运营开发部项目管理细则

2014年版

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布  [ ] 正在修改 | 当前版本： | 1.0 |
| 作 者： | 孙维晟 |
| 完成日期： | 2014-11-19 |

**开发部**

**版 本 历 史**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **版本** | **修订人** | **修订日期** | **修订内容** | **批准人** | **批准日期** |
| 1.0 | 孙维晟 | 2014-11-19 | 新建 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**目录**

目录

[1. 接收需求阶段 4](#_Toc404761584)

[1.1. 工作流程细则 4](#_Toc404761585)

[1.1.1. 接收的需求很简单 4](#_Toc404761586)

[1.1.2. 接收的需求已经模式化 4](#_Toc404761587)

[1.1.3. 接收的需求很简单但实际很复杂需要需求分析师介入整理需求文档 5](#_Toc404761588)

[1.1.4. 接收的需求不是非常简单需要整理需求文档 5](#_Toc404761589)

[1.1.5. 接收的需求直接就是界面设计文档或策划案 6](#_Toc404761590)

[1.1.6. 只需要技术实现的需求 6](#_Toc404761591)

[1.2. 流程图 8](#_Toc404761592)

[2. 立项阶段 9](#_Toc404761593)

[2.1. 工作流程细则 9](#_Toc404761594)

[2.1.1. 确定项目名称和版本号 9](#_Toc404761595)

[2.1.2. 确定项目属性信息 9](#_Toc404761596)

[2.1.3. 确定项目成员 9](#_Toc404761597)

[2.1.4. 确定项目时间 10](#_Toc404761598)

[2.1.5. 发送立项报告邮件 10](#_Toc404761599)

[2.1.6. 立项会议 10](#_Toc404761600)

[2.2. 流程图 12](#_Toc404761601)

[3. 迭代开发阶段 13](#_Toc404761602)

[3.1. 工作流程细则 14](#_Toc404761603)

[3.1.1. 计划会议1 14](#_Toc404761604)

[3.1.2. 计划会议2 14](#_Toc404761605)

[3.1.3. 开发测试 16](#_Toc404761606)

[3.1.4. 需求部门评审 19](#_Toc404761607)

[3.1.5. 迭代总结会 20](#_Toc404761608)

[3.2. 流程图 21](#_Toc404761609)

[4. 部署验收阶段 22](#_Toc404761610)

[4.1. 工作流程细则 22](#_Toc404761611)

[4.1.1. 部署外网测试服务器进行测试 22](#_Toc404761612)

[4.1.2. 需求方外网验收 22](#_Toc404761613)

[4.1.3. 部署正式服务器及正式环境测试 23](#_Toc404761614)

[4.2. 流程图 24](#_Toc404761615)

[5. 结项阶段 25](#_Toc404761616)

[5.1. 工作流程细节 25](#_Toc404761617)

[5.1.1. 结项会议 25](#_Toc404761618)

[5.1.2. 结项报告 25](#_Toc404761619)

[5.2. 流程图 25](#_Toc404761620)

# 接收需求阶段

接收需求阶段是指：从需求方发出需求到立项之间的阶段，这个阶段最重要的工作就是与需求方确定业务规则和业务范围，并且还要判断界面设计和技术实现上是否存在风险。

* **让客户做决定**

比如遇到一个任务，有两种实现的方式。一种比较快但对用户有些限制，另一种对用户没有限制但实现起来比较慢，并且这个任务有时间压力，如果你使用了第一种方法，但在给客户演示的时候，客户对用户的限制很可能会感到震惊，甚至愤怒！结果还得重做。

开发者（或项目经理）能做的一个重要的决定就是：判断哪些是自己决定不了的，应该让企业主做决定。要将各种方案的优缺点以及潜在的风险详细的告知客户，然后让客户做决定，然后接受客户的决定。

正确的做法：“开发者、经理或业务分析师不应该做业务方面的决定。用业务负责人能够理解的语言，向他们详细解释遇到的问题并让他们做决定。”

## 工作流程细则

### 接收的需求很简单

有些需求很简单比如修改界面的文字或图片，再或者是在某个功能中增加简单的判断条件。这类需求的特点是：

* 不需要界面设计或只需要设计某个界面元素
* 因为基本不涉及的业务规则的变更所以不需要整理需求文档
* 涉及变更的系统因为业务简单项目经理可以判断修改不会造成业务规则冲突
* 编写很少的代码或不需要编写代码

工作流程细则：

* 要求需求方发邮件写明工作需求
* 判断是在现有项目版本工作中增加任务或者是发紧急任务，优先考虑在现有项目版本中增加工作任务其次是发紧急任务。
* 回复需求方邮件重复你理解的需求并告知需求方工作方法及时间（项目排期）
* 要求需求方确认你的邮件
* 增加项目中的任务或发紧急任务

（**紧急任务需要在48小时内完成并且不能耽误正在进行项目的进度，紧急任务一般都需要加班赶进度**）

### 接收的需求已经模式化

这种需求是以前已经做过很相似的项目或多次开发相同模式的项目，需求文档已经形成模板只需要需求方填空即可，这种情况多发生在模式化的活动项目中。

这类需求的特点是：

* 界面设计已经形成模式
* 已经多次实现类似的功能
* 需求方的业务规则也形成模式或在模式上有很少的修改
* 业务规则不复杂

工作流程细则：

* 将需求模板文档邮件发给需求方
* 要求需求方填写后邮件回复你
* 回复需求方邮件明确页面制作时间和项目时间（项目排期）
* 进入立项阶段

### 接收的需求很简单但实际很复杂需要需求分析师介入整理需求文档

需求方的信息很简单但在与需求方接触后发现实际业务很复杂或功能实现很复杂，如果需求涉及的界面设计是在已有的系统基础上扩展或需求方无界面设计要求，则可以没有界面设计工作，反之则需要先立界面设计的项目并与需求方确认界面设计是否符合要求，如果功能实现困难应该先立文档项目进行研究如何实现或者是原型Demo项目进行研究，如果确实难以实现应该告知需求方无法实现的结果。

