第14章信息文档管理与配置管理

14.1信息系统项目文档及其管理

14.1.1信息系统项目相关信息（文档□

1. 信息系统项目相关信息（文档□含义

信息系统相关信息（文档□是指某种数据媒体和其中所记录的数据。它具有永久性，并可以由人或机器阅读，通常仅用于描述人工可读的东西。在软件工程中，文档常常用来表示对活动、需求、过程或结果，进行描述、定义、规定、报告或认证的任何书面或图示的信息（包括纸质文档和电子文档□。

2. 信息系统项目相关信息（文档□种类

软件文档一般分为三类：开发文档、产品文档、管理文档。

(1) 开发文档描述开发过程本身，基本的开发文档包括：

·可行性研究报告和项目任务书。

·需求规格说明。

·功能规格说明。

·设计规格说明，包括程序和数据规格说明。

·开发计划。

·软件集成和测试计划。

·质量保证计划。

·安全和测试信息。

(2) 产品文档描述开发过程的产物，基本的产品文档包括：

·培训手册。

·参考手册和用户指南。

·软件支持手册。

·产品手册和信息广告。

G)管理文档记录项目管理的信息，例如：

·开发过程的每个阶段的进度和进度变更的记录。

·软件变更情况的记录。

·开发团队的职责定义。

·项目计划、项目阶段报告。

·配置管理计划。

文档的质量可以分为四级：

(1) 最低限度文档（1级文档□，适合开发工作量低于一个人月的开发者自用程序。该文档应包含程序清单、开发记录、测试数据和程序简介。

(2) 内部文档（2级文档□，可用于没有与其他用户共享资源的专用程序。除1级文档提供的信息外，2级文档还包括程序清单内足够的注释以帮助用户安装和使用程序。

(3) 工作文档（3级文档□，适合于由同一单位内若干人联合开发的程序，或可被其他单位使用的程序。

(4) 正式文档（4级文档)，适合那些要正式发行供普遍使用的软件产品。关键性程序或具有重复管理应用性质（如工资计算□的程序需要4级文档。4级文档遵守GB/T 8567-2006的有关规定。

14.1.2信息系统项目文档管理的规则和方法

管理信息系统文档的规范化管理主要体现在文档书写规范、图表编号规则、文档目录编写标准和文档管理制度等几个方面。

(1) 文档书写规范。管理信息系统的文档资料涉及文本、图形和表格等多种类型，无论是哪种类型的文档都应该遵循统一的书写规范，包括符号的使用、图标的含义、程序中注释行的使用、注明文档书写人及书写日期等。例如，在程序的开始要用统一的格式包含程序名称、程序功能、调用和被调用的程序、程序设计人等。

(2) 图表编号规则。在管理信息系统的开发过程中用到很多的图表，对这些图表进行有规则的编号，可以方便图表的查找。图表的编号一般采用分类结构。根据生命周期法的5个阶段，可以给出如图14-1所示的分类编号规则。根据该规则，就可以通过图表编号判断该图表出于系统开发周期的哪一个阶段，属于哪一个文档，文档中的哪一部分内容及第几张图表。

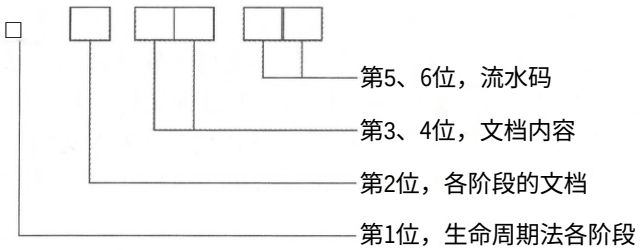

图14-1图表编号规则

(3)文档目录编写标准。为了存档及未来使用的方便，应该编写文档目录。管理信息系统的文档目录中应包含文档编号、文档名称、格式或载体、份数、每份页数或件数、存储地点、存档时间、保管人等。文档编号一般为分类结构，可以采用同图表编号类似

的编号规则。文档名称要书写完整规范。格式或载体指的是原始单据或报表、磁盘文件、磁盘文件打印件、大型图表、重要文件原件、光盘存档等。

(4)文档管理制度。为了更好地进行信息系统文档的管理，应该建立相应的文档管理制度。文档的管理制度需根据组织实体的具体情况而定，主要包括建立文档的相关规范、文档借阅记录的登记制度、文档使用权限控制规则等。建立文档的相关规范是指文档书写规范、图表编号规则和文档目录编写标准等。文档的借阅应该进行详细的记录，并且需要考虑借阅人是否有使用权限。在文档中存在商业秘密或技术秘密的情况下，还应注意保密。特别要注意的是，项目干系人签字确认后的文档要与相关联的电子文档一一对应，这些电子文档还应设置为只读。

