**如意通科技产品支持 - 全局相关联的Scrum敏捷软件开发过程**

# 长文慎入，建议在电脑上阅读！

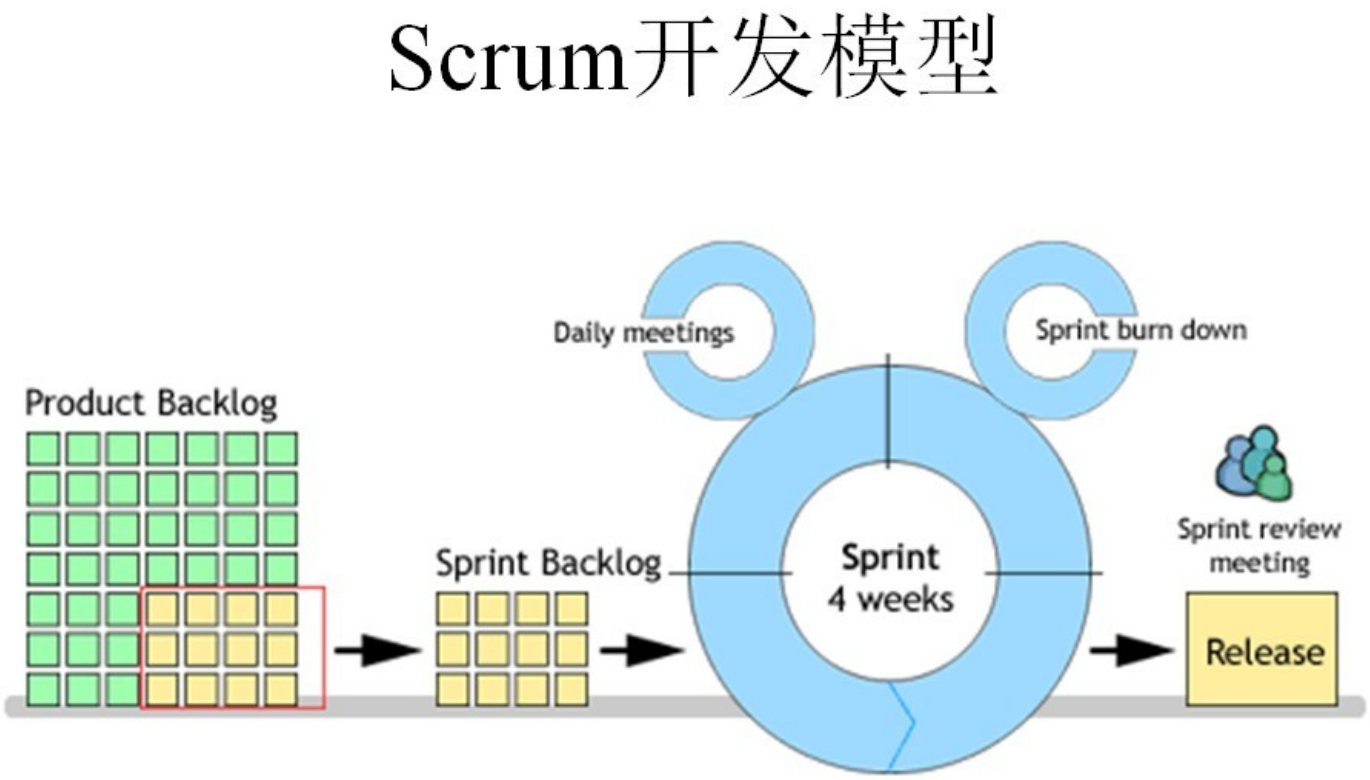
这里说说咱们如意通科技是怎么用Scrum来进行敏捷软件开发的。前面一段摘录了一些文章来介绍标准Scrum（熟悉者可跳过），后面一段介绍了如意通科技在Scrum实践中的一些具体做法。

# Scrum简介

之所以这里还要介绍一下Scrum，一方面给不了解的人看看，另一方面是为了后面把我们如意通科技的Scrum实践和标准的Scrum过程做个对比

Scrum 是一个用于开发和维持复杂产品的框架
，是一个增量的、迭代的开发过程。在这个框架中，整个开发过程由若干个短的迭代周期组成，一个短的迭代周期称为一个Sprint，每个Sprint的建议长度是2到4周(互联网产品研发可以使用1周的Sprint)。在Scrum中，使用产品Backlog来管理产品的需求，产品backlog是一个按照商业价值排序的需求列表，列表条目的体现形式通常为用户故事。Scrum团队总是先开发对客户具有较高价值的需求。在Sprint中，Scrum团队从产品Backlog中挑选最高优先级的需求进行开发。挑选的需求在Sprint计划会议上经过讨论、分析和估算得到相应的任务列表，我们称它为Sprint backlog。在每个迭代结束时，Scrum团队将递交潜在可交付的产品增量。
Scrum起源于软件开发项目，但它适用于任何复杂的或是创新性的项目。

---以上摘自 Scrum中文网




# Scrum角色

---以下摘自 维基百科Scrum词条

Scrum当中定义了许多角色。按照对开发过程的参与情况，这些角色被分为两组，即猪组和鸡组。这个分组方法的由来是一个关于猪和鸡合伙开餐馆的笑话：

一天，一头猪和一只鸡在路上散步。鸡对猪说：“ 嗨，我们合伙开一家餐馆怎么样？ ”猪回头看了一下鸡说：“ 好主意，那你准备给餐馆起什么名字呢？ ”鸡想了想说：“ 叫‘ 火
腿和鸡蛋 ’怎么样？ ”“ 那可不行 ”，猪说：“ 我把自己全搭进去了，而你只是参与而已。 ”

