

StarTeam 最佳实践

说明：本文基于 StarTeam5.3 写成

目录

| | |
|-------------------------------|----|
| 引言..... | 2 |
| 一、StarTeam 概览..... | 2 |
| 二、StarTeam 模型..... | 3 |
| (一) StarTeam 库..... | 3 |
| (二) C/S 体系结构..... | 3 |
| (三) 面向项目..... | 4 |
| (四) 项..... | 5 |
| (五) 项目..... | 6 |
| (六) 视图..... | 6 |
| (七) 文件夹..... | 7 |
| (八) 视图标签..... | 7 |
| (九) 分支视图..... | 7 |
| (十) 合并视图..... | 8 |
| (十一) 链接..... | 9 |
| (十二) 文件状态..... | 9 |
| 三、使用案例..... | 10 |
| I. 发布开发与主开发..... | 10 |
| 范例场景 1..... | 10 |
| 使用一个新视图来解决场景 1..... | 10 |
| 使用标签来解决场景 1..... | 10 |
| 范例场景 2..... | 12 |
| 使用提升状态来解决场景 2..... | 12 |
| II. Bug 修理..... | 14 |
| 使用修理或维护分支来解决场景 3..... | 14 |
| III. 独立的产品开发..... | 15 |
| 范例场景 4..... | 15 |
| 使用一个新的分支来解决场景 4..... | 15 |
| IV. 公共组件库的开发..... | 15 |
| 设置构建过程..... | 16 |
| 按构建标签进行检出..... | 16 |
| 按提升状态检出..... | 16 |
| V. 配置标识报告..... | 17 |
| 创建过滤..... | 17 |
| 显示已修改的文件并生成报告..... | 19 |
| VI. 阶段..... | 20 |
| 阶段 1：为你的阶段创建提升状态..... | 21 |
| 阶段2：为测试阶段创建一个视图..... | 21 |
| 阶段3：为产品化阶段创建一个视图..... | 22 |
| 四、关于StarTeam 5.3的一些提示与技巧..... | 22 |

| | |
|--------------------------------|----|
| 数据库..... | 22 |
| 项目结构..... | 22 |
| 文件夹..... | 23 |
| 备份..... | 23 |
| 安全..... | 23 |
| (一)、从StarTeam Server中进行操作..... | 24 |
| (二)、从StarTeam中进行操作..... | 24 |
| 在StarTeam中维护文件夹和项..... | 25 |
| 标签..... | 24 |
| 视图标签..... | 25 |
| 修订标签..... | 26 |
| 提升状态..... | 26 |
| 变更请求..... | 27 |
| 主题..... | 27 |
| 任务..... | 27 |

引言

当前，越来越多的软件公司已经认识到软件配置管理工具的重要性，它是软件工程领域中至关重要的一个环节，也是Borland公司所宣扬的ALM理念的重要一环，你可以看到，它贯穿了整个ALM周期。StarTeam是一个革命性的软件配置管理工具，使用它可以节省你公司的时间与金钱，帮助维护代码的一致性。自从使用StarTeam至今，我们摸索及总结出了一些关于安装、配置和使用StarTeam的最佳方式，我们很高兴将它们与你分享。本文包括：

1、StarTeam 概览

对StarTeam产品线的一般描述和介绍。

2、StarTeam 模型

对StarTeam中使用的对象模型进行概览，并对面向项目的系统（如StarTeam）和面向文件的系统（如PVCS,VSS和MKS）之间的差异作了比较。

3、软件开发过程

软件开发基本过程概览以及StarTeam如何支持这些过程。

4、能力成熟度模型（CMM）

一些用来适应、支持组织的CMM需要的技术。

5、使用案例

一些使用场景的应对办法。

6、技巧

我们摸索出来的一些有用的技巧。

由于时间仓促，本文必定存在很多不足之处，欢迎你将你的真实范例、体验、评论和建议发送给我们，以便让我改进本文档并促进交流与合作。请发送email至：Michael@OOChina.Org。

声明：未经本人许可，不得以任何形式转载本文内容。

一、StarTeam 概览

StarTeam产品线由StarTeam Server和它的客户端组成，客户端包括：StarTeam、StarDisk、WebEdition和跨平台客户端等。管理员在安装StarTeam服务器时，可以选择不同的数据库来进行服务器配置，数据库包括：MSDE、Microsoft SQL Server、IBM DB2、Informix、Sybase SQL Server、或Oracle（目前在StarTeam5.3版中，由于ORACLE数据库中存在的

内存泄漏问题，所以暂时不予支持(目前仅支持SQL SERVER和MSDE)，DB2也有类似的情况，但只要问题一解决，就能够支持它们了，详见StarTeam5.3的【Release Notes】)。用户可以是位于某个站点或广泛地分布于全球，它们通过上面的客户端之一访问服务器。

StarTeam是一个Windows应用程序，提供了一个直观的GUI显示项目、视图、文件夹和文件等等。分页的面板使得导航和部署变得容易，无论你是在bug修订(变更请求)上工作，还是工作于产品讨论(主题)、工作分配(任务)或开发代码(文件)之上。StarTeam可以与当今许多流行的IDE进行集成，例如：Microsoft's Visual Studio.Net、Jbuilder、C#Builder、Delphi和Oracle。StarTeam可以与PVCS和SourceSafe协同工作，从而允许你转换已存在的SCM项目到StarTeam中。它也提供了一个命令行的接口。

StarDisk可以让用户通过一个虚拟的StarDisk驱动器和TCP/IP协议访问文件修订。StarDisk与Windows的集成，提供了对StarTeam的透明访问。

跨平台客户端使得可以在支持Java版本1.4或更高的平台上使用命令行接口，这可以使得UNIX用户也可以访问StarTeam。

WebEdition通过标准的浏览器方式访问项目库。WebEdition允许用户将文件检入、检出StarTeam、PVCS或VSS库，同时也可以创建、编辑和报告变更请求，还能参与团队讨论。

使用StarTeam SDK还可以创建定制化的客户端。

StarTeam可以适应所有类型的商业需要。

二、StarTeam 模型

为了使得在你的软件开发过程中对StarTeam的使用达到最佳效果，你需要熟悉StarTeam术语。这些术语一开始看上去可能比较陌生，但很快你就会发现StarTeam模型要比上一代的SCM工具更适合你的商业实践。同时，StarTeam模型所具有的弹性可以让你通过一个单一的、紧密集成的系统实现最终管理你的所有信息资产，源代码和文档。

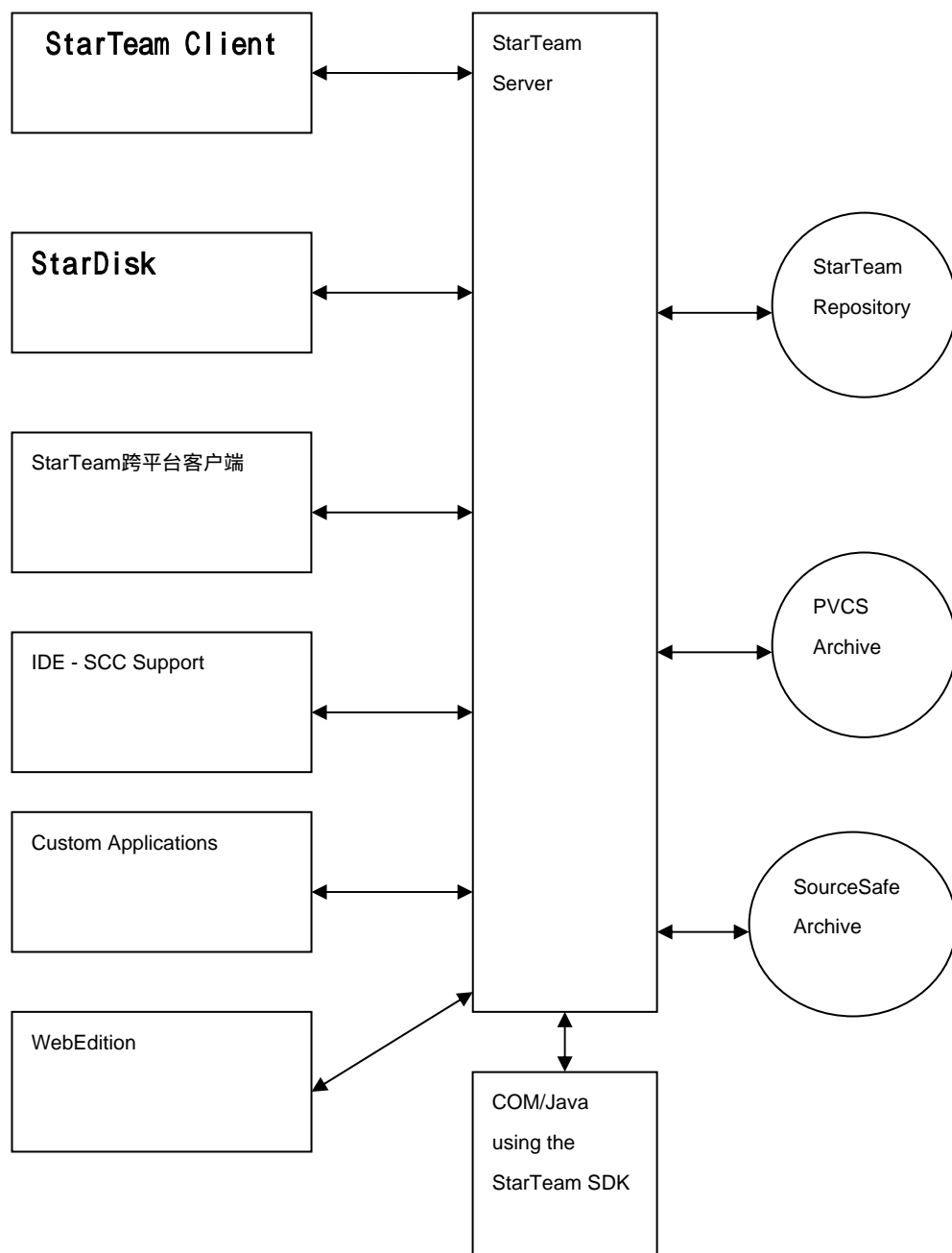
(一) StarTeam 库

StarTeam系统的中心是StarTeam库，它通过StarTeam Server维护。这个库是一个面向对象的数据存储库，支持对象版本化，链接和配置。任何对象，称为一个StarTeam项，存储在库中，具有它的历史记录，因此该项的前面的状态可以被检索并恢复。StarTeam项可以链接到库中的其他项，因此可以维护不同信息资产之间的关系，并将其用于你的工作过程之中。配置工作就是通过StarTeam提供的库服务执行多个项的创建、维护和恢复工作。

(二) C/S 体系结构

对StarTeam库的访问是通过StarTeam Server进行的，这意味着你的归档文件是完全收到保护的。其他某些产品如PVCS和SourceSafe需要以共享文件的方式才能实现归档库被相关人员访问到，这可能会使得这些归档和它们存储的信息资产同时也会受到计算机病毒的攻击或心怀不满的员工攻击。而使用StarTeam，访问这些归档库的唯一途径是StarTeam Server。所有的StarTeam客户端，不管它是StarTeamWindow GUI、命令行接口、IDE集成、StarDisk或者是使用StarTeam SDK建立的定制应用程序，与StarTeam Server 的通讯都是使用TCP/IP协议。StarTeam，作为Windows平台下的应用程序，也可以使用NetBEUI、IPX/SPX 或命名管道协议。由于StarTeam已经为Internet使用作了优化，远程用户可以将数据以压缩和加密的方式来访问StarTeam 库。考察StarTeam 的C/S体系结构时的一个最后考虑是StarTeam可以让你选择使用何种数据库，你可以选择MSDE、Oracle、Microsoft SQLServer、Sybase SQL Server、Informix和IBM DB2等等所有你的DBA所熟悉的工业标准的数据库。从一开始，你就可以挑选适合你的公司标准的数据库来管理你的信息资产。

