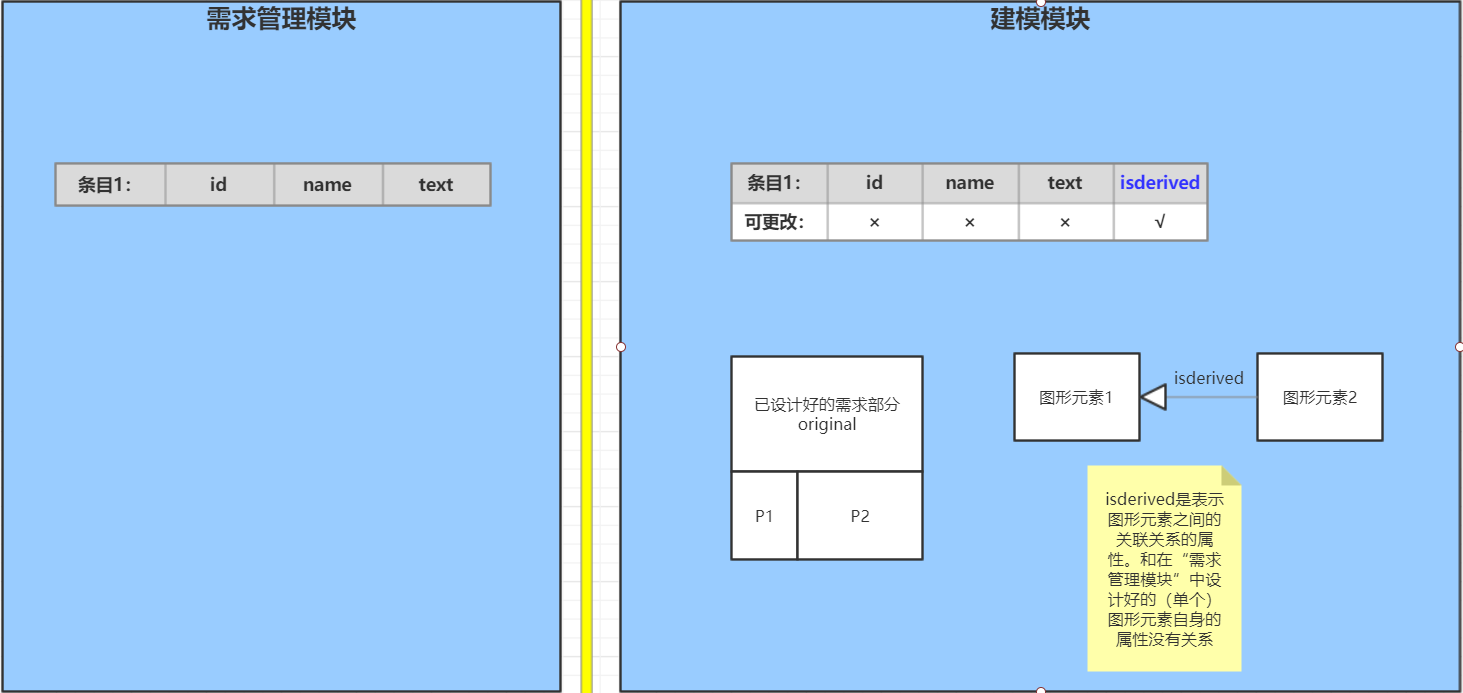
### 需求文档版本管理

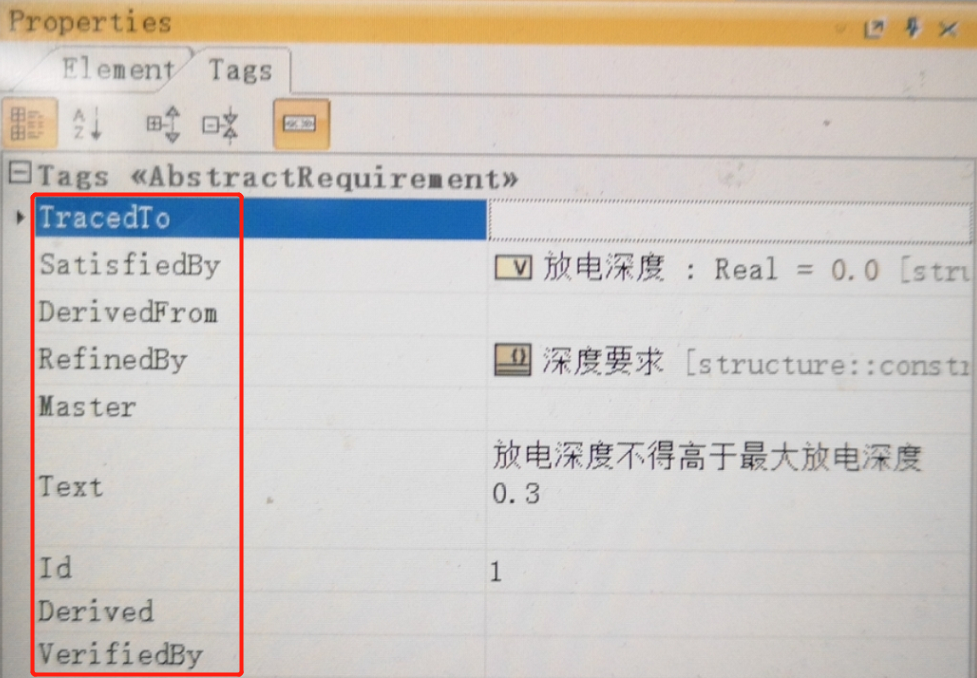
1

在整个系统中“需求管理模块”和“建模模块”之间的版本管理是相对独立的，各自有自己的版本管理。

更改需求，只能在“需求管理模块”中更改。“需求管理模块”可以分享给“建模模块”，并给予不同的权限，分为两种：可读可用、可读可用可改。可改是只能改动除“需求管理模块”中定义的属性，即只能改动如两个图形之间的关联关系这类属性。

举例：如在“需求管理模块”中定义“条目”有id、name、text属性，那在“需求管理模块”中这3个属性不能更改，只能新增一个isderived属性，可以填Verify、Copy、Satisfy等图形元素之间的关联关系。不更改单个“条目”的已有的属性。

