CM Synergy 简介

发布版本 6.3

编号: ICCM-063-010

Telelogic 中国北方分公司

北京市东城区东长安街 1 号

东方广场西三办公楼 909 室　邮编:100738

电话：010 85185130　传真：010 85185136

目录

简介

CM Synergy 是一个综合的，功能丰富的变更管理系统，能够帮助软件开发团队控制软件、文档开发及维护的活动。CM Synergy 支持从中等到大型的开发团队，可以适合混合平台及分布式的计算机环境。

本手册的目的是帮助潜在用户及评估者对 CM Synergy 的术语、概念与方法获得一个基本的了解。CM Synergy 有几个界面。在本手册的一些段落中所讨论的概念与方法是针对特定界面的。

我们假设您对 Windows 与 UNIX 操作系统及其目录文件结构有基本的了解。

从其他工具转换

CM Synergy 能够管理同一文件以档案形式所维护的多个版本的过程以及能够管理其他更多的内容。作为 CM Synergy 的新用户，你或许在以前使用过版本控制工具，如 PVCS (Windows) 或 RCS 或 SCCS (UNIX)。这些工具维护对文件的版本控制，但是缺乏 CM Synergy 的很多功能，例如工作流程管理，基于角色的安全机制，产品生产的可重复性与基于规则的配置更新等等。

虽然 CM Synergy 与其他版本控制工具在对流程的支持上存在着很大的不同，但是熟悉其他工具的用户可以很容易地转移到 CM Synergy 的界面。

约定

本节描述本手册的约定。

CM Synergy 的图形用户界面

CM Synergy 具有以下几种图形用户界面。注意当本文讨论 CM Synergy 产品 (CM Synergy, CM Synergy for Developers, 与 *Active*CM) 时，都使用通用的名字 CM Synergy。当讨论特定的用户界面时，使用下面的名称：

- CM Synergy 原始用户界面

  这个界面为开发人员与开发主管提供配置管理的功能。

- CM Synergy 开发人员界面

  这个界面只适合具有*开发人员*的角色的用户使用。它包含了一部分针对开发人员的特定功能。

- 任务栏界面 (*Active*CM 的一部分)

  这个界面是针对开发人员的简化的配置管理界面。一旦启动，它将会主动地进行所需的操作，它包含了 CM Synergy 开发人员用户界面的一部分功能。

- 浏览器界面 (*Active*CM 的一部分)

  这个界面是任务栏界面的辅助界面。

- CM Synergy 命令行界面

  命令行界面 (CLI) 可以适用于 Windows 与 UNIX 平台。

使用 CM Synergy 的好处

配置管理工具的目的

配置管理系统的目的是帮助开发团队控制源代码修改、开发文档与管理产品的流程。以下重点讨论开发团队所面临的主要问题，以及配置管理工具可以怎样帮助减少或消除多个开发人员修改多个文件时所面临的需要协调一致的挑战。

一个配置管理工具必须能够导入平铺或多层嵌套的目录结构。一旦项目数据被迁移，工具必须能够容易地创建基线—所有接下来的开发工作所基于的一个项目版本。

理想地说，配置管理工具应该是透明的，允许开发人员用通常的方式来操作源文件，并与其他开发人员的变更所造成的不稳定因素所隔离。在某一点上，隔离又应该有所保留以便开发人员可以在需要时将别人的变更包含到他们自己的项目中。因此，配置管理系统应该为开发人员提供具有一定的隔离程度的工作空间，同时又能够共享团队人员变更了的源代码，以利于开发人员能通过团队的方式来工作。

配置管理工具应该能够使团队提高各个阶段的编译的稳定性。同时，工具应该能够提供一种流程使开发能够持续地达到较高的编译水平。这种流程能够确保团队走在正确的产品发布道路上并使他们可以修复造成高水平编译不稳定的任何代码。

配置管理工具必须超越仅仅能够重新生成各种应用软件版本的能力。通常，开发团队并行地对修复补丁版本与新功能版本进行开发。这些发布可能具有不同的编译需求。当修复补丁版本完成后，开发新功能发布的团队一般会把这些变更包含到他们的工作中来。配置管理工具的一个重要的目的就是应该允许同时开发不同类型的项目并为开发人员重用代码提供一种方法，而不管这些代码是在目录的何处所创建的。一旦并行产品被发布，技术支持人员必须能够重新创建过去的阶段性发布以用于客户支持。随着开发团队所完成的并行发布数量的增加，产品版本重新生成的需求也会随之增加。

CM Synergy 的优点

CM Synergy 提供了完整的配置管理环境，可以使开发团队在方便、快捷及安全的环境下工作。本节描述 CM Synergy 能够为软件开发组织提供的功能与好处。

容易使用，直接使用定制好的模板

大多数的开发团队都时间紧迫，只有很少的时间来学习新的工具。Telelogic 能够理解使开发团队很快熟悉新工具的重要性。因此，CM Synergy 提供以下的基本功能：

- 直观、易于使用的图形用户界面与综合的命令行界面
- 简单的基于任务的方法来跟踪变更
- 与许多流行工具和开发环境相集成

- 模板驱动的，对流程的灵活支持
- 无需定制就可以直接使用工具
- 预定义的角色、对象生命周期、安全性及访问权限
- 支持图形界面与命令行界面的帮助文档

快速上手

开发团队在安装 CM Synergy 后，通常在同一天就可以使用它，这是因为 CM Synergy 具有以下功能：

- Telelogic 的快速且自动的迁移工具可以把文件系统中已有的项目置于 CM Synergy 的控制之下。
- CM Synergy 与已有的编译程序兼容，你可以用现有的编译文件来编译产品。

可以使新手快速进步

一旦设置好了 CM Synergy，新的用户就可以直接使用了。一旦 CM 管理员把你添加到 CM Synergy 的用户中，你就可以检出一个项目来创建你自己的工作空间，然后按下列步骤进行变更：

- 选择你将要工作的任务。
- 检出项目并进行文件变更。
- 检入任务，同时也就检入了所有变更了的文件。

这些步骤是使用 CM Synergy 的最基本的步骤。你还需要学习许多其他的功能才能真正用好 CM Synergy。你或许需要参加培训课程来更深入地了解产品。无论如何，你可以马上使用 CM Synergy 来帮助你的团队在截止期限前完成任务。

灵活、自动化的工作流程

CM Synergy 的基于任务的方法为编译与测试软件提供了一种直观的方式，使你能够迅速发现错误并达到你所需的质量水平。基于任务的工作流程能够使你：

- 容易评审变更的原因并辨识所有为实现变更所修改了的文件
- 在隔离的环境下进行变更，并在需要时可以获取其他人的变更
- 可以准确地控制测试空间与软件发布中的变更
- 发现配置冲突，如并行版本或缺失的变更
- 保存或重新生成所需的软件产品
- 自动的编译管理操作
- 管理并行开发

- 容易为不同的团队设置不同的工作流程

安全的团队工程环境

以下的功能可以确保你的开发团队在开发软件项目中无需过多地关注 CM Synergy 工具：

- 数据存储库是可信赖的、商业化的关系数据库管理系统 (RDBMS)。
- CM Synergy 使用开发人员已有的目录结构与工具。
- CM Synergy 为开发人员提供私密、隔离的工作空间，使他们能够完全控制他们检出的文件版本并进行原型开发、编辑、编译与调试，然后再把变更结果共享给他人。
- CM Synergy 通过准确地创建基线配置来提供项目的恢复功能。
- CM Synergy 允许你通过任务关联与检入、检出能力来跟踪文件与项目
- 开发人员可以在需要时利用 CM Synergy 的重新配置过程进行项目更新来使用已测试和检入的文件。
- CM Synergy 通过自动的并行开发支持与内建的安全机制来管理并行开发中的变更。
- 在集成或发布空间进行测试时，开发人员无需停止开发。因为这些空间是与开发人员所正在进行的变更隔绝的。
- 缺省的生命周期可以保证只有经过授权的变更才能被发布。
- CM Synergy（根据你的组织所定义的*用户*、*角色*、*操作*与*数据*）可以控制数据与文件的访问操作的安全性。

高性能的编译机制

ObjectMake, CM Synergy 的面向对象的 make 工具对编译过成提供了完全的控制并与通用的编译文件格式兼容。使用 ObjectMake，你可以：

- 使用你已有的编译知识与程序来快速地把变更通过编译文件进行编译。
- 使用可重复的编译环境来使编译过程更加流畅。
- 根据当前版本与前一版本的编译文件清单内容的不同来迅速判断出编译错误的来源。
- 通过编译文件宏的功能来创建多种独立的编译文件
- 在编译文件中自动生成隐含的依赖关系
- 针对不同平台来控制编译：针对并行或分布式编译，针对远程目标编译

远程控制与信息传递

Telelogic 的分布式变更管理 (DCM) 产品允许你在遍布世界的任意多个数据库中共享软件变更。使用 DCM，你可以：

- 允许开发人员使用他们所熟悉的相同的用户界面（DCM 的信息在相应的对话框中显示）

- 选择适当的方法来定义数据在数据库之间传递的特性与方向。
- 可以发送源对象、项目、任务夹与任务到任何的 DCM 数据库，不受对象分组的限制。
- 自动或手动传输整个或部分数据库
- 在传输完成之前可以预览传输列表
- 可以在不同地点的数据库中进行并行开发，然后使用比较与归并功能来解决冲突

与 Windows 开发环境无缝集成

Telelogic 的配置管理工具可以与许多业界领先的开发环境相集成。这些集成能使你在自己的开发环境中进行源文件控制与配置管理。这些集成很容易安装。Telelogic 支持以下的集成：

- DOORS
- Microsoft Developer Studio for Visual C++ 与 Visual Basic
- VisualAge for Java
- VisualCafé
- Visual Studio .NET
- Microsoft Project
- Sybase PowerBuilder
- Mercury Interactive TestDirector
- Websphere Studio
- 所有支持 Microsoft SCCI 1.x 规约的工具

CM Synergy 的基本概念与术语

本章介绍 CM Synergy 的基本概念与术语。这些概念是按顺序进行介绍的。Telelogic 建议你按顺序阅读本章。

请注意本手册讨论的是所有的 CM Synergy 界面。在可能的情况下，本手册对所有界面使用通用的术语；当讨论特定的用户界面是怎样工作时，例如 CM Synergy 开发人员界面，相关的术语是特指这个界面的。

对于所有界面来说概念与术语是相同的。例如，所有界面都支持作为缺省的基于任务的方法，并且所有界面都建议在开发人员进行一个新的项目时要引入最新的成员来更新工作空间。

CM Synergy 的数据库

CM Synergy 的数据库是一个用于存储所有被控数据的数据存储库，包括源文件与数据文件、它们的属性与它们彼此之间的关系。你可以根据你需要控制的数据大小及组织方式来决定使用一个或几个 CM Synergy 的数据库。

任务与对象

一个*任务*代表需要对软件应用进行的一个逻辑变更。一个任务收集了所有用于完成变更的软件修改并包括变更的描述与负责完成这个任务的人的名字。

一个*对象*是一组数据，如一个文件或目录。举个例子说，源文件、编译文件、测试结果、目录与文档都可以称作对象。为了跟踪的目的，一个对象的每个修订版本都被称作一个*对象版本*。每个对象版本都具有一组属性（例如，名字、所有者、创建时间）来对之进行定义。

例如，一个用户对图形用户界面（GUI）的应用程序报告了一个缺陷。GUI 团队的领导把对缺陷的修复工作分派给了叫做 Hannah 的开发人员。她被分派了一个叫做 *“在 Snap 对话框中修复滚动问题”* 的任务。当她每次使用 CM Synergy 检出一个对象来修复 Snap 对话框中的滚动问题时，这个对象就会与这个任务相关联。

为了说明这些概念，见图 1。

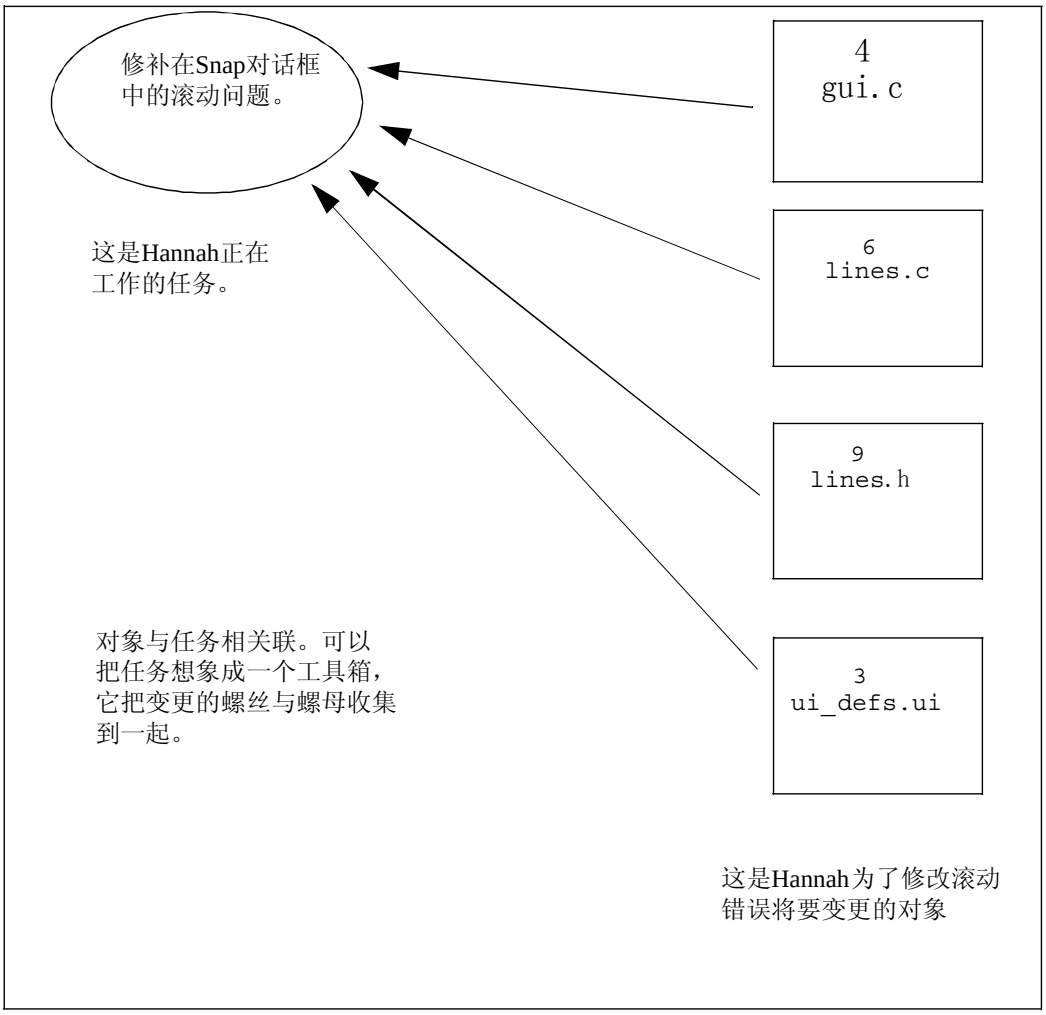

图 1. 任务与对象

任务与对象具有一种关系。由任务所组合的对象被称作与任务*相关*。所有为了修复一个特定问题的对象被组合在一起，由任务名来描述。在图 1 右侧的对象版本与“*修复在 Snap 对话框中的滚动问题*”的任务相关，因为它们包含为了完成这个任务所需的代码变更。在每个对象的上部的数字表示这个对象的版本。

