Release 模型 —— 一种通用的软件开发模型

PMT 徐晓春 1999 年 7 月

开发软件产品的方法很多，实际上没有一种绝对正确的方法可以保证软件产品的开发。很多单位在长期的实践中形成了自己独有的开发组织形式，建立开发模型，创造了许多成功的软件产品。Release模型，作为一种软件开发的通用模型，广泛地应用于软件开发部门，可以根据各部门的实际条件进行灵活的调整，以满足本单位的需要。本文就Release模型作一介绍。

一、Release 模型的基本概念

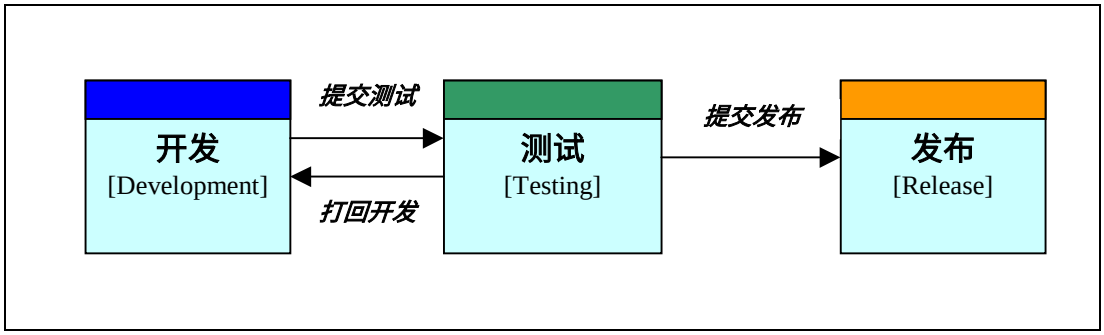
Release模型的重点在于创建具有版本号、项目名和组织名等单一标识的软件产品，一旦产品推出，即以此为起点，接下来的开发工作变围绕着下一个版本开始，这个新版本是一个独立的版本，比如 SOFT 2.0.1 不同于 SOFT 2.1，尽管 SOFT2.1是建立在 SOFT2.0.1的基础上，因为它有许多 SOFT2.0.1没有的新特征和功能。在此类开发环境中， 需要跟踪产品的每一版本，并且要知道那些文件有更改，和更改的原因。在开发下一个版本软件的同时要维护当前软件的任意一个版本。通过Release模型可以解决这些问题。

经典的Release模型有三个基本状态：开发、测试和发布（RELEASE）。每个状态中的更改内容相互独立，以保证开发中的程序修改不影响测试结果，测试内容不影响到已经发布的版本。

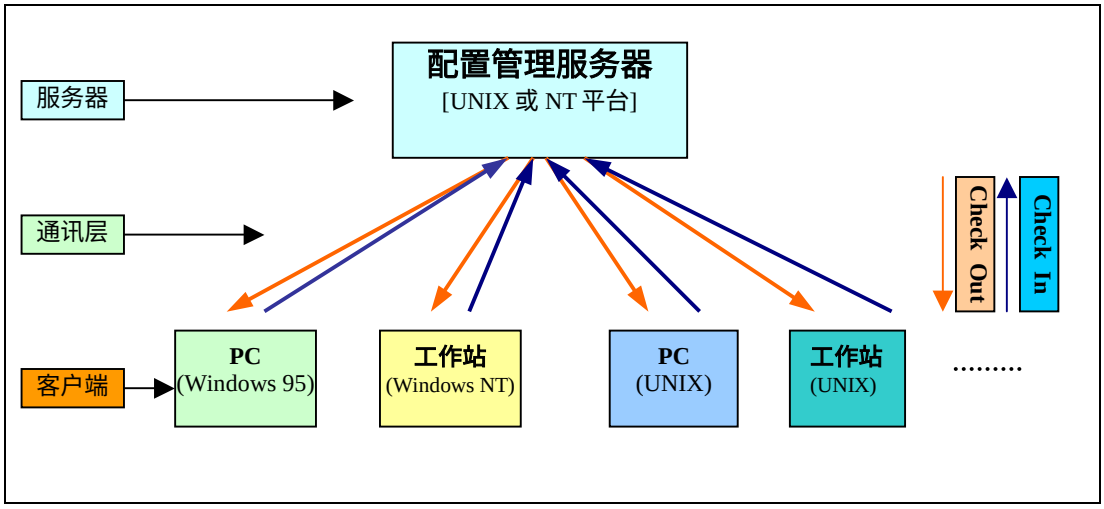
Release模型对应的生命周期开始于需要进行一次软件修补，并且已经创建了一个软件包。该软件包与一个更改需求表格相关联，以详细描述要求进行的更改。任何人可以创建CHANGE包。如果包已经创建完毕，就可以将软件包中的文件检出（CHECK OUT）进行更改。修改完成后，可以将文件检入（CHECK IN）到软件配置管理服务器，该文件版本与该软件包相关联。当所有的更改完成以后，就可以将软件包推进到测试状态。测试完成后，将该软件包推进到发布状态。