图1：StarTeam 客户机/服务器体系结构



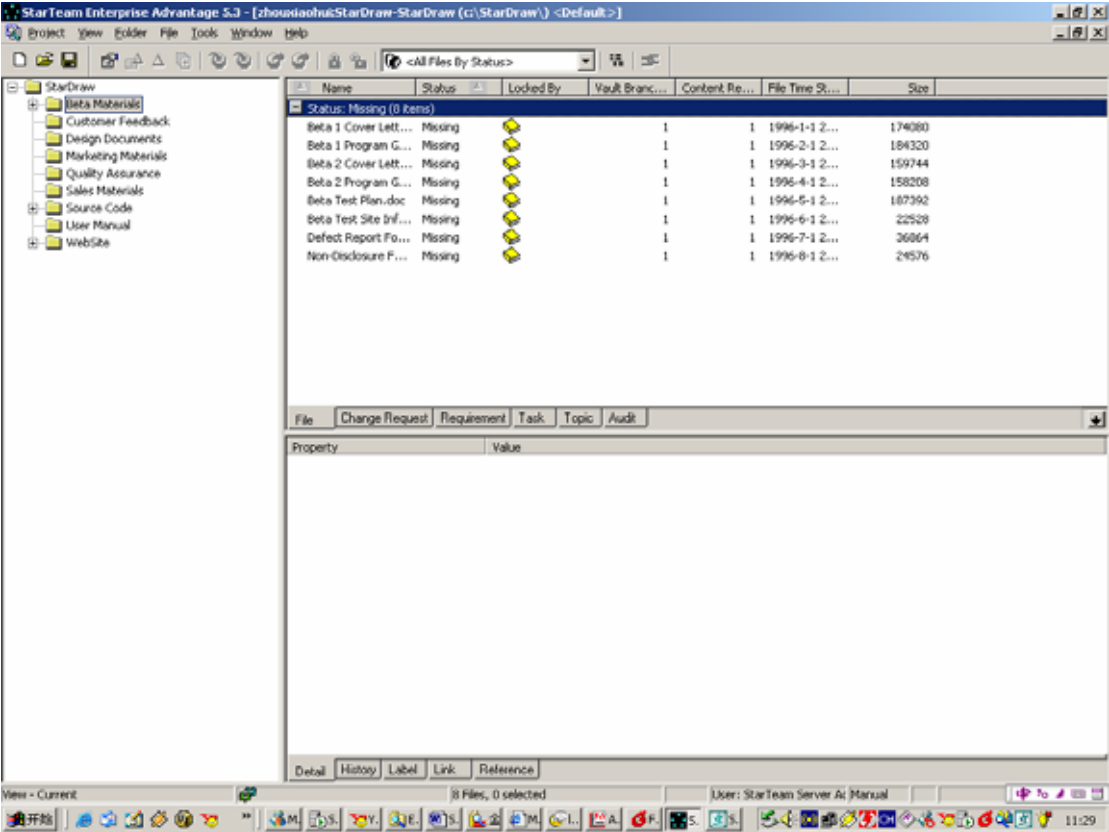
（三）面向项目

旧的SCM应用程序如PVCS和SourceSafe，是直接面向单个文件的。它们称为面向文件的版本控制系统。添加到系统中的每个文件具有它的版本号，存储在一个特定的归档文件中，它们之间的一对一映射与构建应用时的文件放置的位置是无关的。某些产品，如PVCS，并不跟踪记录文件需要检出的目录，而这一信息对正确地重建历史配置文件是必须的。

StarTeam采用面向项目的方法。在这一方法中，源代码和文档文件只是作为组成整个项目的特定项类型。除了具

有旧式产品所具有的面向文件的版本控制特性以外，StarTeam还支持对你的项目所需要的其他项进行版本控制，如变更请求、主题、任务、需求和存储这些项的文件夹结构。面向项目的系统还可以让用户根据他们的角色或项目的即时工作需要以不同的方式查看这些项。面向项目的方法是面向文件方法产品中实现特性的超集。

图2：StarTeam是一个面向项目的SCM工具



（四）项

StarTeam模型使用项，如文件、需求、变更请求、主题、任务和审计条目。大多数常用的项是可以版本化的，就是说，StarTeam存储了项的修订历史并允许你查看和比较不同修订的内容。

项也可以被分支，就是说，它们可以由其它项（那些项就成为了它们的祖先）派生出来。

项可能会有几个完全不同的修订历史，而这些修订历史具有共同的祖先。在文本文件情况下，分支项可以与派生出它的原始项进行合并。例如：为新操作系统开发的产品可以基于为第一个操作系统开发的文件为基础开始进行。

分支的概念在文档管理系统中并不多见。然而，这一能力对软件配置管理来说则是基础。开发员经常需要在保持原有开发路径的同时作出或大或小的变更。

StarTeam的协作性的框架体系结构支持多种类型的项，并可以根据客户的需要开发和添加更多的项。下表列出了StarTeam的当前版本所支持的项的类型：

表1：StarTeam 项类型

| 项类型 | 是否可版本化 | 是否可分支 |
|------|--------|-------|
| 文件 | 是 | 是 |
| 需求 | 是 | 否 |
| 变更请求 | 是 | 是 |
| 任务 | 是 | 否 |
| 主题 | 是 | 否 |

(五) 项目

StarTeam 使用项目、视图和文件夹来组织存储在StarTeam库中的项。一个StarTeam项目可以认为是紧密相关的视图的集合，每个视图代表一个来自库中的项的配置，可以支持在同一代码上的不同开发阶段。文件夹将项分为组，例如：你可能想要检出某个文件夹下的所有文件以工作于具有特定特性的产品上。对位于不同项目中的项并没有限制，只要项在同一个库中，它们就可以在任何视图间移动或共享，而不管项和视图是位于哪个项目中。

项目提供了一个组织的附加层次，它为视图提供了一个层次结构，同时也提供了在项目级分配访问权限的机会。项目如何使用取决于你。

你可能会为你公司生产的每个产品建立一个项目；或者取决于你构建产品的方式不同，你可能更愿意为产品的每个主要组件创建一个项目。为每个产品组件建立一个单独的项目提供了更多的弹性，因为这样一来每个组件可以被容易地标签化，分支化，并通过它自己的提升模型序列来运转。

(六) 视图

当你打开一个StarTeam项目时，你可以选择默认（或主）视图或者选择另外一个视图。项目的默认视图通常包含用于主要开发的配置。其他视图可以派生于这个视图，也就是说是以它为基础创建出来并具有不同的行为。被选中的视图代表了特定配置下的项的集合。

视图通常具有象下面这样的命名：基线、4.0 维护、4.0中国版、5.0 新开发。它们代表项的配置，对基于同一代码基础上的不同开发基线提供了支持。视图可以被比较和合并。例如：你可能想要将【4.0 维护】视图和【5.0 新开发】视图中的文件最终合并到视图【基线】中。

你可以通过创建和使用视图达到：

1、动态显示你的项目中的源代码和文档的变更。

这是项目中当从【View】菜单中选择【Select Configuration】命令后当【Current Configuration】选项被选中时，默认（主）视图的典型使用。这一动态视图显示了所有项的变化，可以用于协同开发。

2、引用原始视图中的项的子集。

它们通常称为引用视图。新视图中所作的任何改变也会改变原始视图中的相同项。这是因为子视图包含对原始视图中的原始项的引用，并且当变更发生时不会产生分支行为。通常引用视图具有如下的命名：【开发视图】或【文档视图】，只显示合适的项给相应的人，如开发员或文档员。

3、只读、基于原始视图特定状态的视图。

这通常是为了方便的需要，以便产品发布中的项的修订可以容易地进行定位。例如：一个【4.1发布】视图可以用于在将来重建4.1版本的产品，或者是允许想要购买你的源代码的公司在签订一个临时协议后查看源代码。

4、允许在新视图对项进行分支

这一视图可以用来修改特定视图状态下的项，而不会影响主开发。它通常通过创建和维护一个维护基线来完成。

视图的一个重要特性是你重新配置它，以显示视图在某个更早的时刻点、或特定的视图标签、或与视图相关联的提升状态时的项。使用视图菜单的【Select Configuration】命令回滚视图。回滚视图是只读的，显示项的精确状态，并且不再允许对它们作出改变。

提示：使用【View】菜单的【Select Configuration】命令可以定位截至特定时间检入的文件修订和变更请求的状态，以及需求、主题和任务。

(七) 文件夹

每一个StarTeam视图包含一个文件夹层次，用来组织它的项。文件夹反映了视图代表的配置的逻辑组织结构。文件夹通常具有如下这样的命名：源代码、计划、用户手册。它们根据谁需要访问哪些项或者是文件之间的紧密相关性对项进行分组，而文件夹可以被组织为任何层次结构（通常遵循文件被检出时的工作文件夹的结构）。

文件夹在你需要创建共享项的不同配置时也是有用的。你可以在视图之间或视图内部共享文件夹、文件、变更请

求、任务和主题，只要这些视图使用同一个服务器配置。文件夹被共享后，两个视图的用户就都可以访问它的内容了，包括子文件夹及其内容。

共享文件夹的设置是设置视图的一个重要部分。例如：假设公司的所有产品都不同程度的使用了公司的公共库，虽然这些库不是由某个产品的开发人员来维护，但该产品是基于这个库中源代码的某个版本完成的，并且必须与之一起编译。因此，这些库文件夹应该被共享给该产品的视图。

使用【Ctrl+Drag】来共享文件夹或项从一个位置到另一个位置。通过共享，你创建了一个对原始文件夹或项的引用。除非被共享文件夹或项的行为被设置为【branch on change】，所有对它的改变将同时修改原始文件夹或项。

被共享文件夹或项的配置（浮动、基于标签、某个提升状态或某个时刻点）初始在两个视图中是同一的。然而，它们可以被分别修改，这意味着共享项在每个视图中可能会有极大的差异，所以在这么作之前请确信对共享有深刻的理解。

被共享的文件夹或项将失去它们在先前视图中的所具有的任何标签。标签不能从一个视图移动到另一个视图。

（八）视图标签

StarTeam视图的另一个特性是视图标签。视图标签用来标识视图中包含项的特定修订的静态配置。当你创建视图标签时，它为视图保存了一个时间戳。视图标签为你保存了它创建时的动态视图的静态快照。

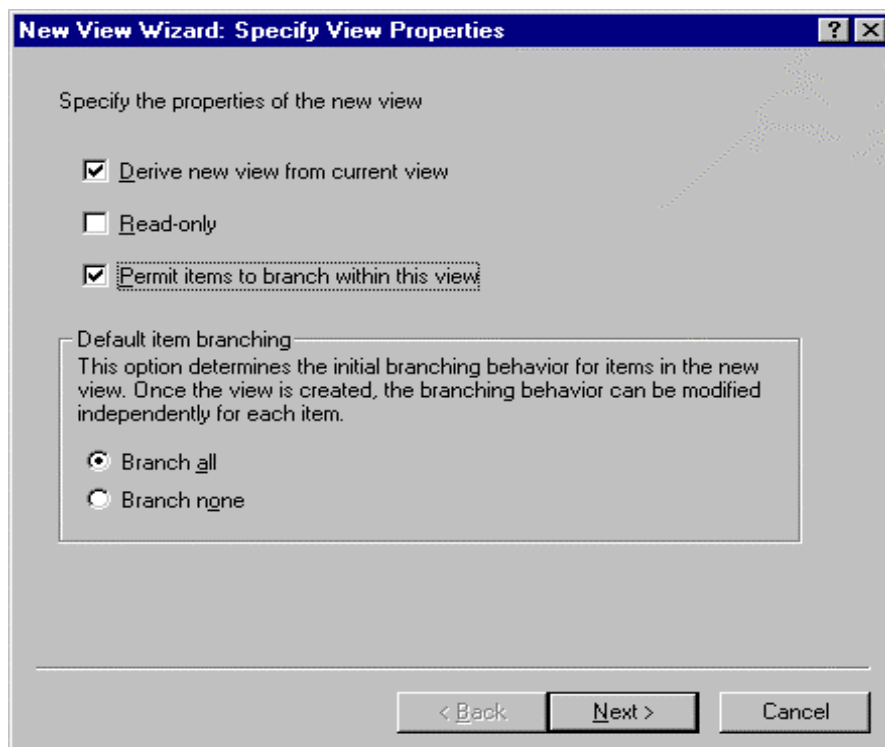
可以通过在标签面板中拖拽标签从项的一个修订到另一个修订来改变与视图标签相关联的项的修订。通常，一个视图标签会包含少量的标签变更，而大多数项修订是由它的时间戳所标识的。

提示：使用视图标签来指示开发里程碑，如每日构建。这可以让你在后来通过使用【View】菜单的【Select Configuration】命令或从命令行使用【CFGL（使用特定标签配置视图）】选项来返回到特定修订的精确配置。

（九）分支视图

软件开发中的一个常见操作是创建一个拥有基于先前状态配置的分支项的新的配置。它通常在用户希望执行对先前构建的系统的维护，但又不希望影响当前的开发时发生。StarTeam通过分支视图对这一活动进行支持。