任务的名字，在这里叫做“*修复在 Snap 对话框的滚动问题*”，是指任务的*提要*。另外，当你创建一个新任务时，CM Synergy 也为它分配一个数字。除了数字与提要之外，任务还包含关于变更的其他信息，例如，解决问题的人的名字。当一个任务被分派给一个开发人员的时候，解决问题的人就自动地被设置为这个开发人员。当你创建任务时，你也可以设置以下的属性：

- **发布**

指定逻辑变更将被用于你的软件的哪个发布。发布的名字可以如 editor/2.0 或 ChangeSynergy/5.0。编译经理将为你的软件设置发布的值。

- 平台

指定逻辑变更将被应用的硬件平台。平台的名字可以如 SPARC-SOL 或 x86-WIN32。 编译经理将根据你的软件来设置平台的值。如果一个任务被用于多个平台，那么你就不必设置任务的平台值。

- 子系统

为任务指定软件的子系统。例如，如果你开发一个客户端-服务器的软件应用，你的子系统可以是“客户端”、“服务器”与“通讯”。如果你开发一个财务软件，你的子系统可以是“AR”、 “AP”或 “GL”。配置管理员将根据你的软件来设置子系统的值。你不必设置任务的子系统；这个值只是为了方便而已。

任务与你将修改的源文件都存储于 CM Synergy 的数据库中。任务没有版本，但是他们有生命周期（详见 13 页）。任务不能包含其他任务。

关于对象的更多内容

在 CM Synergy 数据库中所管理的对象可以被以下的属性唯一地标识：*名称*、*版本*、*类型*与*实例*。

在缺省的情况下，这个由***四部分组成的名字***（也叫*对象描述*或*全名*）的写法是这样的：

名称-版本号：类型：实例

例如, main.c-3:csrc:2 或 draw.c-beta:csrc:7

一个对象的名字可以最多由 151 个字符组成，版本可以是任何 32 个字符的组合。类型可以是任何缺省的类型（例如，csrc, ascii 等）或任何你创建的类型。名称、版本与类型由你来指定，但是实例是由 CM Synergy 通过计算来获得的。

*实例*是用于区分同一名称与类型的多个对象的，但是它们彼此并不构成版本关系。例如，一个项目可以包含 20 个不同的编译文件，每个都叫编译文件，每个都在不同的目录中，并且每个都有许多版本。如果你想用 makefile-4，通过查询你可能会得到 6 个叫 makefile-4 的对象。在这里，实例属性就可以帮助区分你到底想用哪个 makefile 对象。实例的值通常是数字，但有时也可以是字母，例如在分布式的数据库（DCM）中。

你可以在多个目录中使用一个指定的对象版本。你可以通过路径名来引用一个对象版本。但是，文件的位置也许会变化，由四部分名称所确定的唯一标识却是不变的。

检出与检入

为了修改已存在的对象，你必须创建一个可以修改的对象版本。你可以通过*检出*操作来创建一个可以被修改的对象，这也意味着从原有的版本中创建了一个新版本。新版本包含了对原有版本进行复制所获得的所有属性。

检出操作与 PVCS get -l 命令 (Windows) 或 RCS co -l 或 SCCS get -e 命令 (UNIX) 相似。不同的是，在 CM Synergy 中，只有在你准备修改对象时才需要检出对象。如果你只需要查看或使用它，你就不必检出它。你可以检出任何类型的对象（文件、目录、符合连接、可执行文件等等）。

*检入*操作通常会通过使对象不可写来保护对象的版本。一旦被检入，对象就可以被其他用户所获取。检入一个对象会改变其*状态*属性，状态属性定义了谁能修改或使用这个对象。检入操作与 PVCS put -u 命令 (Windows) 或 RCS ci -u 与 SCCS delget 命令 (UNIX) 相似。

在 CM Synergy 中，你可以多次检入文件的同一个版本，而不仅是只检入一次。例如，当你预备测试时，你可以检入文件；当你预备发布时，你可以把它检入到不同的状态。

用我们的开发人员 Hannah 的例子来说，我们知道她将通过修改文件来修复滚动问题的缺陷。她将会检出需要修改的文件（如 gui.c 与 lines.c 等）来完成分派给她的任务。在完成文件的修改与修复缺陷后，她将检入任务以便这些文件可以被用于下一个产品的编译。

在开发周期中，检出与检入的过程是变更控制的重要组成部分。当开发人员检出一个对象（如一个文件）来修改它时，这个开发人员就获取了这个被复制的文件的所有权。在缺省的情况下，CM Synergy 允许另一个开发人员检出同一个文件的另一个版本来进行修改。这就是*并行开发*，这个同一个文件的不同的版本被叫做*并行版本*。工作过程不会因为文件的一个版本已经被使用而被耽搁。

在某一点上，并行版本将需要被归并。CM Synergy 的*归并*功能能够使你融合一个文件的两个并行版本的信息。当你归并两个对象版本时，会生成第三个版本。CM Synergy 使用最新共有的前辈版本来推荐新版本应该包含哪些变更。如果文件中没有冲突，就可以直接使用新版本。如果存在冲突，你必须选择有冲突的行的代码以确定归并版本应该使用的代码。

历史

一个对象的*历史*可以显示对象的所有存在的版本及版本之间的关系。对于*历史*，Telelogic 是指在当前对象版本之前的所有对象版本（叫做*前辈*）与当前对象版本之后的所有对象版本（叫做*后辈*）。

图 2 显示了文件 save.c 的历史。箭头指示哪个版本是从哪个版本检出的。例如版本 2 是从版本 1 检出的。版本 1 是版本 2 的前辈，而版本 3 与 2.1.1 是版本 2 的后辈。版本 3 与 2.1.1 是并行版本。

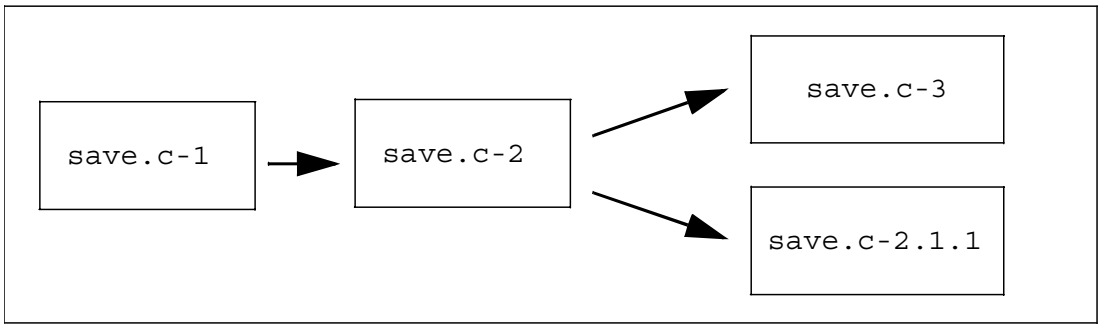

图 2. 历史关系

在第 23 页中给出了 CM Synergy 开发人员界面中“历史对话框”的例子。

属性

一个对象的*属性*可以使它区别于其他对象。一个对象的基本属性包括它的由四部分组成的名字的组成部分（*名称*、*类型*、*实例*、*版本*）以及*所有者*、*状态*、*平台*与*发布*。你可以在 CM Synergy 的属性对话框中或使用 ccm properties 命令来查看属性。

在为编译与测试阶段收集软件版本时，*平台*与*发布*属性是很重要的。CM Synergy 收集那些平台与发布的属性值与编译和测试阶段的配置所要求的属性值相匹配的版本。

缺省任务

*缺省任务*是指你当前工作的任务。（当使用 CM Synergy 开发人员用户界面时，你当前工作的任务叫做*当前任务*。）当你指定一个任务作为缺省任务时，你就相当于告诉了 CM Synergy 在你每次检出对象时，你想要这个对象与缺省任务自动关联。当你为任务而完成了所有的软件变更时，你可以检入这个任务。（在 CM Synergy 开发人员用户界面中，你可以完成这个任务。）检入一个任务也就检入了与任务相关的所有对象版本。

采用任务能够使逻辑变更作为一个单元在 CM Synergy 的生命周期中移动。使用任务的信息，通过指定你想要的逻辑变更你就能够收集用于编译与测试的软件版本。当你用任务来代表软件变更时，你就指明了与任务相关的对象版本应该被一起使用，在一个项目中只用一些版本而不用另一些版本是不行的。有了这些信息，CM Synergy 可以帮助你在软件生命周期的早期发现配置的问题。

角色、生命周期与状态

在使用 CM Synergy 当中，每个用户会使用特定的角色来进行操作。在数据库中，一个用户可以有多个角色，并且可以随时变换角色。在数据库中，用户的*角色*定义了他对数据与操作的权限。CM Synergy 中典型的用户包括*开发人员*—用于开发与测试软件，*编译经理*—用于配置与建立测试空间并预备用于发布的软件。

所有对象都有生命周期。*生命周期*是指对象的可能状态以及基于当前状态对象可以转移到哪个状态。一个对象的*状态*定义了对象在生命周期中的阶段以及可以对它采用的操作，例如谁可以

修改它。

在 CM Synergy 的基于任务的方法中，对象的三个缺省的状态是*工作*、*集成*与*已发布*。图 3 说明了对象的缺省状态的在生命周期中的顺序。

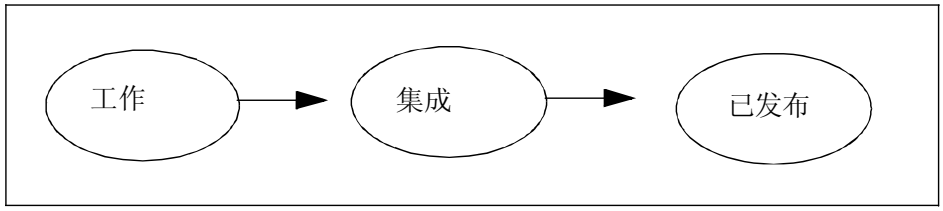

图 3. 对象的生命周期

这些状态是按照下列的方式使用的:

- *工作*状态被用于所有新的对象版本，无论是当它们被创建或从另一个版本中检出时。在*工作*状态下的对象是可以被其所有者所修改的。
- *集成*状态是用于编译-管理集成测试的。处于*集成*状态的对象是不可修改的。
- *已发布*状态是用于已经发布或已经到达一个里程碑阶段的对象。处于已*发布*状态的对象是不可修改的。

你的组织也许会使用其他生命周期的方案。CM Synergy 有几个可选的状态，如*检查点*、*拒绝*、*共享*与*可视*状态。配置管理员也可以为你的组织创建定制的状态。

与任务关联的对象的生命周期是和其任务的状态紧密相连的。是任务的生命周期才使对象能在流程中移动。例如，你可以在任何时候检入对象，但是直到你完成了与其相关的任务时，你才能在集成测试阶段使用它。

在缺省的情况下，任务可以具有以下的状态：*已注册*、*任务已分派*与*已完成*。图 4 说明了任务的缺省状态在生命周期中的顺序。

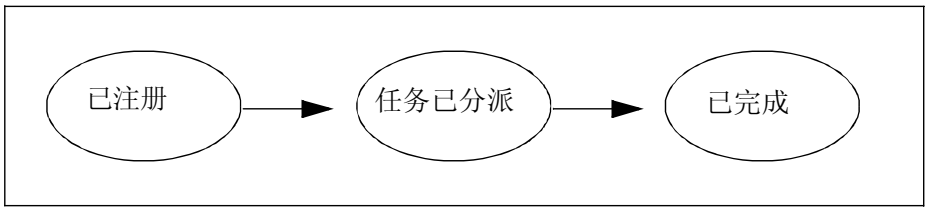

图 4. 任务的生命周期

这些状态是按照下列的方式使用的:

- *已注册*状态是用于任务已经被创建但还没有分派的时候。当任务处在*已注册*状态时，它是可以被修改的并等待被分派给开发人员。在缺省状态下，用户处在*开发者*、*编译经理*与*任务分派者*的角色时可以创建任务。
- *任务已分派*的状态是用于任务已经被分派给开发人员时。当任务处在*已分派*的状态时，它可以被修改并可以被解决者（被分派了任务的人）所使用。在缺省状态下，用户处在

*编译经理*与*任务分派者*的角色时可以分派任务，处在*开发者*角色的用户可以把任务分派给他们自己。

- *已完成*状态是用于任务已经被完成时。当任务处在*已完成*状态时，它已经被检入并且只有配置管理员才能修改。在任务完成之前，所有与任务相关的对象版本都必须被检入。只有任务的解决者才能完成任务。任务解决者可以是任何角色。