当下一个RELEASE开始或需要进行紧急的修复时，可以创建快照（SNAPSHOT）。快照主要用来创建一个新的环境，以便于进行另外独立的开发。例如，现在工作在RELEASE 1.0，正在进行它的Beta测试，你想在Release 2.0上进行开发，就要先创建一个Release 1.0 Beta的快照，并用它创建Release 2.0 的环境。任何在Release 2.0 的环境中修改，在Release 1.0 的环境中都不可见，但是在一个环境中的修改也可以在另一个环境中实现，只要采用交互合并的技术，而且可以选择所需要合并的内容。例如，现在有很多错误修复需要集成到Release 1.0 Beta中，在Release 2.0中已有一半更正已经实现，而且需要在Release 1.0中包含它们，就可以采用交互合并过程将所有Release 2.0中的更改带到Release 1.0中，由于Release 1.0 Beta中的修补将会传播到Release 2.0，当Release 1.0 Beta中的修正全部完成以后，可以采用交互合并将Release 1.0 Beta中的所有更正带到Release 2.0。

Release模型的生命周期的定义如下图：



Release模型有三个状态：开发、测试和发布，而每一个状态有自己的视图，每一状态可进行诸多操作，如检入（CHECK IN），检出（CHECK OUT），合并（MERGE），提交（PROMOTE），打回（DEMOTE）、快照（TAKE SNAPSHOT）包的创建（CREATE PACKAGE）等操作。这些操作均基于客户/服务器的基础上进行的。客户——开发者、测试员、经理等等，在客户端建立与配置管理服务器的连接，在有关的环境下操作，最终完成软件的开发、测试和发布。他们的关系如下图：



二、Release 模型中的操作

一般的配置管理软件都提供了诸如检入、检出之类的基本操作功能，使用户的软件开发纳入到控制管理中，更完善的配置管理软件还提供了软件开发的解决方案，在此基础上有更高级的操作，如提交、打回、快照、包的创建等等。以上所说的检入、检出、合并、提交、打回、快照、包的创建等操作的含义如下：

检入（CHECK IN）：将本地文件拷贝到服务器的用户环境中，或者仅仅释放一个保留文件。服务器端以增量式存储仅保留版本间的差别。可以以更改、释放和保持方式进行

检入操作。

检出（CHECK OUT）：将服务器的用户环境中的文件取到本地的工作环境，或仅仅将一个文件置上保留标志，并不真正将文件拷贝到本地。可以以可更改、只读、同步等方式检出。