## 猪组的成员

猪是在Scrum过程中全身投入专案的各种人物，他们在专案中承担实际工作。他们有些像上边那个笑话里的猪，要把自己身上的肉贡献出来。

**产品负责人(product owner)**

产品负责人代表了客户的意愿。这保证了Scrum团队在做从业务角度来说正确的事情。产品负责人编写用户故事，排出优先级，并放入产品订单。

**Scrum主管（或促进者）(scrum master)**

Scrum主管促进
Scrum过程，他的主要工作是去除那些影响团队交付冲刺目标的障碍。Scrum主管并非团队的领导（因为团队是自我组织的），而是一个负责屏蔽外界对开发团队的干扰的角色。Scrum主管确保Scrum过程被按照初衷使用。Scrum主管是规则的执行者。

**开发团队(dev team)**

负责交付产品的团队。一个团队通常由5至9名具有跨职能技能的人（设计者，开发者等）组成，承担实际的开发工作。

## 鸡组的成员

鸡并不是实际Scrum过程的一部分，但是必须考虑他们。敏捷
方法的一个重要方面是使得用户和利益相关　　者参与到过程中的实践。参与每一个冲刺的评审和计划，并提供反馈对于这些人来说是非常重要的。

**用户**

软件是为了人而开发的。有人说，“ 假如森林里有一棵树倒下了，但没有被人听到，那么它算是发出了声音吗？ ”同样地，人们可以说，“ 假如软件没有被使用，那么它算是被开发出来了么？ ”

**利益相关者（客户，提供商）**

影响项目成功的人，但只直接参与冲刺评审过程。

**经理**

为产品开发团体搭建环境的人。

# Scrum活动

## 产品待办事项列表梳理

---以下摘自 Scrum的五个活动

产品待办事项通常会很大,也很宽泛,而且想法会变来变去、优先级也会变化,所以产品待
办事项列表梳理是一个贯穿整个Scrum项目始终的活动。该活动包含但不限于以下的内容:

- 保持产品待办事项列表有序
- 把看起来不再重要的事项移除或者降级
- 增加或提升涌现出来的或变得更重要的事项
- 将事项分解成更小的事项
- 将事项归并为更大的事项
- 对事项进行估算

产品待办事项列表梳理的一个最大好处是为即将到来的几个Sprint做准备。为此,梳理时会特别关注那些即将被实现的事项。需要考虑不少因素,这包括但不限于以下的内容:

理想情况下,下一个Sprint的备选事项都应该提升“ 商业价值 ”。
开发团队需要能够在一个Sprint内完成每一个事项。每个人都需要清楚预期产出是什么。

产品开发决定了,有可能需要其它的技能和输入。因此,产品待办事项列表梳理最好是所有团队成员都参与的活动,而不单单是产品负责人。

## Sprint计划会议

每个Sprint都以Sprint计划会议作为开始,
这是一个固定时长的会议，在这个会议中,Scrum团队共同选择和理解在即将到来的Sprint中要完成的工作。

整个团队都要参加Sprint计划会议。针对排好序的产品待办事项列表(Product Backlog),产
品负责人和开发团队成员讨论每个事项,并对该事项达成共识,包括根据当前的“完成的定
义”,为了完成该事项所需要完成的所有事情。所有的Scrum会议都是限定时长的。Sprint计划会议推荐时长是Sprint中的每周对应两小时或者更少(译者注:比如,一个Sprint包含2个星
期,则Sprint计划会议时长应为4个小时或者更少)。因为会议是限制时长长的,Sprint计划会议的成功十分依赖于产品待办事项列表的质量。这就是产品待办事项列表梳理十分重要的原因。

在Scrum中,Sprint计划会议有两部分:

1. 决定在Sprint中需要完成哪些工作
2. 决定这些工作如何完成

**第一部分:需要完成哪些工作?**

在会议的第一部分,产品负责人向开发团队介绍排好序的产品待办事项,整个Scrum团队共同理解这些工作。

Sprint中需要完成的产品待办事项数目完全由开发团队决定。为了决定做多少,开发团队需要考虑当前产品增量的状态,团队过去的工作情况,团队当前的生产能力,以及排好序的产品待办事项列表。做多少工作只能由开发团队决定。产品负责人或任何其它人,都不能给开发团队强加更多的工作量。

通常Sprint都有个目标,称作Sprint目标。这将十分有效地帮助大家更加专注于需要完成的工
作的本质,而不必花太多精力去关注那些对于我们需要完成的工作并不重要的小小细节。

**第二部分:如何完成工作?**

在会议的第二部分里里,开发团队需要根据当前的“完成的定义”一起决定如何实现下一个产品增
量。他们进行行足足够的设计和计划,从而有信心可以在Sprint中完成所有工作。头几天的工作会
被分解成小小的单元,每个工作单元不超过一天。之后要完成的工作可以稍大大些,以后再对它 们进行行分解。

决定如何完成工作是开发团队的职责,决定做什么则是产品负责人的职责。

在计划会议的第二部分,产品负责人可以继续留下来回答问题,以及澄清一些误解。不管怎样,团队应该很容易找到产品负责人。

Sprint计划会议的产出
Sprint计划会议最终需要Scrum团队对Sprint需要完成工作的数量和复杂度达成共识,并预期在一个合理的条件范围内完成它们。开发团队预测并共同承诺他们要完成的工作量。
总而言言之:在Sprint计划会议中,开发团队和产品负责人一起考虑并讨论产品待办事项,确保他们对这些事项的理解,选择一些他们预测能完成的事项,创建足够详细的计划来确保他们能够完成这些事项。