这类需求的特点是：

* 需求方实际上并不了解实际业务
* 需求方要求按某个系统进行复制
* 实现功能很复杂超出需求方预期
* 需求方有很多，造成业务规则杂乱
* 需求方有很多，造成业务规则不统一

工作流程细则：

* 需求分析师与负责该业务的项目经理一起与需求方深入沟通
* 编写详细的《需求规格说明书》
* 给需求方讲解《需求规格说明书》的内容
* 要求需求方回复邮件确认《需求规格说明书》是否符合要求
* 判断界面设计是否存在风险，是否需要先立界面设计的项目
* 判断是否需要先对技术实现方案进行研究
* 进入立项阶段

### 接收的需求不是非常简单需要整理需求文档

在确定是非简单项目的情况下要判断是否需要先立界面设计的项目或原型Demo的项目，如果无法实现要尽早告知需求方。

这个类型的项目特点：

* 可能是系统业务复杂
* 可能是界面设计复杂
* 可能是功能实现复杂
* 可能是项目背景复杂

工作流程细则：

* 需求分析师与负责该业务的项目经理一起与需求方深入沟通
* 编写详细的《需求规格说明书》
* 给需求方讲解《需求规格说明书》的内容
* 要求需求方回复邮件确认《需求规格说明书》是否符合要求
* 判断界面设计是否存在风险，是否需要先立界面设计的项目
* 判断是否需要对技术实现方案进行研究
* 进入立项阶段

### 接收的需求直接就是界面设计文档或策划案

这种情况下风险最大的就是业务逻辑无法从页面表现中全部体现并且无法与需求方直接沟通，所以在发生接收的需求就是界面设计的时候应该主动询问需求方信息并与需求方和最终用户进行直接沟通，制作页面或策划案的部门一般都不是需求方或最终用户。

这个类型的项目特点：

* 界面设计或策划案已经完成
* 没有详细的业务规则说明
* 没有长期的业务规划
* 技术实现风险有可能比较高

工作流程细则：

* 与需求方进行沟通了解业务规则和远期规划
* 编写详细的《需求规格说明书》或发送需求模板
* 将整理后的需求文档通过邮件发给需求方
* 要求需求方回复邮件确认
* 与需求方沟通工作流程，希望可以尽早的得到原始需求
* 进入立项阶段

### 只需要技术实现的需求

游戏接口对接、平台接口对接都属于这类需求，这类需求一般没有用户界面，需要注意的是技术实现上可能会遇到难点，比如对方提供的验证方法有语言级的不兼容情况（JAVA 和 .NET的非对称加密）。

这个类型的项目特点：

* 没有用户界面
* 技术实现可能存在困难
* 项目是为其他程序或系统提供服务

工作流程细则：

* 要求需求方发邮件写明工作需求
* 判断是否需要对技术实现方案进行研究
* 回复需求方邮件确认项目时间（项目排期）
* 要求需求方确认你的邮件
* 进入立项阶段

## 流程图

开发需求

模式化？

简单？

编写邮件确认需求

发送需求模板确认需求

紧急任务或

增加项目范围

编写需求文档并邮件确认

进入立项阶段

技术风险？

界面风险？

进入立项阶段

（技术研究项目）

进入立项阶段

（界面设计项目）

# 立项阶段

立项阶段需要成立项目组并确定项目各种信息后告知项目组全体成员（含需求方和高管）。

## 工作流程细则

### 确定项目名称和版本号

项目的名称其实就是系统名称，为了大家可以有统一的项目名称标准所以项目名称是不能随便取的，只能用《项目定义表》中定义好的名字，如果是个新业务或新系统请找部门负责人在项目定义表中添加。项目版本号是4位版本号也不能随便创建，需要从OA中申请版本号，这样多个团队开发同一个系统版本号就不会产生冲突，并且自动部署系统是依赖版本号进行工作的所以版本号一定要从OA中获取。

版本号规则如下：

主版本号. 子版本号. 编译版本号. 修正版本号

主版本号：系统有大的升级改版才能修改并且需要部门负责人审核

子版本号：对系统功能进行的添加修改工作时需要修改此版本号（非紧急任务）

编译版本号：部署时漏传文件再重新部署时需要修改此版本号

修正版本号：紧急任务部署时需要修改此版本号

立项报告中的版本号是：主版本号.子版本号，编译版本号和修正版本号在项目部署阶段才会产生。

### 确定项目属性信息

项目有一些附加属性比如是否需要加班完成（5+1计划），项目是否属于固定人员项目等，项目如果是5+1计划则需要安排加班进行工作，如果是固定人员则优先使用给该系统固定分配的人员否则需要向各个组的职能经理申请人员，这些信息已经记录在《项目定义表》中。

### 确定项目成员

如果是固定人员的项目则从《项目定义表》中确定项目成员，如果是非固定人员项目则需要向各个职能经理申请（项目支持组、程序开发组、软件测试组、Web前端组），在申请人员的时候需要向职能经理说明项目需求、需求方希望的完成时间、需要人员水平、项目开始时间等信息，职能经理和项目经理共同协商后确定项目成员，这时职能经理可先锁定被确定进入项目的人员（但锁定时间不会很长并有可能发生变化），项目经理应该尽快完成立项工作。

项目成员还包含需求方接口人、高管、项目协调人等其他部门的人员，要将项目利益涉及到的人员都看作是项目成员，项目进展中的任何信息应该是项目组所有成员共享（例如每日燃尽图邮件）。