设置需求元素的固定不可修改的基础属性为以下9种（参考CSM）：



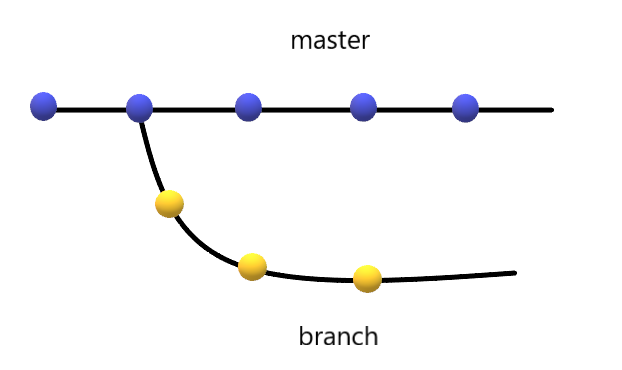
2

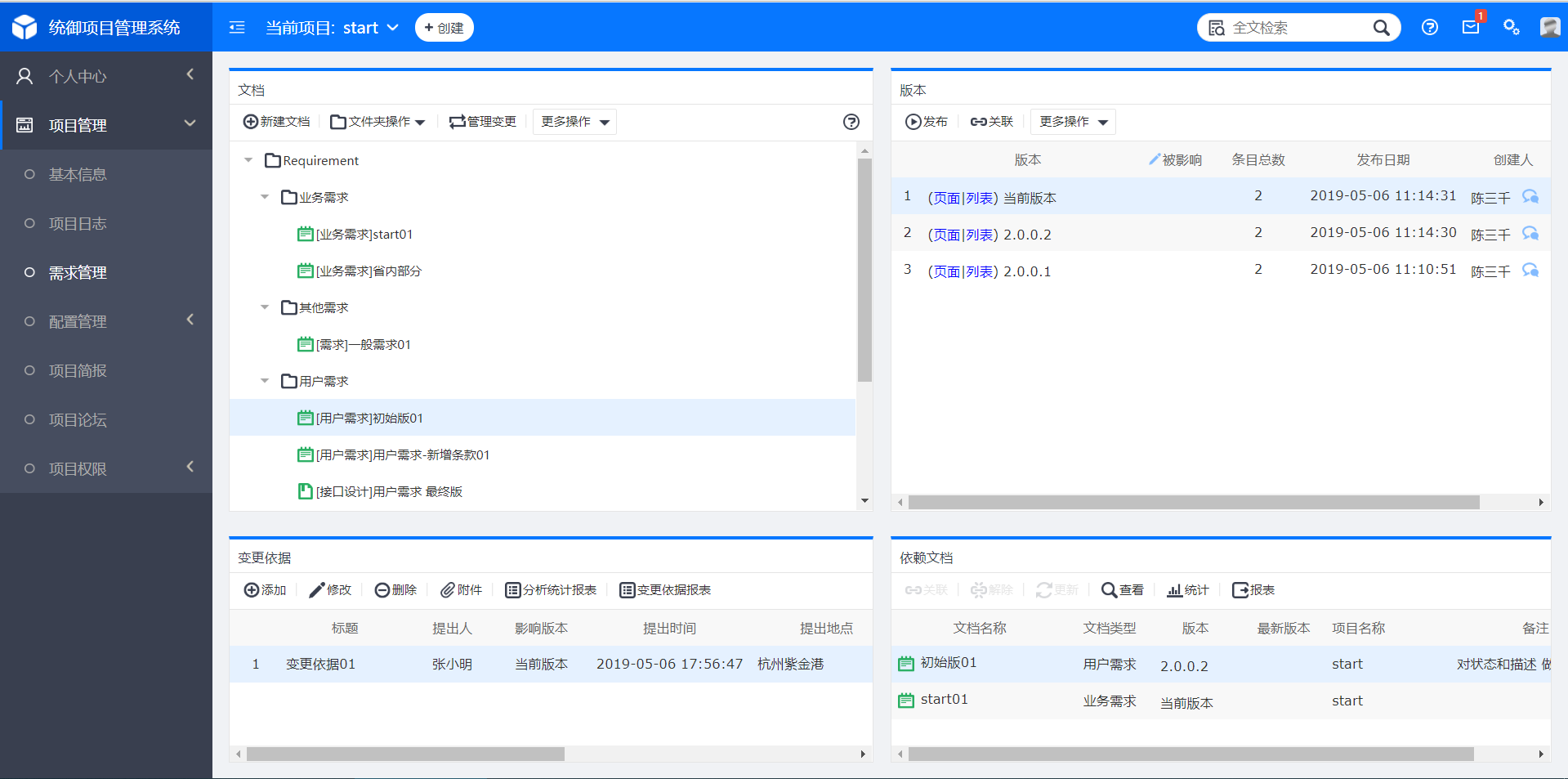
需求表版本分为master和branch。发布新的版本都是从master上发布；可以创建多个 branch，其不可发布但是可以保存在那（即某条branch上可以创建新的需求版本但是不发布）（例：一个需求分析人员可以为自己的需求不断添加branch，最后将某几个branch合并到一个，然后提交到master去，接着发布新版本），如果需要发布branch里的需求，可以将最新的branch提交到master上进行发布。