项目

一个*项目*是用户定义的一组相关文件、目录与其他项目（叫做*子项目*）。一个项目通常代表一个逻辑上的软件组，例如一个库或一个可执行文件，并且它包含文件的目录结构。例如用于实现 editor 应用程序的软件可能被存储在一个叫做 editor 的项目里。

项目的版本就如同任何其他的对象的版本一样。同一个项目的不同版本可以包含不同版本的成员对象，甚至不同的成员。例如，不同版本的 editor 项目可以代表第一个发布版本 editor/1.0，与后续的发布版本 editor/1.1, editor/1.2 及 editor/2.0。对于发布版本是 editor/2.0 的 editor 项目可以包含版本 1.0 不存在的新对象，同时也可以包含同一个对象的许多新版本。在 1.0 版本中含有的文件也可以不必包含在 2.0 版本的项目中。

一个单个对象版本可以是许多项目的成员。虽然同一个对象版本可以出现在不同的项目中，但这个对象版本只会在 CM Synergy 数据库中出现一次。

一个项目可以包含其他的项目。如果一个项目被另一个项目所包含，那么这个项目被称为*子项目*。你可以通过把软件分组成不同的项目来对其进行组织。例如，你可以为每个可执行文件设置项目，并把它们作为代表整个应用程序项目的子项目。

对于任何给定的项目，可以存在几个不同的版本：

- *工作*项目是开发人员用来开发与测试他们的变更的项目。
- *预备*项目是编译经理为测试与发布做软件预备的项目。
- *已发布*项目是软件版本已经被发布或已经到达一个里程碑的项目。

在第 22 页上给出了 CM Synergy 开发人员界面的“项目窗”的一个例子。

目录与候选者

CM Synergy 对目录的控制方法与文件相同。与文件系统中的目录不同，在 CM Synergy 中创建的*目录*会对属于它的文件进行跟踪。对于每个属于目录的文件，目录有一个存放地，叫做*目录入口处*。目录入口处描述属于这里的文件的名字，但不描述版本。例如，如果目录入口处描述了 delete.c，那么目录中就应该有叫做 delete.c 的文件，但是并不特指其某个版本。在目录入口处所描述文件的可用的所有版本被称做*候选者*。

如果你要在目录中添加或删除一个对象，那么这个目录对象必须是可写的。如果你想修改（添加或删除一个成员）处于不可修改状态的目录，CM Synergy 就会为你检出一个新的目录版

本。如果你设置了缺省的任务，新的目录就会自动与缺省的任务相关，并当你在检入任务时，它就会与其他变更被一起检入。

就像源文件一样，目录也会出现并行的版本。CM Synergy 能使你归并并行的目录。当你归并目录时，你会对比目录入口处的差异，并选择相应的目录入口把它包含在归并后的版本中。例如，如果一个用户检出了源目录并在其中加入了叫做 open.c 的对象，而另一个用户检出了这个目录的并行版本并在其中加入了叫做 select.c 的对象，那么归并的操作应该显示两个新的目录入口并且你可以在归并后的版本中把他们都包含进来。

工作空间

*工作空间*是在你检出一个项目时，CM Synergy 把这个项目写到你的文件系统的地方。工作空间可以存在于网络文件系统的任何地方。工作空间中的项目目录树结构与在 CM Synergy 数据库中的项目目录树结构是一致的。

CM Synergy 可以使工作空间与数据库同步。在 UNIX 系统中，工作空间的文件与数据库的文件是*连接*关系。在 Windows 系统中，工作空间文件是从数据库文件*复制*的。

当你创建或变更一个项目时，CM Synergy 会自动且透明地更新你的工作空间—当你添加成员到一个项目时，CM Synergy 把这个新文件更新到工作空间；当你从一个项目中删除成员时，CM Synergy 也会把相应文件从工作空间中删除；你也可以在 CM Synergy 的控制之外，用手工直接在网络文件系统中对工作空间进行更新。

同步（Reconcile）

当你使用 CM Synergy 的操作来更新项目数据时，你的工作空间与 CM Synergy 的数据库都会得到更新并保持同步。但是如果你直接在工作空间下工作，而不使用 CM Synergy 的操作来更新你的文件时，工作空间可能会与 CM Synergy 数据库中的项目不同。你可以使用 CM Synergy 的 reconcile 命令来对比你的工作空间与数据库，并有选择地进行更新。（reconcile 操作在 CM Synergy 开发人员界面中被称做同步（*synchronize*）。）当数据库与工作空间的文件不同时，CM Synergy 会通知你，你可以对比其不同点并有选择地进行更新。你也可以选择对文件进行归并；如果你进行归并，归并工具会为你显示变更并让你进行选择。

如果你需要在 CM Synergy 数据库之外工作，你可以在工作空间中修改文件，无论数据库中的对象版本是否被检出。当你重新连接到数据库与项目进行同步时，CM Synergy 会自动从数据库中检出一个新的对象版本并把你在工作空间中修改的对象加入到其中。

Reconcile 操作对数据的一致性很重要。如果你不把工作空间的文件变更存入到 CM Synergy 数据库中，那么工作空间中的文件的可靠性就只能依赖于操作系统本身了。当你做 reconcile 时，你的文件就变成了 CM Synergy 数据库的一部分并且会随你的数据库一起被备份。

使用、创建、添加、删除或不使用对象

在本章的早期，我们讨论了检出操作，检出会从已有的对象版本创建一个新版本。一个对象可以是一个文件、目录、文档或其他的数据。有许多其他的方式来修改项目的内容：

- **使用一个对象**

  如果在你的项目中你想选用对象的一个不同版本，例如回到一个早期的版本，你可以*使用*你需要的版本。使用（use）操作在调试过程中非常有用。如果你的测试失败了，你可以用一个早期版本替代这个对象来进行问题的调试。你可以使用除了被其他用户检出的版本（例如，处于*工作*状态的对象）之外的任何版本。

- **创建对象**

  当你想创建一个全新的对象而不是一个已有对象的新版本时，你就需要*创建*一个对象。这个操作与 PVCS vcs -I 命令 (Windows) 或 SCCS create command (UNIX) 类似。对象版本必须被以明显的方式来创建后（如必须在 CM Synergy 的控制之下）才能被检入。所创建的对象存在于项目的目录中，并会出现在项目的工作空间中。

  你也可以创建项目与目录。当你创建了一个项目，其工作空间就会随之产生。当你创建了一个目录时，起初它是空的直到你在其中创建或添加对象。

- **添加对象**

  你可以使用*添加*对象操作在项目的目录中添加一个已有的文件、目录或项目。你无需检出对象来把它添加到你的项目中。

- **删除或不使用对象**

  你可以通过删除或不使用操作来消除对象。当你*删除*对象时，它就从数据库中永久地消失了。如果一个对象在数据库中已被其他的项目所使用，你就不能删除它。当你*不使用*一个对象时，你只是把它从项目中移出，但它还是存在于数据库中。如果你日后需要，你还可以把它加回到项目中。

重新配置、基线、任务与重新配置模板

重新配置（在 CM Synergy 开发人员界面中称作更新成员）是在项目或目录中更新对象版本的过程。在项目或目录中的每个对象版本都将被评估，并从 CM Synergy 的数据库的候选版本中选出合适的版本。当开发人员开始工作于一个新任务时，他通常会更新其项目中的成员。他们通过这个操作来引入项目的最新成员。

对于特定发布与用途的一组项目其成员是基于*基线*之上的。*基线*是轻量级的对象，用来跟踪在这个基线中包含那些项目。可以把它想做一个或多个项目及其所含*任务*在某一时刻的快照。它可以代表一个特定的编译、一个里程碑或一个发布。

注意如果一个项目使用重新配置模板，重新配置模板会识别将使用哪个基线。引用了这个配置模板的项目使用基线来识别在更新时使用哪个基线项目。（一个基线项目是这个项目的起始点；每个项目根据基线来找到起始点—这叫做一个基线项目。）例如，如果对于当前发布的*隔离开发*的重新配置模板指明了应该使用 Integration Build 20020913 基线，并且它含有静态的项目 toolkit-int_20020913 与 calculator-int_20020913，那么一个开发人员的 calculator-bob 项目就应该选择 calculator-int_20020913 作为其基线项目。

因此，*重新配置模板*就是用来定义项目将怎样被更新的模式。当你重新配置项目时，它们指定规则用来选择所使用的基线外加一组任务。你的团队将使用重新配置模板来裁剪与协同他们的软件开发与测试过程。你可以在第 41 页的“定制你的团队的工作流程”中获得更多的关于重新配置模板的知识。

重新配置（更新成员）操作也使用基线。这会使对任务的评估更加流畅，也提高了重新配置的性能。使用基线的重新配置只对最新基线之后的任务进行分析，而不对所有的用于发布的任务进行分析。

编译经理通常会创建基线并且设置重新配置模板，然后对于特定的里程碑或发布共享给开发人员。

任务夹

任务夹是一组被命名了的任务。任务夹是用于收集逻辑上被分组的所有任务，例如，可以根据任务的状态、发布、所有者、或这些属性的任意组合来收集任务。任务夹的例子有*所有已分派给 Hannah 的任务*或*所有对于 editor/2.0 发布的已完成的任务*。

你可以用两种方式在任务夹中添加任务：

- 手工选择你所要添加的每个任务。一旦你选择这种方式，任务夹就只能包含这些任务直到你再次对任务夹进行手工修改。
- 指定一个数据库的查询。例如，你可以设置查询来选择对于 editor/2.0 发布的所有你的任务。当你访问这个任务夹时，CM Synergy 就会查询数据库并收集所有符合查询标准的任务。使用查询的优点是在你创建新任务时无需每次都手工来更新任务夹；如果一个任务符合你的查询标准，CM Synergy 就会自动地把新任务添加到任务夹中。

任务夹的另一个用途是它们允许多个用户共享一组任务。例如，一个软件配置通过了集成测试之后，通过了测试的任务可以用任务夹来共享给开发人员。

编译、产品与编译文件

*编译*是从已有的源文件中通过使用工具或编译器或代码生成器来生成文件的过程。

通过处理其他文件来编译成的文件被称作*产品*。任何对象类型都可能是潜在的产品；最常见的产品是可执行文件、库文件或目标文件 (.obj)。

项目可以包含任意个*编译文件*，编译文件是包含用于编译产品指令的文件。

产品可以是不受控的或受控的。*不受控的产品*存在于你的工作空间中，但是不在 CM Synergy 的数据库中。*受控的产品*是在 CM Synergy 中做为对象版本来控制的。因为受控的产品存在于 CM Synergy 的数据库中，用户可以共享它们。

当使用 CM Synergy 的 make 工具，ObjectMake，进行编译或手工标识对象版本作为产品时，一个对象就变成了产品。

ObjectMake 是一个并行的、分布式的编译工具，与通用的编译文件格式兼容。ObjectMake 是与 CM Synergy 的对象系统集成在一起的。你也可以选用第三方的编译工具。

注意：所有被控的对象都可以是产品除了项目、符号连接与目录（因为在编译文件中它们不能作为目标文件）。

迁移、视图与版本

本章将对下列的题目进行简要的介绍：

- 把已有的项目迁移到 CM Synergy 的原始界面
- 在 CM Synergy 中查看你的项目数据
- CM Synergy 是怎样对对象与项目实施版本的

迁移

*迁移*是你第一次把已有的目录与文件从你的文件系统导入到 CM Synergy 的原始界面的过程。你可以使用迁移命令 (ccm migrate) 或 CM Synergy 迁移对话框来导入数据。迁移对话框只有在 CM Synergy 原始界面下才存在。

你可以把整个应用导入到 CM Synergy 的原始界面，或者使用逐步（部分）迁移来导入部分应用。

项目迁移

*项目*是用户定义的一组相关的文件、目录或其他项目（叫做*子项目*）。

项目通常会包含源代码与用于编译产品的编译文件。项目可以有扁平的目录结构或很大的、嵌套的目录结构。

一些项目表示文件被编译的目录的结构，另一些则以一种实用的方式来组织对象，例如表示安装的目录结构。

当你想要把已有的目录树迁移到 CM Synergy 的原始界面时，你需要进行项目迁移。无论所要导入的目录与文件大小如何，你都需要考虑以下几点来达到优化项目配置的目的：

- **项目的大小—**如果你的软件应用的文件结构较小，只有几百个中等大小的文件，你可以无需考虑太多就可以把它们放到一个 CM Synergy 项目中。如果你需要管理的应用大于 2000 个对象，你应该把它们分成一组 CM Synergy 的子项目。
- **项目团队—**你的项目团队的大小与组织结构很重要。如果许多开发人员在一个大的应用下进行工作，并且如果存在自然的文件分隔（例如，许多开发人员不需要别人的文件来编译他们的那部分应用），你可以把应用的结构分成几个 CM Synergy 的子项目。这将减少每个开发人员所需管理的对象数目。

通过把应用分成反映开发努力的不同部分的子项目，你将减少需要用来编译应用的时间与空间。（子项目对于文档的结构非常有用。）

- **变种项目—**一个变种项目是同一个产品的平台变种，例如对于同时具有 UNIX 与 Windows 平台的软件。通常情况下，你将对每个平台所编译的应用拥有一个项目版本。

你可以通过对每个项目对象设置*平台*属性来区分变种项目的配置。

- **可重用的对象**-- CM Synergy 可以提供自足的软件单元（例如，库文件或可执行文件）。但这不是被强制的（例如，你可以随意地把任意多个程序或库放到项目中），把你的应用分隔成小的单元结构的好处是你可以共享或重用这些项目。

逐步迁移

当你从文件系统中把软件迁移到已有的 CM Synergy 原始界面的项目中，你就是进行了*逐步迁移*。你可以用两种方式来进行逐步迁移：

- **迁移项目的一部分**—当你想要添加新的目录与文件到 CM Synergy 数据库中的已有的项目中时，你需要迁移项目的一部分。
- **迁移整个项目**—当你想要控制其他人（例如，供应商或其他团队）在 CM Synergy 数据库之外进行修改的软件时，或者你想要控制他们的变更，很可能是把你与他们的变更进行归并，你将对整个的已有项目进行迁移。CM Synergy 原始界面将比较被迁移的文件与已存在的文件，并只对已变更的文件创建新版本。被迁移的软件版本作为你以前迁移版本的后辈。如果你通过创建新版本来修改任何对象，你的新版本将并行于新的被迁移软件（关于并行版本已在第 11 页中说明了）。这将使你能够把本地的变更与供应商或其他团队的最新版本进行归并。

