需求管理

最佳实践

文档版本	01
发布日期	2024-05-28

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

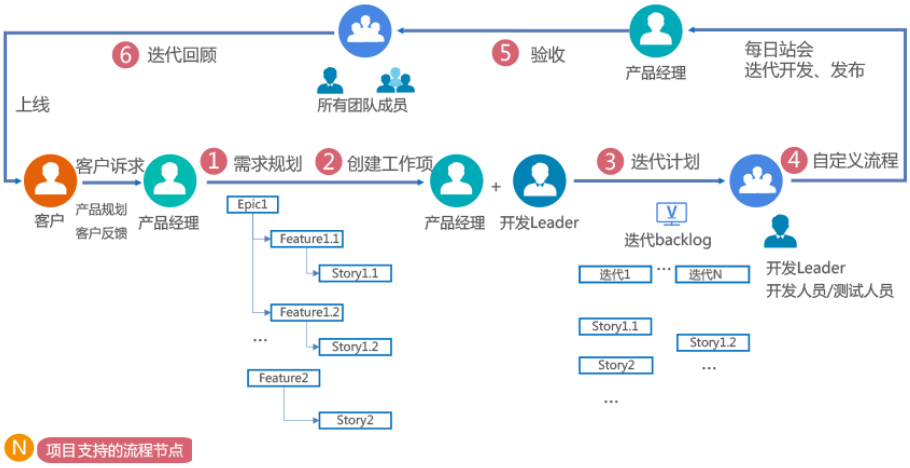
目 录

1 Scrum 项目最佳实践

1.1 Scrum 项目实践概述

Scrum适用于敏捷开发模式的研发管理平台，短周期持续交付，快速响应需求和市场变化。一个完整的Scrum迭代流程大概涉及需求规划、迭代计划、迭代开发、敏捷回顾四个阶段，整体流程图如图1-1所示。

图 1-1 Scrum 迭代流程



本文将基于前期和大量客户交流识别出来的常见问题，总结出Scrum流程中涉及的主要阶段的最佳实践。

1.2 需求管理

1.2.1 如何理解敏捷需求管理的四个关键词

背景

我们常见到Epic、Feature、Story和Task这些和敏捷相关的概念，它们之间的关系是什么？我们如何灵活使用这些概念，从而让敏捷的需求管理更为高效？本文为您详细剖析这四个关键词。

什么是 Epic、Feature、Story 和 Task?

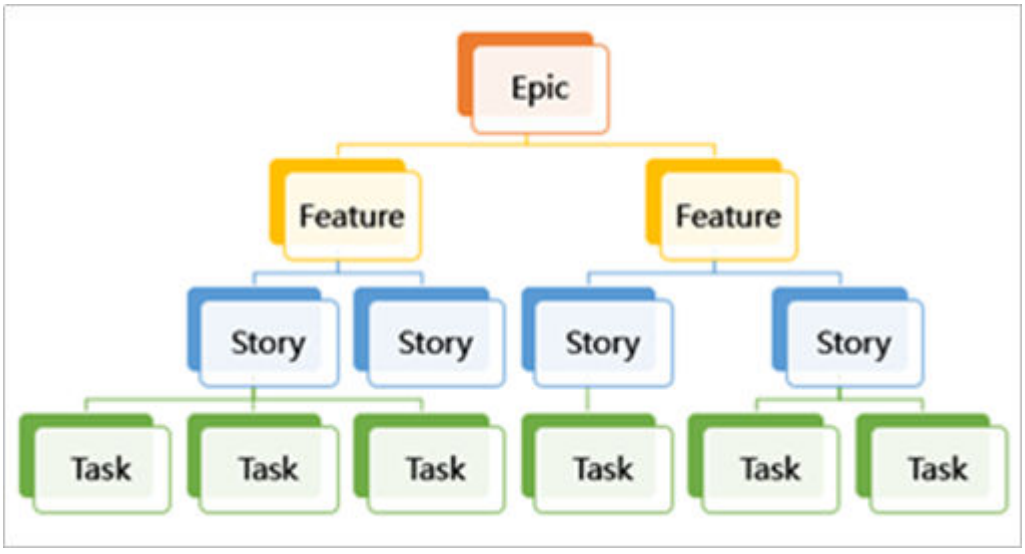
Epic、Feature、Story和Task用来划分需求颗粒度的标签，可以看作需求占位符，分别代表需求颗粒度从大到小。每个层级的需求本身又承载着一些意义，在进行需求划分的时候可以进行参考。

- **Epic**：史诗，是项目的愿景目标。通过Epic的落地达成，使公司可以获得相应的市场地位和回报，具有战略价值。通常需要数月完成。
- **Feature**：可以带来价值的产品功能和特性。相比Epic，Feature更具体，更形象，客户可以感知，具有业务价值。通常需要数周，多个Sprint才能够完成。
- **Story**：通常所说的用户故事，是User Story的简称。Story是从用户角度对产品功能的详细描述，承接Feature，并放入产品Backlog中，持续规划，滚动调整，始终让高优先级Story交付给客户，具有用户价值。Story要符合INVEST原则（**I**ndependent、**N**egotiable、**V**aluable、**E**stimable、**S**mall、**T**estable），通常需要数天，并在一个Sprint中完成。
- **Task**：是团队成员要完成的具体任务。在Sprint计划会议上，将Story分配给成员，然后由成员分解为Task，并预估工时，通常在一天内完成。

Epic、Feature、Story 和 Task 之间关系是什么?

Epic-Feature-Story-Task是一种将需求进行结构化管理的方法，在使用时是从上到下逐层分解，形成自下而上的依赖。如图1-2所示。

图 1-2 Epic、Feature、Story 和 Task 关系图



在实际的开发过程中，需求会发生变化，我们要不断的调整，在调整中避免偏离目标方向，每次新建需求的时候都要记得向上对齐到Epic，保证所添加的Story和Task和它们的上层是有关联的，这样就可以在一定程度上保证团队在朝着目标前进。

更多关于需求结构化管理的内容，请参考**如何进行需求结构化管理**。

我们如何灵活使用这些概念，让需求管理更为高效?

为了加深对Epic、Feature、Story和Task的理解，本文对一个案例进行需求拆分，过程中会结合CodeArts需求管理服务进行展示。

案例:

某大型商超受互联网的冲击，营业额大幅下滑。

为了减少门店消费者流失，保有市场地位和份额，决定用6个月的时间建立自己的网上商城。

- **第一步：Epic确定和创建**

  根据前面的介绍，在进行需求确认的时候先看颗粒度，然后再考虑其承载意义。

  此处需要考虑一个问题：**一个产品是一个Epic吗？产品的每个业务模块是Epic还是Feature?**

  - 产品通常具有战略意义，从这个角度看，产品适合作为一个Epic。但是不是所有的产品都适合，还要看产品是什么，它的颗粒度有多大。在本文的实例中，网上商城周期是6个月，目的是保有市场份额，从颗粒度和战略意义上，网上商城适合作为一个Epic。
  - 每个业务模块具体是Epic还是Feature要分情况。比如：构建智慧城市是一个愿景目标，下面包括智慧交通、智慧政务、智慧社区等，这些每个业务模块都很大，用Epic进行需求占位合适一些。

    在CodeArts创建一个Scrum项目，命名为“某大型商超网上商城”。进入“需求规划”界面，新建Epic:

    图 1-3 新建 Epic

    新建之后，单击进入到详细编辑界面。将描述信息填写完整，可以使用CodeArts提供的模板:

    - **作为**：对于这个Epic来说，用户角色是整个公司。
    - **我想要**：想要的结果就是建造网上商城。
    - **以便于**：目的是想要减少消费者流失，保有市场地位和份额。