最终产生的待办事项列表就是“Sprint待办事项列表(Sprint Backlog)”。

## 每日Scrum会议

开发团队是自组织的。开发团队通过每日Scrum会议来确认他们仍然可以实现Sprint的目标。
这个会议每天在同样的时间和同样的地点召开。每一个开发团队成员需要提供以下三点信息:


从上一个每日Scrum到现在,我完成了什么; 从现在到下一个每日Scrum,我计划完成什么; 有什么阻碍了我的进展。

每日Scrum中可能有简要的问题澄清和回答,但是不应该有任何话题的讨论。通常,许多团队
会在每日Scrum之后马马上开会处理他们遇到的任何问题。

每日Scrum既不是向管理层汇报,也不是向产品负责人人或者ScrumMaster汇报。它是一个开发
团队内部的沟通会议,来保证他们对现状有一致的了解。只有Scrum团队的成员,包括
ScrumMaster和产品负责人人,可以在会议中发言言。其他感兴趣的人人可以来旁听。在必要时,
开发团队会基于会议中的发现重新组织他们的工作来完成Sprint的罒目目标。

每日Scrum是Scrum的一个关键组成部分,它可以带来透明性,信任和更好的绩效。它能帮助
快速发现问题,并促进团队的自组织和自立立。所有Scrum会议都是限定时长长的。每日Scrum通 常不超过15分钟。

### Sprint评审会议

Sprint结束时,Scrum团队和相关人人员一起评审Sprint的产出。所有Scrum会议都是限定时长长
的,Sprint评审会议的推荐时长长是Sprint中的每一周对应一个小时(译者注:比比如,一个Sprint
包含2个星期,则Sprint评审会议时长为2个小时)。

讨论围绕着Sprint中完成的产品增量。由于Sprint的产出会涉及到一些人人的“ 利益 ”,因此一个明
智的做法是邀请他们参加这个会议,这会很有帮助。这个会议是个非非正式的会议,帮助大大家
了解我们罒目目前进展到哪里里,并一起讨论我们下一步如何推进。每个人人都可以在Sprint评审会议
上发表意见见。当然,产品负责人人会对未来做出最终的决定,并适当地调整产品待办事项列表 (Product Backlog)。

团队会找到他们自己的方式来开Sprint评审会议。通常会演示示产品增量,整个小组也会经常讨论他们在Sprint中观察到了什么、有哪些
新的产品想法出现。他们还会讨论产品待办事项列表 的状态、可能的完成日期以及在这些日期前能完成什么。

Sprint评审会议向每个人人展示示了当前产品增量的概况。因此,通常都会在Sprint评审会议中调 整产品待办事项列表。

### Sprint回顾会议

在每个Sprint结束后,Scrum团队会聚在一起开Sprint回顾会议,目的是回顾一下团队在流程人际关系以及工具方面做得如何。团队识别出哪
些做得好,哪些做得不好,并找出潜在 的改进事项,为将来的改进制定计划。所有的Scrum会议都是限定时长长的,Sprint回顾会议的
推荐时长长是Sprint中的每一周对应一个小时(译者注:比比如,一个Sprint包含2个星期,则 Sprint回顾会议时长长为2个小时)。

Scrum团队总是在Scrum的框架内,改进他们自己的流程。

## Scrum文档

--以下摘自 维基百科

### 产品订单

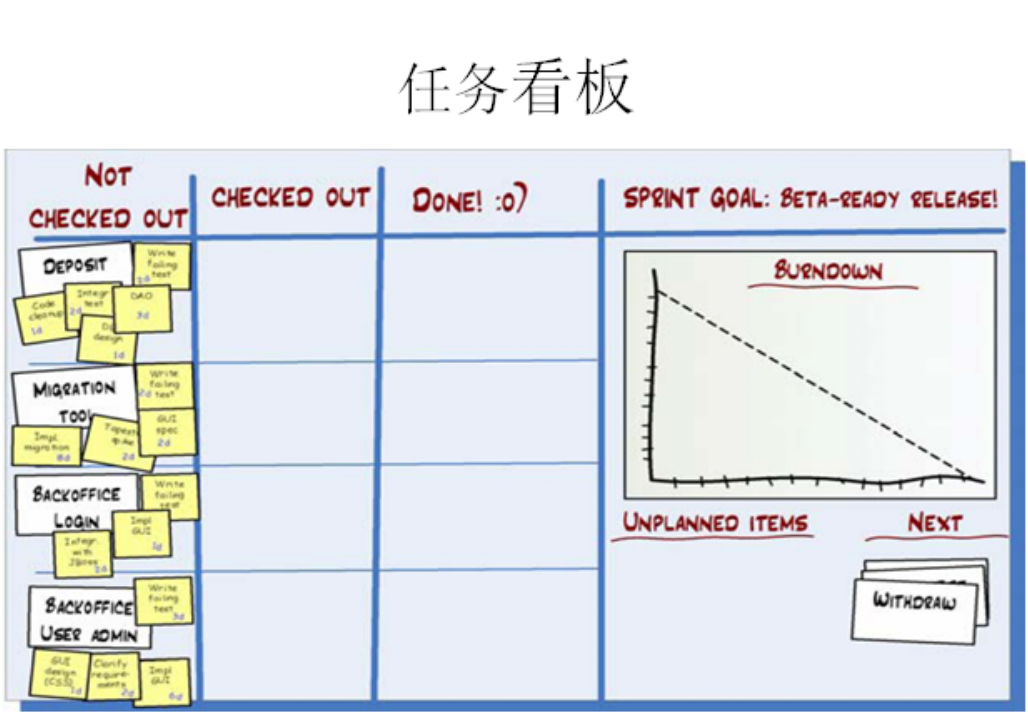
产品订单（product
backlog）是整个专案的概要文档。产品订单包括所有所需特性的粗略的描述。产品订单是关于将要生产什么样的产品。产品订单是开放的，每个人都可以编辑。产品订单包括粗略的估算，通常以天为单位。估算将帮助产品负责人衡量时程表和优先级（例如，如果"增加拼写检查"特性的估计需要花3天或3个月，将影响产品负责人对该特性的渴望）。