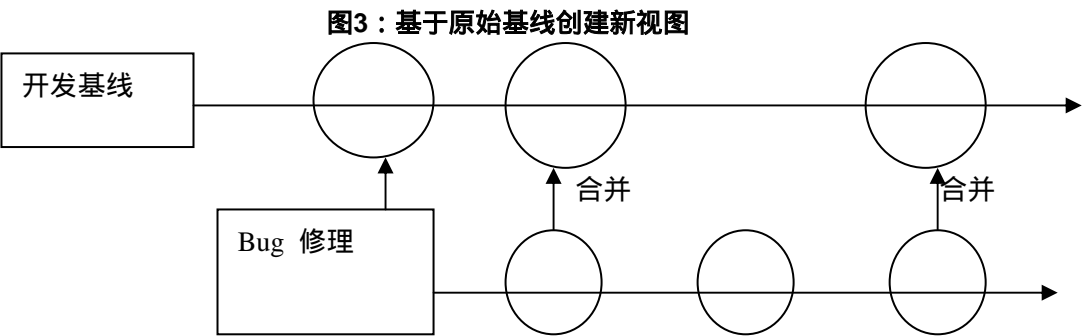
通过从【View】菜单中选择【New...】并选中【Permit Items To Branch Within This View】来创建分支视图。选择这一选项将使得新视图具有不同和独立的视图命名空间且新视图中的项能够被分支。你可以选择让每个项一旦发生改变就产生分支或者你可以推迟这一决定并有选择性地在新的视图中选择要分支的项。



提示：StarTeam的面向项目的方法允许你通过【Branch on Change】选项决定整个配置的分支行为。与面向文件的系统不同，它需要你在每个你想要分支的文件上分别指出，而StarTeam允许你为特定的视图根据想要使用的行为或为该视图作出的工作流程来指出应该发生分支。

对StarTeam模型不熟悉的用户经常会困惑于老视图中的视图标签没有在新视图中发现的事实。这通常是因为他们熟悉面向文件的系统和修订标签的缘故，在这些系统中，修订标签在特定文件归档的所有分支中是同一的。而在象StarTeam的面向项目的系统中，每个配置空间，由一个允许分支的视图所代表，也必须具有一个唯一的视图标签命名空间。这是因为当你创建一个允许分支的新视图时，视图发生了分支。此外，每个视图仅呈现被该视图引用的项的分支历史，而不是该项的贯穿不同分支的整个历史。这使得新视图成为项的独立配置，因此，在原始视图中发现的视图标签不会存在于新视图中。

提示：你无须在每次你需要分支某个项时都创建一个新视图。通过将项从一个文件夹共享(Ctrl+Drag)到另一个文件夹，然后设置行为（Behavior）选项为【Branch On Change】，你就实现了在同一个视图内创建了一个项的分支。这给了你一个在老的版本控制系统如SourceSafe中发现的相同的基于文件的分支能力。



| | |
|--------------------|---|
| Current | 工作站上的文件与视图中的对应文件的顶端修订相同。 |
| Out of Date | 工作站上文件与视图中的对应文件的旧修订相同。 |
| Modified | 自从从视图中检出以来，工作站文件已经被修改了，但在视图中没有发现此文件的更新的修订。 |
| Merge | 自从从视图中检出以来，工作站文件已经被修改了，并且在视图中存在有此文件的更新的修订。 |
| Missing | 工作站上没有发现视图中的此文件。 |
| Not in View | 视图中没有发现工作站中的对应文件。 |
| Unknown | 此文件没有从这个视图中检出的记录，但是在视图中存在一个与对应工作文件夹下文件同名的文件。使用【Update Status】命令让StarTeam去将工作站上的文件与视图中的文件的某个版本匹配，并提供一个准确的状态。 |

当你更新文件的状态时，StarTeam比较工作文件与你检出的修订及和顶端（最近）修订(即三方比较)。例如：文件列表可能说某个文件为【Current】状态，但可能已经有某个人检入了它的一个拷贝，因此你的真实状态应该为【Out Of Date】。

更新文件状态与更新文件是不一样的。例如：假设某个文件不在你的工作文件夹下，更新状态操作将会让你知道该文件的状态为【Missing】。它并不会为了使得状态不再为【Missing】而为你检出该文件。毕竟，你可能并不想该文件检出到你的硬盘上。通常来说，使用文件的状态来确定文件是否应该被检入、检出、加入或忽略。一旦你熟悉了文件的状态后，你就可以使用它来：

- 1、检入文件，如果它的状态为【Out Of Date】、【Missing】或【Merge】的话。
- 2、检出文件，如果它的状态为【Modified】或【Merge】。
- 3、将文件加入到StarTeam，如果它的状态为【Not In View】。
- 4、在检出期间，让你提前知道你需要合并工作站文件。
- 5、运行【Visual Diff】来比较状态为【Out Of Date】的工作文件与顶端修订。这可以让你在检出该顶端修订之前查看由其他团队成员对该文件所作的变更。
- 6、通过回滚到某个特定的视图标签来从某个更早的构建中检出所有的文件(使用【View ->Select Configuration...】，然后返回到当前配置，通过比较检出的文件与它们的顶端修订来查看自从该构建被创建以来所作的每个修改)。
- 7、通过增量回滚视图并查找状态为【Modified】的文件来找出引起大问题的小变更。使用【History】来确定文件是什么时候被改变的。

三、使用案例

Borland意识到，每个公司的工作环境都具有独特的结构、策略、过程和软件开发生命周期。我们的目标是在StarTeam中建立起你的SCM方法学的模型。下面是一些使用案例的简要介绍和使用StarTeam建立SCM模型的最佳方式。

I.发布开发与主开发

你无须在使用StarTeam进行发布活动期间冻结基线。StarTeam可以以多种方式处理发布开发和主开发，下面这些场景你可能会比较熟悉：

范例场景 1

在开发过程中，Joyce和Mike都有为发布1所作的需要检入到StarTeam中的代码变更。Joyce在Mike两周之前完成了为发布1所作的工作。当Mike还在工作于他的变更上时，Joyce需要在不影响发布1的情况下继续为下次发布工作于她的文件上。你如何保证Joyce检入StarTeam的项不会停止对主基线的继续开发？并且，当Mike完成他的变更时，这些文件又如何合并到发布1中？

使用一个新视图来解决场景 1

处理这个场景的一个方法师创建一个新的、但可能是临时的视图，让Joyce检入她所作的另外的变更。以后在适当的时候，这些变更就可以合并回基线中。

使用标签来解决场景 1

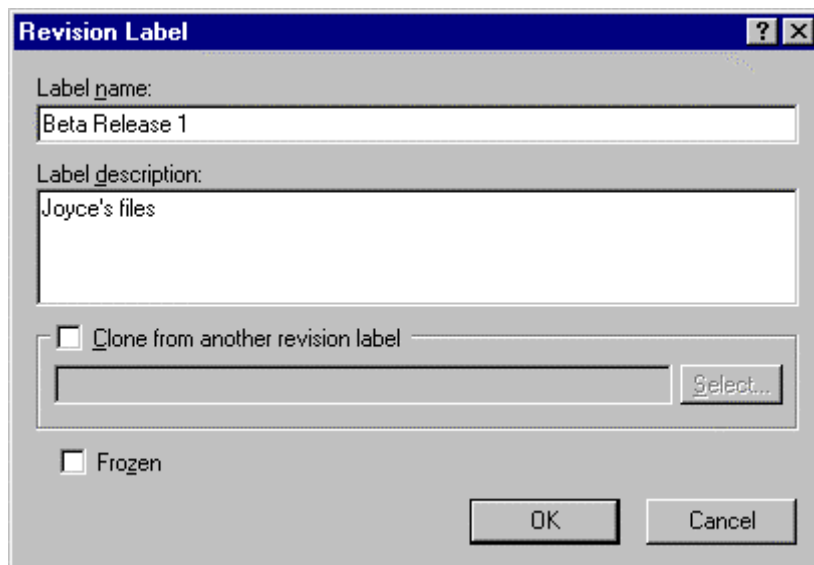
处理这个场景的另一个方法是使用修订标签和视图标签。修订标签提供了一个方便的途径通过使用修订标签的名称来标识一个修订或一小批修订。这种方法主要用于文件。

视图标签应用于视图中的所有项的最新修订。例如：当项目到达某个特定的里程碑时（如 beta），你可能要创建一个视图标签。然后你就可以配置视图返回到应用此标签时的情形，使用此标签成组地检出修订，基于该标签创建一个新视图，或者将该标签指派给某个提升状态。

在当前场景下，Joyce和Mike将执行下列步骤：

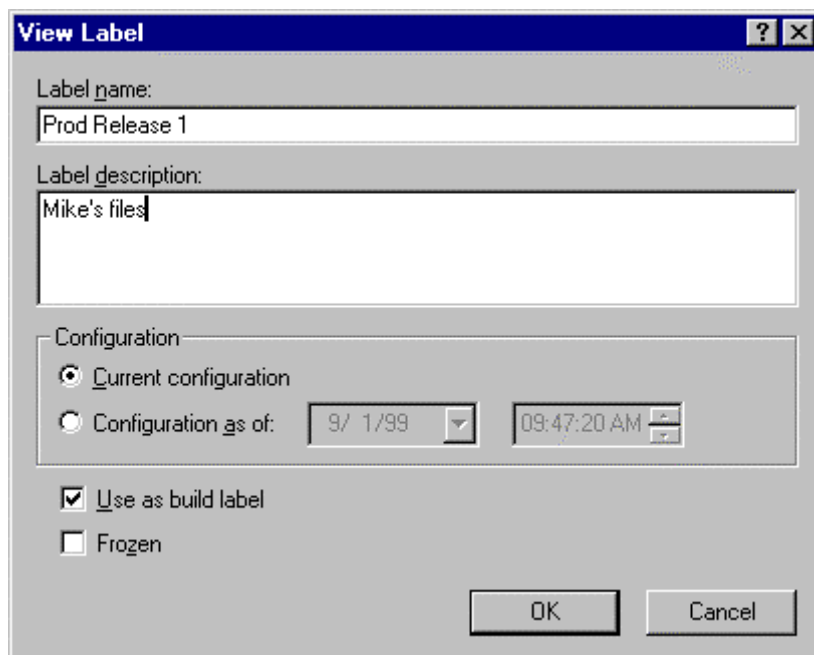
1、在Joyce完成她为发布1所作的变更后，她为她的文件创建一个修订标签（称为【Beta Release】）

1) 并且为下次发布继续开发：



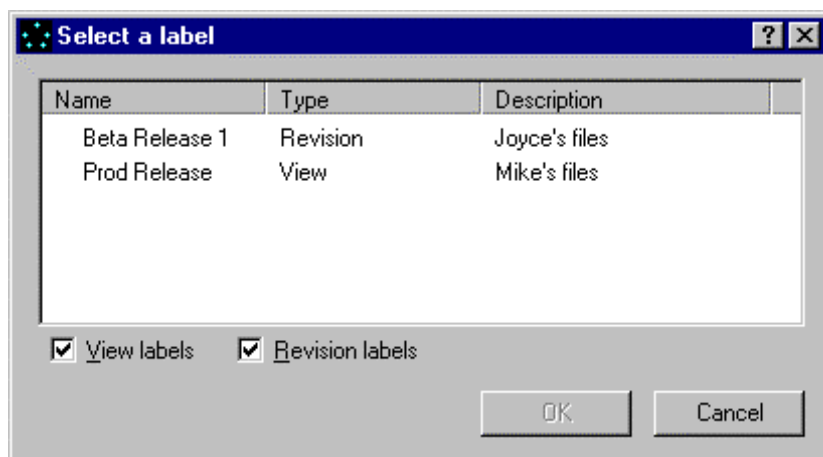
The 'Revision Label' dialog box is shown. It has a title bar with a question mark and a close button. The 'Label name:' field contains 'Beta Release 1'. The 'Label description:' field contains 'Joyce's files'. There is a checkbox for 'Clone from another revision label' which is unchecked. Below it is a text field and a 'Select...' button. There is also a checkbox for 'Frozen' which is unchecked. At the bottom are 'OK' and 'Cancel' buttons.