### 确定项目时间

人员确定后依照以往的经验，项目经理需要评估项目的时间，以下是具体需要评估时间和参考值：

* 立项及计划会议时间：发送立项报告到完成燃尽图的时间，工作内容包含立项会议、计划会议1、计划会议2，参考值：简单的项目几小时，复杂的项目1-2天。
* 迭代工作时间：燃尽图中任务时间的合计×迭代次数，参考值：一个迭代一般为5个工作日以下，复杂项目的迭代次数一般为2-3次，普通项目仅仅一次迭代。这里需要注意：一个项目内迭代周期总是固定并一致的，比如某个项目分为3次迭代进行，第一次迭代为1周那么第二次和第三次迭代肯定必须也是1周。参考值：一次迭代不应超过2周，一次迭代的标准时间是1周。
* 内网集成测试时间：最后一次迭代完成后在内网集成测试，参考值：几个小时足以。
* 外网测试时间：将系统版本部署到外网测试服务器进行集成测试，参考值：几个小时足以。
* 需求方验收时间：这个时间需要与需求方进行协商，参考值：最好一天内完成。
* 版本部署日期：这个时间需要和需求方协商并提前告知系统部的部署人员，参考值：必须1天内完成。

（以上各个阶段的工作流程细则会在以后阐述）

评估的时间如果是从几月几日到几月几日就是明确的时间评估，如果是几个工作日则是估算的时间评估。如果是估算的时间评估则需要在立项邮件发出后48小时内向项目全体成员（特别是需求方）发送明确的项目时间计划邮件，也就是说在这48小时内需要尽快进行计划会议1和计划会议2评估具体的项目时间。如果项目是属于5+1计划则需要计划并明确加班时间，从周1到周5加班时间在8小时以上则视为执行了5+1计划。

### 发送立项报告邮件

到这步就可以完成立项报告的编写了，在发送邮件之前需要私下将项目的时间和主要功能等信息与需求方进行沟通，沟通后将立项报告发送给部门负责人进行审核，邮件需要抄送宋总、项目组内成员，需求方接口人、部门项目助理等所有与项目相关的人员，部门负责人回复邮件同意后便可组织立项会议启动项目。

### 立项会议

在立项报告审核通过后，项目经理应该召集项目所有成员进行立项会议，在立项会议上应该让项目组成员明确以下几点信息：

* 项目组今天成立并开始进行工作
* 项目成员都是谁
* 项目组中每人的角色
* 项目要达成的目标和目的
* 项目的时间和加班计划
* 项目的主要任务
* 项目的需求方是谁
* 项目会议制度（如每日立会）
* 项目组工作风格（根据项目经理不同而不同）
* 迭代的次数与周期和每次迭代的目标（这点需要需求方认可，因为有优先级的问题）

立项会议的形式必须是面对面的开会，因为立项会议要带有仪式性，这样更容易形成团队气氛和凝聚力。**立项会议后建议在RTX中创建以项目名称+版本号的群，这样可以提高项目组内的沟通效率。**

建议在会议上介绍一些敏捷开发的优秀实践：

* **管理自己每天的时间-工作要有节奏**

我们每人每天在下班前都希望完成自己的工作，这是不现实的，但我们可以在工作结束前完成单元测试，并提交一天完成的代码，如果你正在开发小块任务，这种方式非常有助于你管理自己的时间：如果在你工作的时候没有一个固定的最终期限（例如一天的结束），就应该好好想想了。它会让你工作有一个节奏，在每天下班的时候，提交所有的工作，开心地收工。这样，明天就能开始新的内容，解决下一系列的难题。

* **打破砂锅问到底**

正确的方式：“不停地问为什么。不能只满足于别人告诉你的表明现象。要不停地提问直到你明白问题的根源。”

为什么是一个非常好的问题。事实上，在理解一个问题的时候，需要渐次地问5个以上的“为什么”。

注意：当你问“为什么”的时候，也许你会被反问：“为什么你问这个问题？”在提问前，想好你提问的理由，这会有助于你问出恰当的问题。

* **及时通报进展与问题**

不正确的做法是：“管理层、项目团队以及业务所有方，都仰仗你来完成任务。如果他们想知道进展状况，会主动找你要的。还是埋头继续做事吧。”

接受一个任务，也就意味着做出了要准时交付的承诺。不过，遇到各种问题从而导致延迟，这种情形并不少见。截止日期来临，大家都等着你在演示会议上展示工作成果。如果你到会后通知大家工作还没有完成，会有什么后果？除了感到窘迫，这对你的事业发展也没什么好处。

及时通报进展与问题，有情况发生时，就不会让别人感到突然，而且他们也很愿意了解目前的进展状况。他们会知道何时应该提供帮助，而且你也获得了他们的信任。

整个团队可以使用一些黑板（或者BBS、WIKI）来发布他们状态、代码设计、研究出的好点子等内容。现在只要绕着团队的工作区走一圈，就可以学到不少新东西，而且管理层也就可以知道目前的状况如何了。

正确的做法说：“及时通报进展与问题。发布进展状况、新的想法和目前正在关注的主题。不要等着别人来问项目状态如何。”

