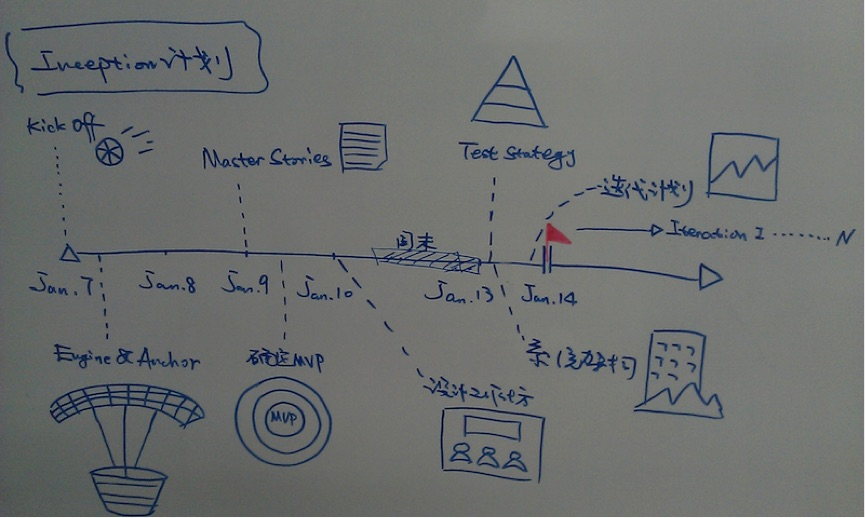
## 【案例】版本升级系统的先启阶段

【案例】版本升级系统的先启阶段

朗读人: 东急之花00′00′′ | 3.6M

我曾经作为咨询师参加了通信领域的一个产品的研发工作，该产品为版本升级系统，主要负责完成对基站软件的版本升级。整个系统的敏捷项目管理与架构设计由我负责，而我对该领域的需求知识可谓一窍不通，于是我决定针对该产品做一次先启活动。由于周期较短，我不能将太多时间耗费到这个先启阶段中，因而制定了一个为期一周的先启计划，如下图所示：



enter image description here

首先确定了产品的相关干系人，包括软件产品部部长、部门的大项目经理、版本管理人员、外场用服、项目组所有成员，然后召集他们召开了此次先启阶段的启动会议。我们在会议中经过充分交流与讨论，一起确定了版本升级系统的愿景与目标，明确了项目的当前状态和我们期望达到的未来状态。本次要开发的版本升级系统其实是对旧的版本升级系统的彻底改造，之前的版本升级系统完全部署在基站主控板中，占用了主控板极为珍贵的 CPU 和内存资源，同时也不利于与后端运行在 Java 虚拟机上网管平台的集成。因此，本次产品开发就是要将与版本升级有关的功能从主控板中彻底剥离出来，建立一个全新的后台架构，同时还要保证系统架构的可扩展性，以应对未来需求的变化。

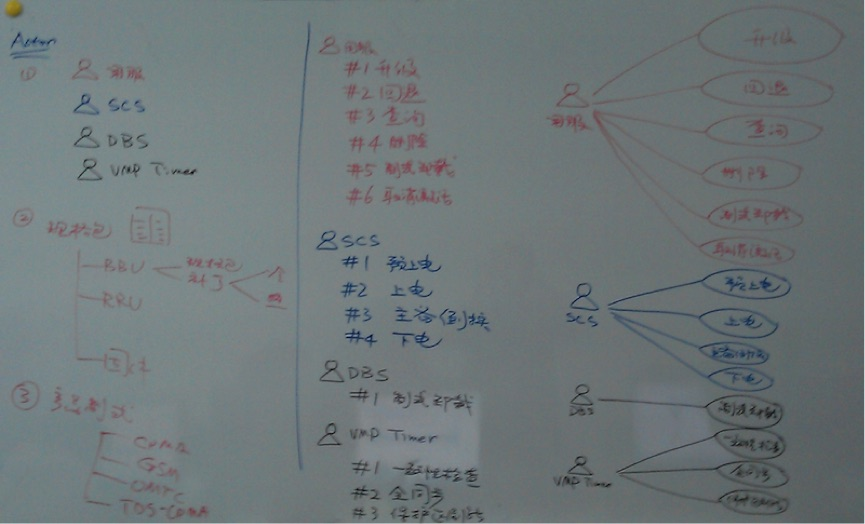
在这个项目中，版本管理人员与外场用服就是我们的客户。此外，项目组的需求分析师和开发人员全都参与了旧产品的研发，他们对整个系统的需求已有比较深入的理解。然而为了完成新产品的愿景与目标，我们仍有必要对产品需求进行全面的梳理。