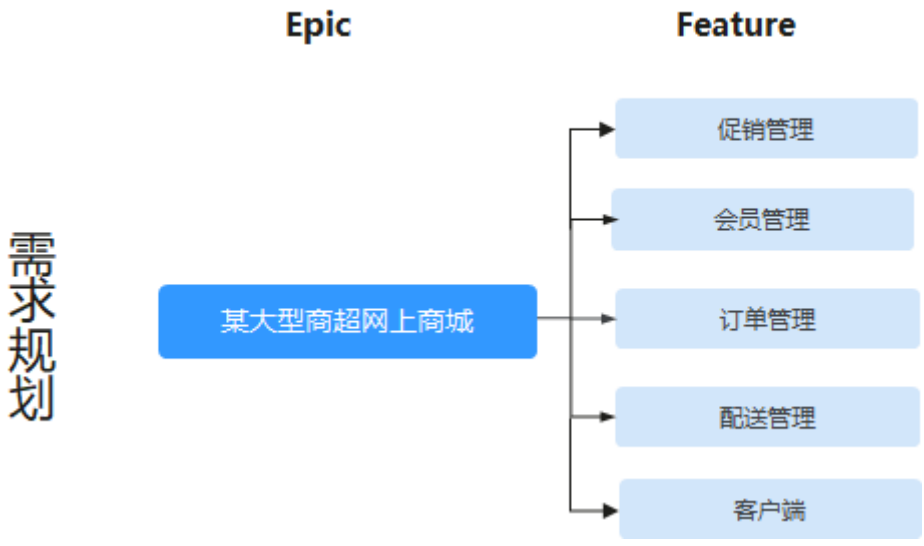
      同时在基本信息中设定这个Epic的起止时间、优先级、重要程度、预计工时等信息。这些信息对于团队理解产品、理解项目起到至关重要的作用，所以要进行详细填写。

图 1-4 编写 Epic

- **第二步：将Epic分解为Feature**

  客户要求在6个月内交付5个功能模块：促销管理、会员管理、订单管理、配送管理和客户端。团队的一个Sprint是2个星期，每个模块大概需要2-3个Sprint完成，从颗粒度和承载的意义，这5个模块适合作为Feature。

  图 1-5 Epic 分解为 Feature



  创建之后，如需要填写详细信息，可以在详细页面进行编辑。界面信息项和前面Epic的相同，此处不再赘述。

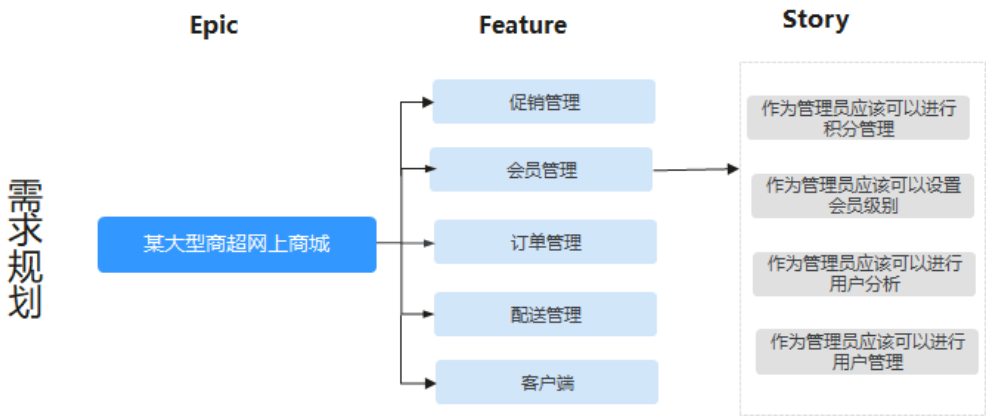
- **第三步：Feature分解为Story**

  敏捷开发是渐进明细的，不要求所有需求在相同时间做到同样详细，只要求当前Sprint和未来的一个或两个Sprint的Story是详细的。将来Sprint的Story可以是一个

大概的情况。进入到当前Sprint的Story要符合INVEST原则。开发团队要在Sprint结束时完成交付。

客户优先级中，会员管理Feature优先级高，会员管理这个Feature就要在需求梳理会议上详细分解为Story放入到产品Backlog中。经过分解后，需要包含和管理员相关的功能：积分管理、会员级别管理、用户分析、用户管理。这些具体的功能就可以作为Story。需要注意的是，我们分解出的Story要尽量在一个迭代内完成交付，如果无法完成就尝试继续分解。因为只有交付的Story才是有价值的，无法交付的Story对于当前Sprint来说就是浪费。分解后的Story如图1-6所示。

图 1-6 Feature 分解为 Story



- **第四步：将Story划分为Task**

  在Sprint计划会议上，团队和PO要共同从产品Backlog中按照优先级顺序选择本次Sprint需要完成的Story，进入到冲刺Backlog中。团队成员认领后，将Story分解为Task，并进行估算。

  此时面临一个问题：**Story和Task如何区分?**

  - **Story**聚焦价值，需要在Sprint中完成，要用数天的时间，要符合INVEST原则。Story的描述是一个名词，如积分管理这个Story的完整描述是：作为管理员，我能够进行会员的积分管理，以此来划分消费等级提供不同增值服务。
  - **Task**聚焦实现价值，通过过程性的任务来实现Story的功能。通常是1～8个小时。Task的描述是一个动作。如积分管理这个Story，功能的实现需要通过业务逻辑开发、积分规则设计和积分数据库设计这几个过程来完成，这些就是Task。如图1-7所示。

图 1-7 Story 分解为 Task

这样，从Epic开始，到Task结束，我们完成了网上商城的需求拆分。

小结

使用Epic、Feature、Story、Task管理需求时，需要注意以下几个方面：

1. 敏捷开发中需求是逐步细化的，遵循自上而下的方式去分解需求。
2. Epic和Feature都是颗粒度比较大的需求，是用户对于产品的期望和功能特性的描述。
3. 分解为Story的时候，目前正在进行的Sprint需要分的更小更细，将来的Sprint需求（可能是3个及以上）就不需要那样细分。当进行到某个Sprint时，再进行分解，细分成一组更小、更细的条目。
4. Task是对当前Sprint的Story进行的分解。
5. 所有这些粗略和详细的Story都放在产品Backlog中，整个列表要遵循DEEP（Detailed appropriately、Emergent、Estimated、Prioritized）原则，定期梳理和排序优化，保证高优先级的需求优先实现和交付。
6. 在整个过程中需要和客户一直保持沟通合作，这样才能保证我们实现的功能是客户真正想要的。

本文通过一个用户场景来帮助大家理解Epic、Feature、Story、Task的含义以及如何使用。在现实业务中，没有相同的项目。因此在开发过程中还要结合产品和业务本身的特点，进行具体问题具体分析。

1.2.2 如何在软件项目需求变更频繁的情况下做好有效的需求管理和规划

概述

围绕项目需求变更频繁，如何做好有效的需求管理和规划，本文将从背景、问题分析、解决措施几个方面进行细致讲解。

背景

不管是项目型软件开发还是产品型软件开发，需求变更频繁都是影响研发效能的第一号因素，在2019年中国DevOps现状调查报告中也可以发现，超半数企业认为需求的频

繁变更是阻碍软件按时交付的主要原因。解决或缓解需求变更频繁带来的影响，是势在必行的重要工作。

问题分析

由于每家企业的情况不同，包括客户合作方式、人员能力水平、研发流程等各方面的差异，同样是需求变更频繁，所体现出来的具体症状却有所不同，导致问题发生的根因也可能不同，所应采取的措施也需要根据实际情况来选择。根据我们的观察以及与企业交流的经验发现，一般都体现为如下几种场景：