这样做的话当经理或同事来询问工作进展、最新设计，或研究状况时，不会感到头痛。

在展示进度状况时，要照顾到受众关注的细节程度。举例来说，CEO和企业主是不会关心抽像基类设计的具体细节的。

## 流程图

登录OA获取版本号

查看《项目定义表》确定项目名称和其他信息

经过确认后的需求文档或邮件

固定人员项目？

与职能经理协商项目组人员

确定项目组成员

与需求方沟通确认项目时间和主要功能

编写立项报告

评估项目各阶段的时间

邮件发送项目立项报告给项目全体成员

估算的时间？

部门负责人审核后组织立项会议启动项目

立项报告邮件发出后48小时内发送项目明确时间计划

举行立项会议

# 迭代开发阶段

因为我们采用敏捷开发的工作风格所以有必要介绍一下敏捷开发的相关知识：

* **敏捷——成功的软件开发方法（取决于你识别和适应变化的能力）**

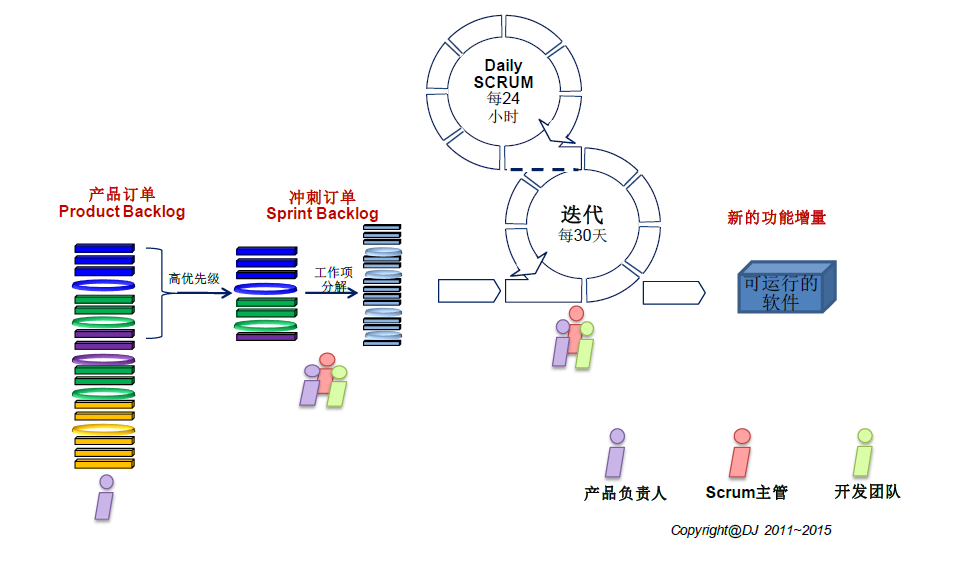
敏捷代表的不是快速！是为了适应变化！现在任何事情都在不停的变化，很多时候是等你做完了一步，发现事情已经变化了，你做的事情已经没有用了，这时不要生气和埋怨，因为事情大多是这样的，敏捷是要将工作分成很多小环节，每个环节都完成一小部分事情（每个环节都经历全体开发流程）并立刻确认这些事情是否符合期望，如果不符合马上修改，这样能最快的对变化进行响应。

* **使用短迭代，增量发布**

迭代开发是指：在小且重复的周期里，你完成各种开发任务：分析、设计、实现、测试和获得反馈，所以叫迭代。如果给我一份详细的长期计划，我就会给你一个注定完蛋的项目。

根据Capers Jones的格言：“……大型系统开发是一件非常危险的事情。”大型系统更容易失败。他们通常不遵守迭代和增量开发的计划，或迭代时间太长。Larman指出，软件开发不是精细的制造业，而是创新活动。规划几年之后客户才能真正使用的项目注定行不通。

对于大项目，最理想的办法就是小步前进，这也是敏捷方法的核心。



## 工作流程细则

### 计划会议1

在计划会议1上需要给项目组成员讲解本次迭代开发的业务规则和业务规划，还有确定开发业务功能的优先级和本次迭代要达到的目标（最好邀请需求方一起参加），因为本次会议只是涉及全部功能中的一部分业务所以要求会议一般不要超过8小时，最关键的是需要每个团队成员对业务完全理解，所以在计划会议1时需要安排业务反讲，业务反将就是让项目组成员在听完业务人员或需求方讲解业务后再通过自己的理解将业务规则复述一遍给需求方或需求分析人员听，以此来验证所有人在业务理解上没有出现偏差。

本次会议要达到的目的：

* 项目组所有成员明确本次迭代的业务范围
* 项目组所有成员明确本次迭代的业务规则
* 项目组所有成员明确本次迭代的业务功能优先级
* 项目组所有成员明确本次迭代的目标

会议的会议记录应该邮件发给全体项目组成员（含需求方），如果开发进行中需要变更则在此邮件基础上进行修改。

会议后要填写计划会议1记录表（非项目经理填写）。

### 计划会议2

计划会议2就是概要设计会议，在计划会议1的基础上团队成员对本次迭代涉及的功能实现方案进行细致的讨论和总结，讨论实现方案的粒度原则上是越详细越好，因为计划会议2讨论的实现越多越细开发与测试时就会越顺利，反之则会议上没说明白的事情越多开发和测试时讨论和沟通的成本也会越高，其实开发测试工作时的讨论和沟通是无法避免的但类似设计或规划方面的事情在前期开会时进行讨论和沟通效果会更好成本更低，在会议中每个岗位需要讨论的内容和优先级如下：

* Web前端开发人员需要讲述页面的概要设计和用户体验方面的优化方法
* 测试人员需要讲述怎么设计和编写测试用例
* 服务器端开发人员需要讲述数据表的设计和程序实现的时序图
* 需求分析人员需要确定实现方案没有改变原始需求

在每个岗位讲述自己的实现方式时，其他岗位人员负责审核和补充。会议记录经过整理可以直接变为概要设计文档，然后邮件发给全体项目组成员，如果开发进行中需要变更则在此文档基础上进行修改。会议时间不应超过8小时，会议后要填写计划会议2记录表（非项目经理填写）。

在计划会议2时有一项重要的工作要完成——合理的规划用户故事和用户故事内的工作任务最后形成燃尽图（**燃尽图必须由项目经理编写**）。用户故事就是用户使用功能的场景，工作任务就是实现用户故事所需要完成的每个任务。编写用户故事的原则：