### 冲刺订单

冲刺订单（sprint
backlog）是大大细化了的文档，包含团队如何实现下一个冲刺的需求的信息。任务被分解为以小时为单位，没有任务可以超过16个小时。如果一个任务超过16个小时，那么它就应该被进一步分解。冲刺订单上的任务不会被分派，而是由团队成员签名认领他们喜爱的任务。
燃尽图[编辑]

### 燃尽图

燃尽图（burn down
chart）是一个公开展示的图表，显示当前冲刺中未完成的任务数目，或在冲刺订单上未完成的订单项的数目。不要把燃尽图与挣值图相混淆。燃尽图可能在一次冲刺的大部分时间内都维持平坦，但计划仍然可以按照既定时间进行。



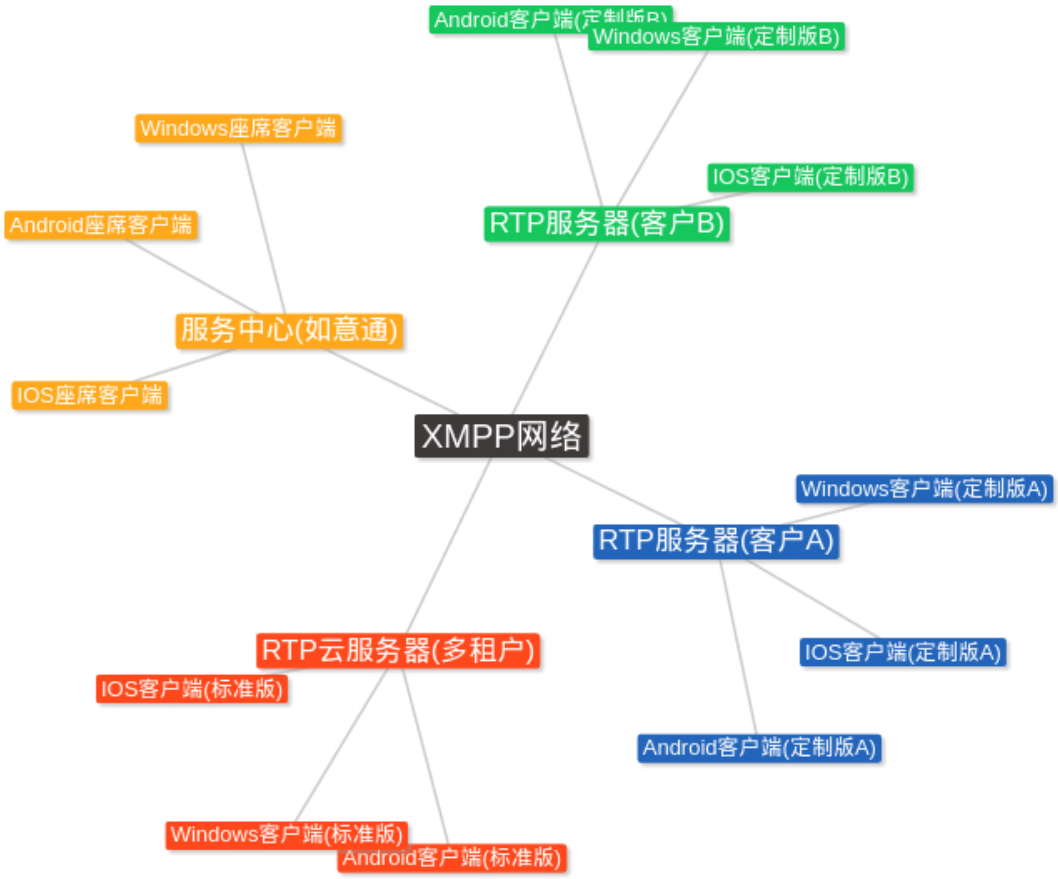

# 如意通科技的Scrum过程

前面讲了Scrum的一些标准做法，接下来说一下我们如意通科技的Scrum特点

- 全局相关联

如意通科技是一家做即时通讯软件的公司，所以从用户的视角来看，我们只是在做 **一个** 软件而已。实际如意通RTP产品系列包括 **至少三个** 不同架构的客户端，IOS, Android, Windows， **至少两个** 不同架构的服务器，RTP服务器， RTP云服务器，以及 **一些** 相对独立的扩展服务，比如我们的服务中心。我们提供给用户的大部分功能需要这些客户端和服务器软件之间的协作才能完成，所以我们既有多个Scrum项目/团队，在开发功能的时候各项目之间又互相依赖。

以下是RTP/RTP云网络示意图，源码参见 http://text2mindmap.com/FMWZs4j




举例来说，消息漫游功能，三个RTP客户端的聊天记录要能够漫游到其他客户端去，而RTP服务器和RTP云服务器都需要保存消息漫游记录，所以开发的时候我们需要先定义一个消息漫游协议，然后在RTP服务器上开发消息漫游功能，接下来在三个客户端上分别开发此功能，最后在RTP云服务器上开发此功能。

