审查和监控、管理层评审等普遍性要求。此外，对 CMMI 三至五级的企业，还要求达到更多的目标：使配置管理过程成为组织范围的已定义过程，收集过程改进数据，建立量化管理过程，不断纠正问题以优化过程。

2．实现特定目标的特定实践

实现配置管理的各个特定目标有相应的不同特定实践，如表 4-1 所示。

表 4-1　配置管理的特定目标和实践对应表

特定目标	特定实践
SG1　建立基线	SP1.1　标识配置项 SP1.2　建立配置管理系统 SP1.3　建立或发布基线
SG2　跟踪和控制变更	SP2.1　跟踪变更请求 SP2.2　控制配置项
SG3　建立完整性	SP3.1　建立配置管理记录 SP3.2　进行配置审计

1）建立基线

为了达到“建立基线”的配置管理特定目标，需要执行的特定实践有以下几点。

- 1.1 标识配置项，其包括如下实践。

- 根据文档标准选择配置项和工作产品。
- 给配置项分配唯一的标识符。
- 说明每个配置项的重要特征。
- 指明处于配置管理下的配置项。
- 给每个配置项指明所有者。

- 1.2 建立配置管理系统，其包括如下子实践。
  - 建立一个配置管理的多重控制级别的机制。
  - 在配置管理系统中能存储和重新取得配置项。
  - 在配置管理系统的不同控制级别中共享和传递配置项。
  - 存储和恢复配置项的存档过的版本。
  - 存储、更新和检索配置管理记录。
  - 从配置管理系统中创建配置管理报告。
  - 保存配置管理系统的内容。
  - 必要时修正配置管理结构。
- 1.3 建立或发布基线，其包括如下子实践。
  - 在创建或发布配置项的基线前获得配置控制委员会（CCB）的认可。
  - 只从配置管理系统的配置项中创建或发布基线。
  - 为包含在基线中的配置项集提供文件。
  - 使当前基线集可以容易地被使用和访问。

2）跟踪和控制变更

为了达到“跟踪和控制变更”的特定目标，需要执行的特定实践有。

- 2.1 跟踪变更请求，其包括如下子实践。
  - 在变更请求数据库中接受并记录变更请求。
  - 分析所建议的变更或修正可能造成的影响。
  - 与相关人员一起评审，并获得同意后再实施变更。
  - 跟踪变更请求的状态，直到结束。

- 2.2 控制配置项，其包括如下子实践。
  - 在产品的整个生命周期中控制配置项的变更。
  - 在把变更过的配置项纳入配置管理系统前获得批准。
  - 从配置管理系统中检出/检入配置项时，要保证其正确性和完整性。
  - 对变更进行审查，以确保变更不会对基线造成非预期的影响。
  - 记录配置项的变更，以及变更原因。

3）建立配置完整性

为了达到“建立完整性”的特定目标，需要执行的特定实践有。

- 3.1 建立配置管理记录，其包括如下子实践。
  - 记录配置管理活动，以便掌握每个配置项的内容和状态，并且能够恢复以前的版本。
  - 确保相关人员能够访问和了解配置项的配置状态。
  - 指明基线的最新版本。
  - 确定那些构成特定基线的配置项的版本。
  - 描述前后基线之间的差别。
  - 必要时修正每个配置项的状态和历史。
- 3.2 进行配置审计，其包括如下子实践。
  - 审核基线的完整性。
  - 确保配置管理记录正确标识了配置项的配置情况。
  - 审查配置管理系统中配置项的结构完整性。
  - 确保配置管理系统中配置项的完备性和正确性。
  - 确保遵循配置管理标准和程序。
  - 跟踪待改进事项，直至其完成。

3．通用目标

配置管理的通用目标包括实现配置管理的特定目标（GG1）和使配置管理过程成制度化（GG2）。实现前一个通用目标只是强调特定目标的实践，后一个目标需要执行如下实践才能达到。

- 建立组织内的执行配置管理过程的策略。
- 制订并维护执行配置管理过程的计划。
- 提供足够的资源去执行配置管理过程。
- 为执行配置管理及相关过程进行分工授权。
- 培训配置管理过程的执行人员。
- 将配置管理过程进行一定级别的控制。
- 在配置管理过程中确定并引入相关人员。
- 对配置管理过程进行监控。
- 客观评估配置管理过程执行的一致性。
- 请上级管理者评审配置管理状态。