- 需求杂乱、经常变更，难以管理。
- 需求优先级的不断调整，打乱了开发计划。
- 需求遗漏。

接下来我们结合这些情况的部分实例来分析：

- **场景一：软件项目前期结构清楚，开发到后期，需求变化多而细，如何管理，如何规划。**

  此场景中，困扰项目成员的是前期需求不明确、不完善，导致后期需改动，需求发生变更。

  出现这种情况，往往跟需求规划未能被正确使用有一定关系，比如需求层次划分不清晰、缺少规范机制等问题。例如，某客户规划一个用户登录功能，按照下图所示规划需求。用户会将其中管理员登录的Task放在第一个版本中发布，后期又增加了一个手机号登录的需求，设置成Task放在第二个版本中发布，这样一个Story里面存在多个不同版本（或迭代）发布的Task，不方便管理。由此可以将这个问题的根因定性为**如何进行需求结构化管理**的问题：

  a. 没有区分跟随项目进展而持续产生的碎片化需求和系统/产品持续完善的功能特性。

  b. 对CodeArts提供的Epic-Feature-Story的需求结构理解有误，未能正确使用。

  图 1-8 需求结构

- **场景二：软件项目进行过程中，领导需要提拉需求，在敏捷研发模式中该如何去操作?**

  提拉需求的意思也就是要将某些需求的优先级提高，要求团队先实现它们，因而可以将此问题定性为**需求优先级管理**的问题。解决此问题，需要了解：

  – 为什么领导会要提拉需求？如果是合理的，那么团队就应该提升响应能力、优化工作安排流程，使得优先级调整对研发进展带来的影响最小化，且能够尽快地响应领导需要，先交付被提拉的需求。

  - 这种情况发生频率有多高？如果是经常发生，那就是一种常态，而且是一种不好的常态，那团队需要去思考是什么导致了这种常态发生，并考虑如何从流程、制度、协作模式或人员能力等方面去做调整，减少过程中提拉需求情况的发生；如果是偶尔发生，那就可以针对具体情况找到解决方案，没有必要为例外情况调整流程、制度，这样反而会加重常态工作的负担。
  - 需要提拉的需求有无共性特点？比如是否都跟某个客户有关，或者跟某个功能域（如退款）有关？如果能够找到共性，那我们就可以针对这些共性去思考针对性的解决方案。

- **场景三：由于外界原因经常会临时增加一些紧急需求，并且这是目前常态**

  临时增加需求，首先是一个如何处理突发需求的问题；紧急需求，也就是说需要马上就做，肯定也是重要的需求，所以这还涉及到需求优先级管理的问题。

  当两种情况合在一起，需要将它定性为是**重要需求遗漏**的问题，反问一句就是——为什么这些紧急重要的需求无法更早预见？同样的，团队需要了解：

  - 具体是哪些外界原因？这些原因是否有共性，有的话，那就针对性处理。
  - 增加的需求有无共性特点？有的话，可以针对性处理。
  - 临时增加有多临时？团队是否有提高或改善响应能力的空间，如果团队可以更快调整和响应，使得这些临时需求对团队产生不了什么影响，那么这个问题也就不再是问题了。
  - 既然是常态，为何团队的流程没有做出调整去应对？是调整过流程或工作方式，还是无法解决问题，还是说不知道该怎么调整流程或工作方式去适应？

解决措施

综合前面几种参考情况经分析后得出了根因，基于这些根因，可以将所要解决的问题重新描述如下：

1. **如何进行需求结构化管理？**
2. **如何进行需求优先级管理？**
3. **如何避免重要需求遗漏？**

- **如何进行需求结构化管理？**

  首先，并不是说任何情况下都需要进行需求的结构化管理。只有在需求较多、且需求之间存在关联，而且即便是已经实现的需求也需要进行一定的管理、维护的情况下，才需要去思考需求结构化管理的问题，此时，团队需要使用CodeArts提供的Scrum项目模板，因为里面有Epic-Feature-Story的需求结构，以及需求规划功能可以辅助团队进行需求的结构化管理。那么团队应该以什么为脉络来建立这个结构呢？这就意味着，**团队的需求结构化管理，需要以产品或系统的功能特性的脉络为依据**。而软件项目管理所需要关注的版本、客户、模块等信息，则可以通过需求的不同属性甚至标签等方式来实现。

  简单来说，可以通过如下三个步骤来完成：

  a. 针对产品或系统建立CodeArts项目。
  b. 确立Epic-Feature-Story的需求结构。
  c. 对不同模块以及版本的管理，可以通过工作项的属性来进行管理。

  更多详情请参考**如何进行需求结构化管理**。

- **如何进行需求优先级管理？**

  需求优先级的管理，其实是为了帮助团队确定先做哪个需求后做哪个需求，从而可以最大化团队的回报、最小化团队的风险或投入。要做好优先级管理，或者更直接来说是优先级顺序管理，我们需要做到如下几件事情：

a. 确定优先级模型：需要考虑的因素以及因素的综合判断原则，比如**Kano模型**。
b. 排定需求优先级顺序：因素的具体量化和排序标准，例如成本收益法是按照收入还是按利润的多少来排序。
c. 调整需求优先级顺序。
d. 改进优先级模型：根据反馈调整模型或模型的落地实施细节，以提升效果。

更多详情请参考**如何进行需求优先级管理**。

- **如何避免重要需求遗漏?**

  根据重要需求遗漏的事前、事中、事后的不同时间点，我们可以采取不同的措施。参照八二原则，我们需要确保常态问题有对应的处理方式，软件项目成员按照既定方案进行处理即可，而特殊情况要有应急机制指导现场处理、事后再复盘总结。

  a. 事中的处理：按照常规做法进行处理，或是特殊情况特殊处理，先解决眼下的问题。
  b. 事后的处理：基于模型或思路进行复盘，并落实为新的常规做法或特殊情况处理方式。
  c. 事前的处理：明确如何区分常规情况或特殊情况，并制定相应的处理方式或应急机制。

  更多详情请参考**如何避免重要需求遗漏**。

1.2.3 如何进行需求结构化管理

为什么要进行需求结构化管理?

并不是说任何情况下都需要进行软件项目需求的结构化管理。如果只是事务性质的管理需求，也就是有需求了能记录、能跟踪状态、实现之后不需要继续跟踪、也不需要维护需求与需求之间的关联，那么不需要思考需求结构化管理这个问题。这种情况下，不管是用CodeArts的Scrum项目模板还是看板项目模板，都可以管理好需求和软件项目。只有在需求较多、且需求之间存在关联，而且即便是已经实现的需求也需要进行一定的管理、维护的情况下，我们才需要去思考需求结构化管理的问题，此时，我们需要使用CodeArts提供的Scrum项目模板，因为里面有Epic-Feature-Story的需求结构，以及需求规划功能可以辅助团队进行需求的结构化管理。

以什么为依据进行需求结构化管理?

需求结构化管理，应该以什么为脉络来建立这个结构呢？软件研发无非是分为项目型软件研发和产品型软件研发两种，项目通常来讲都是临时性的，或者说短期性的，而产品或者软件系统是长期性的，或者说会持续维护、更新其功能特性的。项目复项目，我们很可能通过持续地完善和刷新同一套软件产品或系统来达成项目目标，交付软件项目所要求的功能特性的。这就意味着，**我们的需求结构化管理，需要以产品或系统的功能特性的脉络为依据**。而软件项目管理所需要关注的版本、客户、模块等信息，则可以通过需求的不同属性甚至标签等方式来实现。

使用 CodeArts 进行需求结构化管理的一种方式

接下来，我们介绍推荐的一种方式。

- **第一步：建立CodeArts项目**

  针对一个产品或系统，建立一个CodeArts项目，该产品或系统的所有需求，都在此CodeArts项目里面进行管理。

- **第二步：确立Epic-Feature-Story的需求结构**

  a. Epic要承载业务价值，即Epic需要是对企业本身是有意义的。

  将产品或系统的业务模块作为Epic，比如用户中心、购物车、配送管理等，比如一家货运云商的油卡业务，就适合作为一个Epic，针对油卡的各种功能，就可以作为Feature展开。

  b. Feature要承载用户价值，也即对于用户来说，是可以理解这个Feature，且认可其价值的，通常Feature也是用户可以直接感知、可以操作的。

  针对前面业务模块的具体展开、拆开，就可以作为Feature，也可以简单理解为一个业务流程、用户流程；以前面用户中心为例，用户信息可以是一个Feature、我的订单可以是一个Feature、地址管理可以是一个Feature；或者以油卡为例，购买油卡、我的油卡等就可以作为不同的Feature。