在先启阶段，需求分析的重点是对史诗级故事和主故事的识别与拆分。由于该系统的主要工作是对原有系统进行重写，除了极个别的新需求外，需求功能与范围都是明确无误的。我们引入了精益软件开发的 MVP（最小可用产品）实践，通过它获得产品的发布计划与迭代计划，又需要运用领域驱动设计来指导整个项目的研发，因此，重新梳理需求很有必要。需求的梳理过程要求整个团队的成员都参与，这是一个极佳的通过沟通与协作明确需求的机会。

在梳理需求时， **首先还是从调用者（参与角色）的角度出发** ，这相当于用例中的参与者（Actor），从这个视角去分析需求功能，可以更容易帮助我们去识别需求的价值，并找到系统的边界，该项目的情况比较特殊，从使用者角度看，仅有外场用服是系统的参与者；但是，调用或触发产品功能的角色却不仅限于用服，一些参与的系统也可以视为参与者，其中包括 SCS、DBS 和 VMP Timer。

产品的主要功能是管理版本规格包，包括升级规格包、回退规格包与删除规格包。版本规格包的类型相对比较复杂，分别分为：BBU、RRU 和固件，BBU 和 RRU 还包含了补丁规格包。补丁规格包又分为冷补丁和热补丁，这些规格包可以组合，同时，不同产品制式包含的规格包也有所不同。

在需求分析时，我们将自己的视角切换到参与者的观察角度，可以得出属于该参与者的主要用例，这些用例就成为了主故事列表的重要输入：

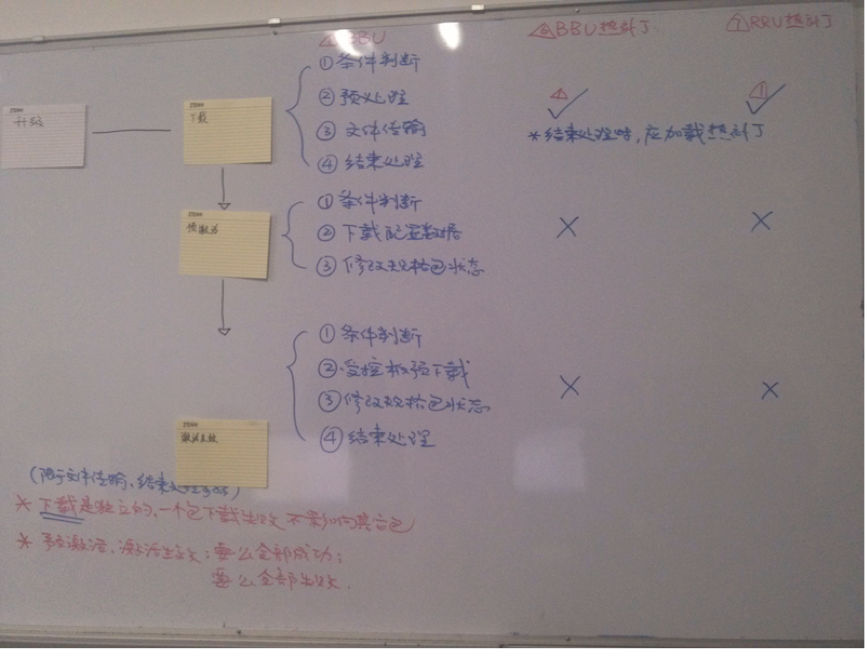


enter image description here

作为主故事的用例有助于我们建立对整个系统的全局感官，但对于系统的核心领域，这些识别出来的主要用例显得太粗犷，需要继续细化。我借助了卡片以更加 **可视化的方式** 来表现功能之间的层次关系。同时，根据规格包的类型，又以表格的方式来展现各种规格包之间的异同。下面的两张图片展现了版本升级功能的需求分解：