预览迁移

使用 CM Synergy 原始界面迁移对话框可以预览你的文件结构并查看在 CM Synergy 原始界面中你的项目将怎样被组织。在预览过程中，你也可以选择忽略某些文件（以使它们不必被上载），为每个文件选择对象类型，并为 CM Synergy 原始界面将怎样迁移目录与文件应用规则。图 5 显示了迁移对话框，预览了 y:\users\pc_demo 中的目录与文件。

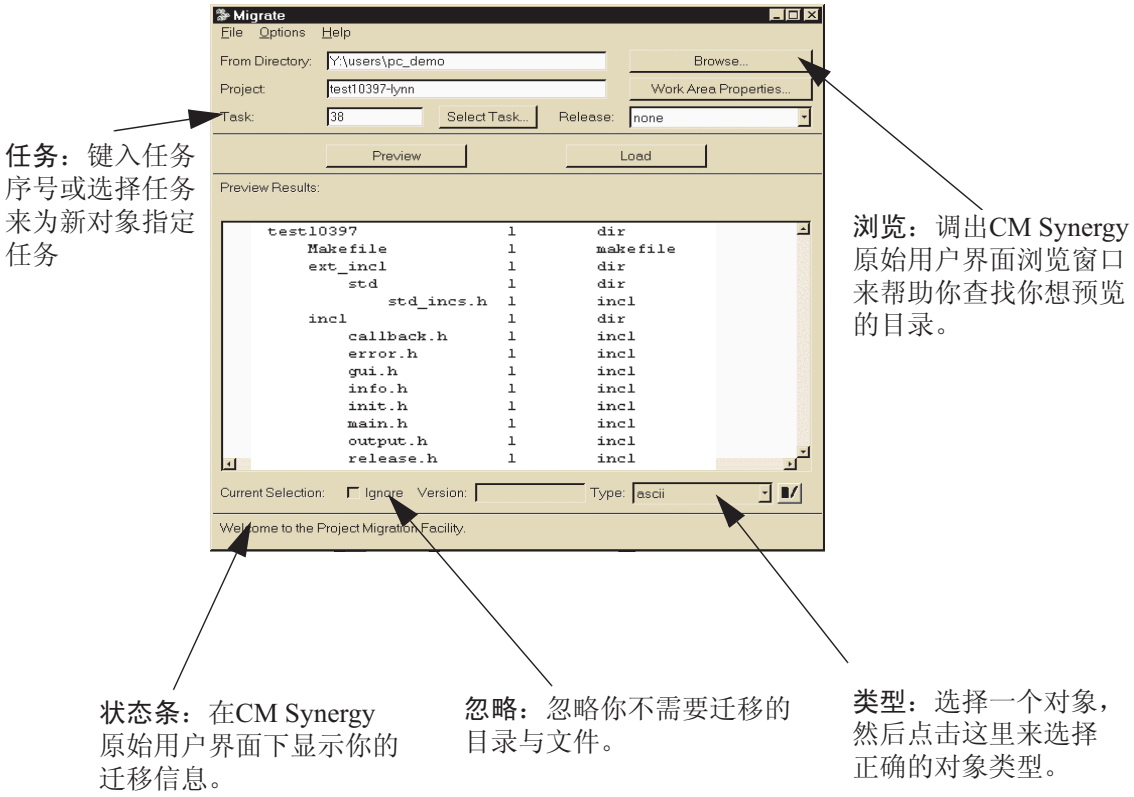

图 5 在预览中的迁移对话框

如果你所上载的文件包含多余 2500 个对象，迁移对话框将在你执行预览时提出警告。这将在上载之前给你一个重新考虑项目结构的机会。

怎样查看你的数据：项目窗口与历史对话框

当你把文件与目录迁移到 CM Synergy 数据库后，你就会希望看到新的顶层项目、根目录与对象成员。本节向你展示怎样在 CM Synergy 开发人员界面中来查看数据。

你可以使用项目窗口与历史对话框用不同的方式来查看你的数据。

项目窗口

每个 CM Synergy 开发人员界面都有项目窗口。在项目窗口中你可以显示任何项目。图 6 显示了项目窗口及相关提示。

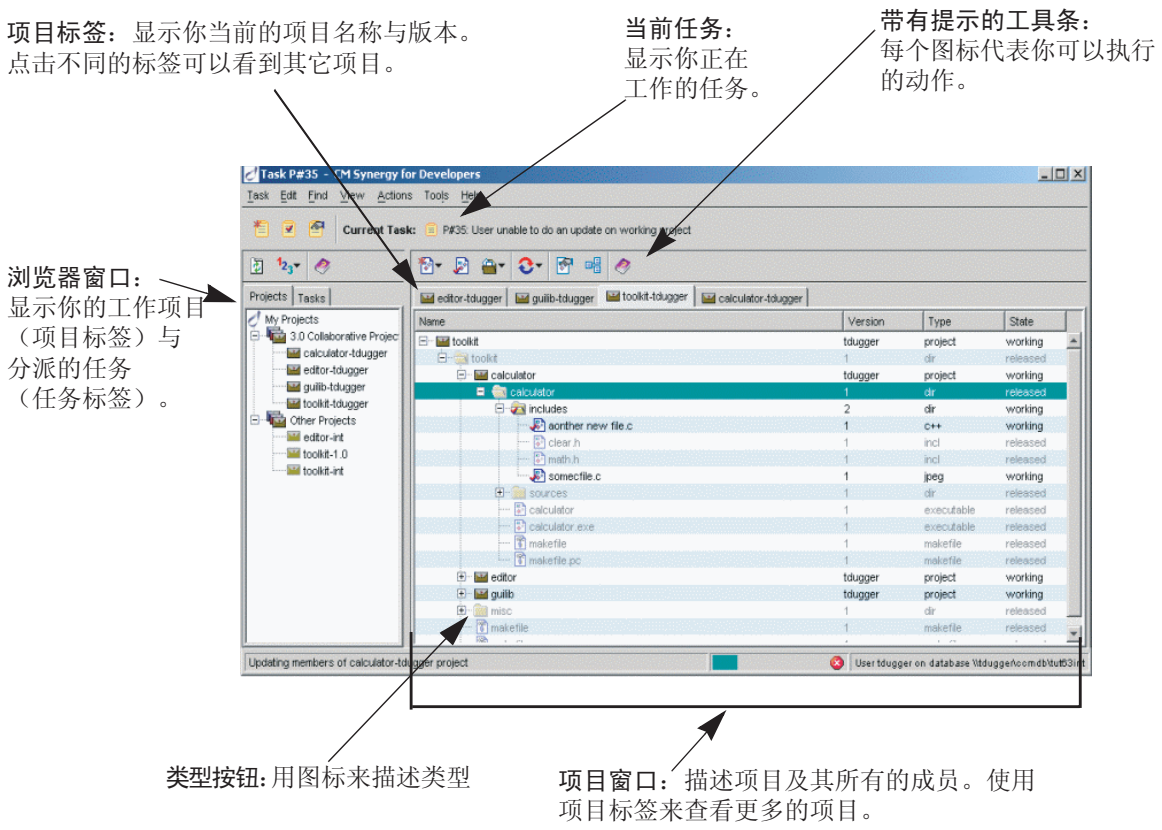

图 6 项目窗口

当前任务是针对 CM Synergy 开发人员界面进程的，而不只是针对项目窗口中所显示的项目。

历史对话框

CM Synergy 开发人员界面的历史对话框使你能够看到一个对象的*历史*，即一个文件、目录或项目的所有版本以及他们彼此之间的关系。

查看一个文件的历史，应该使用历史对话框。例如，用户 Hannah 最新检入了一个叫做 bufcolor.c-3 的对象版本。如果她想查看所有已存在的 csrc 版本，她打开历史对话框就会看到图 7 的视图。

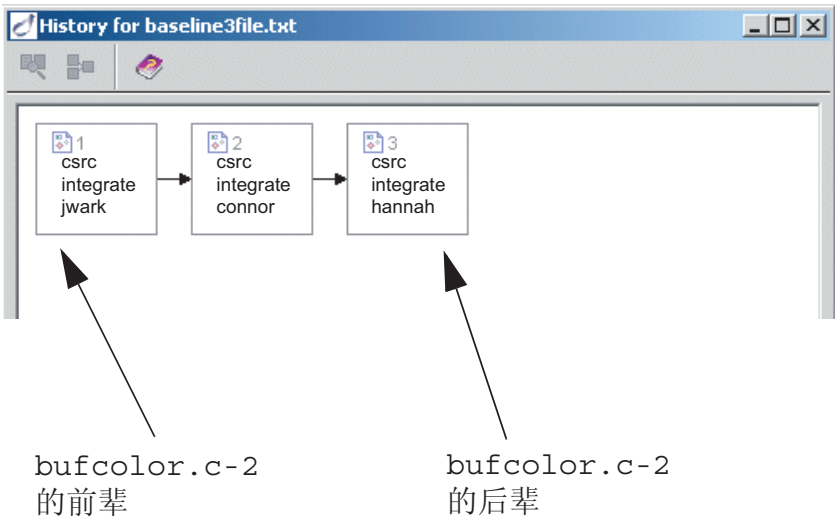

图 7 历史对话框

在命令行中，通过使用 ccm history 命令，用户 Hannah 可以查看 bufcolor.c 的历史。她可以在 DOS 或 UNIX 的命令行下输入下面的命令：

ccm history bufcolor.c

以下信息会显示在屏幕上：

Object:bufcolor.c-1 (csrc:2)

Owner: jwark

State: released

Created: Fri Jan 10 01:15:10 2003

Task:<void>

Comment:

Baseline of source code migrated in for sandbox project.

Predecessors:

Successors:

bufcolor.c-2:csrc:2

****

Object:bufcolor.c-2 (csrc:2)

Owner: connor

State: integrate

Created: Fri Jan 10 03:21:07 2003

Task:25

Comment:

Enhanced to support rgb settings.

Predecessors:

bufcolor.c-1:csrc:2

Successors:

bufcolor.c-3:csrc:2

****

Object:bufcolor.c-3 (csrc:2)

Owner: hannah

State: integrate

Created: Fri Jan 10 03:25:57 2003

Task:26

Comment:

Fixed problem with buffer handling.

Predecessors:

bufcolor.c-2:csrc:2

Successors:

****

分配对象版本与项目

当你检出一个对象时，你可以指定版本。如果你不指定，CM Synergy 就会自动为新对象分配一个版本。

以下章节将描述 CM Synergy 在缺省的方式下是怎样分配版本的。

源文件与目录对象

在缺省的情况下，在 CM Synergy 数据库中对于创建的每个源文件或目录对象，CM Synergy 从 1 开始创建版本的数字。随着每次检出对象，CM Synergy 就增加版本数。

项目与产品对象

根据你的角色与 CM Synergy 的数据库设置来自动分配项目与产品的版本。

当开发人员创建项目或产品对象时，它具有他个人自己的版本；这个版本永远都不会被作为一般意义上的使用。当编译经理创建一个项目或产品对象时，他的目的是为了使大家都能使用它。CM Synergy 根据你的角色与你的用户名来为项目及产品对象创建版本。

- 如果你是*开发人员*的角色，缺省情况下，CM Synergy 用你的用户名来检出版本。从 1 开始逐渐增加构成一个系列。例如，如果用户 Jeff 检出一个叫做 draw.obj-1 的产品对象，新的对象版本就将是 draw.obj-jeff。下一次 Jeff 检出这个产品对象时，新的对象版本就将是 draw.obj-jeff1。

  你可以通过在你的 ccm.ini 文件中设置*初始*选项来覆盖用户名。

- 如果你处在 *tester, build_mgr, type_developer,* 或 *ccm_admin* 角色，CM Synergy 就会

从 1 开始创建版本 (例如 draw.exe-1)。每一次这个对象被检出时，版本就会增加 (例如， draw.exe-2)。

CM Synergy 是怎样辨识并行版本的

当从一个单一的对象中检出两个或多个对象时，*并行版本*就产生了。为了确保每个可能的并行版本都具有唯一的版本号，CM Synergy 对并行版本使用下列的数字方案：初始.分支.索引。*初始*是指对象派生的起源版本。*分支*显示了检出的顺序—每个并行分支都从 1 开始被分配一个顺序数。*索引*是每个分支的计数器。

在图 8 所示的例子中，版本 1.2.1 表示这个版本是从版本 1（其初始版本）中检出的，它是第二个被检出的并行分支（因此是.2），并且它是此分支的第一个版本（即.1）. 版本 1.1.1.1.1 表示这个版本是从版本 1.1.1 中检出的，它是第一个被从 1.1.1 检出的并行版本，并且它是此分支的第一个版本。

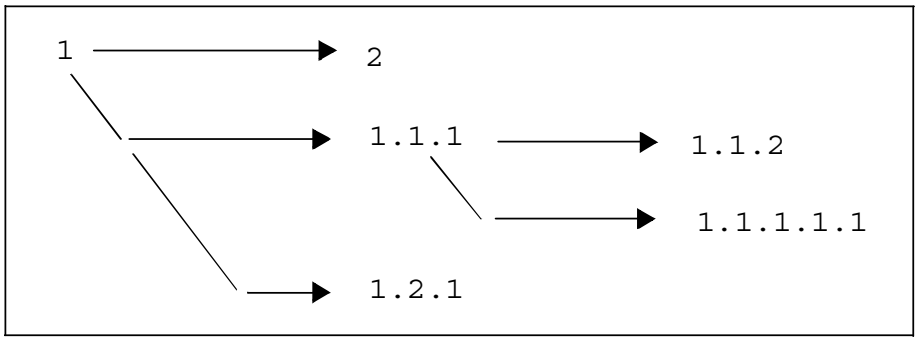

图 8 并行对象的版本

你可以修改一个并行对象的版本以使它对你或你的项目团队更有意义。如果你改变这个版本（对象必须可写），CM Synergy 在检出时会继续使用这个词或一组字母并在检出后的版本后面加上接下来的数字。例如，你有一个对象叫做 main.c，它有两个并行的分支，一个针对用户界面（GUI）而另一个针对命令行（CLI），你可以把分支分别命名为 main.c-gui1 与 main.c-cli1。当你从两者中检出新版本时，main.c-gui1 就将变成 main.c-gui2, 而 main.c-cli1 就将变成 main.c-cli2. 使用这些命名的约定，你一眼就可以看出那个版本是针对 GUI 的，那个版本是针对 CLI 的。