2、当Mike完成他为发布1所作的变更后，他创建一个视图标签（称为【Prod Release 1】）：

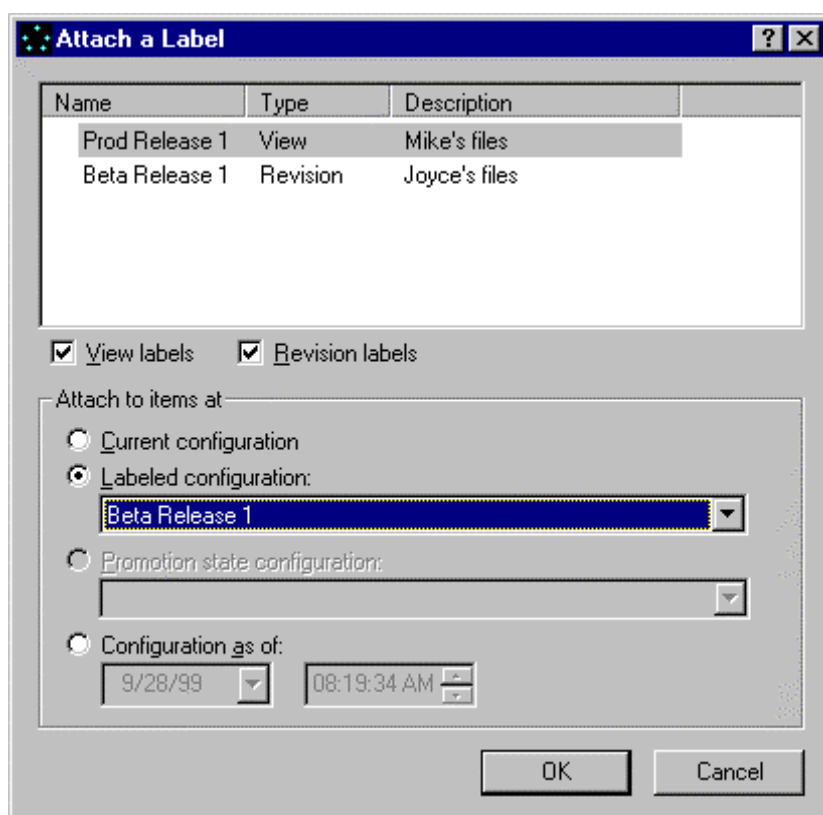


The 'View Label' dialog box is shown. It has a title bar with a question mark and a close button. The 'Label name:' field contains 'Prod Release 1'. The 'Label description:' field contains 'Mike's files'. There is a 'Configuration' section with two radio buttons: 'Current configuration' (selected) and 'Configuration as of:'. The 'Configuration as of:' section has a date field '9/ 1/99' and a time field '09:47:20 AM'. There is a checkbox for 'Use as build label' which is checked. There is also a checkbox for 'Frozen' which is unchecked. At the bottom are 'OK' and 'Cancel' buttons.

3、Mike然后选择【File > Select > By Label...】来找出Joyce的修订标签（称为【Beta Release 1】）：



4、一旦被查询的文件被选中后（Beta Release 1），Mike使用【Attach a Label】对话框将视图标签(Prod Release 1)从这些文件的顶端修订移动到标签为【Beta Release】的修订上：



范例场景 2

Mike 是项目协调员和StarTeam管理员，他的团队要同时发布软件到产品和开发新版本。问：Mike要如何处理项目的发布，同时还要允许其他开发继续工作在主基线上？

使用提升状态来解决场景 2

StarTeam 使用标签、提升状态和分支视图来允许在发布工程活动期间继续基线的开发。这一流程中的步骤是：

- 1、当开发完成时创建一个视图标签。例如：你可能会命名该视图标签为【Release Candidate】：

View Label [?] [X]

Label name:
Release Candidate

Label description:

Configuration

☒ Current configuration

☐ Configuration as of: 9/15/99 18:18:10

☒ Use as build label

☐ Frozen

OK Cancel

2.、选择【View->Promotion】,从【Promotion】对话框中，创建不同的提升状态（例如：Testing、QA和Release）：

Promotion [?] [X]

Promotion States (listed from last to first):

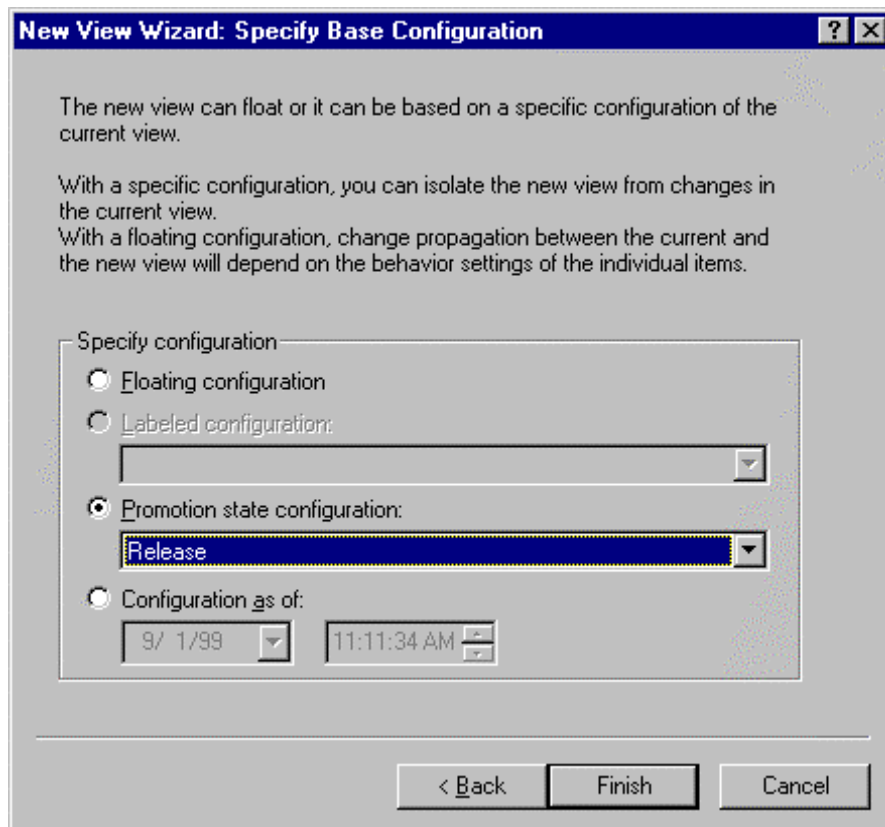
| State | View Label |
|---------|-------------------|
| Release | <current> |
| QA | <current> |
| Testing | Release Candidate |

Add... Edit... Delete Promote... Move Up Move Down

OK Cancel Apply Reset

3.、选择【View->Promotion】来提升【Release Candidate】视图标签通过不同的状态，直到它到达最终状态——【Release state】。

4. 从当前视图中派生一个分支视图，并指派这个配置一个提升状态。在此情况下，提升状态应为【Release】。



这样就为发布集成和管理创建了一个单独的基线，而开发可以继续在主基线中进行。

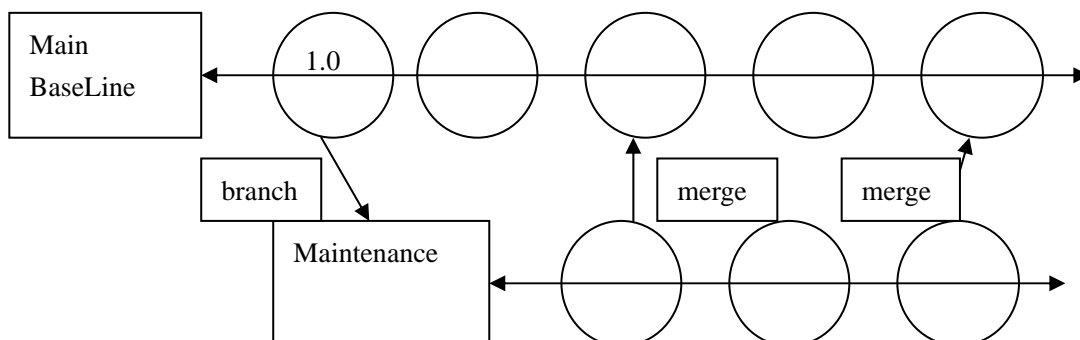
II. Bug 修理

StarTeam为用户提供了从主开发基线中创建修理分支（或维护分支）的能力，以便可以创建服务包或其他故障修理发布。服务包发布后，你可以将修理分支合并回开发的主基线中。

范例场景 3

在开发过程中，Joyce和Mike都有为发布1所作的需要检入到StarTeam中的代码变更。Joyce 意识到她需要对主基线作一些故障修理，然而，她又不想影响主基线的开发。Joyce应该如何进行这些故障修理？

图4：并行维护与开发



使用修理或维护分支来解决场景 3

当维护 and 并行开发需要同时发生时，StarTeam 推荐使用分支和合并，这一过程的步骤是：

- 1、从开发的主基线中，从当前视图中派生出一个新视图。
- 2、允许项分支并分支所有项。
- 3、按照向导进行（就是说：给新视图一个命名和一个新的工作文件夹。基于特定的日期和时间、标签或提升状态选择一个配置）。
- 4、在维护视图中执行你的所有故障修理。
- 5、一旦完成你的故障修理，选择【View->Compare/Merge...】来将你的项合并回开发的主基线中。

III. 独立的产品开发

StarTeam允许用户为独立的开发工作创建单独的开发基线。这个独立的开发基线可以基于主基线中的当前或更早配置。

范例场景 4

开发员Joyce和Mike正在主基线上为产品A开发代码，而Tim和Jill正在为产品A1开发新代码。产品A和产品A1是互相独立的两个产品，分别拥有他们自己的文件、任务、主题、变更请求和标签。Tim和Jill需要从主基线中创建一个允许分支的视图。例如：产品A可能是产品A1的为特定（但不典型）用户开发的带有某些特定特性的版本。

使用一个新的分支来解决场景 4

当基于某个存在的产品开发一个独立的产品时，StarTeam推荐创建一个分支视图。新视图中的每个项发生改变时应该进行分支，实现它的步骤为：

- 1、从主基线的当前视图中派生出一个新视图。
- 2、允许项分支，并且分支全部项。
- 3、按照向导指示进行（就是说：给新视图一个命名和新的工作文件夹。基于特定的日期和时间、标签或提升状态选择一个配置）。
- 4、从新视图中检出文件，适当地修改它们，然后将它们检入。

IV. 公共组件库的开发

通常，公司会有一些产品带有由公共源代码文件的共享库组成的组件。这些公共源代码文件需要包含在使用它们的每个组件的构建中。用户将需要一个项目来专门用来保存这些公共库。将产品分为单独的组件可以大大增强每个集成组件的弹性，这么作可以使得它们容易打标签、进行分支以及与产品中的任何其他组件独立地通过它自己的提升模型序列。

图5：共享公共代码文件的项目层次

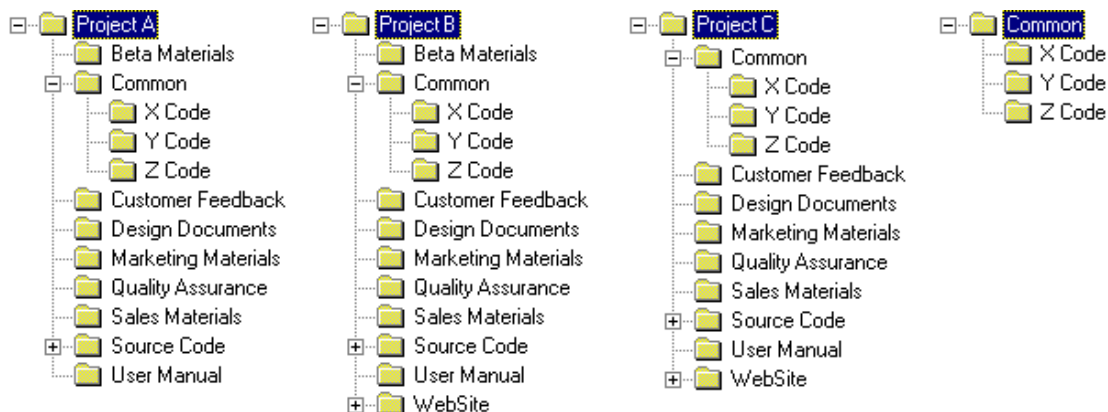


图5描述了三个独立的StarTeam项目¹。1. 【Common】项目包含了项目【A- C】²的共享源代码。你可以从任何项

¹ 注意：它们是项目而不是视图。

² 注意：用户不允许检入文件到共享文件夹中，此外，如果用户想要从公共文件夹中删除某个文件，并不会波及到其他所有的共享文件夹，必须一个一个地删除这些共享文件夹下的该文件。

目中访问这些共享的源代码。

你可以对共享文件夹进行设置，以便在任何共享项目中对源代码所作的任何改变或者是来自【Common】项目的改变立即更新所有的项目。你也可以强制项目开发员对公共代码的修改仅保留在修改代码的项目中。你通过分别设置或不设置【Branch On Change】行为选项来完成这项工作。