enter image description here



enter image description here

通过这样的需求分析活动，团队成员基本上就系统的功能达成了一致意见。现在，我们就可以把最高层次的特性卡排列在白板上，根据大家对系统的认识来确定优先级，并基于功能完备性来选择可以放在同一个 MVP 的特性卡上：



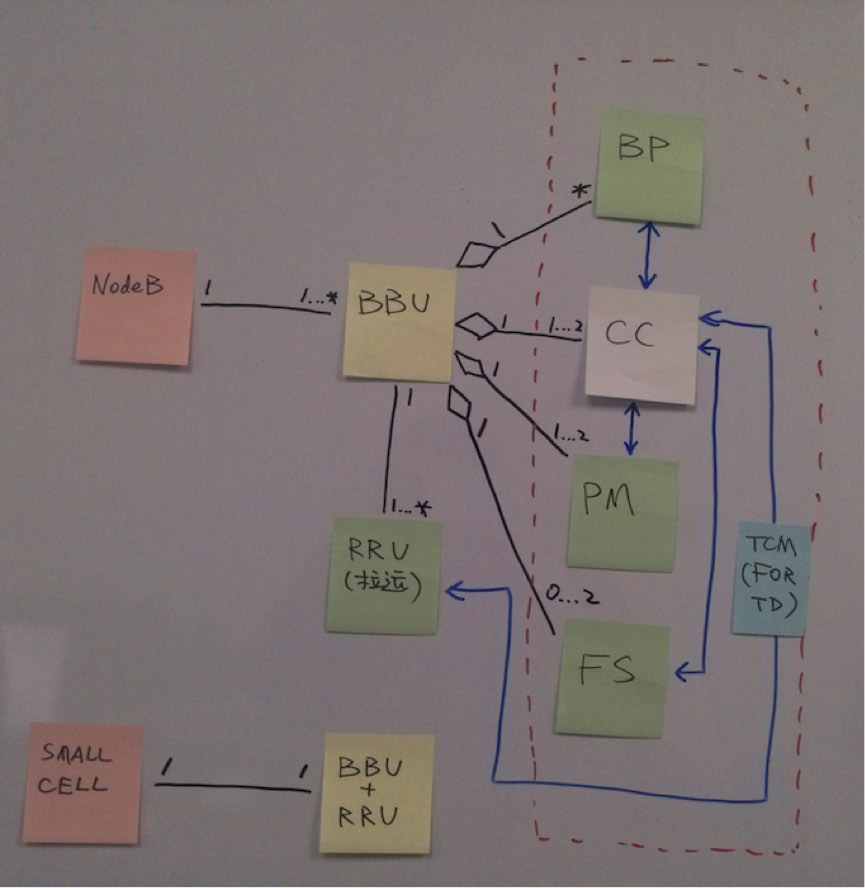
enter image description here

我们将整个项目周期划分为四个 MVP。我们一致认为版本的升级是整个系统最重要也是最主要的功能，它的价值最高，可能存在的风险也最高，完全有理由放在迭代的最前面。最初，我们并没有考虑将“查询”功能放在最高优先级，相较于“升级”和“上电”，甚至于“回退”，它的优先级都有所不如。但当我提出如何在每个发布阶段进行演示时，大家才认识到如果不尽快实现查询功能，可能会使得我们发布的最小版本并不可用。当然也可以考虑命令式脚本进行查询，但这样的开发成本相较于开发 UI 的查询功能而言，并不能减少多少。