CM Synergy 的方法论

以下的章节是按顺序编写的； Telelogic 建议你按顺序进行阅读。

基于任务的方法论

*方法论*是用来管理软件的过程与策略。在 CM Synergy 中，方法论控制了从最初开发、测试、发布与维护的整个开发周期的软件流程。

在 CM Synergy 中，你可以使用几种不同的方法论。本章描述*基于任务的方法论*—CM Synergy 的缺省方法论—它使用任务作为基本的工作单元来跟踪变更到软件应用。一个任务代表一个逻辑变更。下面讨论基于任务的配置管理的好处。

- 基于任务的配置管理很直观。

  开发人员可以很自然地用逻辑变更来思维，并在头脑中会把每个变更对应到所需变更的特定文件。对于大多数的配置变更系统来说，开发人员必须记住检入他们变更的每个文件。基于任务的配置管理方法论可以通过自动跟踪所有相关的变更并一次就把他们全部检入来帮助开发人员按照他们的思维方式来工作。

- 基于任务的配置管理无需通过猜测所要发布的内容来创建发布。

  使用基于任务的配置管理，你可以通过基线加上一组任务的方法来配置你的应用。通过从最新的里程碑或发布并添加特定的修复或增强来创建一个发布是很直观的。

- 基于任务的配置管理会对有潜在冲突的文件发出警告。

  因为基于任务的配置管理记录了比非基于任务系统更多的关于文件之间关系的信息，基于任务的配置管理可以在测试发生之前在你的软件配置中发现冲突。在你更新你的配置时，CM Synergy 可以发现丢失的或部分丢失的任务。

- 基于任务的配置管理为你的发布提供了更多的信息。

  你可以使用任务描述这种有意义的方式来对所要发布的新功能进行罗列，而无需采用仅罗列源文件的方式。

- 基于任务的配置管理可以与变更请求集成在一起

  任务可以与用户提交的缺陷和增强报告相关联，这就在你的变更请求系统与实际的软件变更之间提供了一种紧密地集成。

用户与角色

CM Synergy 用户总是要工作在特定的*角色*之下，例如 *developer, writer*, 或 *tester*。角色是通过配置管理员在数据库层次上设置的。你的角色决定了你在数据库中的操作权限，并且在不同的数据库中这种权限可能会不同。例如，用户 Hannah 在 main_product 数据库中可能具有 *developer, tester* 与 *build_mgr* 的角色，但是在*集成*数据库中只具有 *developer* 的角色。在一个

CM Synergy 的进程中，你可以随时改变到你可以成为的角色。（在 CM Synergy 开发人员界面中，用户的角色永远被设置为 *developer*。）

在 CM Synergy 原始界面中，最常用的被预定义的角色是：

- ***developer*** –使用*开发人员*角色来开发与测试程序或其他类型的对象。
- ***writer*** – 使用*作者*角色来书写及测试文档、帮助系统和相关的对象。
- ***build_mgr*** –使用*编译经理*角色来配置与编译用于测试与发布的预备项目。
- ***dcm_mgr*** –使用*分布式配置管理员*角色来定义目标数据库与传输组，添加对象到传输组并生成传输包。
- ***assigner*** – 使用*分派员*角色来分派任务。
- ***component_developer*** – 使用*组件开发人员*角色来开发软件与发布组件（软件单元）并供其他人员使用。关于更多的信息，参见第 38 页的“基于组件的开发”。
- ***ccm_admin*** – 使用*配置管理员*角色来修改不可修改的对象或做管理变更（如加入新的用户到数据库）。*配置管理员*角色的用户可以对 CM Synergy 数据库中任何对象进行任何操作。这个角色只应该交给负责任的*配置管理*员，并且要尽少地使用。过多地使用了配置管理员角色就意味着你的流程需要有所调整。

因为本章是讨论开发方法论的，所以讨论主要局限在*开发人员*与*编译经理*的角色。

项目与工作流程

一个项目包含一组特定的成员对象并提供一个隔离的工作空间（关于项目的基本信息参见第 14 页中对“项目”的说明）。项目的不同版本可以被用于不同的用途。例如：

- 每个开发人员都有一个工作版本用于开发与测试他们正在进行的变更。
- 一个隔离的项目版本可以被用于收集最新完成的任务以用于集成测试。
- 一个项目版本可以被用于编译一组指定的变更以用于系统测试。
- 另一个项目版本可以被用来保存特定的配置作为发布或里程碑。所有这些用于不同用途的项目能使团队工作于同一个应用。项目及其对变更选择的设置方式定义了你的团队的工作流程。

CM Synergy 提供的缺省的工作流程包含以下阶段：

- 开发人员在他们的工作项目中开发与测试他们的变更。当他们完成一个任务时，这个任务就可以被包含在集成测试的*预备*项目中。当开发人员重新配置他们的项目时，他们不仅保留他们自己检出的版本，并且他们也能够获得集成测试中所通过的测试版本。
- 集成测试的*预备*项目会搜集目前所有已经完成了的任务。这些项目通常会被用于实施“每日编译与快速测试，”—一种最佳的软件开发实践。集成测试的目的是尽快地发现错

误。由编译经理来管理集成测试的*预备*项目。

- 当编译通过了集成测试时，编译经理就会创建一个基线。
- 系统测试的*预备*项目是通过编译特定的一组变更来进行深层测试所使用的项目。编译经理定义与更新一组变更来保证这个项目是与开发人员正在进行的变更所隔离的。单个的修改可以被加入、编译并重新测试直到项目满足团队的质量标准。系统测试的预备项目通常是为发布或里程碑而准备的。
- 在软件被发布或到达了一个里程碑时，项目可以通过设置基线或进行发布来保留配置。被发布的项目可以被作为新发布的基线。

图 9 展示了一个怎样使用项目来实施 CM Synergy 缺省工作流程的例子。箭头指示任务在项目中的流向。

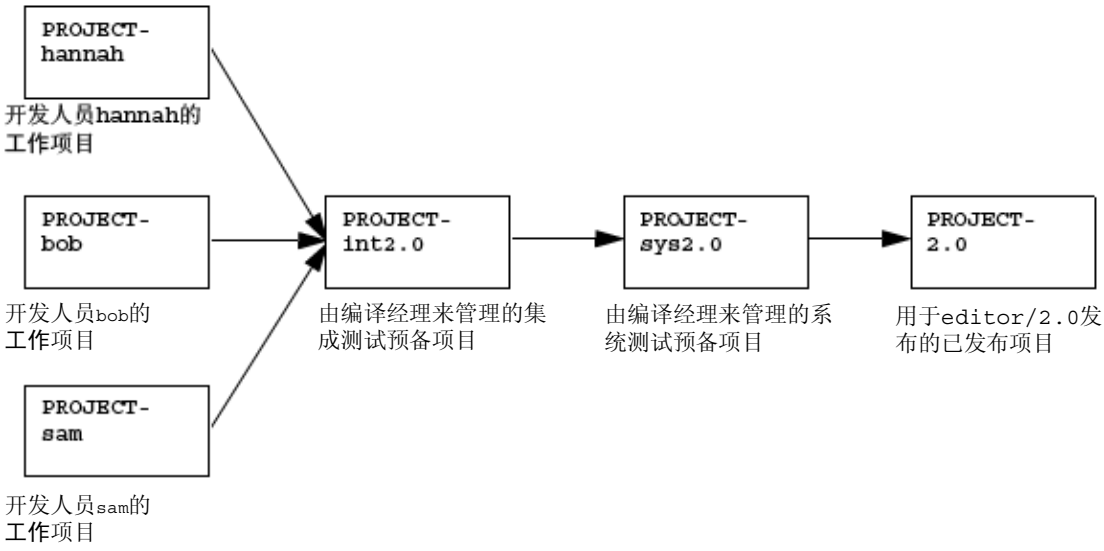

图 9. CM Synergy 缺省工作流程

CM Synergy 可以使你能够直接使用基于任务的方法论。但是，CM Synergy 的过程模型是灵活的，使你能够定制缺省的方法来适应你的团队的流程。关于定制方法的信息，参阅第 41 页的“定制你的团队的工作流程”的内容。

发布

在 CM Synergy 中，你总是要为特定的*发布*来工作。发布（有时叫做*分支*）是表明你的软件应用版本的一个标签。例如，你的软件的第一次发布可以是 editor/1.0，第二次发布可以是 editor/2.0 或 editor/1.1。

当你检出一个项目时，你应该指定它将使用的发布。同样，当你创建一个任务，你应该指定它将被包含在那个发布。发布很重要是因为 CM Synergy 用它来组织你的任务与项目并保证项目中所用的任务与发布的值相匹配。

CM Synergy 可以为你的软件应用保留发布。发布能够使你注明项目、任务及任务夹所针对的特定发布。它也能够帮助你跟踪哪个对象版本是为哪次发布所开发的。

只有处于*编译经理*角色的用户才能够创建或改变发布。你可以在浏览发布对话框或使用 ccm release 命令来看到它们。每个 CM Synergy 数据库具有它们自己的一组发布，虽然说你可以用分布式的变更管理 (DCM) 来在数据库之间转移它们。

一个典型发布的名字可以是下面的任何一种。表 2 举例显示了由编译经理所创建的发布名称。它由组件名称与组件发布组成。发布名称就是你所见的组合。

表 2. 发布名称的例子

发布名称	组件名称	组件发布版本
1.0		1.0
2.0		2.0
2.0_patch		2.0_patch
CM Synergy/6.3	CM Synergy	6.3
editor/2.0	editor	2.0
editor/2.1	editor	2.1

发布名称包含可选的组件名称与发布定义符及组件的发布。组件名称可以代表一个应用或组件的名称，例如 CM Synergy 或 editor。组件的发布版本标明了这个应用或组件的特定发布。

注意，组件名称不是发布必备的部分。在上表的第一行，1.0 组件名称没有组件，CM Synergy 把它作为空白处理。

一旦你检出一个对象，CM Synergy 会自动从缺省的任务中复制发布名称到新的对象。

项目用途

项目*用途*是一种设置用于指定一个项目的用法并把它与用于重新配置过程的一组规则相绑定。CM Synergy 提供下列预定义的项目用途：

表 3. CM Synergy 预定义的项目用途

用途	状态
隔离开发	*工作*
并行共同开发	*工作*
自定义开发	*工作*
集成测试	*预备*
系统测试	*预备*
共享	*共享*
可见	*可见*

隔离开发、**集成测试**与**系统测试**的用途是用于前面所描述的缺省的方法论的。**共享**、**可见**与**并行共同开发**的用途是针对使用各种能够使团队更紧密工作的标准方法的。**自定义开发**的用途是使开发人员能够指定基线与任务来用于他们自定义的项目。

开发人员使用最多的用途是**隔离开发**，**并行共同开发**与**自定义开发**。编译经理使用最多的用途是**集成测试**与**系统测试**。**共享**与**可见**的用途是对于需要对缺省的基于任务的方法有所改变的团队。

当你创建或检出一个项目时，你需要指定其项目用途，CM Synergy 会通过设置项目的重新配置属性来自动为那个用途对新项目进行设置。

另外，状态列显示了项目对这个用途在缺省情况下创建时的状态。例如，只有当项目的用途是可见时，项目在创建时才能处于可见的状态。在下面的用途下，能够创建*工作*状态的项目有：**隔离开发**，**并行共同开发**或**自定义开发**。状态也会保证当你重新配置时，项目会选取正确的成员。

重新配置属性

当重新配置一个项目时，项目会使用一组叫做*重新配置属性*的属性来自动决定选择哪些对象版本到你的工作空间中。重新配置属性与项目一起存储并包含一个基线加上任务列表与任务夹。

所有项目都具有重新配置属性。当你检出一个项目时，你选择的用途（如**隔离开发**、**集成测试**等）与项目的发布值就决定了项目的重新配置属性将怎样被设置。另一方面，你也可以手工设置你的重新配置属性。

当你进行重新配置时，CM Synergy 按照下列方式来更新你的项目：

1. CM Synergy 决定使用哪些任务。它评估在重新配置属性中的每个任务夹并加入其中的任务列表，同时添加你在重新配置属性中直接指定的任何任务。
2. CM Synergy 通过查看不存在于基线中的每个任务来计算的对象版本列表。这对象列表加上基线项目的成员就成为了你重新配置的候选对象列表。

3. CM Synergy 使用一组简单的规则来比较候选对象的属性与项目的属性的匹配程度并选取最佳的对象。

CM Synergy 的缺省工作流程

本节描述 CM Synergy 的缺省工作流程。

任务的使用

*任务*代表了你的应用软件的问题或是对应用软件功能的增强。因为任务组合了所有你对于特定问题或增强所修改的对象，你只需检入组合这些对象的任务而无需单独地检入每个对象。用这种方式，任务为用户作了很多工作。

在缺省情况下，任何 CM Synergy 的用户（开发人员、测试人员，编译经理，项目经理等）都可以创建一个任务。任务也可以根据客户或技术支持工程师提交的问题来生成与分派。

当某人创建了一个任务，如果他知道谁将会解决这个问题，那么他就可以马上*分派*这个任务。否则，任务就始终处在*已注册*的状态直到它被分派为止。任何具有分派角色的用户都可以把任务分派给他自己或另一个用户；创建任务的用户可以把任务分派给自己，而无需考虑他的角色。当一个用户分派一个任务时，他应该设置发布来指明将要包含这个任务的应用软件的版本。

任务被分派后，开发人员使用以下的过程：

1. 选择任务作为缺省任务。（在 CM Synergy 开发人员界面中，这叫做当前任务。）
   你可以选择任何分派给你的任务作为缺省的任务。
2. 为完成这个任务进行所有必须的变更。