你可以回滚共享文件夹实现在给定项目中使用公共源代码的更早版本。然而，你无法根据标签或提升状态来回滚共享文件夹，因为互相独立的项目（和视图）并不共享视图标签或提升状态。

项目A到C可以是同一产品的组件或单独的产品，这取决于你的需要。下面的段落解释当图5中显示的项目A到C在同一个产品下时，如何在一个时刻点同时构建它们³。

设置构建过程

当每个组件是一个单独的StarTeam项目时，你可以很容易地进行打标签、分支和提升项的工作，不用担心会影响其他组件/产品。然而，有些其他的工作需要检出所有的组件，才能创建最终的产品。最容易的方式是使用StarTeam的命令行方式来组合这些项目以实现构建过程。

按构建标签进行检出

图6是一个生成产品构建的批处理文件的例子，该产品由独立的组件组成，每一个存储在不同的项目中。每个组件/项目都是根据它的构建标签进行检出：项目A使用它的Build 5视图标签；项目B则使用它的Build 2视图标签；项目C使用它的Build 3视图标签。检出过程仅检索带有正确构建标签的文件的修订。来自所有项目的全部文件被检出到“C:\production build”中。这个批处理文件编译这些项目并创建一个单一的产品。为了实现重用此构建过程，以便无须记住使用了每个组件的哪个标签，将【build.bat】检入到StarTeam中并给它指派一个构建标签。将来，如果你想要重建这一构建过程，只需要将【build.bat】从StarTeam中检出并执行它即可。记住，如果你有一个公共源代码的共享文件夹，必须要检出该代码到不同的地方三次，那么你必须允许这么作：

Figure 6: Build.bat

```
@ccho off
stcmd co -p "Administrator:Administrator@Whitestar:49201/Project A/Project A" -is -x -q -o -fs -rp
"C:\Production build" -vl "Build 5" *
stcmd co -p "Administrator:Administrator@Whitestar:49201/Project B/Project B" -is -x -q -o -fs -rp
"C:\Production build" -vl "Build 2" *
stcmd co -p "Administrator:Administrator@Whitestar:49201/Project C/Project C" -is -x -q -o -fs -rp
"C:\Production build" -vl "Build 3" *
```

按提升状态检出

你可以通过按提升状态检出文件来快速创建一个你的产品的当前构建。好处是你无须为每个组件使用适当的标签对项进行设置。而这可以为你节省大量的时间，记住提升状态是浮动的。因此，今天用来生成一个构建的同一个批处理文件，明天就可能生成的是不同的构建，取决于分配给提升状态的标签。图7演示了一个为每个项目设置的三步提升模型（关于设置提升模型的更多信息，请查看StarTeam用户指南的第九章：【使用提升状态】）。每一个提升状态都被指派给一个对应的构建标签。为了快速地检出你的产品的当前生产版本，运行一个批处理文件（请见图8：production.bat）来从当前的发布提升状态中检出每个子组件下的所有文件。

Figure 7: Three-step Promotion Model

³ 用户可以只构建A和C，而排除B。该开发模型允许用户构建可互换的组件来产生产品。

| | Project A | Project B | Project C |
|---------|-----------|-----------|-----------|
| Release | Build 7 | Build 4 | Build 5 |
| QA | Build 6 | Build 3 | Build 4 |
| Testing | Build 5 | Build 2 | Build 3 |

Figure 8: Production.bat

```
@echo off
```

```
stcmd co -p "Administrator:Administrator@Whitestar:49201/Project A/Project A" -is -x -q -o -fs -rp  
"C:\Production build" -cfgp "Release" *
```

```
stcmd co -p "Administrator:Administrator@Whitestar:49201/Project B/Project B" -is -x -q -o -fs -rp  
"C:\Production build" -cfgp "Release" *
```

```
stcmd co -p "Administrator:Administrator@Whitestar:49201/Project C/Project C" -is -x -q -o -fs -rp  
"C:\Production build" -cfgp "Release" *
```

运行【production.bat】可以产生一个与【build.bat】同样的检出结果。然而，如果Release提升状态被指派给了一个新的构建标签，然后在运行production.bat将生成一个新的构建，而build.bat则不能做到。为了实现很容易地回到先前的构建，构造和使用这两个批处理文件都很重要。为了重用，要将这些批处理文件检入到StarTeam中。

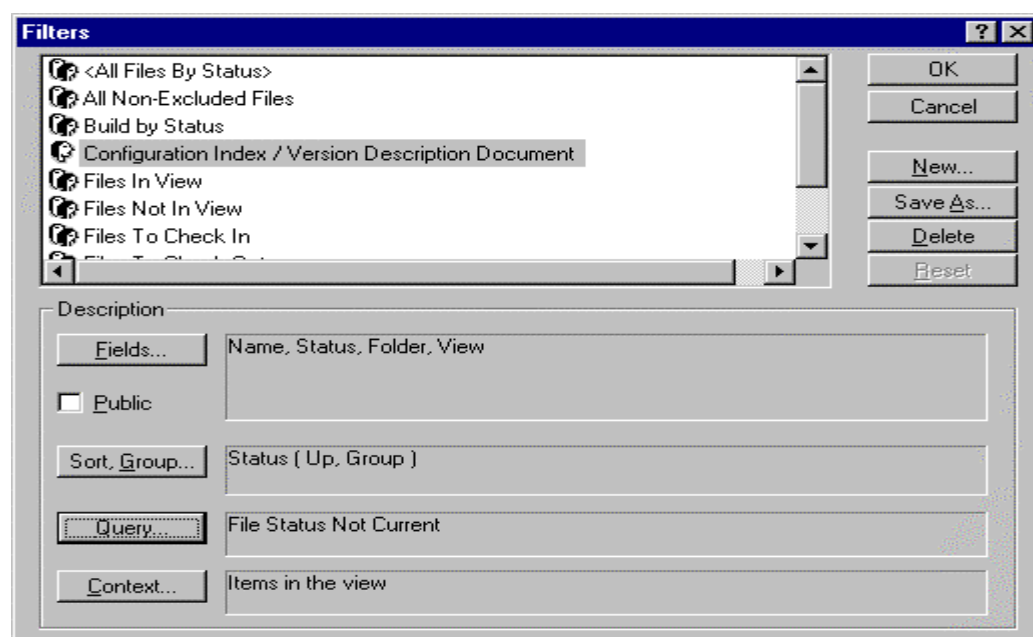
V. 配置标识报告

StarTeam允许它的用户为构建和/或项目创建配置标识报告。配置标识报告说明了自从上次构建以来，什么文件发生的改变，加入了什么文件，删除了那些文件等。

下面的过程将带领你完成创建这样一个报告所需要的步骤。通过创建一个过滤（这个过滤可以被所有配置标识报告重用），在文件列表中显示正确的文件，应用该过滤，然后生成报告。

创建过滤

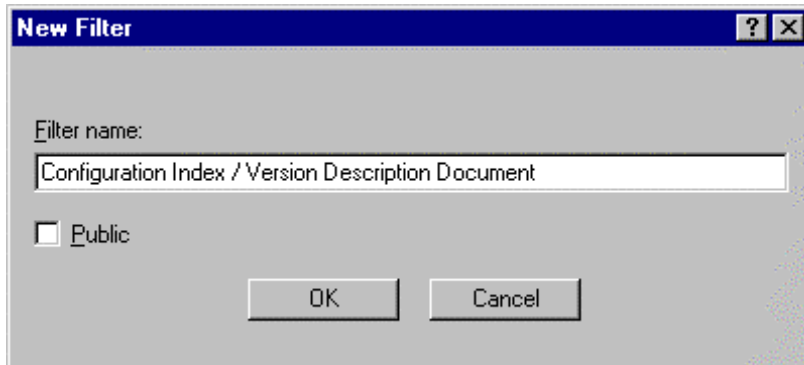
你创建的过滤将会与下面对话框中描述的过滤类似：



- 1、创建一个名为【Configuration Index / Version Description Document】的过滤。
 - a、选择【Files】以显示文件列表。

b、选择【File->Filters->Filters...】显示过滤对话框。

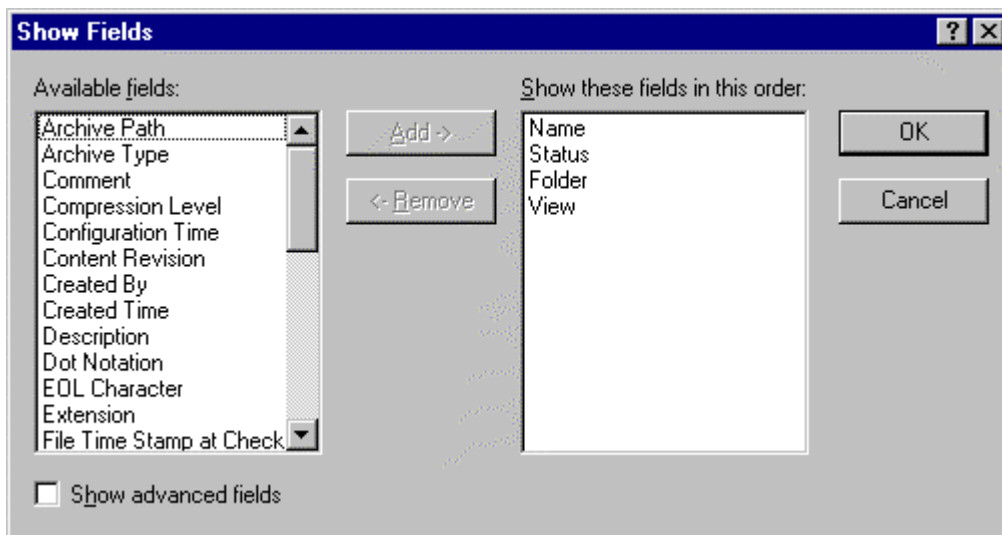
c、点击【New...】显示【New Filters】对话框。



d、命名该过滤；如果你想要让此过滤可以被除你以外的别人使用的化，选中【public】复选框，点击OK按钮。

2、指定要显示的字段：

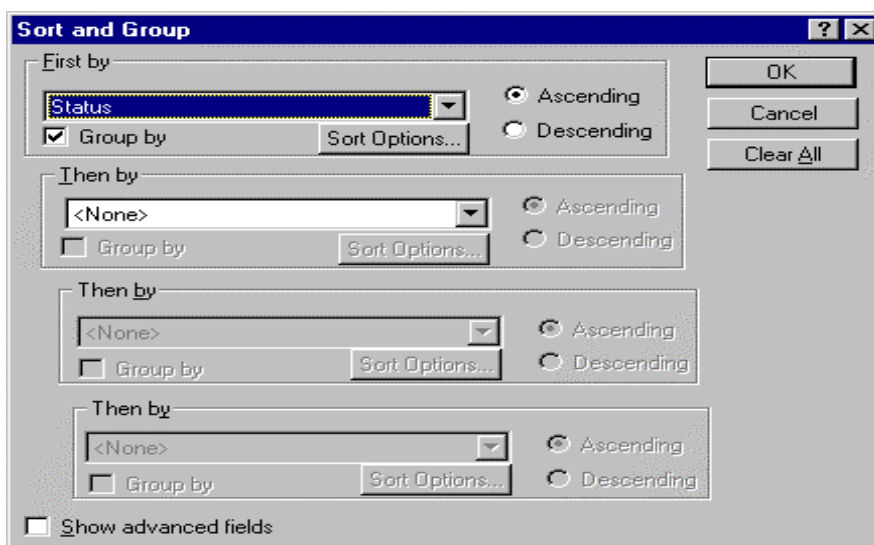
a、点击【Fields...】显示【Show Fields】对话框。



b、选择Name、Status、Folder、View和其他感兴趣的字段，点击OK。

3、按【Status】进行排序和分组：

a、点击【Sort, Group....】显示【Sort and Group】对话框。



- b、按状态对文件进行排序和分组；点击OK。
- 4、创建一个查询，以找出状态不为Current的所有文件：
 - a、点击【Query】，显示【Queries】对话框。
 - b、点击【New...】显示【New Query】对话框。

New Query

Query

Name: ☐ Public

Query tree:

- AND
 - Status Is Not Current

Logical node

Condition node

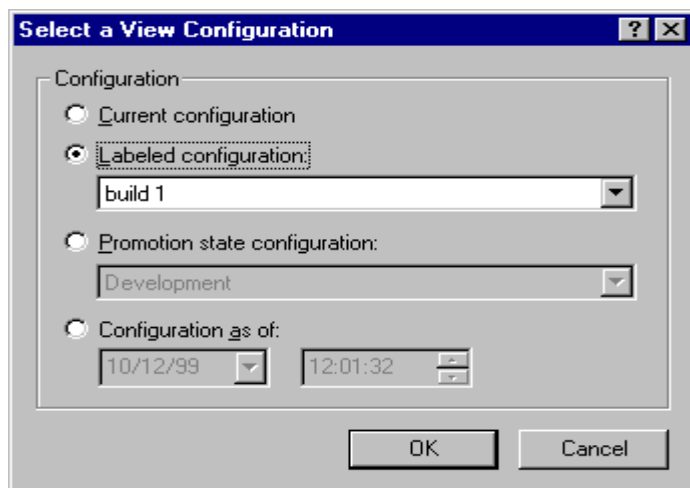
| Field | Operator | Value |
|--------|----------|---------|
| Status | Is Not | Current |

☐ Show advanced fields

- c、创建一个条件，找出所有满足状态不为Current的记录，点击OK。

显示已修改的文件并生成报告

- 1、根据特定的构建（如：build 2）检出所有文件。
 - 注意：**在检出该构建之前，从要用于检出过程的目录/文件夹中删除所有项。
- 2、将视图回滚到先前的构建标签（例如：build 1）。
 - a、选择【View->Select Configuration...】显示【View Configuration】对话框。
 - b、将视图配置到先前的构建标签。

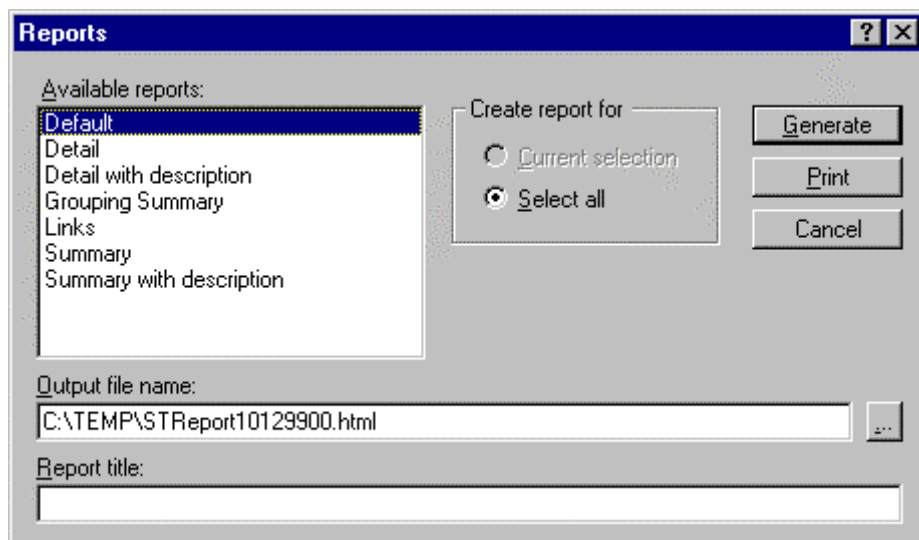


3、对于在构建2中被修改的文件状态将从【Current】改变为【Modified】，刚加入构建2的文件状态将为【Not In View】，而对从构建2中删除的文件其状态则变为【Missing】。

4、应用你的新过滤（【Configuration Index / Version Description Document】）以去掉状态为【Current】的文件（如果有的话）并对文件进行分组，为生成报表作准备。

5、选择【File->Reports...】显示【Reports】对话框。

6、点击【Generate】创建默认报表。



VI. 阶段

许多内容经理使用阶段的概念来帮助管理软件开发生命周期（SDLC）流程。例如：内容经理通常可能会有三个阶段：开发、测试和产品。当内容被开发、测试或者是放到产品服务器上时，他们将内容从一个阶段移动到另一个阶段。这可能会涉及管理三个计算机，每一个服务于一个流程阶段，使用VSS安装在每一台机器上。

取代使用VSS安装在每台机器上，需要管理三台机器的方式，你可以使用StarTeam仅需要管理一台机器就可以完成，它通过使用一个带有三个StarTeam视图（在下面解释）的项目来实现此功能。这样就降低了管理数量与管理任务。

这三个视图分别逻辑地位于不同的网络根文件夹下，并且可以通过单一的web服务器引用它们。文件可以手动从每个视图中检出到独立的工作文件夹/目录下。这无论是对于内部网还是位于防火墙内部的网络应用来说，都是可以接受的实践。由于大量的流量以及客户的网站通常位于防火墙外部，客户的网站服务器通常位于单独的机器上。不管怎样，StarTeam视图都会使得内容管理更轻松。

下面的例子中，对每个发展进程来说都定义了一个对应的阶段。它使用StarDraw 范例项目，你可以在其中创建

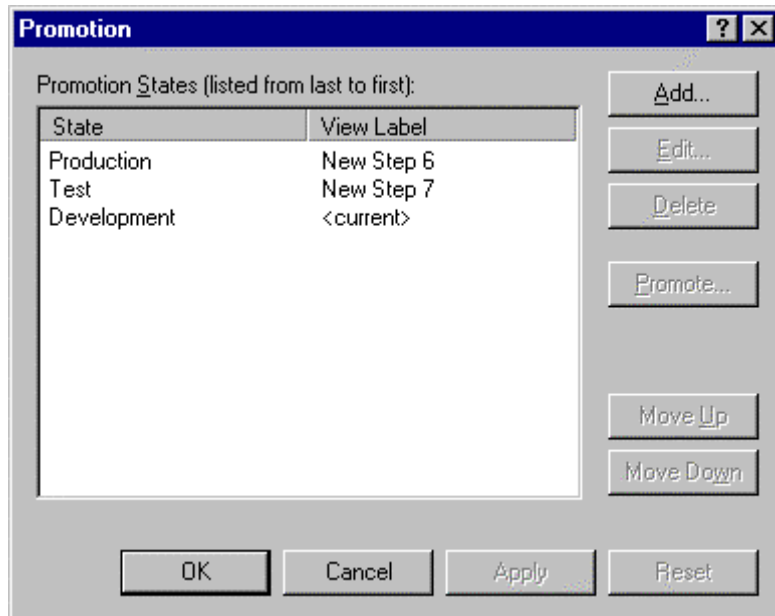
视图和提升状态。

阶段 1：为你的阶段创建提升状态

1、打开StarDraw项目到它的根视图中。

2、在StarDraw中创建下列提升状态（使用【View ->Promotion...】）。 .

- Production —将标签【New Step 6】指派给此状态。
- Test— 将标签【New Step 7】指派给此状态。
- Development— 指派【<current>】给此状态。



注意：当你创建提升状态时，可能会出现当前还没有视图标签存在的情形出现。那么请为这些状态使用【<current>】。你可以在文件与符合该提升状态的某个视图标签相关联时，再将该视图标签指派给该提升状态。如果你愿意，你也可以将指派给提升状态的视图标签从先前的状态改变为没有标签的状态。

提示：

- 使用【Move Up】和【Move Down】按钮来重排提升状态的次序。它们应该是以从最后到第一的次序排列（从最终状态到初始状态）。
- 点击【Reset】从服务器上重新读取提升状态列表。当你打开此对话框或当你最近点击【Apply】按钮时，也将刷新当前可用的提升状态。
- 点击【Edit...】编辑所选择的提升状态的名称、描述或视图标签。
- 点击【Delete...】从列表中删除选择的提升状态。

阶段2：为测试阶段创建一个视图

1、从StarDraw的根视图中，选择【View->New...】创建一个新视图。

2、默认，【Derive New View From Current View】复选框时选中的，点击【Next】。

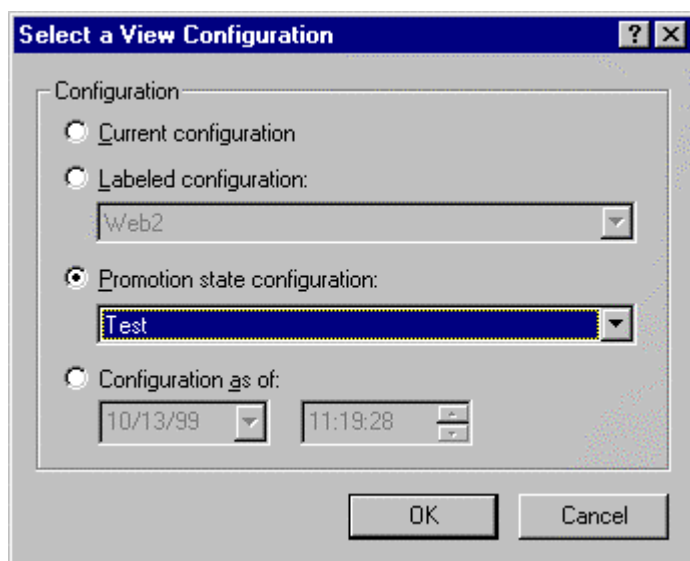
3、命名这个视图为【Test】，然后点击【Next】。

4、默认，StarDraw文件夹及其子文件夹将出现在新视图中，点击【Next】。

5、改变工作文件夹/目录（用于测试组需要用来检出某些文件的情况），点击【Finish】。

6、配置该视图以显示提升状态为Test的所有文件及其他配置项：

- a、选择【View->Select Configuration...】。
- b、选择【Promotion State Configuration】选项按钮。
- c、从列表框中选择【Test】提升状态。



注意：你需要为此视图设置访问权限，以便只有测试组可以查看和修改此视图的变更请求、任务和主题。

阶段3：为产品化阶段创建一个视图

- 1、从StarDraw的根视图中，选择【View->New...】创建一个新视图。
- 2、默认，【Derive New View From Current View】复选框时选中的，保留它为选中状态。
- 3、选中【Read-only】复选框，点击【Next】。
- 4、命名此视图为【Production】，然后点击【Next】。
- 5、默认，StarDraw文件夹及其子文件夹将出现在新视图中，点击【Next】。
- 6、改变工作文件夹，点击【Finish】。
- 7、配置该视图以显示提升状态为Production的所有文件及其他配置项：
 - a、选择【View->Select Configuration...】。
 - b、选择【Promotion State Configuration】选项按钮。
 - c、从列表框中选择【Production】提升状态。

注意：你需要对此视图设置某些权限，以便只有产品组的成员可以查看此视图中的文件。

四、关于StarTeam 5.3的一些提示与技巧

StarTeam是一个非常弹性的工具，可以适合许多不同的工作类型。你需要打造它使之最适合你的团队的需要。每一个可能的项目选项都具有它正面和反面的效果。本文解释一些你需要考虑的基础信息。

数据库

除非你对某个数据库有特殊需要，Borland产品专家推荐你使用你最熟悉的数据库。如果你对任何数据库都不熟悉，他们推荐使用MSDE数据库。StarBase存储它自己的文件在StarTeam中，使用StarTeam Server和MSDE数据库可以支持超过50个用户。这个MSDE数据库运行与大多数SQL数据库一样稳定。此外，它更容易管理。虽然，MS Access数据库有1.2GB的限制，然而这一限制很少会达到，因为StarTeam将数据文件存储在文件夹中，称为 StarTeam vault 或 repository，而不是在数据库中。

项目结构

Borland产品专家为你的每个产品使用一个StarTeam项目。一个项目通常包含一个产品的源代码、文档和其他辅助文件。缺少布局规划而将多个产品放在单个StarTeam项目中，将使得很难独立地对每个产品进行版本化。当你在某个项目中创建一个视图标签时，该标签标记了当前项目视图中地每一个项。（这包括StarTeam文件夹、文件、变更请求等等）。如果你将多个产品放在同一个视图中，那么它们将具有同一个标签，这将在对构建标签化和处理变更请求

时引起混乱。

文件夹

在大多数情况下，每个StarTeam文件夹和项目视图与一个唯一的物理位置相对应，称为工作文件夹。工作文件夹存储了检出的文件，并且是将新文件加入StarTeam的版本控制系统的地方。要避免给出两个或多个StarTeam文件夹或者十查看同一个工作文件夹（或文件夹结构）——虽然StarTeam允许这么作。工作文件夹和StarTeam文件夹之间的一对多关系将导致文件状态问题。

如果你需要从StarTeam的许多文件夹中检出许多文件到一个公共的物理文件夹下（例如：为了使用一个构建环境），你应该使用命令行工具完成此项工作。使用【/fs】参数可以忽略文件的状态，这将节省大量的时间。