{{mermaid(消息漫游开发流程示例)
graph TB
开始(确定开发消息漫游功能) --> 设计协议
设计协议 --> 在RTP服务器上开发
在RTP服务器上开发 --> 在IOS客户端上开发
在IOS客户端上开发 --> 验证协议{协议有无问题}
验证协议{协议有无问题} -->|有| 设计协议
验证协议{协议有无问题} -->|无| 交付IOS客户端和RTP服务器
交付IOS客户端和RTP服务器 --> 在Android客户端上开发
交付IOS客户端和RTP服务器 --> 在Windows客户端上开发
交付IOS客户端和RTP服务器 --> 在RTP云服务器上开发
在Android客户端上开发 --> 结束(完成消息漫游功能)
在Windows客户端上开发 --> 结束(完成消息漫游功能)
在RTP云服务器上开发 --> 结束(完成消息漫游功能)
}}

接下来说说我们在Scrum过程中的一些做法

# Scrum角色

## 猪组的成员

包括开发各项目的开发人员，产品负责人等，也就是干活的人。

#### 共用的产品负责人

因为前面说的“ 全局相关联 ”
，我们产品线的几个Scrum项目是共用同一个产品负责人的，RTP各客户端和RTP服务器/RTP云服务器都是一个，我们把这个人叫做“ 产品经理 ”。

我们的产品负责人会做以下的事情:

1. 自己创建 Product backlog，也就是提出新需求
2. 对其他相关人员创建的 Product backlog 排出优先级，也就是对别人提出的新需求做一些管理
3. 跟进 Sprint backlog 的执行情况，也就是跟进当前进行中的迭代

我们的产品负责人无权做以下的事情:

1. 无权删除别人的 Product backlog，也就是不能删除别人提出的需求
2. 无权向正在进行的 Sprint 中加入新的 Sprint backlog，也就是不能在当前的迭代中插入新的需求

**共用的Scrum主管**

同样因为前面说的“ 全局相关联 ”
，我们产品线的几个Scrum项目是共用同一个Scrum主管(这里叫促进者似乎更合适)，也就是Boss，以确保Scrum过程被按照初衷使用。

**交叉的开发团队**

各项目的开发团队的主要成员是各自独立的，部分通用开发成员是兼职的。例如：

1. 因为技术架构的原因，各项目的核心开发团队是独立的，例如Android开发工程师就属于Android客户端项目
2. 测试人员是部分跨项目的，但是并不一定会跨所有项目，比如做服务器测试的人通常不会去做UI测试
3. 美工是各项目通用的，所以他是跨项目的
4. 原型制作人员是各项目通用的，所以他是跨项目的，具体到我们公司，这个工作是由 ”产品负责人“ 兼任的
5. 协议设计人员，这取决于协议涉及的功能，可能由关联性最大的那个项目的开发人员来设计（通常是架构设计师/技术团队负责人）

## 鸡组的成员

包括用户和利益相关　　者。不一定参与每一个迭代，但至少可以自主创建 Product backlog，参与测试并提供反馈。

**用户**

这里指的用户，主要是内部的别的部门的使用者，因为他们可以直接在内部协作系统创建 Product backlog
，当然实际上他们很少这么干，基本上他们口头或在公司内部群上提出自己的想法，然后由 "产品负责人" 创建 Product backlog

**利益相关者**

这里包括销售，售前，售后，营销的人员，这些人会通过客户反馈或者市场调研来产生新的需求，也就是他们会创建 Product backlog
，有的时候他们也参加 Sprint 计划会议或评审会议（视具体情况而定）

# Scrum活动

## Product backlog 列表梳理

也就是 Product backlog 的梳理，产品待办事项通常会很大,也很宽泛,而且想法会变来变去、优先级也会变化,所以产品待
办事项列表梳理是一个贯穿整个Scrum项目始终的活动。这方面我们和标准Scrum差不多。

- 保持 Product backlog 列表有序
- 把看起来不再重要的 Product backlog 降级（不可删除别人创建的 Product backlog ）
- 增加或提升涌现出来的或变得更重要的 Product backlog
- 将事项分解成更小的 Product backlog
- 将事项归并为更大的 Product backlog
- 将跨项目的 Product backlog 拆解到各个项目中去

在平时，这种梳理活动是我们的 "产品负责人" 来做的

不过，对于如意通科技来说，真正的 ”Product backlog 梳理“ 要等到 Sprint
计划会议时所有团队成员一起进行,而不单单是产品负责人。

## Sprint计划会议

每个Sprint都以Sprint计划会议作为开始， 我们称之为迭代，实际上如意通科技大部分的 Sprint 计划会议，是紧接在上一个 Sprint
的评审会议之后举行的, 这样工作的连续性比较好。

猪组的全部和鸡组的部分人员要参加 Sprint 计划会议。主要的工作，是针对排好序的 Product backlog 列表,

产品负责人和开发团队成员以及利益相关人员（比如销售人员和售后支持人员）讨论每个事项,并对该事项达成共识,包括根据当前的“完成的定义”, 为了完成该事项所需要完成的所有事情。

次要的工作是对于未纳入本此迭代的 Product backlog 进行集体讨论和梳理

**锁定需求**

在 Sprint 计划会议中，我们首先要决定本次迭代要完成哪些 backlog, 凡是纳入到本次 Sprint 的 backlog , 就变成了 Sprint backlog , 我们把这个过程叫做 ”锁定需求“ ，这是沿用旧的说法，意思是本次迭代就干这些活了，迭代进行中不应该再往里加入任何新的 backlog (bug工单除外，这个后面再说)