14.2配置管理

配置管理是为了系统地控制配置变更，在系统的整个生命周期中维持配置的完整性和可跟踪性，而标识系统在不同时间点上配置的学科。在GB/T 11457—2006中，将“配置管理”正式定义为：“应用技术的和管理的指导和监控方法以标识和说明配置项的功能和物理特征，控制这些特征的变更，记录和报告变更处理和实现状态并验证与规定的需求的遵循性。”

尽管硬件配置管理和软件配置管理的实现有所不同，配置管理的概念可以应用于各种信息系统集成项目。

配置管理包括6个主要活动：制订配置管理计划、配置标识、配置控制、配置状态报告、配置审计、发布管理和交付。

14.2.1配置管理的概念

1.配置项

GB/T 11457—2006对配置项的定义为：“为配置管理设计的硬件、软件或二者的集合，在配置管理过程中作为一个单个实体来对待。”

以下内容都可以作为配置项进行管理：外部交付的软件产品和数据、指定的内部软件工作产品和数据、指定的用于创建或支持软件产品的支持工具、供方/供应商提供的软件和客户提供的设备/软件。典型配置项包括项目计划书、需求文档、设计文档、源代码、可执行代码、测试用例、运行软件所需的各种数据，它们经评审和检查通过后进入配置管理。

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。

在信息系统的开发流程中需加以控制的配置项可以分为基线配置项和非基线配置项两类，例如，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。

所有配置项的操作权限应由CMO（配置管理员□严格管理，基本原则是：基线配置项向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.配置项状态

配置项的状态可分为“草稿”“正式”和“修改”三种。配置项刚建立时，其状态为“草稿”。配置项通过评审后，其状态变为“正式”。此后若更改配置项，则其状态变为“修改”。当配置项修改完毕并重新通过评审时，其状态又变为“正式”。

配置项状态变化如图14-2所示。

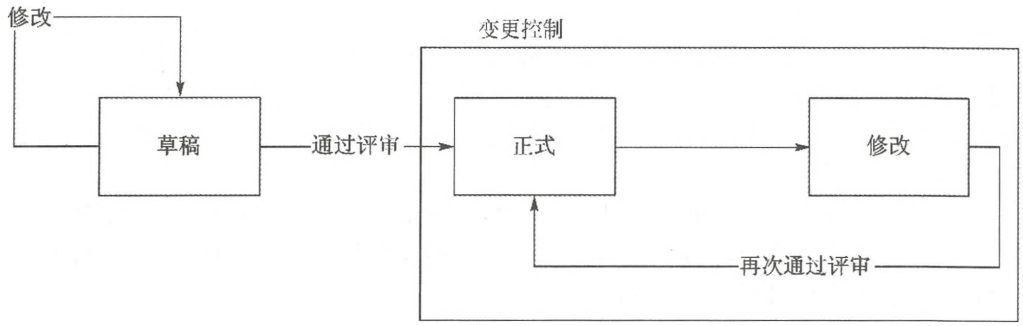

图14-2配置项状态变化

3. 配置项版本号

配置项的版本号规则与配置项的状态相关。

(1) 处于“草稿”状态的配置项的版本号格式为O.YZ，YZ的数字范围为01~99。随着草稿的修正，YZ的取值应递增。YZ的初值和增幅由用户自己把握。

(2) 处于“正式”状态的配置项的版本号格式为X.Y，X为主版本号，取值范围为1~9。Y为次版本号，取值范围为0~9。

配置项第一次成为“正式”文件时，版本号为1.0。

如果配置项升级幅度比较小，可以将变动部分制作成配置项的附件，附件版本依次为1.0，1.1，…。当附件的变动积累到一定程度时，配置项的Y值可适量增加，Y值增加一定程度时，X值将适量增加。当配置项升级幅度比较大时，才允许直接增大X值。

(3) 处于“修改”状态的配置项的版本号格式为X.YZ。配置项正在修改时，一般只增大Z值，X.Y值保持不变。当配置项修改完毕，状态成为“正式”时，将Z值设置为0,增加X.Y值。参见上述规则（2)。

4. 配置项版本管理

配置项的版本管理作用于多个配置管理活动之中，如配置标识、配置控制和配置审计、发布和交付等。在项目开发过程中，绝大部分的配置项都要经过多次的修改才能最终确定下来。对配置项的任何修改都将产生新的版本。由于我们不能保证新版本一定比