备份

要将定期对你的项目作完整备份作为你的基础原则。要确保在同一时间备份了所有需要的文件和文件夹。

《StarTeam 管理员指南》(附录C,“备份StarTeam”)中描述了正确的备份过程。然而，它没有足够强调数据库与StarTeam归档库必须同步备份的重要性，如果StarTeam文件归档库与StarTeam数据库时在不同时间进行备份的，而用户在这段时间内又访问了StarTeam，则归档库与数据库将失去同步。这在当使用的数据库是如MS SQL或Oracle时是很容易发生的情形，这些数据库常常是与StarTeam Server 运行在不同的机器上，并且可能是按照它们自己的备份计划，而恢复失去同步的归档库与数据库将可能在StarTeam中引起操作问题。虽然有一个库验证实用程序可以帮助重新同步数据库与归档库，但还是可能会有一些不可避免的数据丢失。为了避免发生这种情况，你应该执行下列步骤之一以确保同步备份：

- 1、在整个备份过程中，使用StarTeam命令行或服务器管理工具锁定服务器。
- 2、关闭StarTeam Server。

安全

默认，所有用户有权访问StarTeam中的一切事物，这也包括删除项目。为了避免意外的删除，要尽早地设置访问权限。

有关安全方面的事项包括：

1、StarTeam是从最低的层次（项层次）到最高层次（项目层次）来检查访问权限的。例如：如果你是在单个的文件上设置访问权限的，则文件夹、视图和项目级别的访问权限将不会再去检查该文件的访问权限。

2、如果你在【】对话框的某个页面上为某个用户或组设置的访问权限，则在该页面上所有不包括在内的用户将会被拒绝该页面上的该级别的所有访问权限。

通常，如果你在任意上设置访问权限，你必须确保要为相关的用户或组也在这个页上设置访问权限，因为他们需要访问被此页涉及权限所影响的对象。

3、访问权限可以被下列情况取代：

（1）用户是该对象的拥有者：通常来说，拥有者就是创建该对象的人。拥有者信息存储在服务器配置数据库中，但是并不显示在StarTeam中。在StarTeam 5.3中，这一取代因素可以被设置为忽略。

（2）已经给包含该用户的组赋予了特权：特权是从StarTeam Server中预先设置好的一组权利。

在StarTeam 5.3中，这一取代因素可以被忽略。

默认，管理员组具有完全的特权（有权作任何事情），而安全管理员和【All Users】组则没有特权。

4、当用户隶属于多个组时，该用户具有他所属的任何组所赋予的权限。换句话说，该用户具有他/她所隶属于的全部组所赋予的最大组合访问权限。

5、访问权限是不继承的。不管怎样，它们附着于文件夹或项一起，知道这些文件夹或项发生分支。

同一项目中的每个视图具有相同的项目级的访问权限。当你从一个已存在的视图中派生出一个分支视图时，子视图并不从它的父视图中继承访问权限。然而，子视图中的在父视图中同样存在的每个文件夹或项将具有它的父视图中同样的文件夹级或项级的访问权限，直到该文件夹或项发生分支，然后该文件夹或项就不再具有任何文件夹或项级的访问权限了。如果你具有查看访问权限的权利，你就可以从父视图和子视图中查看发生分支前的这些文件夹级或

项级的权限。在两个视图中的任一个视图中改变这些权限也将在另一个视图中改变它们，因为你是同一个文件夹或项上改变权限。

6、访问权限随同文件夹或项的移动或共享而一起移动：

当文件夹从一个视图移动或共享到另一个视图中时，它们会将分配给它的文件夹级的访问权限一起带到新视图中。类似，从一个视图移动或共享到另一个视图的项（文件、变更请求、任务和主题）也会将分配给它的项级的访问权限带到新视图中。共享的文件夹和/或项可以被分支，一旦发生分支，它们就丢失了它们所具有的访问权限。

（一）、从StarTeam Server中进行操作

在StarTeam Server中，使用【User Manager】为每个服务器配置（该服务器配置应该处于运行之中）创建用户和组。请使用下列指导方针：

- 1、不要改变【All Users】组或【Administrators】组的特权。要知道这些组特权取代了项目级的任何安全设置。
- 2、不要在管理员组下面创建组，因为它们将从管理员组中继承特权，从而成为管理安全的最困难的地方。
- 3、在【All Users】组下创建组或者互相容纳。例如：你可能需要创建与下面类似的组：Developers，QA，和Documentation。
- 4、创建用户并将他们分配给组。要取保至少有两个用户为管理员，以应付某个管理员因为某种原因被锁定的情况。

（二）、从StarTeam中进行操作

从StarTeam中，你可以按照下面的方法设置访问权限：

- 1、项目级的访问权限设置，选择【Project-> Access Rights...】。
- 2、视图级的访问权限设置，选择【View->Access Rights...】。
- 3、文件夹级的访问权限设置，选择【Folder->Advanced->Access Rights...】。
- 4、单独的项级的访问权限设置，选择【<item>->Advanced->Access Rights...】（项是文件、变更请求、任务和主题）。通常不在这个级别设置访问权限，但是如果你有需要，也可以这么作。

设置访问权限时使用下列指导方针：

- 1、首先在项目级别设置访问权限。

（1）、在每个页上为每个组（All Users组除外）设置访问权限（这些页是：Project, Views, ChildFolders, Files, Change Requests, Tasks, 和Topics）。

- 2、在视图级设置访问权限。

（1）、如果你为某个页设置的访问权限，记住在该页上没有赋予访问权限的所有用户将会被拒绝此页上的这个级别的所有访问权限。如果你在某页上设置访问权限，你必须在该页上为每个需要在这个级别具有访问权限的用户或组进行设置。（页是：Views, Child Folders, Files, ChangeRequests, Tasks, and Topics）。这一规则也适用于文件夹级和项级的访问权限设置。

- 3、只有在你真正需要时才在文件夹级设置访问权限：

记住当文件夹成为新视图的一部分时，该文件夹上的访问权限设置也会随同一起移动或共享（直到该文件夹在新视图中发生分支为止）。记住文件夹仅在当它们的属性发生改变时才发生分支，而它们的属性往往很少发生变化。

- 4、尽可能避免在项上设置安全。

记住：如果以后某个时候从此视图中派生的某个视图中包含此项，这些安全设置将一直伴随该项，直到它发生分支为止。同时，如果该项被移动或被共享，这些安全设置也会随同该项一起移动。

- 5、你可以拒绝授权，也可以对它们授予权限，但最好是采用仅授予权限的方式。

如果你采用了拒绝授权的方式，要注意观察以确保遵循下面的两条准则：

- （1）总是确保任意页上的拒绝授权记录在该页的所有授权记录之前。
- （2）永远不要在【Access Rights】对话框的任意页上只有拒绝授权记录的情况出现。

在StarTeam中维护文件夹和项

文件夹和项（文件、变更请求、任务和主题）可以被移动、共享和删除。它们可以在同一个视图内或视图间移动和共享，只要这些视图隶属于的项目使用的是同一个服务器配置即可。

移动文件夹或项时可能会带来某些复杂情况的发生：

1、如果移动文件夹或项到视图的外面，然后回滚该视图（到该文件夹或项还存在于该视图中的某个日期），这些文件夹或项也不再会出现。这是因为文件夹或项一旦被移动，它们就不再附着于它们原来所附着的视图了，而删除或共享文件夹和项则不会这样。

删除文件夹和项并不能使你的归档库的大小减少，也不会增强性能。删除实际上并没有真正删除任何东西，它是退休功能的增强，被删除的文件夹或项只有当你将视图回滚到它们被删除前的时间点才能被重新看见。

当文件夹或项变为作废时，可以考虑将其移动到同一视图的另一文件夹（可能其名称为【Obsolete】）下或者删除它。

如果你需要某个项也位于另一个视图中，则考虑将其与另一个视图共享。

2、如果你将文件夹或项从一个视图移动到另一个视图中，则附着于它之上的标签并不会随它一起移动到新视图中，而在同一个视图中移动则附着于其上的标签仍将继续附着在它上面。

3、如果你将某个变更请求从一个视图移动到另一个视图中，则它的工作流处理会受到如下的影响：

（1）如果该变更请求中的【Last Build Tested】和【Addressed In Build】字段具有构建标签作为它们的值（也就是说，这些字段非空且不包含值【Next Build】），则被移动的变更请求仍将保持这些值。在新视图中，这些值可以被改变，但只能改变为新视图中存在的构建标签的名字。

（2）如果移动的时候【Addressed In Build】字段的值为【Next Build】，则【Next Build】值将会被原始视图中创建的下一个构建标签的名字所替代，而不是新视图中创建的下一个构建标签，即使该变更请求在新视图中发生了其他改变也是如此。

（3）如果移动的时候该变更请求的【Last Build Tested】和【Addressed In Build】字段还没有值，则它的工作流将仅采用新视图中的规则。

本段落中关于工作流的注解也适用于父视图和从它派生出的分支视图中的变更请求。然而，在这里，影响工作流的是发生分支时的变更请求中的【Last Build Tested】字段和【Addressed In Build】字段的值，而不是它被移动时的值。

当你合并变更请求时，你可以控制属性最终在视图中采纳何值。不管怎样，当合并变更请求时，你也会遇到上面提到的同样的复杂性问题。例如：被指派为【Next Build】值的视图中的下一个视图标签将变为【Addressed In Build】字段的值，而不管合并后的变更请求现在驻留于哪一个视图中。

共享文件夹或项也会导致复杂性问题：

1、应避免循环共享。循环共享时文件夹中的某个东西或项，如某个文件，被共享会它的原始位置。例如：你可能会共享同一个文件夹从第一个视图到第二个视图，然后从第二个视图到第三个视图，最后又从第三个视图回到第一个视图，则回到第一个视图的共享就是循环共享。

2、删除原始的共享文件夹或项并不会孤立它的共享后的文件夹或项。记住：在任何时候，StarTeam并不真正删除任何事物。

3、避免冗余共享。冗余共享是指该共享与另一个共享是相同的。例如：如果你将同一个文件夹从第一个视图中共享到第二个视图中两次，你就创建了冗余共享。糟糕的是，相同的共享文件夹将使得文件使用相同的名字保存在相同的工作文件夹下，当从一个文件夹检出文件时，将覆盖从另一个文件夹中检出的文件，从而导致文件状态的错误。

4、如果你共享文件夹或项到某个引用视图中，你事实上是将它共享到了该引用视图的父视图中。如果该父视图是此引用视图的源，则这又是一个循环共享。

5、如果项目的三分之一是有共享文件夹或项组成，那么你应用重新思考该项目的结构。它可能比需要的更复杂化了，同时，大量的共享还会引起某些性能问题。

标签

StarTeam具有两种类型的标签：视图标签和修订标签。视图标签隶属于创建它时所在的视图，在该视图的外面，此视图标签是没有意义的。视图标签是它被创建时StarTeam应用于该视图中的每个文件夹和项上的时间点标记。通常来说，当你想要获得特定视图的快照和特定时间点时，就创建一个视图标签。根据视图标签回滚视图到过去的重要时间点是一个很容易的方式。这并不是视图标签的完整准确的描述，因为视图标签的功能要比这更多一些，但是这里对视图标签的用途作了一个概念性的描述。

修订标签允许你标识某个项的特定修订或一组项的特定修订。修订标签的最常见使用是对视图中的文件进行分组以方便对它们的检出。修订标签可以从一个视图或项目移动到另一个视图或项目中，但仅在你克隆该标签时。通常，修订标签是视图的一部分，不会随项一起移动到新视图中。

如果你创建一个引用视图，则视图标签和修订标签将在新视图中仍然可用，因为引用视图只是对同一视图的不同查看方式而已，它并不是一个不同的视图。然而，如果你创建的是一个分支视图，则将没有标签会自动从父视图中来到新视图中。

同时，如果你移动或共享一个项（如一个文件）从一个视图到另一个视图中，则附着于其上的标签不会跟随它。项所在的新视图中的标签可以附着于被移动的项上，但不能附着到被共享的项上，除非被共享的项在新视图中发生了分支。

视图标签

在项目中，创建分支视图，使它们包含你想要使用同一个视图标签的所有项。如果不想每个事物都收到所给定的视图标签，那么这些成分就应该在另一个视图中。

视图标签的最常见使用是用来对项目的重要日期或里程碑打标签。例如：你可以使用一个视图标签来指出哪些文件被包含在你的项目的构建中，该标签的名字可能如【Build 425】。创建构建标签后，带有该标签的文件就可以为进行构建而被检出，这样作可以防止构建中包含后来加入的或过去某个时间检入的未知文件。

你可以在创建视图标签后加入修订到该视图标签中，或将修订从视图标签中移走。这可以让你将该视图标签创建时还不存在于项目中的文件加入到标签中，或者将逻辑上不属于此标签集合的文件从标签中移走 - 虽然该标签创建时它们呈现于该视图中。当视图根据该标签回滚时，将只显示附着了该标签的文件，它是该视图图片的细微调整。这样的调整可能会非常有用，即使该视图标签不再表现为某个确切的时间点时整个视图的情况。