和标准的 Sprint 计划流程不同的是，我们不是先决定本次迭代要做哪些 backlog，再针对每个 backlog 讨论如何工作，而是针对每一条 backlog，先决定是否要做，再决定怎么做，然后去到下一个 backlog

也就是说，我们不是先预设本次迭代的所有目标，再分解工作任务，而是把每一个 backlog 按优先级依次讨论确定是否要做以及怎么做。

我们之所以这么做，是因为我们的产品在开发新功能的时候，有很大的机会采用新技术，导致我们单纯根据 backlog （也就是需求）是无法估算出工作量的。

例如，我们要增加一个在APP上阅读PDF文档的功能，那么我们开发人员可能会使用系统提供的组件，也可能引入一个全新的PDF阅读组件，所以只有我们和开发人员讨论过如何开发之后才能估算到这个工作量。

{{mermaid(Sprint计划会议流程示例)
graph TB
开始(评估backlog-A) --> 取舍A{是否要做}
取舍A -->|是| 分解A的任务{评估时间}
分解A的任务 --> |时间够|创建任务T1
分解A的任务 --> 评估backlog-B
评估backlog-B --> 取舍B{是否要做}
取舍B -->|是| 分解B的任务{评估时间}
分解B的任务 --> |时间够|创建任务T2
分解B的任务 --> |时间够|创建任务T3
分解B的任务 --> 评估backlog-C
评估backlog-C --> 取舍C{是否要做}
取舍C -->|是| 分解C的任务{评估时间}
分解C的任务 --> |时间不够|结束(结束会议)
}}

**梳理待办事项**

在计划会议的第二部分, 团队应该一起讨论未纳入本次迭代的 Product backlog ，相当于集体充当了产品负责人的角色，这是唯一可以把不合理的 Product backlog 删掉的时机

## 没有每日Scrum会议

我们没有每日Scrum会议这种东西。通常，在产品负责人或开发团队中的人在项目进行的过程中，遇到意料之外的阻碍的时候，我们才会召集临时的 Scrum 会议来确认他们是否仍然可以实现Sprint的目标。

因为我们使用了 Redmine 中的 Sprint 插件来实现辅助管理（请参考 Redmine技术交流#Scrum ），系统自带了任务看板的功能。所以我们Scrum团队的所有人（也鸡组的人）随时随地可以了解到项目的进展，咱们不用一堆人傻乎乎地站在那里往白板上贴小纸条了。

## Sprint评审会议

Sprint结束时,Scrum团队和相关人人员一起评审Sprint的产出。在如意通公司，Sprint 评审会议之后紧接着就是新 Sprint 计划会议，人员也大致是相同的。

猪组的全部和鸡组的部分人员要参加Sprint计划会议。讨论围绕着Sprint中完成的产品增量。由于Sprint的产出会涉及到一些人的“ 利益 ”,因此一个明 智的做法是邀请他们参加这个会议,这会很有帮助。

由于如意通科技的CI管理非常完善(参见 {{article(20)}} ) ，所有人都能第一时间拿到产品的最新版本，所以大部分时间我们不会特别地去演示本次迭代的工作成果。

很多时候，本次迭代的预期时间结束的时候，并未完成所有的 Sprint backlog，原因可能有：

1. 遇到意料之外的技术问题
2. 过去的版本产生的需要优先处理的bug太多
3. 等待依赖的其他产品更新
4. 利益相关人员的额外工作（比如客户强行要求插队）

此时我们有两种选择：

1. 选择延期发布新版本，这通常是因为本次迭代的主要目标尚未完结
2. 选择把未完成的 Sprint backlog 转到新的 Spint ,本次迭代仍然结束，产品发布新版本

当我们选择开 Sprint 评审会议，也就是表示我们打算强行发布新版。所以，我们的 Sprint 评审会议还会把 未完成的 backlog 转移到新的 Sprint ,这样在接下来的新 Sprint 计划会议中我们要把这些遗留的 backlog 的工作量考虑进去。

### 没有Sprint回顾会议

暂时来讲，我们还没有专门的Sprint回顾会议制度。通常我们会在每周的技术例会上讨论一些开发，流程，工具方面的问题,并找出潜在的改进事项。

## Scrum文档

除了 Product backlog 和 Sprint backlog ,我们还有bug和task

### Product backlog

在如意通科技，我们把 backlog 叫作 ”需求场景“ ，当一个 backlog 处于 Product backlog 阶段的时候，我们认为它属于 ”需求收集“ 阶段，当一个 backlog 处于 Sprint backlog 阶段的时候，我们认为它是一个 ”已锁定的需求“ ，隶属于 ”某某迭代“ 。

具体到我们的 Redmine 协作系统中，一个 backlog 就是一个类型为 "需求场景" 的工单。一个 backlog ，在被创建的时候，它总是位于 ”需求收集“ 这个特殊的Sprint里，此时这个 backlog 是一个 Product backlog 。当我们某次 Sprint 计划会议的时候，确认了这条 backlog ，我们就把这个 ”需求场景“ 工单的 "Sprint" 属性改成当前Sprint ，此时它就是一个 Sprint backlog 。当然在 Product backlog 转成 Sprint backlog 的时候我们需要很多加入很多细节 。

在 Product backlog 阶段的时候，这个工单永远是处于 ”打开“ 状态的，如果这条 backlog 被否决，则工单会被删除。在 Sprint backlog 阶段，开始工单是打开的，等到开发和测试完成之后，则由产品负责人来关闭此工单。