旧版本“好”，所以不能抛弃旧版本。版本管理的目的是按照一定的规则保存配置项的所有版本，避免发生版本丢失或混淆等现象，并且可以快速准确地查找到配置项的任何版本。

5. 配置基线

信息系统的开发过程是一个不断变化着的过程，为了在不严重阻碍合理变化的情况下来控制变化，配置管理引入了“配置基线（ConfigurationBaseline）”这一概念。

配置基线（常简称为基线□由一组配置项组成，这些配置项构成一个相对稳定的逻辑实体。基线中的配置项被“冻结”了，不能再被任何人随意修改。对基线的变更必须遵循正式的变更控制程序。

一组拥有唯一标识号的需求、设计、源代码文卷以及相应的可执行代码、构造文卷和用户文档构成一条基线。产品的一个测试版本（可能包括需求分析说明书、概要设计说明书、详细设计说明书、已编译的可执行代码、测试大纲、测试用例、使用手册等□是基线的一个例子。

基线通常对应于开发过程中的里程碑（Milestone），一个产品可以有多个基线，也可以只有一个基线。交付给外部顾客的基线一般称为发行基线（Release），内部开发使用的基线一般称为构造基线（Build）。

对于每一个基线，要定义下列内容：建立基线的事件、受控的配置项、建立和变更基线的程序、批准变更基线所需的权限。在项目实施过程中，每个基线都要纳入配置控制，对这些基线的更新只能采用正式的变更控制程序。

建立基线还可以有如下好处。

·基线为开发工作提供了一个定点和快照。

·新项目可以在基线提供的定点上建立。新项目作为一个单独分支，将与随后对原始项目（在主要分支上□所进行的变更进行隔离。

·当认为更新不稳定或不可信时，基线为团队提供一种取消变更的方法。

·可以利用基线重新建立基于某个特定发布版本的配置，以重现已报告的错误。

6. 配置库

配置库 Configuration Library)存放配置项并记录与配置项相关的所有信息，是配置管理的有力工具，利用库中的信息可回答许多配置管理的问题，例如：

·哪些客户已提取了某个特定的系统版本？

·运行一个给定的系统版本需要什么硬件和系统软件？

·一个系统到目前已生成了多少个版本，何时生成的？

·如果某一特定的构件变更了，会影响到系统的哪些版本？

·一个特定的版本曾提出过哪几个变更请求？

·一个特定的版本有多少已报告的错误？

使用配置库可以帮助配置管理员把信息系统开发过程的各种工作产品，包括半成品

或阶段产品和最终产品管理得井井有条，使其不致管乱、管混、管丢。

配置库可以分开发库、受控库、产品库3种类型。

(1) 开发库 Development Library),也称为动态库、程序员库或工作库，用于保存开发人员当前正在开发的配置实体，如：新模块、文档、数据元素或进行修改的已有元素。动态中的配置项被置于版本管理之下。动态库是开发人员的个人工作区，由开发人员自行控制。库中的信息可能有较为频繁的修改，只要开发库的使用者认为有必要，无需对其进行配置控制，因为这通常不会影响到项目的其他部分。

(2) 受控库 Controlled Library),也称为主库，包含当前的基线加上对基线的变更。受控库中的配置项被置于完全的配置管理之下。在信息系统开发的某个阶段工作结束时，将当前的工作产品存入受控库。

(3) 产品库 Product Library)，也称为静态库、发行库、软件仓库，包含已发布使用的各种基线的存档，被置于完全的配置管理之下。在开发的信息系统产品完成系统测试之后，作为最终产品存入产品库内，等待交付用户或现场安装。

配置库的建库模式有两种：按配置项类型建库和按任务建库。

(1) 按配置项的类型分类建库，适用于通用软件的开发组织。在这样的组织内，往往产品的继承性较强，工具比较统一，对并行开发有一定的需求。使用这样的库结构有利于对配置项的统一管理和控制，同时也能提高编译和发布的效率。但由于这样的库结构并不是面向各个开发团队的开发任务的，所以可能会造成开发人员的工作目录结构过于复杂，带来一些不必要的麻烦。

(2) 按开发任务建立相应的配置库，适用于专业软件的开发组织。在这样的组织内，使用的开发工具种类繁多，开发模式以线性发展为主，所以就没有必要把配置项严格地分类存储，人为增加目录的复杂性。对于研发性的软件组织来说，采用这种设置策略比较灵活。