和okit上的比，多了branch，主要是为了：

* + 1. 可以创建多个需求版本草稿（如发布了的有version1.1、version1.2，未发布的分支有Mikebranch0.1、branch0.11、Jackbranch0.2）；
    2. 若只有一个master，那么想要把多个不成熟的还不能发布的需求表版本放上去是实现不了的，因为没编辑完当前需求版本，想编辑另一个需求版本时得先发布出去才能再去编辑另一个版本；若可以创建多个branch那就可以实现。
    3. 可以按类别/需要创建多个分支；

【版本号】master上的需求表版本号只能由数字和 . 组成，且为递增。Branch上的版本号只能有大小写字母、数字和 . 组成。





3

关于发布的新版需求表，“建模模块”和“需求管理模块”的关系是：“建模模块”去 “需求管理模块”里获取。

“需求管理模块”获取最新版本需求表的方式是：通过“建模模块”提供的接口主动扫描查看有无更新并同步最新版本的需求表到自己模块。查看/获取更新的基本单位是需求表。

“需求管理模块”同步最新版本的需求表有两种方式：

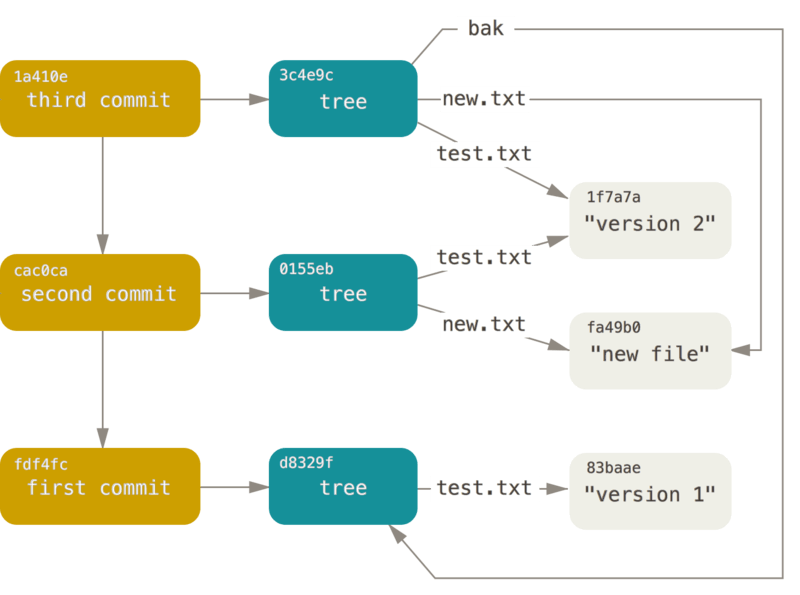
1. “建模模块”在启动的时候扫描一遍 “需求管理模块”，看有无更新，有更新就下载来导入requirement table中。
2. 界面上放置一个refresh按钮，用户可以主动点击refresh按钮去同步最新版需求表，若有更新点击后会自动同步过来。这里面涉及到权限操作。“建模模块”分发权限给“需求管理模块”，分发的权限有大有小。
   * 如果只有“需求管理模块”没有“建模模块”：那么“需求管理模块”有自己独立的一套版本管理；
   * 反之同理；
   * 如果既有“需求管理模块”也有“建模模块”：那么“需求管理模块”这边所有发布了的版本归“需求管理模块”这边的版本控制管理。需求表被“建模模块”调用导入requirement table生成了图形元素，那些图形元素归“建模模块”那边的版本控制管理。

**4实现**

文档版本管理是以“需求条目”为单位进行的。具体实施采用JGit（git的Java实现版本）那一套。即将一条“条目”装载在一个 block里，用一个tree来指向block，一个文档就是由索引指向一个commit，再由commit指向tree，最后由tree指向对应block（block就是装载条目的容器）。如下图：



举例：



其中 .txt（如test.txt或new.txt）就相当于一个“条目”（即一条需求），version 1、 new file和version 2是条目里的内容。

每个小块左上角的数值是用作标识的哈希值。