虽然所有规格包的升级功能在重要性方面都要高于“回退”以及“全同步”，但由于升级功能的工作量最多，所有规格包的升级功能开发完毕大约会占用超过 1/2 的时间，从最小可用的角度来看，我们选择了整体功能的完备性。因此将固件与补丁包的升级放到了优先级低的 MVP 3。而且我们认识到，一旦开发完 BBU 和 RRU 规格包的升级，对于固件与补丁包而言，实现就变得简单了。

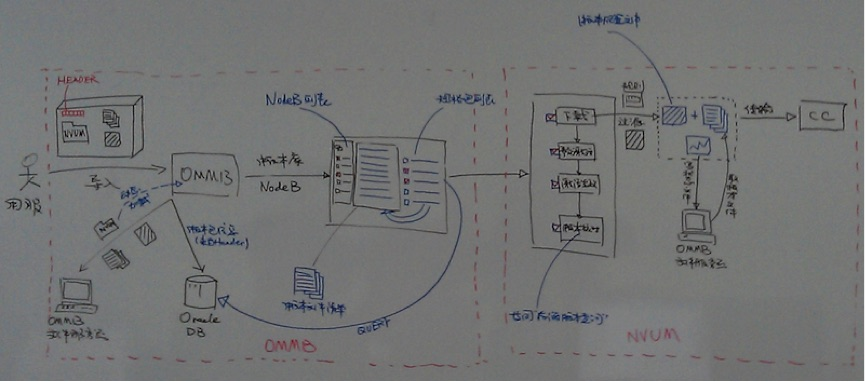
我们之所以考虑将“全同步”与“预上电”功能放到了 MVP 2，与它们的重要程度无关，从重要程度讲，它们甚至要低于升级与回退。但由于 **这些功能的开发需要与其他团队协作** ，若能将这部分功能的迭代周期提前，将有利于我们更好地与其他团队进行协作。毕竟，与外部团队的协作存在较大风险。

在分析需求时，团队还统一了主要的领域术语，确定了各个领域对象之间的关系，为了更加形象地可视化模型，我使用了不同颜色的即时贴来表现模型，但我并没有采用四色建模法，这里的颜色特征没有设计意义，只有业务意义。例如，绿色即时贴代表受控板，白色即时贴为主控板，即 CC。确认基站的术语经过了一番讨论，因为基站又可称之为网元，但最后我们一致确认用领域概念 Node B 来表示基站。在讨论过程中，团队成员提出了还有一种特殊的基站，这种基站的 BBU 和 RRU 是在一块单板上的，最后我们将其称之为 Small Node B：



enter image description here

通过用例建立的主故事列表以及利用可视化手段建立的领域模型都是一种静态模型，并不足以清晰地表达整个产品的核心功能。从用服参与者开始，我们模拟了整个版本升级的过程，这类似作战时使用沙盘进行演练一般。最终，我们以流程图的形式来生动表达这个过程：



enter image description here

可以看到，在版本升级系统的整个先启阶段，我们不断地针对领域需求与相关干系人和团队成员进行有效地沟通。在沟通过程中，我们常常通过使用各种颜色的即时贴、卡片和白板这种极为方便的工具来帮助我们理解需求，并借助如绘制用例图、流程图、领域模型等可视化手段来分析问题域，澄清需求，避免误解，最终达成领域知识的共识。为了真实还原整个先启阶段，我保留了当时在白板上绘制的用例图、MVP 发布阶段划分、核心领域模型与流程图。

下一章

互动评论



评论



sicun7 个月前

图片看不清 严重影响阅读

[鼓掌回复](javascript:void(0))

评论



江文奇8 个月前

图片无法放大

[1回复](javascript:void(0))

评论

查看更多

## 更多资源下载交流请加微信：Morstrong,加入永久会员,网盘更新更快捷！

# 本资源由微信公众号：光明顶一号，提供支持