7.配置库权限设置

配置库的权限设置主要是解决：库内存放的配置项什么人可以“看”、什么人可以“取”、什么人可以“改”、什么人可以“销毁”等问题。

配置管理员负责为每个项目成员分配对配置库的操作权限，如表14-1所示。

表14-1配置库的操作权限

权　限	内　容
Read	可以读取文件内容，但不能对文件进行变更
Check	可使用[check in]等命令，对文件内容进行变更
Add	可使用[文件追加]，[文件重命名]，[删除]等命令
Destroy	有权进行文件的不可逆毁坏，清除，rollback等命令

针对受控库，项目相关人员的操作权限通常设定如表14-2所示。

表14-2受控库的权限设置

权限 \ 人员		项目经理	项目成员	QA	测试人员	配置管理员
文档	Read	V	V	V	V	V
	Check	V	V	V	V	V
	Add	V	V	V	V	V
	Destroy	x	x	x	x	V
代码	Read	V	V	V	V	V
	Check	V	V	x	x	V
	Add	V	V	x	x	V
	Destroy	x	x	x	x	V

说明：V表示该人员具有相应权限，X表示该人员没有相应权限

针对产品库，项目相关人员的操作权限通常设定如表14-3所示。

表14-3产品库的权限设置

Release (产品库□						
权限^ \ 人员		项目经理	项目成员	QA	测试人员	配置管理员
Read		V	V	V	V	V
Check		V	V	V	V	V
Add		x	x	x	x	V
Destroy		x	x	x	x	V

说明：V表示该人员具有相应权限，x表示该人员没有相应权限

8.配置控制委员会

配置控制委员会 Configuration Control Board，CCB)，负责对配置变更做出评估、审批以及监督已批准变更的实施。

CCB建立在项目级，其成员可以包括项目经理、用户代表、产品经理、开发工程师、测试工程师、质量控制人员、配置管理员等。CCB不必是常设机构，完全可以根据工作的需要组成，例如按变更内容和变更请求的不同，组成不同的CCB。小的项目CCB可以只有一个人，甚至只是兼职人员。

通常，CCB不只是控制配置变更，而是负有更多的配置管理任务，例如：配置管理计划审批、基线设立审批、产品发布审批等。

9.配置管理员

配置管理员 Configuration Management Officer，CMO)，负责在整个项目生命周期中进行配置管理活动，具体有：

·编写配置管理计划。

·建立和维护配置管理系统。
·建立和维护配置库。
·配置项识别。
·建立和管理基线。
·版本管理和配置控制。
·配置状态报告。
·配置审计。
·发布管理和交付。
·对项目成员进行配置管理培训。

10.配置管理系统

配置管理系统是用来进行配置管理的软件系统，其目的是通过确定配置管理细则和提供规范的配置管理软件，加强信息系统开发过程的质量控制，增强信息系统开发过程的可控性，确保配置项（包括各种文档、数据和程序□的完备、清晰、一致和可追踪性，以及配置项状态的可控制性。

14.2.2配置管理的目标和方针

1. 确定配置管理目标

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。高级项目经理应确保以下配置管理目标得以实现。

·确保软件配置管理计划得以制订，并经过相关人员的评审和确认。
·应该识别出要控制的项目产品有哪些，并且制定相关控制策略，以确保这些项目产品被合适的人员获取。
·应制定控制策略，以确保项目产品在受控制范围内更改。
·应该采取适当的工具和方法，确保相关组别和个人能够及时了解到软件基线的状态和内容。

2. 确定配置管理的方针

为了实现配置管理目标，高级项目经理应确定软件配置管理过程文件得以制订，项目组成员应严格按照配置管理过程文件规定的要求执行，履行配置管理的职责应被明确分配。相关人员得^软件配置管理方面的培训。管理层和具体项目主管应该明确他们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，这需要在客户，管理层和具体项目主管之间协商。软件配置管理应该实施于如下产品：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。