1. 每个故事必须使用业务语言编写，而不是技术术语，这样需求分析人员或需求方才能确定用户故事的优先级别。
2. 用户故事应该提供最合适的产品行为描述。

用户故事的例子：比如“用户搜索职位”是一个用户故事，“程序将用连接池来连接数据库”就不是一个好的用户故事，因为需求方不关心这样的技术细节。

编写用户故事后需要进一步细化比如“用户搜索职位”这个用户故事，可以通过以下问题进一步细化：

* 用户会按照什么来搜索？州？城市？职位简述？关键字？
* 用户必须成为网站的会员吗？
* 搜索参数可以保存吗？
* 找到职位后要显示什么样的信息？

很多这样的细节都可以作为新增的故事，用户故事的颗粒度为：能够让一个开发和一个测试半天到3天内完成。如果一个故事太庞大就会像史诗一样。这样的史诗可以被划分为2个或多个更小的故事。例如，故事“用户可以搜索职位”可以被划分为如下的故事：

* 用户可以按照地址，薪酬范围，职位，企业名称和职位的发布时间。
* 用户可以查看匹配到的职位的信息。
* 用户可以查看发布职位的企业详细信息。

但是，当一个故事包含了所有最终的细节时它就不能再分解了。例如，故事“用户可以查看匹配到的职位的信息”是一个合理的现实的故事。我们没有必要再把它划分成如下的样

子：

* 用户可以查看职位描述。
* 用户可以查看薪酬范围。
* 用户可以查看工作地点。

用户故事编写完后需要编写工作任务，一个用户故事下的工作任务就是实现该用户故事（场景）所需要实现的工作任务，任务可以是制作页面、开发代码、编写测试用例等，但每个任务不能超过8小时，实在细分不了也不能超过16小时。评估每个任务完成所需要的时间建议采用以下方法：

多个人对同一个任务进行评估，如果相差时间较大则每个人说出自己评估时间的理由，然后再进行评估直到每个人评估时间相差不多时将每个人评估的时间相加算平均数作为该任务评估的最后时间。这里需要注意每个任务在编写燃尽图任务列表时不要确定执行这个任务的人员，任务的执行应该是在每日立会时由项目组成员自行领取。

计划会议2结束后燃尽图应该马上完成并在会议后邮件给全体项目组成员。

* **让设计指导而不是操纵开发**

设计满足实现即可，不必过于详细。（不要理解为不用想就开始编码）

设计可以分为两层：战略和战术。前期的设计属于战略，通常只有在没有深入理解需求的时候需要这样的设计。他应该只是描述总体战略，不应深入到具体的细节。而战术设计应该在项目开发的时候再具体展开。不要一开始就进行战术设计，他的重点是集中在单个方法或数据类型上，这时，更合适讨论如何设计类的职责。因为这仍然是一个高层次、面向目标的设计。事实上，CRC（类-职责-协作）卡片的设计方法就是用来做这个事情的。每个类按照下面的术语描述：

1. 类名

2. 职责：他应该做什么？

3. 协作者：要完成工作他要与其他什么对象一起工作？

如何知道一个设计是好的设计？代码自然地为设计的好坏提供了最好的反馈。如果需求有了小的变化，他仍然容易去实现，那么他就是好的设计。而如果小的需求变化就带来一大堆基础代码的破坏，那么设计就需要改进。

### 开发测试

**关于每日立会：**

开发测试时要重视每日立会，每日立会有一些原则：

* 会议准时开始：对于迟到者团队常常会制定惩罚措施（例如罚款、做俯卧撑、在脖子上挂橡胶鸡玩具等）。
* 欢迎所有人参加，但只有项目组工作人员可以发言。
* 不论团队规模大小，会议被限制在15分钟以内。
* 所有出席者都应站立（有助于保持会议简短）。
* 会议应在固定地点和每天的同一时间举行。
* 在会议上，每个团队成员需要回答三个问题：
* 今天（昨天）你完成了那些工作？
* 明天（今天）你打算做什么？
* 完成你的目标是否存在什么障碍？（项目经理需要记下这些障碍）。

会议后要填写每日立会记录表（非项目经理填写）。

立会后根据收集的工作情况尽快绘制最新的燃尽图并邮件给需求方、项目成员、部门项目助理和部门负责人，如果从燃尽图上看出项目可能延期就要尽早进行调整并马上通知需求方。

立会上还要询问并记录大家在部署工作方面（数据库和配置文件的修改）涉及的内容并总结为《部署清单》，进入最后部署阶段时需要将《部署清单》提交系统部进行操作。

* **定期安排会面时间**

也许你个人很讨厌开会，但沟通是项目成功的关键。我们不只要跟客户谈话，还应该与开发人员进行良好的沟通。

立会（站着开的会议）是将团队召集在一起，并让每个人了解当下进展状况的好办法，顾名思义，参与者们不允许在立会中就坐，这可以保证会议快速进行。一个人坐下来之后，就由于感到舒适而让会议持续更长的时间。

要保证会议议题不会发散，每个人都只回答下述3个问题：

* 今天（昨天）你完成了那些工作？
* 明天（今天）你打算做什么？
* 完成你的目标是否存在什么障碍？

每个人大约2分钟，不要讨论详细的问题，可以在立会之后，再召集相关人员进行深入的讨论。通常立会在每个工作日的早些时候举行，但不要把它安排为上班的第一件事情。一般来说，在大家到公司之后的半个小时到一个小时内举行。

正确的做法：“立会可以让团队达成共识。保证会议短小精干不跑题。”

大家都盼望着立会。希望彼此了解各自的进度和手上的工作，而且不怕把各自遇到的问题拿出来公开讨论 。

**在编码方面应该遵循的原则**：

在新项目刚开始着手开发的时候，她的代码很容易理解和上手。然而，随着开发过程的推进，项目不知不觉中演变为一个庞然大物。发展到最后，往往需要投入更多的精力、人力和物力来让他继续下去。解决这个问题最简单的方式就是在开发过程中细心的照看代码。在编写代码时，每天付出一点小的努力，就可以避免代码“腐烂”，并且保证应用程序不至于变的难以理解和维护。