合并（MERGE）：将代码的许多分支合并到主干，形成新的版本，有并行合并和交互合并两种形式。

提交（PROMOTE）：将一个包从生命周期中的一个状态推进到一个新的状态，以执行新的过程。

打回（DEMOTE）：将一个包从生命周期中的一个状态退回到上一个的状态，再次执行原过程。

快照（TAKE SNAPSHOT）：记录软件开发过程中的某一点的信息如最新版本、修改记录和修改日期等，形成该点的视图。

包的创建（CREATE PACKAGE）：创建虚拟的软件包，许多操作与包相关联，如检入，检出，合并等。

每一种操作在不同的阶段的执行许可都不尽相同，以下是一般情况下各状态所能进行的操作过程、过程类型及执行许可：

状态名	过程	过程类型	执行许可
开发	包的创建	创建包过程	任何人
	可更改的检出	检出过程	开发者
	只读的检出	检出过程	任何人
	文件项的检入	检入过程	开发者
	提交到测试状态	提交[推进]过程	开发者，开发经理
	版本列表	版本列表过程	任何人
	删除版本	删除版本过程	开发经理
	删除文件项	删除文件项过程	开发经理
	跨环境合并	跨环境合并过程	开发经理
	交互合并	交互合并过程	开发者，开发经理
	显示版本间的区别	显示版本间区别过程	任何人
	软件包发布报告	UDP过程	开发者，开发经理
测试	包的创建	创建包过程	任何人
	只读的检出	检出过程	任何人
	提交到发布状态	提交[推进]过程	测试经理
	打回到开发状态	打回过程	测试经理
	版本列表	版本列表过程	任何人
发布	只读的检出	检出过程	任何人
	快照	快照过程	开发经理

注：UDP过程：用户自定义过程

三、Release 模型下的一般开发方法

要在Release模型的环境中工作，首先得在开发状态创建一个包或者从问题跟踪的环境中一个包移动到Release模型的环境的开发状态。如果要作修改，则文件必须从相应的包中以可更改方式检出，当文件从相应的包中以可更改方式检出之后，该文件即被置上保留标志——Reserve Tag，则其他任何人不能修改该文件。当文件修改完毕检入回去之后，保留标志即被释放，其他人才可以将该文件检出修改。

如果开发工作进行完毕，则可以进行测试。在该点，包和修改过的文件即被提交到测试。在测试状态下，可以看见文件新的版本，测试可以用这些版本的文件构造新的产品。如果在测试的时候发现错误，可以将包打回到开发状态，以使新的更新可以合并进去，然后再提交到测试。测试完毕以后，将包提交到Release[发布]状态。

以上的情形是常见的，以不同的方法开发代码，如果没有对怎样进行工作进行通盘的考虑是很难描述开发过程的。下面的情形经历同样的过程，但是开发过程中所发生的事情的描述更加具体。该情形仅作为一例，实际工作并不一定完全如此。

工程的主管收到设计文档之后，为每一个需更改的功能创建一个包，与此同时，评审组提出十个新的问题需要包的相关开发人员来解决；接着有些人开始解决问题，有些人则将新的特征合并到下一个Release，当开发人员开始工作时，他们将特定包中的文件以可更改方式检出，当文件被检出之后，即被置上保留标志，也就是说被锁定了，其他开发人员只有在该文件被检入回服务器之后，才能够将该文件检出更改。开发者将文件检出更改，并编译，如果没有编译错误，则可以进行该代码上的一次基本单元测试，结束之后，将这些修改过的文件检入到服务器中，该文件即被释放，其他人可以使用了。开发以这样的形式持续进行，直到所有的代码都已开发完毕，可以进行集成测试。如果集成测试中发现了任何错误，这些文件需要再次检出，进行必要的修改。当集成测试结束时，该软件就可以交给测试组进行测试了。在该点上，项目主管决定那些包可以提交到新的状态，并且运行测试，得出与保留的包的差异报告，看他们之间有没有冲突。差异报告着重于被提交的包的列表和那些文件被修改了，还有所有保留的包以及包中被修改的文件。如果保留包中存在文件比提交的包的版本要低，则存在差异，需要产生差异报告。

一旦所有的差异问题解决了，这些包可以提交到测试组。当测试组收到提交通知，就将所有的文件检出，则创建测试构造，采用构造进行查证和确认测试。只要测试过程中发现了问题，就要测试组和开发人员共同来决定是否需要将该包打回以进行修补、合并，或创建新的包。

测试完成之后，在测试状态创建新的包并将可执行文件检入，所有在测试状态的包将提交到发布状态。发布组将可执行文件从服务器中检出创建发布媒体，发布的代码可以采用一种软件发布包的形式进行分发，例如用PLATINUM的Safer program。

这种模式还有其他很多用法，但是关键的一点是每个组之间要保持代码的版本的独立性，这样就可以在开发时不影响测试和发布。

四、Release 模型下的并行开发方法

在以上情形下项目主管是由项目主管产生差异报告。该差异报告尤其重要，因为开发人员将代码检出之后要修补Bug #15，这也包括对Bug #9的修补，将Bug #9更改之后的文件与Bug #15合并，并检入回服务器。接着，其他的开发人员将该文件检出以修补Bug #2，因为Bug #15的修补是先进行的，所以Bug #2的修补将基于Bug #15修补进行。现在要准备进行测试，但是，Bug #15 和 Bug #9还没有完成，Bug #2准备提交到测试。由于，Bug #2基于Bug #15，而Bug #15基于Bug #9，如果现在提交Bug #2，测试构造将要失败，因为Bug #15 和Bug #9的文件project.h有变化，该文件必须是正确的才能进行正确的编译。