14.2.3日常配置管理活动

1. 制订配置管理计划

配置管理计划是对如何开展项目配置管理工作的规划，是配置管理过程的基础，应该形成文件并在整个项目生命周期内处于受控状态。配置控制委员会负责审批该计划。

配置管理计划的主要内容为：

(1) 配置管理活动，覆盖的主要活动包括配置标识、配置控制、配置状态报告、配置审计、发布管理与交付。

(2) 实施这些活动的规范和流程。

(3) 实施这些活动的进度安排。

(4) 负责实施这些活动的人员或组织，以及他们和其他组织的关系。

2. 配置标识

配置标识 Configuration Identification)也称配置识别，包括为系统选择配置项并在技术文档中记录配置项的功能和物理特征。

配置标识是配置管理员的职能，基本步骤如下。

(1) 识别需要受控的配置项。

(2) 为每个配置项指定唯一性的标识号。

(3) 定义每个配置项的重要特征。

(4) 确定每个配置项的所有者及其责任。

(5) 确定配置项进入配置管理的时间和条件。

(6) 建立和控制基线。

(7) 维护文档和组件的修订与产品版本之间的关系。

3. 配置控制

配置控制即配置项和基线的变更控制，包括下述任务：标识和记录变更申请，分析和评价变更，批准或否决申请，实现、验证和发布已修改的配置项。

1) 变更申请

变更申请主要就是陈述：要做什么变更，为什么要做，以及打算怎么做变更。

相关人员如项目经理填写变更申请表，说明要变更的内容、变更的原因、受变更影响的关联配置项和有关基线、变更实施方案、工作量和变更实施人等，并提交给CCB。

2) 变更评估

CCB负责组织对变更申请进行评估并确定以下内容。

·变更对项目的影响。

·变更的内容是否必要。

·变更的范围是否考虑周全。

·变更的实施方案是否可行。

·变更工作量估计是否合理。

CCB决定是否接受变更，并将决定通知相关人员。

3) 通告评估结果

CCB把关于每个变更申请的批准、否决或推迟的决定通知受此处置意见影响的每个干系人。

如果变更申请得到批准，应该及时把变更批准信息和变更实施方案通知给那些正在使用受影响的配置项和基线的干系人。

如果变更申请被否决，应通知有关干系人放弃该变更申请。

4) 变更实施

项目经理组织修改相关的配置项，并在相应的文档或程序代码中记录变更信息。

5) 变更验证与确认

项目经理指定人员对变更后的配置项进行测试或验证。

项目经理应将变更与验证的结果提交CCB，由其确认变更是否已经按要求完成。

6) 变更的发布

配置管理员将变更后的配置项纳入基线。

配置管理员将变更内容和结果通知相关人员，并做好记录。

7) 基于配置库的变更控制

信息系统在一处出现了变更，经常会连锁引起多处变更，会涉及到参与开发工作的许多人员。例如，测试引发了需求的修改，那么很可能要涉及到需求规格说明、概要设计、详细设计和代码等相关文档，甚至会使测试计划随之变更。

如果是多个开发人员对信息系统的同一部件做修改，情况会更加复杂。例如，在软件测试时发现了两个故障。项目经理最初以为两故障是无关的，就分别指定甲和乙去解决这两个故障。但碰巧，引起这两个故障的错误代码都在同一个软件部件中。甲和乙各自对故障定位后，先后从库中取出该部件，各自做了修改，又先后送回库中。结果，甲放入库中的版本只有乙的修改，乙放入库中的版本只有乙的修改，没有一个版本同时解决了两个故障。

基于配置库的变更控制可以完美地解决上述问题，如图14-3所示。

现以某软件产品升级为例，简述其流程。

(1) 将待升级的基线（假设版本号为V2.1)从产品库中取出，放入受控库。

(2) 程序员将欲修改的代码段从受控库中检出（Checkout)，放入自己的开发库中进行修改。代码被Check out后即被“锁定”，以保证同一段代码只能同时被一个程序员修改，如果甲正对其修改，乙就无法Check out。

(3) 程序员将开发库中修改好的代码段检入（Checkin)受控库。Checkin后，代码

的“锁定”被解除，其他程序员可以Check out该段代码了。

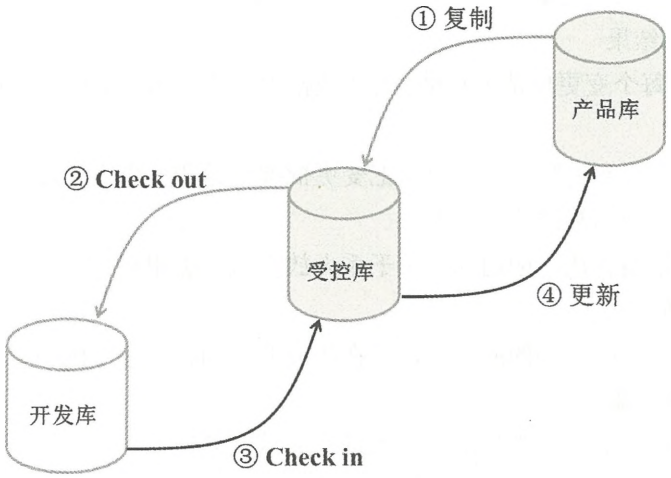

图14-3基于配置库的变更控制

(4)软件产品的升级修改工作全部完成后，将受控库中的新基线存入产品库中（软件产品的版本号更新为V2.2,旧的V2.1版并不删除，继续在产品库中保存)。

4.配置状态报告

配置状态报告 Configuration Status Reporting)也称配置状态统计 Configuration Status Accounting)，其任务是有效地记录和报告管理配置所需要的信息，目的是及时、准确地给出配置项的当前状况，供相关人员了解，以加强配置管理工作。