* **代码要清晰地表达意图**

Hoare谈软件设计

设计软件有两种方式。一种是设计得尽量简单，并且明显没有缺陷。另一种方式是设计得尽量复杂，并且没有明显的缺陷。

我们大概都见过不少难以理解和维护的代码，而且（最坏的是）还有错误。当开发人员们像一群旁观者见到UFO一样围在代码四周，同样也感到恐怖、困惑与无助时，这个代码的质量就可想而知了。如果没有人理解一段代码的工作方式，那这段代码还有什么用呢？

开发代码时，应该更注重可读性，而不是只图自己方便。代码阅读的次数要远远超过编写的次数，所以在编写的时候值得花点功夫让他读起来更加简单。实际上，从衡量标准上看，代码清晰程度的优先级应该排在执行效率之前。

如果别人给你的代码很容易理解，接下来的工作就省心多了。要尊重这个黄金法则，你欠他们一份情，因此也要让你自己的代码简单、便于阅读。

PIE原则（PIE=Program Intently and Expressively，即意图清楚而且表达明确地编程。）

代码必须明确说出你的意图，而且必须富有表达力。这样可以让代码更易于被别人阅读和理解。代码不让人迷惑，也就减少了发生潜在错误的可能，一言以蔽之，代码应意图清晰，表达明确。

Int result = val << 1;

用位移做乘法，是对代码进行不必要而且危险的性能优化。Result = val \* 2;看起来更清晰，也可以达到目的，而且对于某种给定的编译器来说，可能效率更高。不要表现的好像很聪明似的，要遵循PIE原则：代码要清晰的表达意图。

在编写代码时，应该使用语言特性来提升表现力。使用方法名来传达意向，对方法参数的命名要帮助读者理解背后的想法。异常传达的信息是哪些可能会出问题，以及如何进行防御式编程，要正确地使用和命名异常。好的编码规范可以让代码变得易于理解，同时减少不必要的注释和文档。

正确的做法说：“要编写清晰的而不是讨巧的代码。向代码阅读者明确表明你的意图。可读性差的代码一点都不聪明。”

使用符合当时情形的耦合。例如，通过散列表进行松耦合，这种方式适用于在实际状况中是松耦合的组件。不要使用散列表储存紧密耦合的组件，因为这样没有明确表示出你的意见。

* **保持简单**

Andy曾经认识一个家伙，他对设计模式非常着迷，想把他们全部都用起来。有一次，要写一个大概几百行代码的程序。在被别人发现之前，他已经成功地将GoF那本书中的17个模式，都运用到那可怜的程序中。

问题在于，许多开发人员倾向于将投入的努力与程序复杂性混同起来。开发人员更应该为自己能够创建出一个简单并且可用的设计而骄傲。

简单不是简陋，“简单性”这个词汇被人们大大误解了。他并不是意味着简陋、业余或是能力不足。恰恰相反，相比一个过分复杂、拙劣的解决方案，简单的方案通常更难以获得。

怎样才算优雅？

优雅的代码第一眼看上去，就知道他的用处，而且很简洁。但是这样的解决方案不是那么容易想出来的。这就是说，优雅是易于理解和辨识的，但是要想创建出来就困难得多了。

正确的做法：“开发可以工作的、最简单的解决方案。除非有不可辩解的原因，否则不要使用模式、原则和高难度技术之类的东西。”

当你觉得所编写的代码中没有一行是多余的，并且仍然能交付全部的功能时，这种感觉就对了，这样的代码容易理解和改正。

注意：一个人认为简单的东西，可能对另一个人就意味着复杂。

* **做代码复查**

不正确的做法是：“用户是最好的测试人员。别担心——如果哪里出错了，他们会告诉我们的。”

代码复查和缺陷移除

要寻找深藏不露的程序BUG，正式地进行代码检查，其效果是任何已知形式测试的两倍，而是是移除80%缺陷的唯一已知方法。——Capers Jones的《估算软件成本》

管理层担心进行代码复查所耗费的时间。他们不希望团队停止编码，而去参加长时间的代码复查会议。开发人员对代码复查感到担心，允许别人看他们的代码，会让他们有受到威胁的感觉。这影响了他们的自尊心。他们担心在感情上受到了打击。

Venkat最近参与了一个日程安排非常紧凑的项目，团队不少成员都是没有多少经验的开发者。通过严格的代码复查过程，他们可以提交质量极高而且稳定的代码。当开发人员完成某项任务的编码和测试后，在签入源码控制之前，会有另一名开发人员对代码做彻底的复查。

捡拾游戏：

当某些代码编写完成、通过编译、完成测试，并已经准备签入时，其他开发人员就可以“捡拾”起这些代码开始复查。类似的“提交复查”是一种快速而非正式的方式，保证代码在提交之前是可以被接受的。为了消除行为上的惯性，要在开发人员进行轮换，这是一种很有效的技术。

* **结对编程**

在极限编程中，不存在一个人独立进行编码的情况。编程总是成对进行的：一个人在键盘旁边（担任司机的角色），另一个人坐在后面担任导航员。他们会不是变换角色。有第二双眼睛在旁边盯着，就像是在进行持续的代码复查活动，也就不必安排单独的特点复查时间了。

最基本的检查列表：

* 代码能否被读懂和理解？
* 是否有任何明显的错误？
* 代码是否会对应用的其他部分产生不良影响？
* 是否存在重复的代码？
* 是否存在可以改进或重构的部分？

正确的做法：“复查所有的代码。对于提升代码质量和降低错误率来说，代码复查是无价之宝。如果以正确的方式进行，复查可以生产非常实用而高效的成果。要让不同的开发人员在每个任务完成后复查代码。”