  c. Feature往往还是有些大有些复杂，那就需要拆成颗粒度更小的Story，用来承载一个具体的用户操作。

  例如可以查看到所有订单、可以过滤订单、可以修改用户昵称、可以自定义头像等功能。

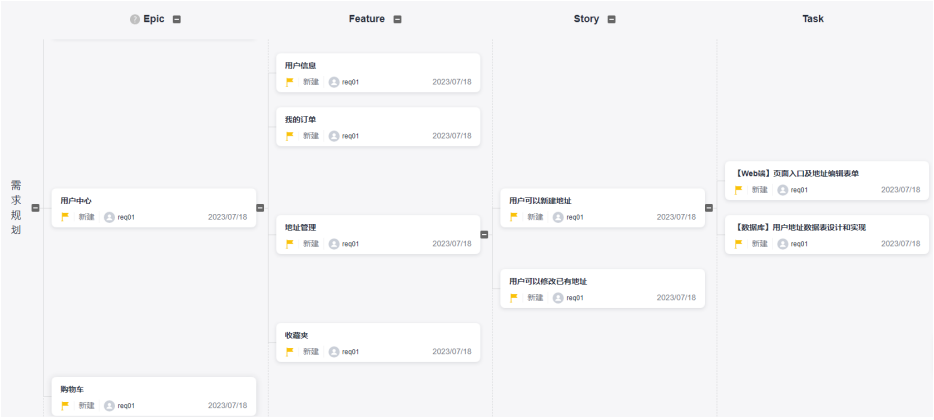
  d. 再往下一级的Task，就主要是为了分工协作，也即是说，如果Story可以包干到人，那么不再拆分Task也是可以的。

  Task往往是关于工程师需要具体做的工作，也就跟业务价值、用户价值、用户单步操作都没有了什么关系，通常都是把Story按照具体的组件、模块进行拆分，例如前端、后台、数据库之类的，或者是按照工作流程分工来拆分，例如UCD、开发、测试、部署等。

  如下图所示，各层级为：

  - Epic：用户中心。
  - Feature：地址管理。
  - Story：用户可以新建地址。
  - Task：【Web端】页面入口及地址编辑表单、【数据库】用户地址数据表设计和实现。

  图 1-9 需求规划

- **第三步：通过工作项的属性，对于不同模块以及版本进行管理**

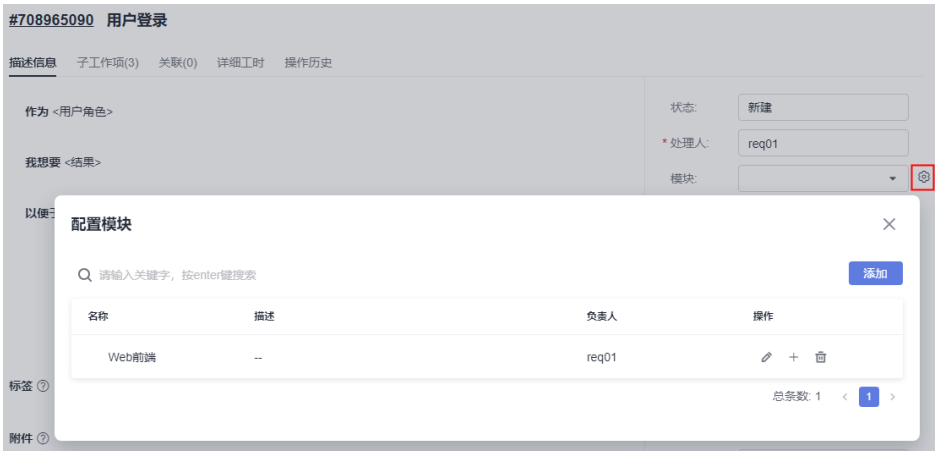
  工作项详情页面对应属性如图1-10所示：

  - 模块：Web端。
  - 发布版本号：1.0.1.2。

图 1-10 属性示例

对于模块清单的维护，可以在工作项编辑状态，单击“模块”字段右侧的，即可在弹出窗口进行操作，可以添加、修改、删除模块。

图 1-11 编辑模块



在工作项管理的Backlog视图下，通过“设置显示字段”增加“模块”字段后，既可以很方便地看到工作项相关的模块，也可以进行过滤。

1.2.4 如何进行需求优先级管理

需求优先级管理四步走

需求优先级的管理，其实是为了帮助我们确定先做哪个需求后做哪个需求，从而可以最大化我们的回报、最小化我们的风险或投入。要做好优先级管理，或者更直接来说是优先级顺序管理，我们需要做到如下四件事情：

1. 确定优先级模型：优先级看起来像是一个简单直接的值，但实际上它是一个基于多种因素进行综合判断之后得出的一个值，这些因素和判断原则，就是我们所说的优先级模型。
2. 排定需求优先级顺序：将需求代入优先级模型进行计算，得出每个需求的优先级顺序。
3. 调整需求优先级顺序。
4. 改进优先级模型：如果经常发生需要调整需求优先级顺序的情况，那么应该对这些情况进行一定的复盘分析，如有必要，修正或改进当前的优先级模型，让它可以适应实际情况，以避免调整优先级顺序的情况反复发生；另外就是需求可能已经交付或发布上线，但是该功能的实际用量或价值不吻合预期，则需要反思我们对这些需求的分析和判断，究竟是分析判断有误还是优先级模型有误，并进行相应的调整。

确定优先级模型

成本收益分析就是最简单的一种优先级模型，重要/紧急的四象限也是一种优先级模型，**Kano模型**也是一种优先级模型，它们都可以帮助我们去确定需求的优先级顺序。模型可以简单也可以复杂，根据企业实际需要来确定即可。

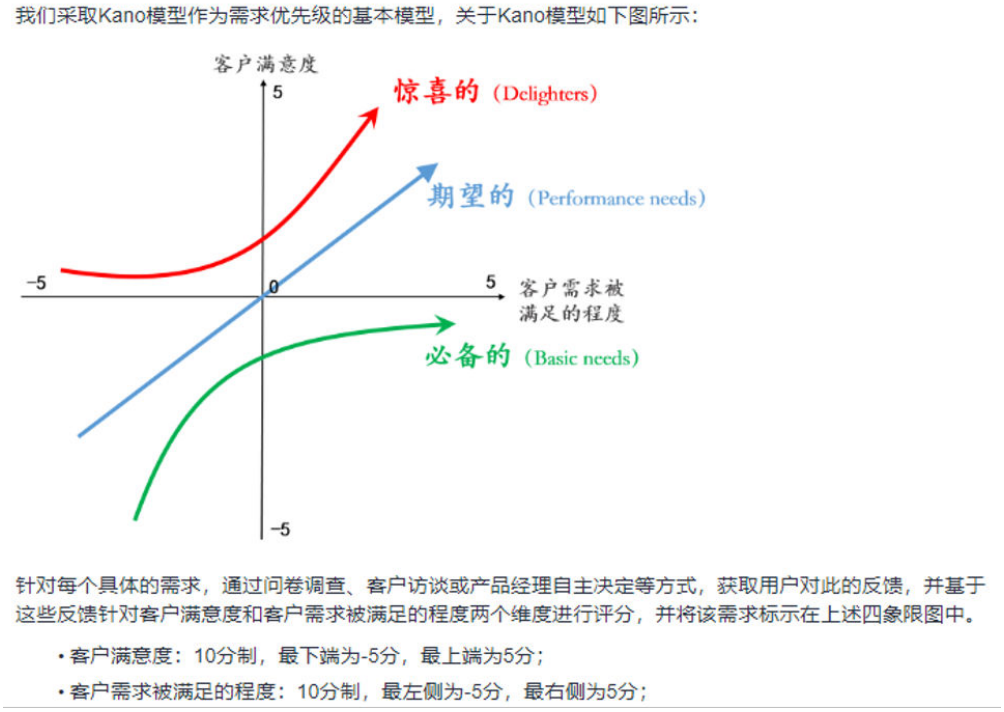
务必切记优先级模型一开始不应过于复杂，那样会导致优先级管理的管理开销过高，喧宾夺主，反而影响了需求的开发和交付。如果较为简单的模型就可以满足需要，就应该首选使用较简单的模型。企业可以从简单开始，逐渐完善，不需要也不应该在一开始就追求过于复杂的模型。

- 简单可以体现在考虑的要素更少，比如成本收益分析只考虑两个要素，就比考虑更多要素的模型简单。

- 简单还可以体现在要素的取值范围更窄或精度要求更低，比如预计利润只要求评估高/中/低，就比要求以万元为单位评估预计利润更简单。

优先级模型确定后，可以进行存档管理，注意该模型宜供所有人或相关人员查阅学习，比如录入到CodeArts的知识库就是一个很好的做法。

图 1-12 优先级模型



排定需求优先级顺序

比如成本收益分析，可以是把预期市场收入作为收益值，把预期研发投入作为成本值，计算差值，或计算ROI均可。假设需求A预计收益为10万元，研发投入按人天折算预计3万元，那么预计利润就是7万元，预计ROI是233%；需求B预计收益为5万元，研发投入折算预计4万元，那么预计利润1万元，预计ROI为25%。那么需求A的优先级顺序就要比需求B更靠前。这种相差悬殊的情况往往不难判断，我们假设还有需求C预计利润也是7万元、预计ROI是50%，以及需求D是预计利润1万元、预计ROI是500%。那么A、B、C、D这四个需求的具体顺序怎么排定呢?