在信息系统项目开发过程中，配置项在不停地演化着。配置状态报告就是要在某个特定的时刻观察当时的配置状态，也就是要对动态演化着的配置项取个瞬时的“照片”，以利于在状态报告信息分析的基础上，更好地进行控制。

配置状态报告应该包含以下内容。

(1) 每个受控配置项的标识和状态。一旦配置项被置于配置控制下，就应该记录和保存它的每个后继进展的版本和状态。

(2) 每个变更申请的状态和已批准的修改的实施状态。

(3) 每个基线的当前和过去版本的状态以及各版本的比较。

(4) 其他配置管理过程活动的记录。

配置状态报告应着重反映当前基线配置项的状态，以向管理者报告系统开发活动的进展情况。配置状态报告应定期进行，并尽量通过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。

配置状态报告中提供了许多有用的信息，可以用来回答如下问题。

·程序P13的1.6版在哪个备份中可以使用？

·在发行基线5.1和发行基线5.2之间实现了哪些变更请求？

·在发行基线5.2中哪些程序更改过了？

·在变更请求671中要对哪些配置项进行更改？在变更前和变更后，这些程序单元的版本是什么？是否所有的变更都完成并入库了？

配置状态报告还可供项目经理和CCB追踪变更的情况，可以用来回答如下问题。

·某个变更请求是否已被批准？

·某个已批准的变更请求目前处于什么状态？

·某个已完成的变更投入了多少时间和工作量？

·某个配置项与哪几个变更请求有关？

5. 配置审计

配置审计（ConfigurationAudit)也称配置审核或配置评价，包括功能配置审计和物理配置审计，分别用以验证当前配置项的一致性和完整性。

配置审计的实施是为了确保项目配置管理的有效性，体现了配置管理的最根本要求——不允许出现任何混乱现象，例如：

·防止向用户提交不适合的产品，如交付了用户手册的不正确版本。

·发现不完善的实现，如开发出不符合初始规格说明或未按变更请求实施变更。

·找出各配置项间不匹配或不相容的现象。

·确认配置项已在所要求的质量控制审核之后纳入基线并入库保存。

·确认记录和文档保持着可追溯性。

1) 功能配置审计

·功能配置审计 Functional Configuration Audit)是审计配置项的一致性（配置项的实际功效是否与其需求一致)，具体验证以下几个方面。

·配置项的开发已圆满完成。

·配置项已达到.配置标识中规定的性能和功能特征。

·配置项的操作和支持文档已完成并且是符合要求的。

2) 物理配置审计

物理配置审计 Physical Configuration Audit)是审计配置项的完整性（配置项的物理存在是否与预期一致)，具体验证如下几个方面。

·要交付的配置项是否存在。

·配置项中是否包含了所有必需的项目。

6. 发布管理和交付

发布管理和交付活动的主要任务是：有效控制软件产品和文档的发行和交付，在软

件产品的生存期内妥善保存代码和文档的母拷贝。

(1) 存储。应通过下述方式确保存储的配置项的完整性：

·选择存储介质使再生差错或损坏降至最低限度。

·根据媒体的存储期，以一定频次运行或刷新已存档的配置项。

·将副本存储在不同的受控场所，以减少丢失的风险。

(2) 复制。复制是用拷贝方式制造软件的阶段。

·应建立规程以确保复制的一致性和完整性。

·应确保发布用的介质不含无关项（如软件病毒或不适合演示的测试数据□。

·应使用适合的介质以确保软件产品符合复制要求，确保其在整个交付期中内容的完整性。

(3) 打包。应确保按批准的规程制备交付的介质。应在需方容易辨认的地方清楚标出发布标识。

(4) 交付。供方应按合同中的规定交付产品或服务。

(5) 重建。应能重建软件环境，以确保发布的配置项在所保留的先前版本要求的未来一段时间里是可重新配置的。

14.3文档管理、配置管理工具

14.2.1工具综述

项目文档一般作为配置管理的一部分，放在配置管理工具中进行管理，所以此处仅列出几种常见的配置管理工具。

常用的软件配置管理工具分为两大类，一类是付费商业软件，一类是开源软件。

(1) 常用付费软件配置管理工具有：

- Rational ClearCase
- Perforce
- CACCC
- Havest Merant PVCS
- Microsoft VSS, CVS