   因为 CM Synergy 会自动把你所变更的所有对象与缺省任务关联起来，所以任何你对之有所操作 (例如，检出或添加对象) 的对象都与缺省任务相关联。

   进行单元测试以便你知道是否需要更多的修改。
3. 检入缺省的任务。

   当你检入任务时，CM Synergy 首先检入与之关联的对象，然后检入已完成的任务。处于*已完成*状态的任务可以被编译经理用于集成与系统测试。当已完成的任务通过了集成测试之后，编译经理会把它们共享给其他开发人员。

开发流程

每个开发人员所正在工作的项目都有一个此项目的*工作版本*。开发人员通过复制编译经理的集成测试*预备*项目来得到他们自己的*工作*项目。

通常情况下，开发人员并不检入*工作*项目。如果一个开发人员在其工作项目中进行并测试了一个变更，当他检入任务时，CM Synergy 会自动检入实施了变更的各个对象版本；开发人员并不检入项目本身。*工作*项目就象容器—它们可以在各个发布之间被重复使用。当开发人员重新配

置时，项目的内容每次都会变化。

当开发人员开始工作于一个新任务或准备重新配置他的项目来引进最新的变更时，他就会对项目进行重新配置。当需要时，他通过重新配置来使一个项目保持最新状态，并且对于项目的成员的每次变更，他都使用*工作*项目的同一个版本。

每个开发人员对使用*工作*项目来对他的变更进行单元测试而负责。开发人员应该重新配置他的项目，在检入任务之前重新测试他的变更来验证此变更与其他开发人员的最新变更的兼容性。当单元测试完成后，开发人员应该检入任务，并由此来检查所有对象的彼此依赖关系。这保证了所有必备的对象都可以用于编译经理的集成测试。

在缺省情况下，当一个开发人员重新配置他的*工作*项目时，CM Synergy 会收集对于当前发布的所有已分派给他自己的及已完成的任务，再加上用于此发布的已通过集成测试的最新的任务。对于正在开发 editor/2.0 发布的开发人员 Hannah 来说，她所工作的项目的重新配置属性应该如下设置：

- 项目的基线应该被设置成最新的里程碑版本或项目的已发布版本的项目，例如 1.0 版本。
- 项目包含叫做“Hannah 对于 editor/2.0 发布的已分派或完成的任务”的任务夹。这个任务夹使用查询来选择 Hannah 作为任务解决者的所有对于 editor/2.0 发布的任务。
- 项目包含叫做“对于 editor/2.0 发布的集成测试任务”的任务夹。这个任务夹包含所有对于 editor/2.0 发布的已通过集成测试的任务。这个任务夹不使用查询；在集成测试后，由编译经理来更新它。
- 项目包含叫做“对于 editor/2.0 发布的对于 Hannah 的其他任务”的任务夹。这个任务夹不使用查询；它通常是空的，但是如果开发人员想包含一个指定的从另一个开发人员而来的任务，他可以把它加入到这个任务夹。所有开发人员都有一个自己的对于其他任务的任务夹，但是，通常只有高级开发人员才使用它。

Hannah 的团队使用重新配置模板。当她重新配置她的项目时，会发生以下的动作：

1. 这个项目重新评估它应该使用那个基线项目。它检查模板来查找基线，然后检查这个基线来辨识这个基线版本是否就是其自身。如果在基线中存在那个项目的多个版本，它就比较平台属性值来选择与之匹配的项目。
2. CM Synergy 从上面描述的任务夹中收集所有任务（不包含已经存在于基线中的任务），并创建这些任务相关对象的列表。这些相关对象就将是 Hannah 的项目所选择的候选对象。
3. CM Synergy 从基线项目中收集所有对象。这些对象也被加入到候选对象列表中。
4. 对于每个在 Hannah 项目中的对象，CM Synergy 对候选对象进行评估。它比较候选对象并选择最匹配的对象。

只有当开发人员想要保存项目的当前状态时，他才需要检入项目。在这种情况下，他把他的项目检入到*检查点*状态。这能够使开发人员定期地保存项目版本用于与以后的项目进行比较。开发人员也能够把其他对象（如源文件与产品）检入到*检查点*状态。当不再需要处于*检查点*的版本时，开发人员可以销毁它。当开发人员*销毁*版本时，他也从数据库中删除了这些版本以及检查点版本的最近处于可写状态的后续版本。这可以节省磁盘空间并缩短版本的历史。

集成测试循环

集成测试循环的目的是在开发周期的早期尽可能地发现错误。前面提到，*预备项目*是编译经理为测试与发布而使用的预备软件。*集成测试预备项目*收集最新检入的变更并编译它们以便进行集成测试。因为集成测试*预备*项目包含许多最新的变更并且用户也正不断地检入新的对象，所以集成测试空间通常是不稳定的。这种不稳定是可以理解的，因为集成测试的目的就是发现错误。

在缺省情况下，当编译经理重新配置*预备*项目来编译它们用于集成测试时，CM Synergy 收集对于当前发布版本的所有已完成的任务。编译经理不希望包含开发人员已分派的任务是因为开发人员还在进行修复并且对于集成来说这些对象还没有准备好。

对于负责 editor/2.0 发布进行重新配置集成测试*预备*项目的编译经理来说，这些*预备*项目的重新配置属性应该如下设置：

- 每个项目的基线应该被设置成项目的最新的里程碑版本或已发布版本，例如 1.0 版本。
- 每个项目都包含叫做“对于 editor/2.0 发布的所有已完成的任务”的任务夹。这个任务夹使用查询来选择所有对于 editor/2.0 发布的处于完成状态的任务。

编译经理定期（通常在晚上）重新配置他的集成测试*预备*项目来编译用于集成测试的软件。当编译经理重新配置项目时，叫做“对于 editor/2.0 发布的所有已完成的任务”的文件夹就会使用其查询来从数据库中选择开发人员正在工作的 editor/2.0 发布的所有已完成的任务，同时使用与那些任务相关的对象版本来更新项目。然后，编译经理就可以编译应用软件了。

如果编译没有成功，编译经理可以做两件事：他可以创建任务并把这些任务分派给引起编译失败的开发人员，或者他可以告诉这些引起编译失败的开发人员应该修复这些错误，然后由每个开发人员来创建他们自己的任务。

一旦编译经理成功地编译了这个产品，他就会创建一个新的基线。当开发人员重新配置他的项目时，就可以保证他们能够获得最新通过测试的变更。

通常情况下，编译经理并不需要检入对于集成测试的预备项目。这些项目就像容器—它们可以在各个发布之间被重复使用。当编译经理重新配置并引入开发人员最新已完成的任务时，项目的内容每次都会变化。

系统测试循环

一旦开发团队达到一定程度上的稳定编译或里程碑时，编译经理通常会为质量保证 (QA) 团队

编译一个可以安装的软件来用于系统测试。系统测试的目的是为一个里程碑（例如一个发布）准备应用软件。*系统测试预备项目*代表了所有开发人员已经准备好了的对于系统测试的工作；它包含了用于系统测试与发布所准备的文件、目录及产品版本。

因为开发人员在开发与测试他们的变更时还要继续检入任务，集成测试*预备*项目会继续引入开发人员最新的变更。编译经理需要准备更稳定的系统测试预备项目作为与开发人员最新检入的变更的一个隔离的空间。

当编译经理重新配置这个*预备*项目来为系统测试进行编译的时候，他可以指定将用于测试的一个精确的任务列表。通过管理这个将包含在应用软件中的精确的任务列表，团队能够修复、编译与重新测试软件直到它能够满足团队的质量标准。系统测试循环通常是迭代式的—团队可以在软件达到期望的质量标准前多次编译、测试、修复与添加任务。

对于负责 editor/2.0 发布进行重新配置对于系统测试*预备*项目的编译经理来说，这些预备项目的重新配置属性应该如下设置：

- 每个项目的基线应该被设置成项目的最新的里程碑版本或已发布版本，例如 1.0 版本。
- 每个项目都包含叫做“对于 editor/2.0 发布的所有系统测试的任务”的任务夹。这个任务夹不使用查询是因为编译经理为了更好的控制而采用手工更新其内容的方式。在系统测试循环的初始阶段，项目经理最初可以通过复制“对于 editor/2.0 发布的集成测试任务”的任务夹的内容来充实这个任务夹，但是在测试开始之后，他通过手工添加所需得到确认的任务来维护这个任务夹。

当编译经理充实了准备用于新的系统测试循环的任务夹之后，系统测试通过下列方式来迭代：

1. 编译经理重新配置与编译对于系统测试的*预备*项目。
2. 质量保证团队测试已完成的软件。
3. 质量保证团队评审发现的任何问题并决定它们中的哪些应该在这个循环中被修复。
4. 质量保证团队为在步骤 3 中得到批准的问题创建任务并把它们分派给开发人员。
5. 在开发人员完成了批准的任务后，编译经理就把它们加入到“系统测试任务夹”中。这个过程又从第一步重新开始。
6. 重复 1-5 步直到没有更多的问题需要修复。
7. 创建基线。

发布系统测试预备项目

一旦软件测试成功，编译经理就会马上把用于编译软件的系统测试*预备*项目转换成发布状态，并创建基线以便用于未来的发布。处于*已发布*状态的所有对象都是不可改变的，这就确保了软件将被保存下来并且在日后需要的时候可以被重新创建。

使用 CM Synergy，你能够只用一步就可以把项目与其成员进行发布。

在你发布项目之前，你或许想要把项目的版本改成一个更有意义的名称或数字，例如 2.0 或 rel6b。

从已发布的项目来创建基线并发布这个基线。

编译经理也可以选择使任务夹处于不可修改的状态。因为已发布的项目是不可以被重新配置的，所以这一步是不需要的，但是对于将来的配置（例如重用任务夹）可能会有用。

为下次发布做准备

一旦编译经理发布了一个项目，它就可以被用作下次发布的基线。以下是为下次发布做准备的步骤：

- 编译经理添加新的发布名称到发布列表中并为新的发布设置重新配置模板。
- 开发人员更新他们的工作项目来使用新的发布名称以便他们可以重新使用这些项目。
- 编译经理更新集成测试*预备*项目以便使用新的发布名称。因为集成测试*预备*项目没有被检入，所以编译经理只需重新配置这个项目就可以重新使用新的发布的集成测试项目。
- 编译经理最终会从基线中检出系统测试*预备*项目的新版本，因为最新的项目已经被发布了并且不再能够被使用了。

小结

我们现在已经覆盖了整个开发周期，从开发人员在他们的工作项目中完成任务、到编译经理收集与测试任务、到最终的项目发布以及为下次发布建立基线。图 10 总结了这些项目是怎样被用于实现工作流程的。箭头显示了任务是怎样在各种项目间流动的。考虑到空间的限制，在图中的一些开发人员的任务夹名称被缩写了。图中的关键点是：

- 每个开发人员的项目都包括叫做“对于 editor/2.0 发布的集成测试任务”的任务夹。在已完成的任务通过了集成测试后，编译经理要更新这个任务夹。每个开发人员也有一个个人的任务夹（例如，“Hannah 的对于 editor/2.0 发布的已分派的或已完成的任务”）来收集他或她的对于指定发布的任务。每个开发人员也有一个叫做“其他任务”的任务夹来用于添加其他开发人员的任务以便进行早期的集成测试。
- 编译经理使用集成与系统测试预备项目来进行测试。集成测试预备项目使用一个叫做“对于 editor/2.0 发布的所有已经完成的任务”的任务夹并使用查询来收集这些任务。因为用于系统测试的任务夹的内容只应该包含被批准的变更，编译经理会手工添加任务到这个叫做“对于 editor/2.0 发布系统测试任务”的任务夹。
- 测试完成之后，项目就被检入到发布状态，并且编译经理会为下一个里程碑或发布创建一个基线。

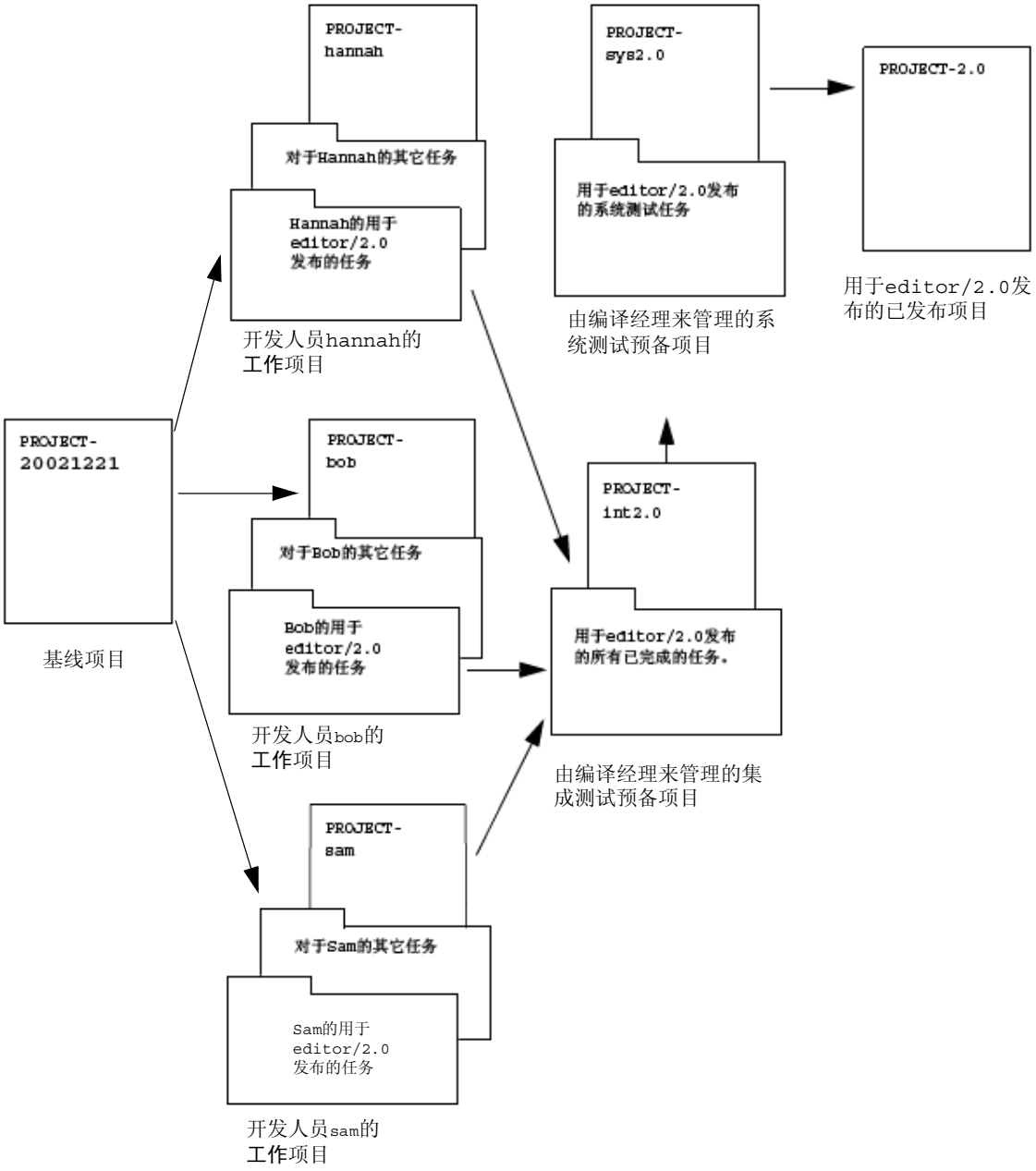

图 10. 项目与工作流程

并行开发

并行开发是同时开发一个对象的一个以上的版本。在缺省情况下，CM Synergy 允许你并行开发任何对象类型（例如，源文件、库等）。理解并行对象版本必须具有一些特征（例如，属性）来彼此区分这一点很重要，这可以使 CM Synergy 能够在一个项目中选择正确的版本。在重新配置时，这种特征决定了对象版本将怎样被评估。用来区分并行对象版本的属性叫做*并行开发属性*。