如果真的出现这种情况，那就更复杂一些了，需要考虑引入权重，然后计算出一个综合值，这个值按照某种规则（例如从大到小）排列出来就是最终的优先级顺序，比如：

需求名称	预计收入（万元）	预计成本（万元）	预计利润（万元）	利润权重	利润加权值	ROI（%）	ROI权重	ROI加权值	综合值	优先级顺序
需求A	10	3	7	0.1	0.7	233%	1	2.33	3.03	2

需求名称	预计收入（万元）	预计成本（万元）	预计利润（万元）	利润权重	利润加权值	ROI（%）	ROI权重	ROI加权值	综合值	优先级顺序
需求B	5	4	1	0.1	0.1	25%	1	0.25	0.35	4
需求C	21	14	7	0.1	0.7	50%	1	0.5	1.2	3
需求D	2	1	1	0.1	0.1	500%	1	5.0	5.1	1

根据上述表格中所得出的结果，我们就应该依序将需求D、需求A、需求C、需求B排入开发计划。优先级顺序，在CodeArts中，可以使用工作项的“优先级顺序”字段来实现，该字段取值范围1~100。

图 1-13 Story 工作项优先级顺序展示

调整需求优先级顺序

调整顺序本身非常简单，只要在CodeArts中重新设定该需求的“优先级顺序”字段的值就可以。但重要的是，需要将优先级顺序调整这件事情记录下来，包括为什么要调整、具体如何调整的、调整背后的具体考虑等信息都记录下来，同样，建议记录在知识库中。用于后续的复盘回顾中作为参考信息，比如每个Sprint/迭代结束时的回顾会议上拿出来进行探讨。

图 1-14 Sprint 回顾会议记录

改进优先级模型

市场在变化，用户在变化，产品在变化，优先级模型自然也必须跟随着发生变化。我们可以定期或不定期的安排对需求优先级模型进行复盘分析，找出可以改进或优化的点，并跟进落实。可以是定期开展，例如每个月进行一次复盘，把这个月所涉及的需求都拿出来审视，或者是其中有调整过优先级顺序或者出现过问题的需求拿出来审视均可；也可以是不定期，以问题驱动的方式，比如某天进行了大量需求优先级的调整，那么当天或第二天就可以临时召集一次复盘会议，分析为什么会发生这种情况。

复盘要有好的效果，就必须尽可能的复原问题发生当时的情况，所以前面提到的知识库记录就变得非常重要。复盘会议应提供尽可能多的相关信息以便参会人员了解情况，充分探讨。

复盘过程中，我们要定位出正确的根因，是模型本身设计有问题（例如要素和尺度），还是取值、加权有问题（比如某类需求的预计收入就是非常难估算），还是过程管理的问题（比如过早进行估算，因为缺乏必要信息，导致估算得出的优先级顺序不准确），并进行针对性地改进。

1.2.5 如何避免重要需求遗漏

避免重要需求遗漏的思路

避免重要需求遗漏，首先我们需要反问一句——为什么这些紧急重要的需求无法更早预见？同样的，我们需要了解：

- 具体是哪些外界原因？这些原因是否有共性，有的话，那就针对性处理。

- 增加的需求有无共性特点？有的话，可以针对性处理。
- 临时增加有多临时？我们是否有提高或改善响应能力的空间，如果我们可以更快调整和响应，使得这些临时需求对我们产生不了什么影响，那么这个问题也就不再是问题了。
- 既然是常态，为何我们的流程没有做出调整去应对？是调整过流程或工作方式，还是无法解决问题，还是说不知道该怎么调整流程或工作方式去适应？

具体操作方法

具体操作，可以按照事前、事中、事后各个阶段来采取不同的措施处理。

1. **事中的处理**

   根据具体情况不同，在发现需求遗漏的当时，可以采取如下一些做法：

   - 重要需求遗漏，不紧急：既然不紧急，按照常规做法增加进去即可，但如果经常出现遗漏，就要考虑是否是需求分析和规划的实践做法有问题，才会导致问题持续出现，这种情况，应强化需求结构化管理，从全局出发进行思考和规划，避免因为思考的片面化和局部性导致的遗漏。
   - 重要需求遗漏，紧急：既然是又重要又紧急的需求，那么必然就得调整当前开发工作的顺序，把这个遗漏的重要紧急需求排进去，把工作安排下去；然后就要考虑从需求的优先级和需求的结构化管理两个方面入手复盘，并切实改进，避免类似情况再次发生。
   - 需求遗漏：如果是不太重要的需求遗漏，按照常规做法处理即可；可以根据其紧急程度和影响，决定是否调整工作顺序让这个需求插队；如果这种情况反复出现，那建议可以考虑进行复盘，从需求结构化管理的角度进行分析，并商讨改进措施。

2. **事后的处理**

   事后其实就是复盘，复盘的关键是要基于盘来推演和分析，这个盘就是事前制定的模型和规范。是我们有模型有规范，但执行出了问题？还是说这几个需求情况特殊，模型比较简单没有覆盖到这些特殊情况？还是说模型和规范都没问题，就是人员能力不足，导致判断偏差大？只有找到正确的根因，才能够真正有效的解决问题，所以我们不复盘则已，要复盘就务必要认真严格地进行复盘。

   怎么复盘？复盘也是有方法，业界也有相关书籍可供我们参考借鉴。例如温伯格在《 成为技术领导者 》中提出的MOI模型就可以用作复盘的一种思路。

   - M：激励（ Motivation ），是不是人们没有动力去做这件事情？
   - O：组织（ Organization ），是不是无组织无纪律、一片混乱，人们不知道自己或别人该做什么？
   - I：想法或创意（ Idea/Innovation ），是不是缺少如何解决这些问题的点子或创意，不知道有什么办法解决这个问题？

   复盘时要注意，受限于能力、经验以及出问题次数多少的影响，我们可能无法得出一个准确的结论和必然有效的解决方案。此时，一方面需要秉持持续改进的心态，我们可以先落实当前已经比较明确的改进措施，后续再观察效果，持续复盘、持续改进即可。另一方面我们也可以先采取一些临时措施。

   a. 预留时间：比如，如果确实很难分析清楚为什么总是会遗漏需求，无法进行非常有针对性的处理时，也可以采取较为模糊应对的方式。可以拉取过去一段时间的工作记录，评估这段时间每个迭代的突发需求所消耗的工作量投入，可以取个平均值，然后在后续进行迭代工作安排的时候，固定的预留出一定量的时间，用于应对极有可能会出现的突发需求。
   b. 需求拆细：当出现突发需求，导致我们需要调整工作顺序时，很有可能会因为需求颗粒度大以至于腾挪余地有限，而难以避免突发需求带来的影响，因