除了在第二级的配置管理过程域中明确描述了成熟度模型中对配置管理的要求外，CMMI 中其他许多过程域中也隐含了配置管理和相关的要求，例如以下内容。

- 在二级过程域“需求管理”中要求的“对需求进行基线化”、“需求变更管理”和“维护需求与工作成果之间的双向可追溯性”等实践与配置管理直接相关。
- 在二级过程域“项目规划”中要求“估算工作量和成本”、“规划项目资源”、“协调工作与资源的配置”等实践与软件开发过程的任务、人员、产品等管理要素及配置管理过程相关。
- 在二级过程域“度量分析”中要求的“确定数据收集和存储规程”、“收集度量数据”、“分析度量数据”等实践也与配置管理紧密相关。
- 在三级过程域“产品集成”中要求的“建立产品集成规程和集成准则”、“管理产品和构建的内部外部接口的定义”、“组装产品构建”、“确认产品集成已经就绪”、“打包和交付产品或构建”等实践都依托于配置管理过程。
- 在四级过程域“组织过程绩效”中的“建立过程性能度量定义”、“建立质量和过程性能目标”和“建立过程性能基线”等实践也需要以配置管理和开发过程管理数据为依据。

可见，软件配置管理是 ISO 9001 和 CMMI 中的重要内容，也是完善软件开发过程管理和保证软件产品质量的核心环节。

4.6　定制配置管理系统以支持软件开发过程

传统的软件配置管理系统大多侧重于产品代码的管理，缺乏对软件开发过程的完整支持。纯粹依靠传统的软件配置管理系统会使软件组织的各个部门之间或角色之间没有连接关系，组织所拥有的只是相互独立的信息资源，从而形成了信息“孤岛”[3]。理想的软件配置管理系统将提供过程支持，使组织之间能通过系统的过程驱动力建立紧密的联系，便于数据共享和开发人员协作。

为了完善配置管理系统以加强对开发过程的支持，需要在传统配置管理系统的基础上，完善对人员和任务的管理，强化变更过程，并提供对团队开发、过程模式、构造发布和项目管理等方面的支持。下面来分别看看软件开发项目在这些方面的实际需求。

4.6.1　角色与职责

常见的配置管理系统中通常只区分普通用户和系统管理员两种角色。为了加强过程的支持，至少必须在系统中定制出下列与过程相关的角色和职责。

- 项目经理：负责项目的创建、协调和进度管理，包括任务分配、风险分析和生成向上级主管汇报的报表。
- 变更控制委员会（CCB），负责分析评判需求和变更，使开发任务与项目的商务需求保持一致。
- 集成人员：负责将产品源码按一定的方式集中起来，并构建出产品的可安装执行代码，向整个项目组发布构建结果和相应构建中的完成任务列表。
- 开发人员：负责接收分配的开发任务，按确定的优先级别进行系统或模块设计和编码。
- 开发小组长：负责协调管理由多个开发人员组成的开发小组，调节小组成员的分工，关注小组开发进度，分析和协调小组成员提出的变更请求，并在必要时提出项目组级变更请求。
- 测试人员：负责对发布的构建进行有计划的测试，根据测试结果关闭缺陷报告或提交新的缺陷报告。
- 测试小组长：负责协调管理由多个测试人员组成的测试小组，设计测试方案和测试策略，选择测试工具，对小组内的测试结果和测试报告进行评审。

4.6.2 任务和变更管理

在任务和变更管理方面，必须能够定义一个完整的任务和变更的生命周期，包括任务或变更的请求、审批、实施、验证、审核、结束；同时还应定义任务及变更请求的状态集合、变更请求各状态间所能进行的转移操作的集合，以及各个角色在变更的不同状态下的操作权限。最后还需要自动收集变更执行后生成的版本列表（变更集）。具体包括以下内容。

- 方便地提交、修改和查看变更信息。
- 变更实现软件版本演化的根源与过程，即反映原始需求、需求变化、增加功能、修改错误等各种开发任务。
- 定义任务和变更的状态转换流程，以规定任务和变更从创建到关闭的生命周期，同时也与开发项目的生命周期相对应。
- 实现对变更实施的必要决策与控制（如评审、分配、批准等环节）并确定各个控制点的负责人。
- 通过变更集记录和控制对软件配置项的每一次修改，并将这种修改内容和构建过程衔接起来，实现变更任务、配置项版本和构造结果的三向可追踪性。
- 提供建立任务间关系的支持，如父子关系、参照关系、模块关系，使任务关系完整系统地反映项目的状况，同时能从任务关系数据中提炼出反映开发过程质量和产品质量的度量数据（Metrics）。
- 实现变化传播机制，使产品一个版本中的修改能方便地传播到其他版本中。