CM Synergy 支持下列类型的并行开发：

- 当多个开发人员从同一个对象检出他们自己的工作版本时就会发生*并行共同开发*。每个开发人员将很有可能工作于代码的不同部分。一旦他们完成了工作，代码了两个版本就应该被归并。
- 当多个开发人员工作于对于不同平台（通常叫做变体）的同一个对象的不同版本时就会发生*并行变体开发*（也叫做*并行平台开发*）。对象的不同版本（例如一个版本用于 Windows 而另一个版本用于 UNIX）通常不需要归并。
- 当组织需要同时生产软件产品的多个发布版本时就会发生*并行发布开发*。这种情况的一个例子可能是不同的开发人员工作于下一个发布、一个当前发布的补丁与一个维护发布，所有这些都是基于当前发布的。并行发布通常会在其中一个发布完成后进行归并。例如，当用于当前发布的补丁完成后，它就被归并到维护发布中并且当维护发布完成后，它就会被归并到下一个发布中。

以下的段落描述 CM Synergy 是怎样管理这些类型的并行开发的。

并行共同开发

CM Synergy 使用*状态*与*所有者*的属性来管理并行共同开发版本。当一个对象版本处于*工作*状态时，只有所有者才能把它包含在他的项目中。重新配置过程把*工作状态*对象版本的所有者与进行重新配置项目的所有者进行比较，以确保选择正确的版本。一旦两个并行工作版本（及其相关任务）被检入，他们就必须被归并，因为没有单一的版本可以包含所有的变更。如果版本被检入但是没有被归并，包含两个任务的项目的更新会选择最新创建的任务，并且项目会显示一个并行的冲突。

并行平台的开发

平台属性可以辨识一个项目或对象是对于那一个特定平台的。如果你为多个平台来编译你的软件，你就需要对每个平台指定一个平台项目，并且每个项目版本都需要设定其平台属性。例如，如果你编译一个 snap 项目用于 UNIX 与 Windows，你就需要两个版本：一个版本的平台属性被设置成 unix，另一个的属性被设置成 win。当你使用基线、任务夹与任务来重新配置你的项目时，就会由你的重新配置属性来选择候选对象。如果你的候选对象包含用于不同平台的并行版本，你必须设置候选对象的平台属性以便重新配置操作可以选择与项目的平台属性相匹配的版本。

例如，对于含有 line.c 的 snap 项目。这个项目版本的平台值被设置成了 win32，指明了它是为 32 位 Windows 平台而编译的。line.c 对象有两个版本：一个被标识为 win32，另一个被标识为 unix。因为它们两个都被包含在项目的任务夹或任务中，它们都可能成为候选对象，选择的规则就会选取与其平台匹配的候选对象。

并行发布开发

前面提到*发布*属性可以辨识一个项目或任务所针对的特定发布。如果你正在为多个发布开发软件，你的每个发布都需要一个项目版本，并且每个项目版本的*发布*属性应该做相应的设置。在重新配置时，这可以保证每个项目选择与其匹配的子项目与任务。

你的任务必须用当前的*发布*属性来标识以便它们可以被与其匹配的项目所选择。

注意：你必须创建独立的任务来进行变更以便应用于多个发布。

基于组件的开发

组件是一个或多个库或可执行文件以及表明怎样使用它的支持文件（例如头文件、帮助与兼容性与依赖关系的信息、硬件或软件需求、设计信息、测试用例等等）。组件也可以包含源文件。

在 CM Synergy 中，组件可以由单个文件或项目来代表。

因为 CM Synergy 中的文件版本是可以重用的，它们可以在一个项目中被创建或编译而在另一个项目中被使用。例如，叫做 ccmscci.dll 的库文件可能是在包含其源文件的 ccmscci 项目中被编译的，但是同一个文件也可以作为成员被包含在 visual_studio_integration 与 va_java_integration 的两个项目中。（每个项目可以包含这个文件的一个不同的版本，这取决于其自身的需要。）

另外，你可以选择创建一个新的项目来包含几个文件以便作为组件进行一起发布。例如，你可以创建一个新项目叫做 ccmserver_ext（与 ccmserver 项目相似，但是供外部使用），这个项目可以包含文件 ccmserver.jar, ccmserver.properties 与 ccmserver.html。你也可以拥有 ccmserver_ext 项目的多个版本，每个都包含这个组件的含有兼容文件的被发表的版本。

你也许会考虑把整个源文件项目作为一个组件，如果组件的用户需要查看或修改代码的话。

因为组件可以由其他组件来构成，你甚至可以用整个项目层次来代表一个组件。

管理组件

在 CM Synergy 中，一个*发布*指明了你的应用软件的发布标签，例如 CM Synergy/6.3 或 ChangeSynergy/4.3。发布与版本很类似，但是是对于整个软件产品的。一个发布可以代表你已经交付（或已发布）的一个产品，或者它可以代表你正在开发的一个发布。每个项目与任务通常会被标明是对于哪个发布的。

每个组件应该具有其自身的发布分支。例如，对于一个开发 GUI 库及两个应用（一个计算器与一个编辑器）的团队来说，他们设置的发布名称可以类似下面的例子。

表 4. 用于不同组件的发布表的例子

组件	发布分支
GUI 库	gui_lib/1.1, gui_lib/1.2, gui_lib/1.3
编辑器应用	editor/1.0, editor/2.0, editor/2.1, editor/3.0, editor/4.0
计算器应用	calc/1.0, calc/2.0

为每个组件设置一个不同的发布分支的目的是降低不同组件间的偶合程度以便它们可以彼此独立。这可以使组件拥有不同的发布时间表，并可以使开发团队使用他们所选择的不同流程。

发表组件

通过发布（或检入到不可修改的状态）代表组件的文件或项目就可以发表组件。CM Synergy 是基于角色的系统，不同的用户具有不同的角色来进行他们的工作。角色提供了安全性并且提供了与这种类型用户相应的工作流程。CM Synergy 提供了几种角色，但是开发人员与项目经理所使用的典型角色包括：

- *developer*，用于开发软件的用户
- *component_developer*， 用于开发软件并发表组件供其他用户使用的用户
- *build_mgr*， 用于为集成或系统测试而收集和编译软件的用户。

如果组件是由使用严格测试程序的结构化的团队来开发的，组件的发表通常由编译经理来负责。相反地说，如果组件是由一个小的、非正式的团队或个人开发的，开发人员就需要具有 *component_developer* 的角色才能发表组件。

在发表组件时，发表者（编译经理或组件开发者）会选择与之相关的任务以便其他人可以引用它。在组件被发表时，如果一个团队想要被提醒，他们可以定义触发器。

引用组件

组件可以与任务相关。任务可以使组件的使用者指定需要使用的组件版本。通常情况下，组件的每个版本都会有不同的任务与之关联。

发表组件的人可以创建一个任务并使之与相应的文件或项目相关。组件的使用者也可以创建一个任务并把它与他需要的文件相关。一个给定的组件可以与几个任务相关。

由使用者来创建任务的好处是他可以对组件的新版本进行单元测试、进行任何需要的额外变更，并使之与同一个任务相关。这可以把对新组件版本的所有的逻辑变更保留在一起，而其余的团队人员不会受到新组件的影响。（注意：一个给定组件的版本可以与多个任务相关：一个与开发它的团队相关，另一个可以与每个使用它的团队相关。）

CM Synergy 也能够使你把一组任务放到任务夹中。这意味着你可以把多组兼容组件版本收集到任务夹中以便一起使用。这种任务夹可以被不同的想要重用这些组件的用户应用共享使用。

过程模式

基于组件的开发可以确保在降低成本、提高质量与提高用户满意度的同时加速软件解决方案的交付。这种实践横扫了软件开发工业；但是，当今的大多数的工具与技术忽略了通过团队在一起管理、发表与共享组件的关键问题。

CM Synergy 为在团队或个人之间共享组件提供了强大的框架与流程。它使你能够管理、发表、重用与分发组件，同时其内建的工作流程能够支持并且也会促进应用软件开发的最佳实践。

CM Synergy 可以以一种灵活的方式来支持基于组件的开发，它提供了多种流程模式以供选择。这种过程模式在《*Managing Component-Based Development with CM Synergy*》的技术白皮书中进行了论述。白皮书可以在 Telelogic 的网址下 http://support.telelogic.com/synergy/kb 找到。

高级的话题

本章描述以下的高级话题：

- 定制你的团队的工作流程
- 管理项目与产品

本章的内容主要是针对编译经理、CM 管理员与想要对 CM Synergy 的灵活性有所了解的评估者的。开发人员可以略过本章。

定制你的团队的工作流程

“CM Synergy 的方法论”描述了 CM Synergy 的缺省方法论，这种方法论所实施的工作流程可以支持许多软件开发的最佳实践并且对很多团队都很有用。但是意识到对于流程来说一种方法并不能满足所有的需求，CM Synergy 的设计考虑到了使每个团队都能够配置合适的软件开发流程。

采用 CM Synergy，你能够用以下的方式来配置你的开发与测试流程：

- 添加或删除测试阶段。

例如，除了在前面章节中提到的集成测试与系统测试阶段之外，你也可以添加阶段来支持测试循环，例如性能测试、回归测试等。

- 为开发人员配置隔离程度。

在有些团队中，开发人员需要在任务通过集成测试之前彼此隔离，这在缺省的方法论中已经描述过。但是，在另一些团队，开发人员可能想要获得其他开发人员检入的任务，或者甚至在任务被检入之前就获得彼此的变更。

- 为每个发布使用不同的流程

例如，一个工作于新功能发布的团队可能只需要一个测试阶段—集成测试阶段，而在此团队中的开发人员想要尽快共享彼此已完成的变更。同时，工作于同一个应用软件的维护发布的另一个团队想要一个非常隔离的环境以使开发人员在变更通过几个测试循环后才能看到彼此的变更。

- 在发布过程中自动变更团队的流程。

编译经理可以添加或删除测试阶段，可以在发布过程中配置开发人员的隔离程度并自动地把对这些过程的改变应用到整个团队中。

你可以通过下面的内容来详细了解通过用项目用途与重新配置模板来定制 CM Synergy 的工作流程。

添加或删除测试阶段

在 CM Synergy 的缺省工作流程中，有两个测试阶段：集成测试与系统测试（见第 28 页图 9对工作流程的总结）。（如果我们包括单元测试的话，实际上有三个测试阶段。单元测试发生在每个开发人员的工作项目中。）这些测试阶段对应于 CM Synergy 的预定义的项目用途列表。

你可以通过更新项目用途列表（关于用途，请参见第 29 页的“项目用途”）来添加或删除测试阶段。例如，在你的工作流程中添加一个回归测试阶段，你应该添加一个叫做回归测试的新的项目用途。

在从你的项目用途列表中删除一个测试阶段之前，你应该考虑是否在将来会需要它。用途列表应该是你的团队所需的所有用途的超集；单个的团队可以选择他们将会使用的用途。

配置开发人员的隔离程度

CM Synergy 能够使编译经理配置开发人员的隔离程度。团队可以选择在其他人员的变更经过测试后来共享变更，也可以选择在变更被检入后甚至在被检入前来共享变更。

开发人员间的隔离程度取决于在他们重新配置项目时所选择的对象。重新配置模板是一种模式，它通过自动设置项目的重新配置属性来定义按某种用途所创建的项目的更新方式。例如，CM Synergy 提供了以下已经定义好了的重新配置模板。

- 一个对于*工作*项目的模板，它指明每个开发人员的工作项目应该选择这个开发人员自己检出的对象版本加上最新通过集成测试的版本。
- 一个对于集成测试*预备*项目的模板用来选择与最新检入任务相关的对象。
- 一个对于系统测试预备项目的模板用来选择由编译经理所定义的指定的任务列表。

这些重新配置模板对应于**隔离开发**、**集成测试**与**系统测试**的项目用途。通常情况下，对于每个项目用途都有一个缺省的重新配置模板。

为了配置开发人员的隔离程度，编译经理可以对**隔离开发**项目修改重新配置模板。通过创建重新配置模板，他也能够配置任何他所添加的测试阶段。例如，如果编译经理为**回归测试**创建了一个新的项目用途，他也应该创建相应的重新配置模板来定义在缺省情况下**回归测试**项目应该怎样被更新。