而还应该尽可能地采取拆细需求的方式，将颗粒度比较大的需求拆分为较小颗粒度的需求，可以增加调整需求工作顺序时的灵活性。

确定要预留多少时间，可以利用CodeArts的Epic-Feature-Story结构，把突发需求汇集在一起，便于统计。例如创建一个特殊的Epic“突发需求”，下一级是为每个迭代创建的Feature，用来承载各个迭代里面具体的那些突发需求（体现为Story），并做好工时的记录，迭代结束后，就可以来计算出现了多少个突发需求、投入了多少工作量了。

图 1-15 规划迭代

也可以采用“模块”字段来辅助记录和统计突发需求的数据。例如，新建一个模块，取名“突发需求”，所有突发需求都标注为这个模块，那么后续就可以基于模块进行筛选或查看报表等方式来统计突发需求所消耗的工作量了。

图 1-16 迭代内容

3. **事前的处理**

事前的处理放到最后来介绍，是因为已经出现了问题就需要在当时尽快处理，所以先介绍了事中的处理。但当我们处理完问题也完成了事后复盘，就需要考虑未来的事前，尽可能的避免问题发生。 简单来讲，事前的话，就是要做好需求的结构化管理和需求的优先级管理，以及做好相关规范的宣导、人员的分配和能力的培养，这样就能够有效的避免或减小突发需求带来的影响了。

1.3 迭代计划

1.3.1 如何合理规划 Sprint 时间盒

背景

一个7人左右的团队采用Scrum框架工作。Sprint的长度，团队目前采用时间盒为一周。团队经常会出现在Sprint结束时不能完成当初设定的Sprint目标，很多工作项需要跨Sprint才可以完成。

问题分析

目前Sprint中存在的主要问题是Sprint目标完成不好，解决障碍，Sprint目标按承诺完成即可。

团队成员的工作内容中包含很多探索性工作项，对工作内容领域不熟悉，需要投入一些学习成本，导致工作项的实际完成用时要比正常多。每个用户故事的工作量也比较大，多数超过24小时。PO对工作项完成标准的要求非常高，评审严格，不合格的工作项在Sprint中经常返工。团队当前的Sprint时长为一周，并且四大事件按照Scrum框架执行，其中Sprint计划会议和Sprint回顾会议平均持续时长为2小时左右。

从分析中归纳影响Sprint目标完成的几个主要因素如表1-1所示。

表 1-1 影响 Sprint 目标因素表（一）

序号	影响因素	具体分析
1	探索性工作项多	无法改变现状，学习成本的投入是必须项。可以考虑团队成员间知识共享，随着能力提升学习成本会逐渐减少。
2	工作领域不熟悉，需要学习成本	
3	用户故事比较大	由于工作项的特殊性，用户故事普遍比较大，可以考虑拆分为更小的故事，或者在每项用户故事下建Task。Sprint的目标可以按Task级别来平衡考量完成度。
4	PO完成标准高	进一步理解PO完成标准，在计划会议上需要明确验收标准，包括AC（Acceptance Criteria）和 DoD (Definition of Done)。
5	Sprint周期短	团队初始确定的周期长度为一周，经过几轮Sprint后，如果Sprint目标完成不理想，根据工作项特殊性考虑到周期有点短，适当调整周期。
6	Sprint计划会议和Sprint回顾会议持续时间略长	每项会议其实是有建议的时长，正常一个月的Sprint周期，建议Sprint计划会议为8小时，那么按比例一周的Sprint，建议计划会议为2小时时长。同样，一个月的Sprint周期，建议Sprint回顾会议不超过3小时，显然，对于一周Sprint时长的回顾会议用掉2小时是严重超时的。

解决措施

针对以上问题的分析，建议这种情景下将Sprint的时间盒由一周改为两周。Sprint的时间过短，团队成员会忙于准备计划会议、评审会议及回顾会议，真正完成工作项的时间较少。对于突发事件的应对能力减弱，不利于形成团队稳定的节奏。Sprint的时间过长，失去了短时间盒的优势，失去了时间盒的意义。因此，如果团队在时间盒为一周的Sprint中经常不能完成Sprint目标，可以试着把时间盒调为两周。同时要注意优化用户故事的大小，提高四大事件的效率。

Sprint的时间盒由一周改为两周后影响因素会有所改善，具体如表1-2所示。

表 1-2 影响 Sprint 目标因素表（二）

序号	影响因素	Sprint由一周改为两周后的相应缓解
1	探索性工作项多	有相对充分的缓冲时间应对学习、探求性工作以及突发情况。
2	工作领域不熟悉，需要学习成本	
3	用户故事比较大	时间相对宽裕后，计划会议开得更充分，用户故事拆分为更小的故事，或者在每项用户故事下建Task。Sprint的目标可以按Task级别来平衡考量完成度。
4	PO完成标准高	时间相对宽裕后，理解PO完成标准更加充分，在计划会议上明确验收标准，包括AC（Acceptance Criteria）和 DoD (Definition of Done)。
5	Sprint周期短	调整周期后，团队成员氛围更加好，每个Spring目标完成度得以改善，成员更加自信。
6	Sprint计划会议和Sprint回顾会议持续时间略长	时间相对宽裕后，每项会议质量有所提高，在合理的时间范围内可以高质量完成会议内容。

其情况Sprint的时间盒长度建议如下，您可根据团队现况选择适合的时间盒。

- 一周到两周

  新产品研发团队，产品具有时效性的特点，业务需求迎合市场随时调整，灵活多变，快速响应业务需求，经常性的自检。
- 两周

  团队节奏相对稳定，故事点较大，不好拆分，需要更多的时间评审和返工修正。
- 三周到四周

  节奏稳定，团队稳定，需求稳定，团队有固定的节奏，浪费较少。

了解更多：时间盒

在Scrum框架中，工作在建议时间长度为一个月或者更短的迭代或循环中进行，这个迭代或者循环叫做冲刺。冲刺在一个时间盒（Time Box）内，也就是冲刺有固定的开始和结束时间。

- **时间盒的优点具体内容如下**：
  a. 时间盒是设定WIP（work in process）数量限制的技术。WIP是已经开始但还没有完成的工作清单。开发团队只开发自己认为在一个冲刺内可以开始并按时完成的工作事项，因此时间盒是为每个冲刺设定WIP数量限制。
  b. 时间盒可以强制排列优先级。我们需要先执行高优先级的工作，时间盒可以强制我们按优先级排序执行小批量工作，这样我们的注意力可以更集中于快速完成高价值的工作。
  c. 时间盒可以展示进度。时间盒可以展示团队需要多少个时间盒才能完成大特性的进度，帮助团队准确知道为交付整个特性还需要做多少工作。

d. 时间盒可以限定结束日期。每个冲刺中，时间盒限定了一个固定的结束日期，通过这种方式强制结束可能无休止的工作。

e. 时间盒可以促进结束。冲刺有明确的最后期限，这个期限不允许修改，这可以激发Scrum团队成员全力以赴按时完成冲刺内的工作。如果没有时间盒的限制，Scrum团队成员就不会有完成工作的紧迫感存在。

f. 时间盒可以增强预测性。团队可以不预测后续长时间段内可以完成的工作，但是预测下个冲刺内能够完成的工作是可以做到的。

- **每个冲刺持续期短有很多好处。**持续期短的冲刺更容易规划，反馈快，错误有限，投入产出比（ROI）高，有助于团队成员保持较高的参与热情，检查点多。具体好处如下：

  a. 持续期短的冲刺更容易规划。为短时间的工作范围做规划所需要的工作量比给长时间的工作范围做规划的工作量要小得很多，结果更准确，可执行性更强。

  b. 持续期短的冲刺可以产生快速的反馈。快速反馈可以去掉不适宜的产品路径和开发方法，避免产生更多的差产品质量成本（COPQ，Cost of Poor Quality）。最重要的是快速反馈可以使团队更快速地发现和利用稍纵即逝的商机。

  c. 持续期短的冲刺所犯的错误也是有限的。在短短一周或两周的时间内，就算出现失误，也只是失去了短短的一周或两周的时间，不会带来巨大的损失。坚持持续期短的冲刺能够进行频繁地试错，协调和反馈。

  d. 持续期短的冲刺投入产出比（ROI）更高。持续期短可以更早、更频繁地交付，有更多的机会快速投入生产，产生收入。

  e. 持续期短的冲刺有助于保持较高的参与热情。团队参与工作的热情，兴趣和兴奋程度会随着时间的拉长而越来越弱。如果一个项目时间过于长，人们看不到结果，那么显然人们会逐渐失去兴趣。持续期短的冲刺通过频繁快速的交付可用的工作产品，让参与者有满足感，有较高的参与热情，使团队成员恢复兴趣并渴望继续完成冲刺的目标。

  f. 持续期短的冲刺能提供多个有意义的检查点。传统瀑布式开发有里程碑，例如分析、设计、编码、测试和运行。这些里程碑其实是一些不太准确的指标。Scrum在每个冲刺结束时会有一个有意义的检查点（冲刺评审会议），团队中的每个人可以根据展示的可以工作的特性做出判断和决策。有更多的检查点来检验和修正，我们就能更好地应对复杂的项目。