(2) 常用的开源免费的软件配置管理工具有：

- SVN。
- GITo
- CVS。

目前在国内IT领域使用最广泛的，当属开源免费软件SVN，在一些大型的企业也有使用Rational ClearCase的案例，在一些互联网行业也有组织使用开源免费软件GIT，所以下面就仅对这三个工具进行介绍。

14.3.2 SVN

SVN是Subversion的简称，是一个开放源代码的版本控制系统，此工具是在CVS的基础上，由CollabNet提供开发的，SVN是集中式版本控制之王，也是目前在国内软件企业中使用最为普遍的配置管理工具。

SVN服务器有两种运行方式：独立服务器和借助apache运行。两种方式各有利弊，用户可以自行选择。

SVN的优点：

(1) 支持重命名，这对Java开发来说非常重要。为了得到更好的代码，开发中需要经常进行重构，重构就经常涉及到文件的重构名，而重命名CVS中是不被支持的。

(2) 开发的时候不一定要锁定。一方面导致重构不方便，另一方面，不能离线开发，使用SVN就不同，可以带回家继续开发，回来后，提交就行了。

(3) 多平台。可以支持多个平台下的操作。

(4) 更好的客户端支持。一^"h在Windows下用的SVN客户端TortoiseSVN比较方便使用。

(5) 更好地与外围工具集成。各种各样的外围工具（主要是服务器端□，满足多种需要。如果有需要，也可以自己写插件或管理脚本，开放的架构，允许我们这样做。

(6) 方便。– 个例子：部署应用的时候，以前的做法是找出一个项目中修改过的文件，更新到服务器上去，可以在服务器上执行svn export命令，把代码库中的最新版本导出，完成部署（也可以替换回老版本)。

(7) 速度与稳定性看起来都不错。

14.3.3 CC

ClearCase (简称：CC)是IBM Rational公司的旗舰产品之一，是全球领先的软件配置管理工具，它广泛地应用于众多的企业级软件工程实践之中，拥有众多的企业级用户。CC提供C/S和B/S两种架构的配置管理解决方案，提供了全面的软件配置管理功能。

CC的特点如下。

·独有的存储库 VOB (Version Object Bases)。

·可视化的文件版本树。

·并行开发。

·版本历史记录。

·自动的比较和版本间的合并。

·工作空间管理。

ClearCase是集中式版本控制工具，是SCM管理工具其中的一种。是RATIONAL公司开发的配置管理工具，类似于VSS，CVS的作用，但是功能比VSS，CVS强大得多，而且可以与Windows资源管理器集成使用，并且还可以与很多开发工具集成在一起使用。但是对配置管理员的要求比较高，安装比Windows还大，运行比蜗牛还慢，能用ClearCase的一般是世界500强。

14.3.4 GIT

GIT是一个开源的分布式版本控制工具，GIT最初由Linus Torvalds编写，用于Linux内核开发的版本控制工具。GIT与常用的版本控制工具CVS、Subversion等不同，它采用了分布式版本库的方式，不必服务器端软件支持，使源代码的发布和交流极其方便。GIT的速度很快，这对于诸如Linuxkemel这样的大项目来说自然很重要，GIT最为出色的是它的合并跟踪 merge tracing)能力，GIT在国外已经非常普及，国内并未普及(在慢慢普及)，越来越多的开源项目已经转移到GIT。

GIT的优势主要有：

(1) 更方便的Merge。分布式管理必然导致大量的Branch和Merge操作。因此分布式版本控制系统都特别注意这方面。在传统的CVS里面制作Branch和Merge简直就是噩梦，Subversion作为一个用于替代CVS的系统，专门改进了 Branch操作。然而似乎人们没有注意到，Branch是轻松了，可是Merge呢？如果不能很方便地Merge回来，做Branch仍然是噩梦。事实上，我就经历过在开发团队里面由于队友操作不对而在Merge的时候把我的许多代码都覆盖掉了。当时正是使用的subversion。虽然源代码仍然在历史里面，但是要去一个一个地找出被覆盖掉的文件并恢复过来确实是一件很难忘的事情。

(2) 更方便地管理。传统的版本控制系统使用中央仓库，一些仓库相关的管理就只能在仓库上进行。赋予开发团队每一个人中央仓库的管理权限是非常不好的。但是有时候确实会比较不方便的地方。