4.6.3 过程支持

软件开发过程的贯彻实施需要相应系统的支持才能实现合理化。正如 Grady Booch 在《统一软件开发过程》中所说的，“开发过程如果不考虑怎样将它自动化，则开发过程只会具有理论意义；软件开发工具如果不知道它们用来支持什么样的过程，则开发工具可能是做无用功”[6]。软件开发过程需要相应工具来支持和强化。常用配置管理工具对软件开发过程只起到零星松散的支持作用。新的工具必须从整体上系统地支持软件开发过程，并尽可能将过程自动化。

- 能够支持软件生命周期的全过程，包括构想、原型、设计、实现、发布和维护等各个环节。
- 能够不定制或尽量少定制以支持不同的过程模式，例如瀑布模式、迭代模式、敏捷模式等，以适应不同客户的需要。

- 能够方便地支持过程中的里程碑或迭代周期的定义和维护，例如将它们与待实施的变更关联起来。
- 既有预定义过程，也支持可选过程，并保证过程中定义的每一步均由授权的人员按正确的顺序执行。
- 支持过程中的关键概念，包括角色、工作组、任务、任务分组、状态、权限控制机制和触发器机制等。

4.6.4　团队开发

如前所述，软件开发过程通常涉及许多的角色，而每个角色都可能由多个人员担当。如何使各个角色和人员相互协调配合，以一个有机的团体而不是多个孤立的个体进行软件开发，这就有求于支持过程的配置管理系统的团队支持能力。完善的系统应该从如下方面支持团队开发实践。

- 反映开发团队的组织结构并支持任务分配机制，以促进开发人员之间的分工与合作。
- 支持并行开发，方便合理地实现多个开发人员同时开发一个项目，或项目中的一个功能；
- 工作区管理，使不同的开发人员拥有独立的、互不影响的工作空间。
- 分布式开发，使开发团队能够分布在地理上相距较远的地点，甚至越洋团队。
- 不同开发人员之间的信息共享，使整个团队能在一个统一集成的平台上获取设计、变更、缺陷、代码、基线、构造、发布等信息。
- 对有衔接关系的过程实现自动邮件通知，在任务的每个操作之后自动通知后一个操作的执行者和相关人员。
- 确保可追踪的依赖性，可通过每个变更或每个版本查找与之对应的构造，或发布的版本，或设计文档的版本。
- 便于进行变更影响分析，从一个部分的修改可查找出相关的修改，并能用于分析可能影响产品的那些其他部分。

4.6.5　构造和发布

构造和发布过程是软件开发项目进行产品交付的重要环节。没有工具支持的纯手工的代码集成、构造、发布过程很难保持交付产品的正确性、完整性和一致性。支持完整开发过程的配置管理系统应该具有如下对产品构造和发布的支持。

- 实现代码集成的自动化，使集成过程中代码版本、基线、变更和构造版本能自动地建立联系，并将这种联系记录在数据库中。
- 通过对构造脚本和构造环境的版本控制，增进构造过程的自动化和可重现化。
- 在构造过程中自动生成构造变更列表，并随构造结果一同发布，便于开发、测试和管理人员协同工作，提高他们的工作效率。
- 为复杂产品的构造提供独立的构造稳定（Stabilizations）分支，以便在修正构造问题的同时允许开发人员提交新的代码。
- 便于实现持续集成（Continuous Integration），及时生成可测试版本。
- 方便实现对历史版本的重新构造，以重现问题或在此基础上生成补丁。
- 在项目发布或达到里程碑时，便于生成产品发布说明（Release Notes）。
- 能为不同的用户提供不同的版本，避免版本混乱。

4.6.6 项目管理

软件开发项目比其他行业的项目有更高的复杂性，项目管理的难度也更高。好的项目管理离不开好的工具，而支持开发过程的完善的配置管理系统就是对软件项目进行管理的重要工具。这种工具可以从如下方面为项目经理或其他管理者提供软件项目管理支持。

- 用于开发任务的评估、量化、分解、排序、分配和状态报告。
- 用于对开发人员的工作负荷和任务完成进度进行客观的度量和评价。
- 用于对项目的完成情况和对产品的质量演化趋势进行整体把握，进而预计出产品的实际交付时间，或做出必要的资源调整。
- 通过对任务和构造结果的汇总，了解当前发布版本中真正实现了哪些功能，修正了那些缺陷，有哪些组成部件，便于与客户或上级管理者沟通。
- 方便地收集前一个发布版本遗留的功能或缺陷，定义需求基线，用于制订下一个版本发布或迭代计划，以及项目总体计划。

总之，一个功能完善的软件配置和开发过程管理系统能有效地支持软件开发项目，使之在人员、任务、过程、产品等方面实现减化流程、优化管理、提高效率等功效。