1.3.2 如何移动迭代中需求变更后看板中的任务卡片

背景

围绕迭代中需求变更后，如何移动看板中的任务卡片，本文将从正常情况下的移动和需求变更情况下的移动两个方面进行细致讲解。

正常情况下的移动

使用看板主要意图之一是控制在制品数量（WIP，work in process），需要拉动式的移动，以有效控制在制品数量，防止工作项过多积压。

另外需要提示一点，在整个移动过程的前提是需要制定一套移动规则，根据团队自身情况定义规则，可以根据团队意见进行调整，最后团队满意就是合适的规则。

需求变更情况下的移动

- **不接受变更**

  当一个Sprint的Sprint Backlog和Sprint目标确认后，为了保持团队在很短的时间内，全力以赴的向着Sprint目标冲刺，一般情况下不接受PO提出的需求变更。在很短的周期内，PO是有责任负责整理好Sprint Backlog的，进一步说，PO至少应该整理好接下来1-3个Sprint需要做的Product Backlog，然后按优先级，挑选出最近一个Sprint的Product Backlog形成Sprint Backlog，因此经常性的需求变更建议团队不接受，另一方面也是一个好习惯的养成，促进PO对需求的把控能力。所以这种情况下，团队正常移动看板中的卡片就好。

- **拥抱变更**

  完全拒绝需求变更是不现实的，有的时候高优先级的需求一定要满足变更的要求。比如，有市场时效性的，本Sprint不能完成，不能抢占市场先机，但变更需要遵循“NO CHANGE”原则。接到需求变更后，首先不是直接接受或者拒绝，而是先对需求进行分析，分析对当前迭代的影响。一般分析结果为以下几种情况：

  - 无价值需求

    与PO沟通协商，对于无价值需求果断拒绝，看板中的卡片不做任何移动。至于这些无价值的需求怎么来的，情况比较多，这里不做讲解了。

  - 变更少，影响小的需求

    高优先级的，对Sprint影响小的需求变更，可以柔和接纳，但要评估工作量，做等价交换。简单说，就是把未做的优先级低的需求从看板中替换出来，移动到Product Backlog中，这也是Product Backlog Refinement的过程，然后看板中加入高优先级需求的卡片就好。如果是交换已经产生工作量的需求就需要分情况处理：一种是移回到Product Backlog列，这种情况多于以完成特性需求为目标，更符合敏捷。另外一种是移到Done列，这种情况多见于物理看板中对统计度量数据比较看中的团队，团队需要对工作量进行有效统计。第二种情况在有些电子看板中也可以灵活统计来满足团队需求，那么就可以直接移动到Product Backlog列。

  - 变更多，影响很大的需求

    高优先级的，对Sprint影响很大的需求变更，需要停止当前Sprint，重新规划新Sprint。这里的影响很大情况是指当前Sprint中的需求可能再做下去也没有价值，这时果断停止当前Sprint，另外一种情况也可能是变更的需求本身确实需要很大的工作量才能完成，也需要停止当前迭代。这时根据最新的Sprint Backlog布置看板中的卡片就好。

1.4 迭代开发

1.4.1 如何在软件开发团队中管理突发性任务

背景

开发团队如何管理突发性工作?

企业的一些软件开发团队经常出现类似培训支撑等突发性工作，开发团队不清楚如何管理好这样的工作。

解决突发性工作的问题被很多开发团队所重视，它直接影响开发团队工作的进度和效率，间接影响迭代目标是否能完成，甚至整个项目的成败。所以降低或解决突发性工作对开发团队非常重要。

问题分析

我们先将场景大致分成以下两种：

场景特点	分析
支撑工作与开发工作混合，出现突发性工作是常态。	支撑的工作和正常的开发工作混合在一起，开发人员会经常临时切换工作内容，难免会对管理增加难度，频繁切换工作内容也会造成时间的浪费，因为开发人员需要梳理新工作，新工作完成后还要继续回想之前的工作做到了哪里。因此，问题的根源在于开发团队模式上，在此模式下需要考虑时刻应对问题、风险。
开发工作为主，伴随着出现突发性应急工作，也就是正常的需求变更。	有了新的应急工作，开发团队成员不知道怎么应对，什么时候去接受这个工作，谁来决定是否要做，谁去做，没有一个公开、透明的规则。没有一个可以承载这些新工作的地方。问题的根源在于如何管理好Backlog。再进一步理解，当有新的工作项加进来时，如何更新Backlog，也就是我们常说的有了需求变更（临时增加任务也算需求变更）怎么办?

- **在第一种场景中**，支撑工作与开发工作混合，没有明显的优先级，需要随时接受和完成新的支撑任务。从管理上来说，这些突发的琐碎任务和开发任务同时管理增加了很大的难度。
- **在第二种场景中**，开发团队正常进行开发工作的时候，客户经常会提出一些干扰开发团队冲突任务。而且突发性的工作是由客户重点提出并需要优先处理的，所以开发团队经常会额外付出工作量去处理，造成迭代目标不能达成的风险。

开发团队主要想解决如何处理琐碎、突发性的支撑工作，如何提高工作效率的问题。

解决措施

敏捷方式是趋势，越来越多的开发团队开始接触和使用，CodeArts产品也是基于敏捷思维设计的，以下内容均以敏捷方式叙述，包括但不限于Scrum框架。

- 对于场景一，可以从人员分工角度思考，建议开发团队人员要有工作侧重点的划分。一部分人员精力侧重于开发，另一部分人员侧重于应对突发工作。应对突发工作的人员时刻准备着问题风险的对应（比如开发工作尽量领取简单、松耦合的）。应对开发工作的人员专心完成开发内容。不管工作分工是哪种类型的开发团队，再有新的工作进来时，都需要遵循开发团队制定的规则，也就是管理好工作项的规则。
- 上表中的场景二是我们很多开发团队中经常遇见过的，也是本文着重描述术的情况。从根本解决工作项优先级的问题，系统地学习怎么样应对需求变更才是根本。

对于场景二的解决方案思路如下：

由于管理好工作项是解决问题的核心，因此在形成工作项之前，我们需要解决谁对工作项负责或者说工作项来源的问题，然后要针对工作项做好工作计划，这样开发团队才能很好的执行。

1. 明确产品经理，做到需求来源唯一。
2. 梳理产品待办列表，高优先级的工作项先做。

3. 重新定计划，确保开发团队容量适合，合理更新迭代目标。
4. 回顾总结，选出改善点，下个迭代做得更好。
5. 几个迭代下来，度量分析，不断改善。另外，同步需要进行的是人才的培养，向跨职能型团队努力。

解决方案思路示意图如下：

图 1-17 解决方案思路示意图

1. **明确需求责任人**

   明确需求责任人，做到需求来源唯一。在CodeArts中一般是**产品经理**充当这个角色。需求责任人至少同时要面对两个方向。

   - **方向一**：需求责任人必须很好地理解项目中的利益干系人、客户和用户的需要（包括前面提到的突发工作项）及其优先级。从这个角度理解，一般是产品经理充当需求责任人。
   - **方向二**：需求责任人必须与开发团队交流要构建的特性及其构建顺序。需求责任人还必须保证特性的接收标准已有明确说明，让开发团队可以确定在什么情况下需求责任人可以认为特性完成了。在这个角度理解，一般是业务分析人员和测试人员的角色。