文件	版本	包名
Project.c	V2	Bug#9
Project.c	V3	Bug#15
Project.c	V4	Bug#2
Project.h	V3	Bug#15
Project.h	V4	Bug#9

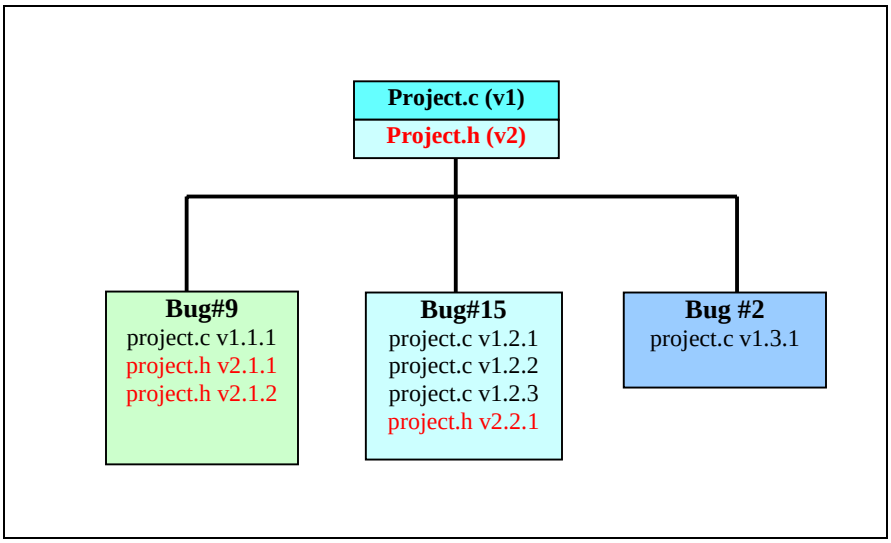
[开发视图]

文件	版本	包名
Project.c	V2	Bug#9
Project.c	V3	Bug#15
Project.c	V4	Bug#2

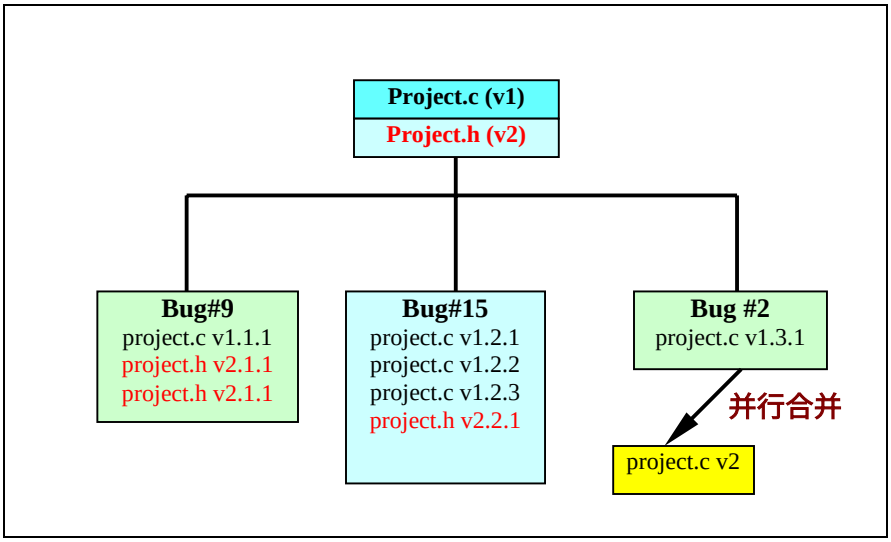
[测试视图]

人们常常在这种类型下实施并行开发。并行开发允许将每个包作为独立的分支存在。在为下一个软件版本发布而进行合并时，需要将分支上的代码合并到主线上。如果有两个或两个以上的包修改了同一个文件，合并的问题一定要解决。