代码复查随着开发活动持续进行，而且每次针对的代码量相对较少。感觉复查活动就像是项目正在进行的一部分，而不是一种令人畏惧的事情。

不进行思考、类似于橡皮图章一样的代码复查没有任何价值。

代码复查需要积极评估代码的设计和清晰程度，而不只是考量变量和代码格式是否符合组织标准。

同样的功能，不同开发人员的代码实现可能不同。差异并不意味着不好。除非你可以让某段代码明确变的更好，否则不要随意批评别人的代码。

如果不及时跟进讨论中给出的建议，代码复查是没有实际价值的。可以安排跟进会议，或者使用代码标记系统，来标识需要完成的工作，跟踪已经处理完的部分。

要确保代码复查参与人员得到每次复查活动的反馈。作为结果，要让每个人知道复查完成后所采取的行动。

* **保持可以发布**

保持你的项目时刻可以发布。保证你的系统随时可以编译、运行、测试并立即部署。

下面是一个简单的工作流程，可以防止你提交破坏系统的代码。

* 本地开发完成后保证代码编译通过
* 把服务器上最新代码同步到本地并解决冲突（如果存在）后确保代码仍能编译通过
* 此时才可提交代码

**关于测试工作方面：**

关于测试工作方面我们需要强调的是测试应该以用户故事为单位进行用例编写和测试工作，并且越早进行集成测试越好。

* **提早集成，频繁集成**

在产品的开发过程中，集成是一个主要的风险区域。让你的子系统不停地增长，不去做系统集成，就等于一步一步把自己置于越来越大的风险中。

集成和独立不是互相矛盾的，你可以一边进行集成，一边进行独立开发。——可以使用mock对象来隔离对象之间的依赖关系这样在集成之前就可以先做测试，用一个mock 对象模拟真实的对象或子系统。

独立开发和早期集成之间是具有张力的，也就是说需要平衡，独立开发时开发速度比较快，生产率更高，但不意味着你要避免或延迟集成。

正确的做法是：“提早集成，频繁集成。代码集成是主要的风险来源。要想规避这个风险，只有提早集成，持续而有规律地进行集成。”

通常，每天要和团队其他成员一起集成代码好几次，比如平均每天5～10次，甚至更多。但如果你每次修改一行代码就集成一次，那效用肯定缩水。（集成也不能过于频繁）

### 需求部门评审

每次迭代开发测试工作结束后都需要邀请需求方对本次迭代的工作成果进行验收，也就是说每次迭代的工作成果必须是可以演示的。

* **使用演示获得频繁反馈-需求就是流动的黑墨**

你时常会听到一些人想要冻结需求。但是世界中的需求就像流动着的黑墨。你无法冻结需求，正如你无法冻结市场、竞争、知识、进化或者成长一样。就算你真的冻结了，也很可能是冻结了错的东西。如果你希望用户在开始之前，就能给你可靠和明确的需求，那就大错特错了，快醒醒吧！

就算客户已经告诉你他们想要的东西了，他们的期望和想法还是在不停地进化——特别是当他们在使用新系统的部分功能时，他们才开始意识到它的影响和可能发生的问题。这就是人的本性。

作为人类，不管是什么事情，我们都能越做越好，不过是以缓慢而逐步的方式。你的客户也是一样。在给你需求之后，他们会不停地研究这些功能，如何才能让他们变的更好用。

每一次把新的功能给客户演示，并收集客户的反馈。都是一次修补你做的软件和客户想要的软件之间误差的行为。这种修补越多最后结果误差就会越小，反之到最后客户说：“这不是我想要的”。

这样做对客户也有好处，他们可以知道项目的真实的进度，他们会根据真实的进度来调整他们的工作的优先级。

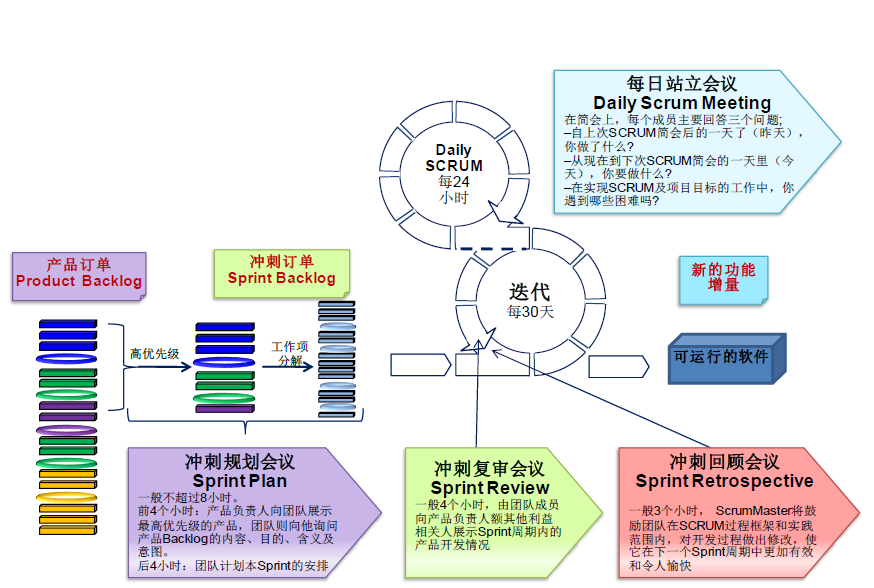
正确的做法是：“清晰可见的开发。在开发的时候，要保持应用可见（而且客户心中也要了解）。每隔一周或两周，邀请所有的客户，给他们演示最新完成的功能，积极获得他们的反馈。”注意：缺少功能或不完善的软件，不应该拿来给客户演示，这样只能让客户愤怒。

在需求部门审核后可能会对本次迭代的结果提出修改的意见，这样的结果很正常，我们在下次迭代时实现需求方的修改意见就好了，需求的变化得到了快速的反应，这样做总比项目都做完了需求方再提出一大堆的修改意见强，项目经理需要邮件邀请需求方对本次迭代的结果进行审核，需求方需要邮件回复审核结果。