(3) 更健壮的系统。分布式系统一般情况下总是比单服务端的系统要健壮，因为当服务端一旦挂掉了，整个系统就不能运行了。然而分布式系统通常不会因为一两个节点而受到影响。

(4) 对网络的依赖性更低。虽然现在网络非常普及，但是并不是随时随地都有高速网络，甚至有时候根本没有网络可以访问。低速的网络会让人心情烦躁，有时候就呆呆地盯着屏幕上的commit进度，什么事情也干不了。而没有网络连接更是致命的：你无法commit ！这表示你进行任何改动以前都必须小心翼翼，否则你可能再也找不回你曾

经写的一些代码了。

(5)更少的“仓库污染”。有时候你要做一个模块，它不是太大，所以没有必要为它新建一个branch，但是它又不是那么小，不可能一次提交就做好。于是便会提交一些不完整的代码到仓库，有时候会导致整个程序无法运行，严重影响团队里其他人的开发。大多数人在这种情况下的解决办法都是写完之后再提交。但是作为习惯了版本控制的人来说，进行不计后果的大幅修改是经常的事情，到后来突然发现自己先前的代码没有提交，就后悔莫及了。如果是分布式系统的话就不会存在这样的问题，因为本地仓库的修改不会影响到别人的仓库。当你完成并测试以后，就可以在邮件列表里面说：我已经把这个模块做好了。然后感兴趣的人就可以从你这里pull你的成果了。

GIT和SVN的比较：

·在很多情况下，GIT的速度远远比SVN快。

· SVN是集中式管理，GIT是分布式管理，分布式和集中式最大的区别在于：在分布式下，本地有个代码仓库，开发者可以在本地提交；而集中式版本控制，只有在服务器才有一个代码仓库，只能在服务器进行统一管理。

· SVN使用分支比较笨拙，GIT可以轻松拥有无限个分支。

· SVN必须联网才能正常工作，GIT支持本地版本控制工作。

·旧版本的SVN会在每一个目录置放一个.svn，GIT只会在根目录拥有一个.git。

14.4本章练习

1. 选择题

(1) 在进行项目文档及配置管理时，引入“基线”这一概念的目的是_____。

A.保证成果的完整与正确　　B.合理分配权限

C.保证成果相互依赖性　　D.合理控制变更

参考答案：D

(2) 在软件配置管理中，_____ 不是配置项。

A.程序　　B.文档　　C.过程　　D.数据

参考答案：C

(3) 软件配置项的属性一般不包含_____。

A.源代码　　B.日期　　C.标识符　　D.作者

参考答案：A

2. 简答题

(1)请简述你在项目中的一次变更处理过程。

答：变更处理过程中需要包含的活动有变更申请，分析和评价变更，批准或否决申请，实现、验证和发布已修改的配置项。

(2) 请介绍下日常配置管理活动有哪些？

答：日程配置管理活动包括制订配置管理计划、配置标识、配置控制、配置状态报告、配置审计、发布管理和交付。

(3) 请简述一个软件开发项目的文档有哪些？

答：软件文档一般分为三类：开发文档、产品文档、管理文档。

① 开发文档描述开发过程本身，基本的开发文档是：

·可行性研究报告和项目任务书。

·需求规格说明。

·功能规格说明。

·设计规格说明，包括程序和数据规格说明。

·开发计划。

·软件集成和测试计划。

·质量保证计划。

·安全和测试信息。

② 产品文档描述开发过程的产物，基本的产品文档包括：

·培训手册。

·参考手册和用户指南。

·软件支持手册。

·产品手册和信息广告。

③ 管理文档记录项目管理的信息，例如：

·开发过程的每个阶段的进度和进度变更的记录。

·软件变更情况的记录。

·开发团队的职责定义。

·项目计划、项目阶段报告。

·配置管理计划。

3.判断题

(1) 管理信息系统文档的规范化管理主要体现在文档书写规范、图表编号规则、文档目录编写标准和文档管理制度等几个方面。（V)

(2) 基线中的配置项构成一个相对稳定的逻辑实体。基线中的配置项被“冻结”了，不能再被任何人随意修改。对基线的变更必须遵循正式的变更控制程序。（V)

(3) 配置审计（ConfigurationAudit)也称配置审核或配置评价，包括功能配置审计和物理配置审计，分别用以验证当前配置项的一致性和完整性。（V)

(4) SVN是一个开放源代码的分布式版本控制系统，此工具是在CVS的基础上，由CollabNet提供开发的，也是目前在国内软件企业中使用最为普遍的配置管理工具。（X)