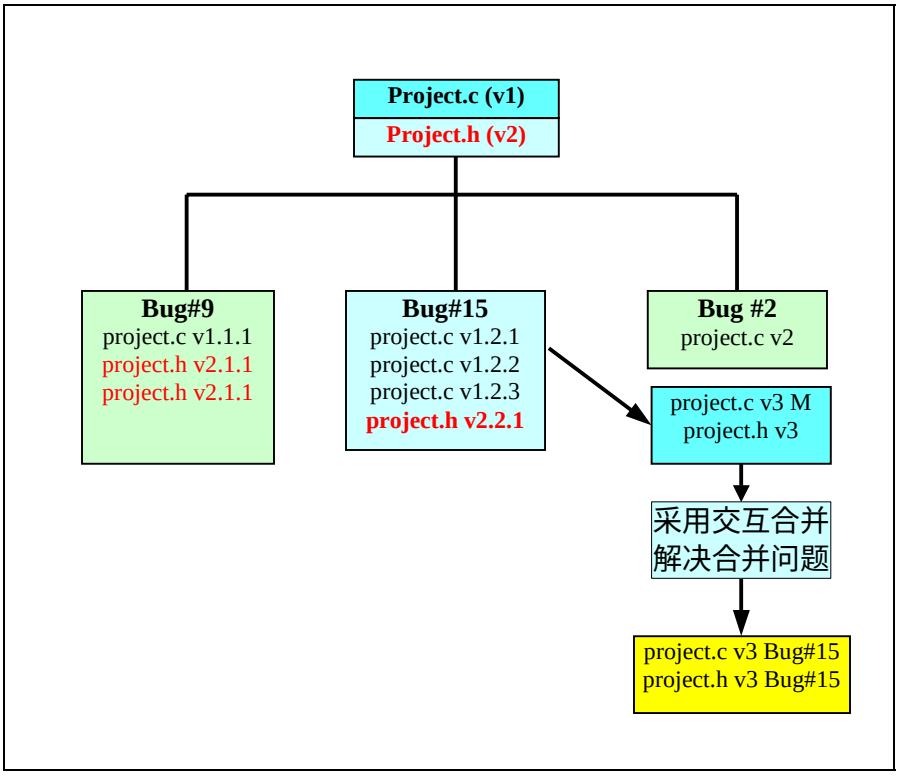
在上例中，将project.c的版本1以并行模式检出，首先，从包Bug #9以可更改的并行方式将该文件检出。这样就创建了第一个分支。而后，可以从包Bug #15以可更改的并行方式将同一个文件检出。在这种情形下，并不需要等待Bug #9完成编码，因为已经创建了一个新的分支，Bug #15创建了第二个分支。现在同样可以将project.c的版本1检出，以修补包Bug #2，这是第三个分支。



现在开始准备提交包Bug #2。先运行并行合并过程之后，创建了project.c的版本2，提交包Bug #2。



后来准备提交包Bug #15，同样运行并行合并过程之后，这样就创建了一个带合并标志（Merge Tag）M的project.c的版本3，但是版本3显示合并冲突，然后执行交互合并过程将Bug #15和Bug #2中的更改合并到一起。交互合并之后，版本3的合并标志将被除去。因为Bug #15同样影响project.h，所以在执行并行合并过程之后，同样创建了project.h的版本3，而没有合并的报告。当合并全部完成之后，可以将Bug #15提交到测试。



五、Release 模型之外的其他模型[题外话]

在众多的配置管理软件中，一般都提供Release模型所需要的基本元件，用户可以根据自己实际条件配置软件开发的环境，建立起Release模型。实际上，有些软件还提供了诸如问题跟踪模型、并行开发模型、产品模型和版本控制模型等很多模型，用户可以选择一种适合与自己的开发模型，组织本单位的软件开发。有关配置管理软件的资源可以在 *http://www.iac.honeywell.com/Pub/Tech/CM/index.htm* 等网页中查找到。

参考资料（网络资源）：

1．http://www.configuration.org
2．http://www.iac.honeywell.com/Pub/Tech/CM/index.htm
3．Platinum CCC/Harvest Process Models　　PLATINUM Technology Inc. 1997年9月

本文已发表于《中国计算机报》1999 年 7 月 12 日第 49 期 E9 软件版