StarTeam可以让你将视图回滚到过去的任意时刻点，基于特定的时间、标签或提升状态。例如：你可以将该视图回滚到【Build 425】标签时的状态。基于特定的标签回滚可能比根据特定的时间回滚更有用，因为标签可能是经过调整的。

构建标签于变更请求

视图标签可以指派为构建标签。构建标签与其他视图标签的唯一不同之处是：它们会影响变更请求的工作流。当你创建构建标签时，StarTeam将查找【Addressed In Build】字段的值为【Next Build】的所有变更请求，并将值【Next Build】改变为刚才创建的构建标签的名字，这可以让测试员知道为了测试该变更请求所要测试的构建。

关于创建新的分支视图时对变更请求工作流的影响，或变更请求的移动或合并对变更请求工作流的影响，请查看【变更请求】段落。

视图标签和项目进度

你可以使用视图标签来显示项目的进度。例如：你可以在准备进行下一阶段的代码开发时创建一个视图标签，比如你可能会命名该视图标签为【Test】或【Production】。更进一步，你还可以协同使用提升状态和视图标签，请查看段落【Promotion States】。

使用问题

问题：某些用户试图超越视图标签的设计用途扩展视图标签的功能，例如：假定你只想对StarTeam文件夹层次的一个分支打标签，某些用户就创建了一个影响整个视图的视图标签，然后将该视图标签从不想应用的分支上删除（操

作方法有两个：将该视图标签从根文件夹上删除并将它添加到指定的分支上；后者分别从其他的分支上删除）。这并不是一个好的方式。当你创建了一个视图标签然后删除它时，程序将创建一个巨大的排除列表（说明除列表中的项外，所有的事物都具有此标签，这种情况将发生在视图的大量地方甚至是整个视图之中，巨大的排除列表将引起性能问题。

解决方案：如果你需要独立于视图中的其他部分对某个特定的分支打标签，则该分支最好是分离到它自己的分支视图中。或者，你可以使用修订标签来仅对StarTeam文件夹层次中的一个分支或少量分支打标签。你不能回滚视图到某个修订标签，但是你可以选择所有具有该修订标签的文件或者基于该修订标签将这些文件全部检出。

修订标签

修订标签的最常见使用是帮助开发员定位文件修订的子集，可能的情况有：

- 1、这些文件是同一逻辑组的成员，但位于不同的文件夹下。
- 2、这些文件虽然在给定的文件夹下，但为了实现特定的构建需要被快速地隔离。

修订标签使得一次检出所有具有该修订标签的文件成为可能，而不管该文件是否在视图内被移动。

修订标签除了用于文件外，还可以用于项。例如：测试员可以对一小组变更请求打标签，然后以后基于标签选择这些变更请求。

使用问题

用户往往认为修订标签是标识文件夹或项（如文件）的唯一方式。他们认为修订标签会一直附着在文件夹或项上，即使他们将该项移动到了另一个视图或项目中也是如此，而StarTeam并不是以这种方式进行工作的，真实情况是：只要这些文件夹或项还位于同一个视图中，修订标签就会一直附着在该文件夹或项的特定修订上。不管怎样，用户（要具有正确的访问权限）可以从项上分离修订标签或重新将该修订标签附着于项的不同修订上。

你必须意识到修订标签是附着于文件夹或项的特定修订上的，例如：如果文件有十个修订，则修订标签将只能附着于它们中的一个上，如【Revision 5】。当你要基于该文件的修订标签将该文件检出时，你就将该文件的【Revision 5】检出来了。而其他配置管理应用程序检出的是已标签文件的最近修订，这可能会使使用过其他配置管理应用的用户产生混淆。

提升状态

大多数产品会经过某种发布或产品周期，一个最简单的例子就是：一个从开发到测试到市场的软件产品周期。通常软件文件是持续加入到产品中，并且根据测试员通知给开发员的缺陷来对文件进行纠正。最后，软件的某些构建或版本为了全面使用而进行发布。

提升状态对项进行了分组，通常是已经达到开发的某个阶段的文件。它提供了一个方便的机制以确保正确的人在正确的时间可以获得正确的文件（或其他项）。例如：如果软件管理员创建了提升状态【Test】和【Release】，那么测试员工作的文件可以被指派为【Test】状态，而准备发布的文件则可以被指派到【Release】状态。

每个提升状态都是基于某个视图标签，该视图标签通常（但并不是必须）是一个构建标签。视图可以根据提升状态进行回滚，这样，工作于回滚后的文件修订之上的用户就处在他想要的正确开发阶段上。然后与提升状态相关联的视图标签在适当的时候被改变，例如：某周可能是【Build 07】被指派给了【Test】状态，而下周则可能变成了【Build 08】。使用提升状态的好处是用户总是使用同一个提升状态，他们不用关心特定的视图标签。

提升模型是一个概念，在StarTeam中可以实现不同的提升状态。StarTeam的提升状态特性允许管理员创建提升状态，并且它们中的每一个都可以关联到某个视图上。提升状态的视图标签可以在适当的时候被改变，它也可以从某个状态提升到随后的状态。例如：测试员可能总是使用【Test】提升状态下的文件，这些文件可能在某周来自于【Build 07】，而在下周则来自于【Build 08】。

用户通常会按提升状态为它们的工作分配配置项目视图，而不是按视图标签。例如：测试员会将视图配置为【Test】提升状态。

变更请求

变更请求或者是请求某个缺陷的修补，或者是对产品的增强建议。它们可能来自于打电话给产品支持的用户，或者是来自于内部的测试员。变更请求并不是一个工作指令，它只是一个注解，对认识到的提供了解释。通常你应该作下面两件事情：

- 1、对所有员工作一个关于如何书写变更请求的指南培训；
- 2、将系统中的变更请求进行优先级划分，并设置其可能的状态为：“Open”、“Deferred”、或“ As

Designed”。状态为“Open”的变更请求被分配给实现该变更请求的团队成员。

你不能为变更请求定制它的工作流。当你提交一个变更请求时，它的状态默认为“New”。你可以将它的状态改为“Open”、“Fixed”、“As designed”、“InProgress”、“Documented”等等。从某个给定状态，你只能将其改为少数可能的下一个状态。例如：当变更请求的状态为“Fixed”时，你只能将它的状态改变为“Open”或“Verified Fixed”，接着你只能将它的状态改变为“Open”或“Closed”。变更请求的状态不能直接从“Fixed”变为“New”。

如果你使用的是StarTeam Enterprise企业版，那么你可以创建定制字段来替换当前的“Status”字段，然而新创建的字段不能控制变更请求的工作流，不管怎样，你可以根据它对变更请求进行分类和生成报告。你月可以编辑“Status”字段以改变可能的状态的名称。例如：你可能会将“New”改为“Entrant”，或将“Closed”改为“Finalized”。不管怎样，你无法改变状态值的次序，也不能使状态失效。

主题

StarTeam将线索对话与StarTeam的文件夹层次关联到一起。通过主题组件，可以发起关于项目的一般问题，或者是启动关于某个问题的非常针对性的讨论，如某个特性的实现；而主题的响应可以造成这些问题的解决，对话的历史记录对于项目来说甚至显得非常重要，未来的团队成员可以使用它：

- 1、更好的对决策重新评估；
- 2、避免重新尝试先前已经发现为错误的解决方案；
- 3、理解为什么某个特定的解决方案对问题来说是必须的，并且没有替代的方案能够满足所有需要满足的条件。

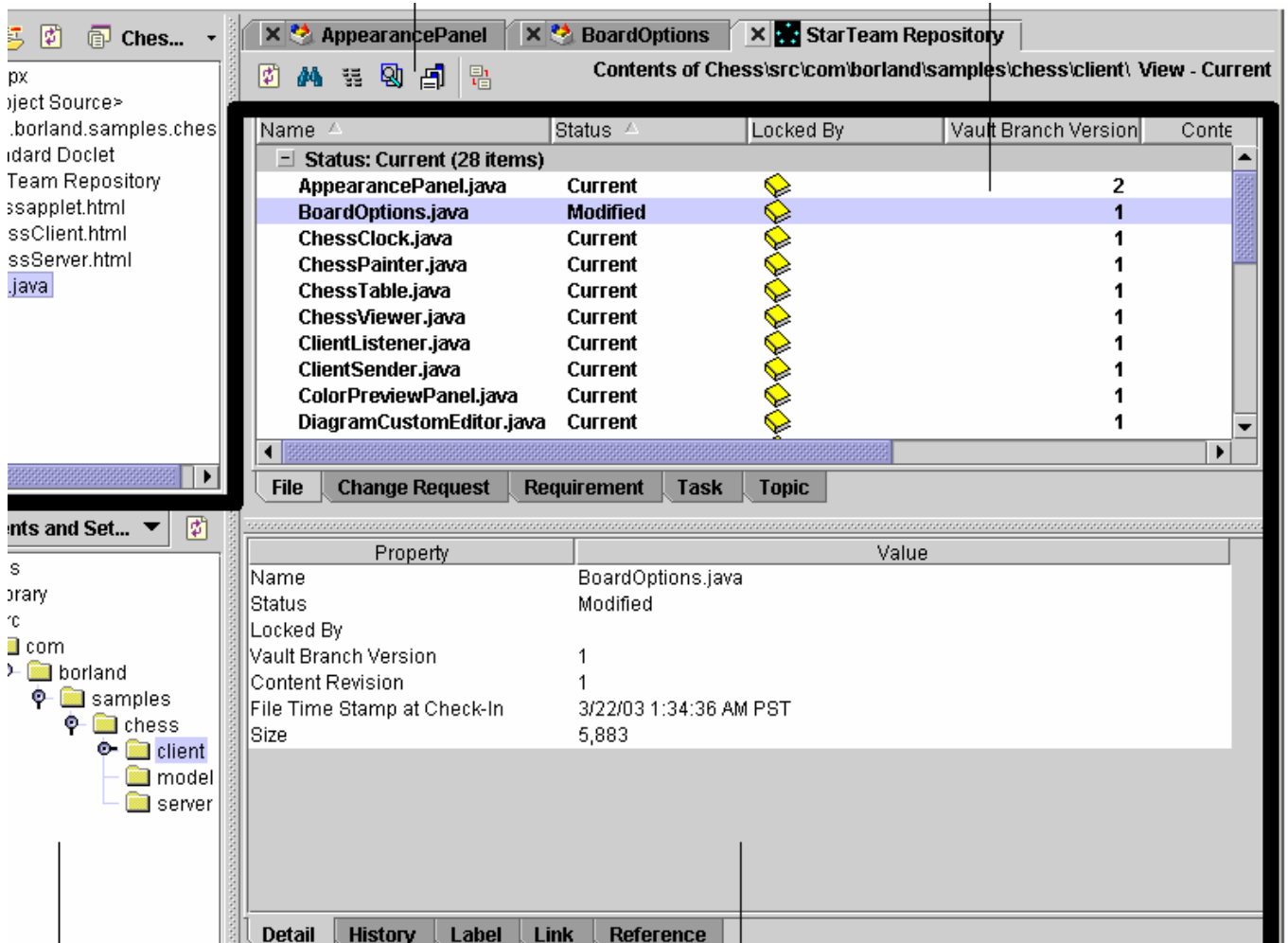
与新闻组类似，主题为讨论提供了一个社区，从而项目中的每一个人可以互相交流而无须使用不可跟踪和非线索的e-mail消息。主题驻留在一个被版本化的集中性的区域中，经过正反双方或经过多方的讨论后，就可以进行相应的任务分配了。

任务

StarTeam的任务组件仅在企业版以上的版本中才可用。它可以让本地和远程用户汇报分配给他们的工作任务的情况。任务组件可以与MS Project和StarTeam的其他组件进行协同工作，或者作为独立的组件与StarTeam的组件一起工作。然而，上述两种方式下的任务组件的使用方式是不同的，当为前者时，某些常用的操作可能只能从MS Project中执行。

后记

StarTeam 作为 Borland ALM 战略的重要一环，必将与它的其他产品越来越进行紧密的集成，下图是JBUILDER9 中 StarTeam 与之集成的图片，大家可以看到，它几乎已经进行了完美的集成：



此外，大家可以看到，Borland 公司是一个技术与平台中立的公司，所以他提供的产品将会保证你的资产最大可能的保值，而且随着他所提倡的 ALM 概念的不断在产品中体现，你的团队的生产率也会成倍的增加。因此，选择 StarTeam 不失为一个明智的选择。