### 迭代总结会

每一个迭代完成后，都要举行一次迭代总结会议，在会议上所有项目组要对这个迭代的工作进行总结。举行迭代总结会议是为了进行持续过改进工作方法，会议的时间限制在4小时。会议上应该总结出本次迭代中有益处的实践（行为）和需要改进的实践（行为），以便在下次迭代中进行改进和完善。会议后需要发邮件将会议结果告知团队所有成员。

## 流程图



冲刺 = 迭代

冲刺规划会议 = 计划会议1 + 计划会议2

产品订单 = 完整的需求文档

冲刺订单 = 由用户故事和工作任务组成的燃尽图

每日站立会议 = 每日立会

冲刺复审会议 = 需求方评审

冲刺回顾会议 = 迭代总结会议

# 部署验收阶段

较为重要或复杂的项目需要需求方在内网最后一次迭代验收通过后再进入部署验收阶段，项目较为简单或网络原因导致内网无法验收的项目可以在迭代集成测试通过后进入部署验收阶段，让需求方在外网测试环境进行项目的整体验收工作。（**不建议直接让需求方在外网进行验收，因为如果需求方验收不通过还需要回滚已经完成的部署工作，这样做风险十分高。**）

部署工作需要十分细心因为外网正在运行的系统直接关系到公司的收入和游戏玩家的游戏体验，所以外网部署一般分为两步进行：部署外网测试服务器和部署外网正式服务器。

需求方验收是在外网测试服务器进行验收，需求方验收通过后在约定的时间再部署正式服务器。

部署不是简单上传程序，部署也包括更新数据库和修改配置文件等工作，这些工作的顺利实施需要靠每日立会时总结的《部署清单》，《部署清单》上应该详细的记录数据库和配置文件的修改内容，这样系统部的同事才能正确的执行部署工作。（很多项目的部署工作较为复杂，项目经理应该每日跟踪整理《部署清单》的内容才能避免遗漏和错误）。

程序的部署工作尽量靠自动部署系统进行，这样能快速并准确的上传程序的，避免人为操作失误的可能。

## 工作流程细则

### 部署外网测试服务器进行测试

外网测试服务器是一台和外网正式服务器环境一样的服务器只是不对玩家开放，但公司网络是可以直接访问的，首先要将系统部署到这台服务器上进行外网集成测试，外网环境和内网开发环境或多或少的存在差异，所以外网测试服务器的集成测试十分重要，很多部署工作的问题都是因为环境差异所造成的。外网测试服务部署成功后需要邮件告诉QA人员在外网测试环境下进行集成测试，QA测试人员测试完成后需要回复邮件告知项目组成员外网测试环境测试通过。（外网测试环境测试出现BUG多半是因为部署不完整）

因为部署外网环境需要系统部支持所以需要事先将部署时间及《部署清单》邮件给系统部相关人员进行确认。

### 需求方外网验收

外网测试服务器测试通过后邀请需求方在外网进行验收工作，需求方人员访问外网测试服务器最简单的方式就是让需求方修改HOST文件进行访问，如果网络环境十分复杂可以请需求方人员到开发部进行验收工作。外网验收通过后需要需求方邮件回复验收结果，验收不通过的风险很小因为每次迭代都会找需求方进行验收，这正是敏捷开发带来的好处，万一需求方验收不通过就只能变更项目计划再增加一次迭代进行修改了。

### 部署正式服务器及正式环境测试

部署正式服务器其实就是将外网测试服务器的部署工作在正式环境下再操作一次，因为所有的外网正式服务器都与一台源服务器进行同步所以只要在源服务器上进行部署工作即可。因为部署外网正式环境很多时候都是有时间要求的，基本上没有部署不成功的余地，所以外网正式服务器的部署工作要求是：不能失败。

外网正式环境部署成功后QA人员还需要在正式环境下再进行一次集成测试，这次测试主要是验收部署外网正式环境的工作是否正确，外网正式服务器测试通过后QA要发邮件通知项目组所有成员（含需求方）。

## 流程图

部署外网测试服务器

（开发部、系统部）

需求方内网验收通过邮件（最后一次迭代）

外网测试环境集成测试通过？

需求方外网验收

验收通过？

修改项目计划增加迭代回滚部署工作

部署外网正式服务器

（开发部、系统部）

外网正式环境集成测试通过？

进入结项阶段

# 结项阶段

结项阶段是总结项目过程和汇报项目结果的阶段，善于总结和汇报是所有人必备的技能。

## 工作流程细节

### 结项会议

在结项会议上项目经理要向项目组总结项目的整体情况比如：项目计划与实际结果的差异及原因、整个项目内产生的严重BUG及原因，还要评价每个成员在项目中的表现（有好有坏），最后项目组要讨论一下在这次项目工作中好的工作方法和需要改进的地方，项目经理应该在会后将会议记录邮件给项目组全体成员，结项会议后该项目组解散并告知职能经理，职能经理会将项目组成员分配进入下个项目。

### 结项报告

项目经理要检查项目必要的文档在VSS上是否齐全，项目必要的文档有：立项报告、燃尽图、数据库设计。有些文档是非必要的文档：项目计划文档（立项报告中项目时间为估算时间时才会有）、部署清单（如果没有修改数据库和配置文件的操作则该文档可以没有）。最后项目经理可以编写结项报告了，项目结项报告完成后需要邮件发给项目组全体成员和需求方还有部门助理、部门负责人和宋总。

有三张纸制的记录表也要交给部门助理：每日立会记录表、计划会议1记录表和计划会议2记录表。

## 流程图

会议记录邮件

结项会议

QA外网正式环境测试通过邮件

纸质的会议记录表

结项报告邮件

编写结项报告

**上述项目管理中产生的所有文档和邮件模板均在VSS:\2014年项目管理文档\ 中可以找到。**

参考资料：

敏捷开发的用户故事.pdf

敏捷开发知识体系(ADBOK)V1.0.pdf

Scrum.Checklists.Chinese.pdf