对于每一个 Sprint backlog ,我们在分解工作任务之后可以给这条 backlog 会估计一个 ”难度值“ ，这个难度值是综合了工作时间，开发难度，复杂度等等的一个综合值。理想情况下，一个特定的团队在一个固定时间的 Sprint 内完成的所有 Sprint backlog 的 ”难度值“ 的综合应该是一个相对固定的值。也就是说，下一个 Sprint 我们在评估工作量的时候可以参考这个值。然而经过一段时间之后我们发现这个功能用处不大，目前为止我们仍然使用 ”拍脑袋“ 和 ”吵架“ 两种传统方式来决定当前 Sprint 到底该做哪些功能点和新增多少功能点。

### Sprint backlog

Sprint backlog 就是我们打算做的功能点，但是它仍然是来自用户视角的工单，也就是本质上它是个 ”需求“ 。要满足一个需求，那么开发人员可能需要一个或多个任务才行。

在 Redmine 系统中，我们会为每一个 Sprint backlog 工单创建一个或多个 ”子任务“ ，这些子任务是给开发人员的，这些子任务工单会说明该怎么干。

产品负责人要跟进所有 Sprint backlog ,如果一个 Sprint backlog 名下的所有子任务都完成了，那么就代表这个 Sprint backlog 也完成了，产品负责人可以关闭此 Sprint backlog

#### 技术攻关backlog

通常产品开发中我们总有机会遇到一些比较底层的技术问题。比如 Windows 显卡兼容问题， 基于这个问题可能会产生很多的 backlog 和 bug 。我们会为此单独创建一个 backlog 和对应的任务工单。而对于那些需求驱动的 backlog 和 bug ，则只是关联到 “ 技术攻关 ” 的 backlog ，而不是各自创建任务工单。

也就是说，可能多个需求或错误是因为一个技术问题导致的，但是我们的 Redmine 并不能让多个backlog 工单关联到同一个子任务工单上。所以我们把这个问题单独创建一个 backlog 并分配一个任务。所以这个 backlog 看起来不像一个客户需求，而是一个技术攻关的需求。

因为这个技术问题导致的其他派生的backlog和bug，我们会关联到这个 “ 技术攻关 ” 的backlog 。这些backlog和bug我们在计算工时的时候就不另外计算了，如果因为 “ 技术攻关 ” 未能在本次迭代周期内解决，则这些backlog和bug就顺延到下一次迭代中去。

## 任务工单

任务，这个概念在 Sprint 标准中并未特别提及，有时候人们会把 backlog 当成任务。

然而并不是这样，任务是分派给开发人员的工单，是面向开发和测试人员的。而 Sprint backlog
是面向用户和需求方的，是我们的工作目标。一个 Sprint backlog 可以有一个或多个子任务工单（我们简称为 ” 任务 “ ）。在 Redmine
系统中实现了从 backlog 派生出任务工单的功能。

当一个 Sprint backlog 的所有子任务都完成之后，产品负责人就可以关闭它了，以下图示

{{mermaid(Sprint backlog任务执行示例)
sequenceDiagram
participant Sprint计划会议
participant 开发人员1
participant 开发人员2
participant 测试人员
participant 产品负责人
Sprint计划会议-->> Sprint计划会议: 锁定Sprint backlog-A
Sprint计划会议-->> Sprint计划会议: 分解成任务A1和任务A2
Sprint计划会议->> 开发人员1: 分配任务A1
Sprint计划会议->> 开发人员2: 分配任务A2
Sprint计划会议->> 产品负责人: 分配跟进Sprint backlog-A
开发人员1-->> 开发人员1: 完成任务A1
开发人员1->> 测试人员: 请求任务A1的测试
activate 测试人员
测试人员-->> 测试人员: 完成任务A1的测试
测试人员-->> 测试人员: 关闭任务A1
deactivate 测试人员
开发人员2-->> 开发人员2: 完成任务A2
activate 测试人员
开发人员2->> 测试人员: 请求任务A2的测试
测试人员-->> 测试人员: 完成任务A2的测试
测试人员-->> 测试人员: 关闭任务A2
deactivate 测试人员
产品负责人-->> 产品负责人: 检查任务A1已关闭
产品负责人-->> 产品负责人: 检查任务A2已关闭
产品负责人-->> 产品负责人: 关闭Sprint backlog-A
}}

## bug工单

bug，这个概念在 Sprint
标准中也未特别提及。在如意通科技，以及在我们的Redmine系统中，明确了bug工单这个概念，bug基本上分两种：

- 当前迭代中的 backlog 和 任务 在测试时出现的bug，此时的bug我们采用在 "任务" 工单中新增备注然后 “ 反馈 ” 回开发人员的方式来解决，这样就只是现有工单的流转
- 以前发布的版本的bug，在当前迭代的时候才被发现，此时的bug我们视为一个类型为 "错误" 的backlog (区别于 “ 需求场景 ” 类型的 backlog) ,根据bug的紧急程度来决定它是进入 "需求收集" 还是插队到当前迭代

{{mermaid(错误工单示例)
graph TB
开始(发现以前版本的新bug) --> 判断{是否紧急}
判断 -->|是| 新增错误工单并加入当前迭代
新增错误工单并加入当前迭代 --> 分解任务工单
判断 -->|否| 新增错误工单并加入需求收集
}}