对于每次发布使用不同的流程

为了更加灵活，你可以为每次发布设置不同的重新配置模板，以便使工作于不同发布的团队能够使用不同的流程。

例如，对于工作于一个叫做 snap/3.0 的新功能发布的团队来说，他们将只使用一个测试阶段—集成测试，而团队中的开发人员需要尽快共享彼此已完成的变更。这个团队需要两个重新配置模板：一个用于**并行共同开发**用途，一个用于**集成测试**用途。用于 snap/3.0 发布的**并行共同开发**用途的重新配置模板将设置开发人员的*工作*项目以便在这些项目被更新时能够包括所有已完成

的任务。

同时，对于工作于一个叫做 snap/2.1 的同一个应用软件的维护发布的团队来说，这个团队需要一个非常隔离的环境，在哪里开发人员对彼此的变更不可见直到这些变更全部通过了三个不同的测试循环：**集成测试**，**系统测试**与**回归测试**。这个团队需要四个重新配置模板，每个对应于他们将使用的各自用途：**隔离开发**、**集成测试**、**系统测试**与**回归测试**。（注意，在这个例子中，他们添加了一个新的用途来用于**回归测试**，因为在缺省的情况下，CM Synergy 没有提供这个用途。）snap/2.1 发布的对于隔离开发用途的重新配置模板将设置开发人员的*工作*项目以便这些项目在更新时能够只包括被批准的任务。

对于每个发布，你必须设置各自的重新配置模板来用于各自的用途。下表显示了多个重新配置模板在 CM Synergy 中是怎样同时存在的例子：

表 5. 对于用途与发布的重新配置模板

项目用途	snap/2.0 发布	snap/2.1 发布	snap/3.0 发布
隔离开发	X	X	X
集成测试	X	X	X
系统测试	X	X	不适合
回归测试	不适合	X	不适合

标有“X”符号的条目表明工作于一个发布的团队能够为他们的用途（或阶段）定义重新配置模板。例如，snap/2.1 发布团队使用所有四个阶段，而 snap/3.0 的发布团队只使用两个阶段。因为你能够为单个发布定义重新配置模板，所以工作于不同发布的团队能够使用不同的流程。

项目用途与发布值的组合可以成为重新配置模板的单一标识。例如，用于 snap/2.0 发布的集成测试的重新配置模板可以用 **2.0:集成**来引用。

相反地，一个项目可以用它的用途与发布值来单一地标识它所使用的重新配置模板。当你检出一个项目并设置它的用途为集成测试而且它的发布值为 snap/2.0 时，你的新项目就将基于 **2.0:集成**的重新配置模板。

如果你改变了项目的发布或用途值，或它的状态（这会影响它的用途），它就会自动转换到使用对于新的发布/用途的重新配置模板。因此，它将使用对于发布与用途而定义的任何过程。

图 11 显示了项目与重新配置模板之间的关系。

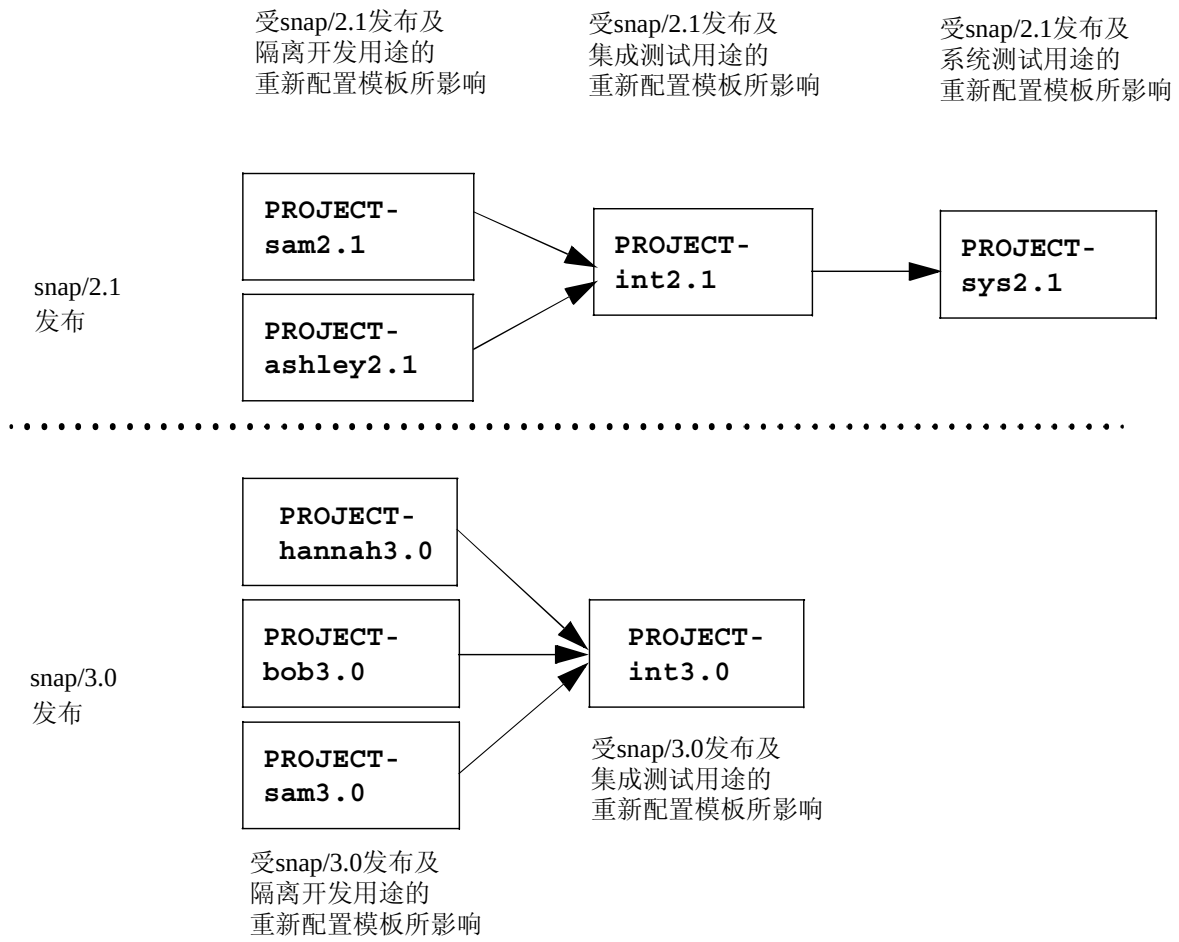

图 11 项目与重新配置模板之间的关系。

对于一个特定的发布来说，开发人员可能有几个不同的工作项目。所有这些项目都会受到对于那个发布的隔离开发的重新配置模板的影响。

更新一个团队的流程

你可以在发布过程中通过修改发布的重新配置模板来改变一个团队的流程。例如，在某一点上开发人员会需要在他们的项目中从包含彼此完成的任务转换到包含通过集成测试的任务。编译经理能够通过改变这个发布的重新配置模板来自动对所有开发人员的流程进行改变。

更多的关于重新配置模板的信息

当你创建了一个项目，它的重新配置模板就定义了在更新时它的行为将会怎样。这是通过定义新项目的重新配置模板属性来实现的。

在前面提到过一个项目的重新配置属性包含一个基线加上任务列表与任务夹（参见第 30 页的“重新配置属性”）。重新配置模板只是以一种更通用的方式定义了项目的重新配置属性。

基线

基线是一组不可修改的由编译经理保存的具有特定名称、发布与用途的项目。这组项目定义了重新配置操作的起点。例如，编译经理可以创建叫做 Integration Build 20020913 的基线，它包括项目 toolkit-int_20020913, calculator-int_20020913 等等。

重新配置模板能够指定其项目使用特定的基线；引用这个重新配置模板的项目使用基线来辨识在更新时使用哪个基线项目。例如，如果对于当前发布的隔离开发配置模板指定了应该使用 Integration Build 20020913 基线，开发人员的 calculator-bob 项目就将选用 calculator-int_20020913 作为其基线项目。

任务夹与任务夹模板

项目的重新配置属性包含指定的任务夹列表。例如 calculator-hannah2.0 项目的重新配置属性可能包括下列任务夹：

- *Hannah 的对于 editor/2.0 发布的已分派或已完成的任务*
- *对于 editor/2.0 的所有已完成的任务*

就像重新配置属性一样，重新配置模板可以包含指定的任务夹列表。例如，对于 editor/2.0 发布的隔离开发项目的重新配置模板可能包含“对于 editor/2.0 的所有已完成的任务”的任务夹。这可能意味着对于 editor/2.0 发布的所有开发人员的项目都应该包含这个任务夹。

另外，重新配置模板能够包含任务夹模板。就像重新配置模板对于项目是一种模式一样，任务夹模板对于任务夹来说也是一种模式。任务夹模板具有一组可以应用于每个任务夹的属性：任务夹名称、谁能够写这个任务夹、谁能使用它、是通过手工还是通过查询来进行更新，并且查询将被用于选择任务。任务夹模板的属性可以使用关键词而不是实际值。关键词是将来被替换为实际值的模式。例如，关键词%owner 将被替代为所有者的名字。当一个用户检出或使用重新配置模板来创建一个项目时，CM Synergy 为每个包含在重新配置模板中的任务夹模板创建任务夹。（如果以前已经创建了恰当的任务夹，那么它们将被重新使用。）

当任务夹模板创建了任务夹，在模板中的名称与查询字符串的任何关键词就会被转换成恰当的值。例如，一个叫做“All %owner's Completed Tasks for Release %release”的任务夹模板会在 Hannah 检出对于 editor/2.0 发布的项目时创建叫做“ All Hannah's Completed Tasks for Release editor/2.0”的任务夹。

从模板中所创建的每个任务夹会继续基于这个模板。如果编译经理改变了模板，所有从这个模板中所创建的任务夹就会根据那些变更立即进行更新。更多关于使用项目用途与重新配置模板来定制你的团队的工作流程的信息请参阅《Build Manager’s Guide》， 它可以在 Telelogic 文档网页上找到 (http://support.telelogic.com/synergy/docs)。

管理项目与产品

本节讨论 CM Synergy 是怎样管理项目与产品的。关于项目与产品的基本信息请参看第 14 页的“项目”与第 17 页的“编译，产品与编译文件”。

在 CM Synergy 中，产品与项目并不使用与源文件对象一样的生命周期。开发人员可以检入源文件对象来把它们共享给编译经理及其他用户，但是他们不能检入他们自己的项目或产品。有这种限制的原因是为了更加小心地控制用于测试与发布准备的编译环境。开发人员可以通过多种方式来改变他们的环境；例如，开发人员可以不使用最新的版本来进行编译，或使用 CM Synergy 控制之外的库的复制版本来编译。如果开发人员能够检入在他自己的环境中所编译的整个项目或产品，就很有可能把这些未加控制的变更带入到正式的软件发布中。通过使开发人员在他们自己的工作空间进行工作并只能检入他们的源文件，CM Synergy 就可以把测试环境与没有被控制的变更隔离开来。

虽然开发人员不能把*工作*项目或产品检入到*集成*状态，但是他们能够把它们检入到*检查点*状态来保存产品或项目的副本以便将来引用。处于*检查点*状态的对象还是属于个人的；除了所有者之外，没有人能够使用它们。它们会受到保护以防变更，但是在不需要时可以被删除。

自动创建的任务

因为开发人员并不检入项目与产品，项目与产品也不与开发人员工作与检入的任务相关联。所以，CM Synergy 会自动创建叫做自动任务的特殊任务来帮助管理项目与产品。

当开发人员检出一个项目时，CM Synergy 会自动把它与开发人员的对于那个发布的自动项目任务相关联。同样，当开发人员检出一个产品时，CM Synergy 会自动把它与开发人员的对于那个发布的自动产品任务相关联。每个开发人员的工作项目的重新配置属性包含他的对于那个发布的自动项目任务与自动产品任务，以确保在他重新配置时他的工作项目会被选择。

同样，当编译经理检出一个项目或产品时，它会自动把编译经理用于那个用途与发布的自动项目任务或自动产品任务与之相关联，确保在预备项目的结构层次中的恰当的项目会被选择。

自动任务被设计成是透明的；他们的目的是自动管理你的项目与产品。但是，如果你查看项目的重新配置属性或如果你在 CM Synergy 数据库中查询任务时，你能够看到自动项目。

就像项目与产品一样，自动任务通常不被检入。

共享产品与项目

对于团队来说，共享软件是很常见的。例如，一个团队可能工作于几个库而另一个团队使用这些库来开发可执行的程序。因为第二个团队不需要改变库，他们想要使用通过了集成测试的最新的库的版本。通过在每个成功的测试循环后检入产品并且把恰当的自动任务添加到开发人员的重新配置属性中，编译经理能够使团队的流程自动化。编译经理通过更新对于那个发布的隔离开发重新配置模板就能够设置这个流程。当更新他们的项目的成员时，工作于这些产品版本的开发

人员将保留他们自己的版本，但是没有工作版本的开发人员将在更新成员时获得最新检入的版本。

如果一个团队需要一组产品，但是不需要包含这些产品源文件的项目的工作版本，编译经理能够设置叫做**外部项目**的特殊项目，外部项目只包含产品以及需要读取它们的其他文件（例如，头文件）。这个团队可以在这些新项目中引用这些产品而不必引用包含源文件的原始项目。

外部项目具有如下好处:

- 他们可以节省开发人员的时间，因为开发人员无需复制与更新含有他没有改变源代码的项目。
- 他们可以帮助提高代码的模块化程度及隐藏信息，这两者对软件开发来说都可称得上是最佳的实践。

不利的因素是编译经理必须做额外的工作来管理外部项目，并且工作于底层与顶层项目的开发人员必须管理他们自己的外部项目的工作版本。

再次强调，编译经理可以通过在每个成功的测试循环后检入外部项目并且把恰当的自动任务添加到开发人员的重新配置属性中来使流程自动化。没有外部项目的检出版本的开发人员在他们更新成员时会得到最新的检入版本，并提供给他们通过了集成测试的最新的产品版本。