## backlog跨项目关联

因为如意通科技的产品之间关联度很高，比如很多功能需要各客户端之间以及客户端和服务器之间的配合，Redmine
为我们提供了跨项目工单关联功能。

对于和别的产品中的 backlog 存在关联的本产品 Product backlog
，我们都会把优先级提到最高，基本上会在最近的下一次迭代中去实现它。

### 燃尽图

燃尽图（burn down chart）是一个公开展示的图表，显示当前 Sprint
中未完成的数目。实际它主要是配合每日Scrum会议来用的，是为了让团队中的每个人快速了解当前的状况。拜技术进步所赐，
Redmine 系统中自带了此功能，我们所有人随时可以看到它，我们不是不需要它，只是它变得唾手可得，仅此而已。

## Scrum迭代频率

### 高频迭代Scrum

典型的 Scrum 项目是高频迭代的，它们是持续迭代的，并且迭代时间不长（我们一般是两周）。这类 Scrum
项目主要用于我们的三个标准客户端。通常我们的关联 backlog 都是从这些项目发起的，然后催生出跨项目的关联 backlog 。

### 低频迭代Scrum

低频迭代的 Scrum 项目，主要用于我们的服务器类产品，比如 RTP服务器/RTP云服务器。它的特点是一个 Sprint
成果发布之后，并不一定会立刻启动新的 Sprint 。我们之所以仍然使用 Scrum 过程来管理这些项目，有两个原因：

1. 作为一个产品，应该有迭代，在 Redmine 中把 backlog 分解成子任务 ，然后摊在看板上，这种迭代管理方式十分方便
2. 如意通RTP系列产品之间是广泛联系的，我们的Scrum是 ”全局相关联的Scrum“ ，大家都使用 Scrum 方便跨项目协作

## 番外

这里说一些理论上没有但是实践中会遇到的事情

### Scrum十分依赖CI

Scrum 的快速迭代模式，十分依赖于高效的CI(持续集成)，对头，我们公司正巧拥有非常高效的CI，参见 {{article(20)}}

好吧，其实是因为我们实施了 Scrum
，才倒逼着我们不停地折腾CI，最后我们的CI做到了保证你在任何时候都可以拿到一个最新版本，而不只是在迭代完成发布的那一刻

- 当开发人员完成一个任务之后，他把任务工单分配给测试人员的那一刻，我们就需要一个可运行的版本，示例如下

{{mermaid(开发-测试快速流转示例)
sequenceDiagram
开发人员-->> 开发人员: 完成任务A
开发人员->> 测试人员: 请求测试
Note right of 开发人员: 分配任务A
测试人员->>CI: 请求打包新版本
CI-->> CI: 打包新版本
CI-->> 测试人员: 返回成功
测试人员-->>测试人员: 手机APP检测到新版本并自动更新
测试人员-->>测试人员: 完成任务A的测试
}}

- 当迭代完成之后，我们也能拿到一份最新的发布版，但其实它和迭代过程中的中间版本没什么本质区别，只是刚好在这个时间点打包出来而已

### 产品开发的早期

这里指的是全新的产品，不是客户定制项目，也不是旧产品增加新功能

#### 需求规划阶段

通常，新产品的启动，都有一个契机，比如我们的Windows客户端最早是Delphi开发的，因为Delphi日渐式微，我们需要采用新技术来开发一个全新版本。我们的服务中心产品，则是因为客户需要在线客服功能而我们又有一些很好的点子可以形成一个独特的创新产品
。

在确定要做新产品之后，我们会做一些市场调研，开一些会，然后形成一个“ 需求规划 ”
文档，这个文档会有新产品的核心功能和流程，有产品的发展路线（短期，中期和长期目标）。

#### 可行性研究阶段

这一步骤并不总是需要，只有在采用全新技术的时候才需要，比如我们的 Windows 客户端的界面开发，我们对于 C++ ,QT4 ,QT5

的技术选型是做了可行性研究的，最终选择了 QT5 。

实际我们大部分的新产品只是在公司的技术能力范围内选择一种技术路线，比如我们的服务器或服务组件，我们有Erlang和Java两种选择，任选一种即可。

**架构设计**

实际上经过了需求规划和可行性研究之后，我们就可以进入 Scrum 过程了，根据需求规划和技术路线我们就可以组建团队了。

在第一次 Sprint 会议上，我们会把基于“ 需求规划 ” 和“ 可行性研究 ”的成果来讨论，此时我们一般会提出架构设计 backlog 。

架构设计，我们视为最重要和最优先的 backlog
，因为我们要根据产品的中长期目标来进行架构设计，这是整个产品的技术基础，架构设计决定了这个产品的极限在哪里。比如最早的RTP服务器架构设计中没有考虑集群，所以它的并发在线数就有自己的极限。当我们在RTP服务器中加入集群功能，实际上我们重新起了一个新的项目，某种意义上相当于一个全新的产品。

**原型/效果图设计**

如果该产品有人机交互界面的话，在第一次 Sprint 会议上，我们还会提出原型或效果图 backlog 。

原型/效果图设计，是有UI的产品中最重要和最优先的 backlog
，这是UI程序开发的基础。不过这里的原型设计主要针对我们产品的短期目标，因为UI不像程序那样有机会通过迭代来进化。

## 例外的客户定制项目

目前我们还没有在客户定制或协作开发项目中引入 Scrum ，估计在可预见的将来也无法引入，你懂的。

**文件**

| | | | |
|---|---|---|---|
| ScrumModel.jpg | 59.9 KB | 2017-01-04 | 如意通 科技 |
| dailyscrumconference.png | 446 KB | 2017-01-05 | 如意通 科技 |
| burndownchart.png | 231 KB | 2017-01-05 | 如意通 科技 |
| rtpnetwork.jpg | 166 KB | 2017-01-09 | 如意通 科技 |