   同时，需求责任人还要在版本、迭代和Backlog层面都能够持续做出良好的经济决策，管理经济效益。为了统一叫法、便于理解，后面需求责任人都由产品经理充当（类似Scrum框架中的产品负责人）。

   产品经理的主要职责总结如下图所示：

图 1-18 产品经理主要职责

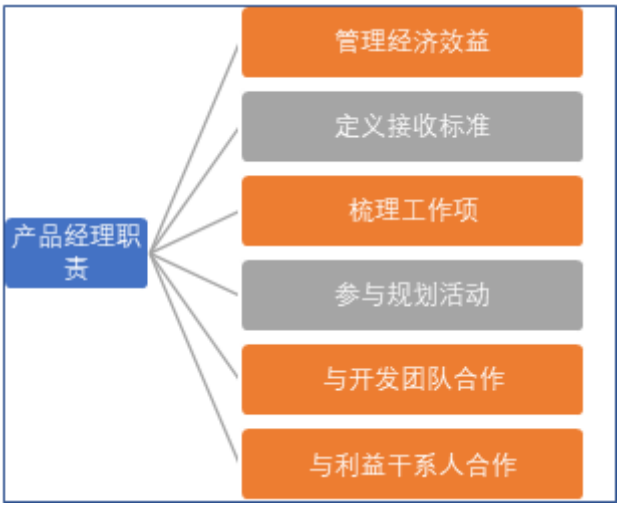


图1-18中标亮部分的产品经理职责就与迭代中对应这些突发性工作相关了。产品经理需要与利益干系人充分沟通，确定这些突发的工作优先级别，同时从业务价值和管理经济效益等维度考虑，明确是否一定要在本迭代中完成。一旦确定这些突发工作属于高优先级，必须在本迭代中完成，那就需要重新梳理工作项了。

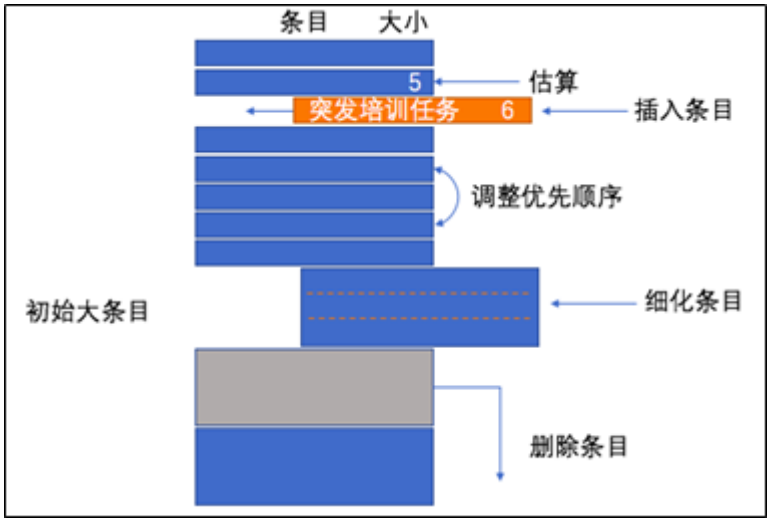
2. **梳理Backlog**

   梳理Backlog，高优先级的Backlog放到迭代待办列表即迭代中（更多关于迭代待办列表的内容请参考**了解更多：迭代待办列表**）中先做。

   - **首先**，我们一起先来了解什么是Backlog。Backlog是一个按优先顺序排列的、预期产品功能的列表。
   - **其次**，再来理解什么是工作项。工作项是Backlog中的待办事项。对用户和客户来说，大多数的工作项都是有实际价值的特性和功能，也包括一些需要缺陷修复、技术改进、知识获取等工作以及产品负责人认为有价值的任何工作（有价值的突发性工作也属于工作项）。
   - **然后**，什么是梳理？梳理是指三大重要活动。第一，确立并细化工作项；第二，对工作项进行估算；第三，为工作项排列优先顺序。

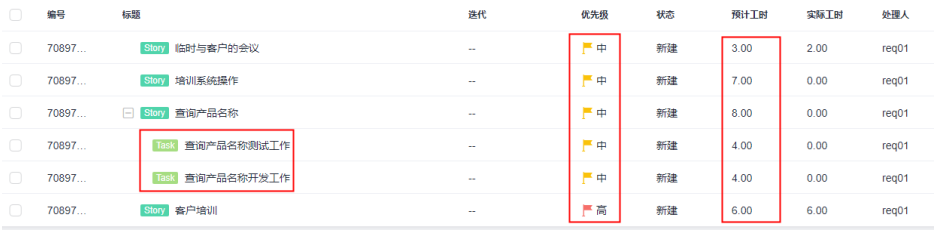
     通过梳理活动改变Backlog的结构：

     图 1-19 Backlog 结构

梳理后，对应CodeArts中的体现，在CodeArts中梳理活动结果如图1-20所示。

图 1-20 需求梳理



编号	标题	迭代	优先级	状态	预计工时	实际工时	处理人
70897...	Story 临时与客户的会议	--	中	新建	3.00	2.00	req01
70897...	Story 培训系统操作	--	中	新建	7.00	0.00	req01
70897...	Story 查询产品名称	--	中	新建	8.00	0.00	req01
70897...	Task 查询产品名称测试工作	--	中	新建	4.00	0.00	req01
70897...	Task 查询产品名称开发工作	--	中	新建	4.00	0.00	req01
70897...	Story 客户培训	--	高	新建	6.00	6.00	req01

经过产品经理梳理后，如图1-20红色框线部分，突发培训任务的工作项得到了合理的优先级，而且是经过工作量估算，同时将原来的查询产品名称任务进行了拆分细化。产品经理是梳理活动的最终决策者，也就是说突发性的工作由产品经理主导梳理。优秀的产品经理能够充分协调利益干系人安排足够的时间，根据开发团队的特点和项目类型来开展梳理活动。同时开发团队还要估算新加入突发工作的工作量，帮助产品经理根据技术依赖关系和资源约束来排列工作项的优先顺序，如果新加入突发性培训任务的工作项优先级高，那么就会纳入本迭代代办列表中。

3. **重新定计划**

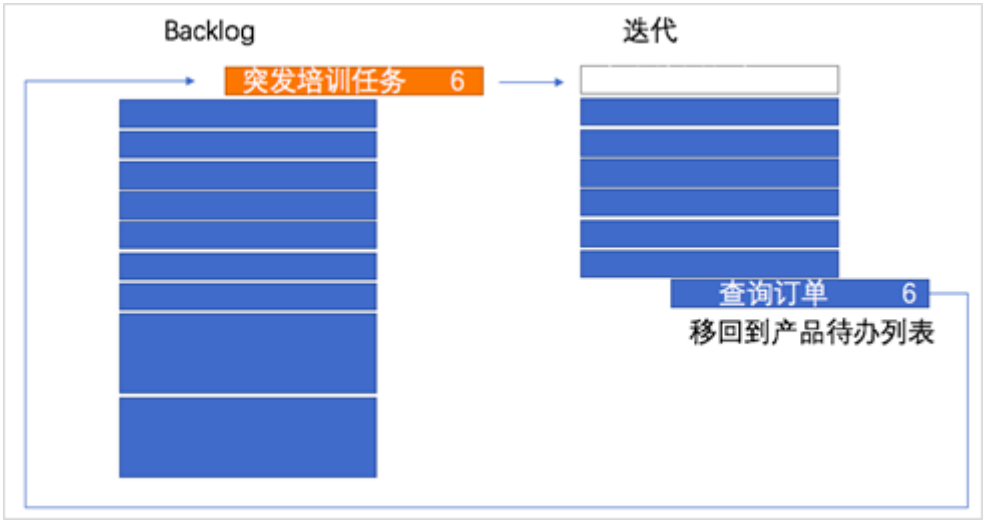
   重新定计划，确保开发团队容量适合，合理更新迭代目标。在重新定计划之前，我们一起来了解下什么是敏捷下的计划，敏捷提倡的计划和传统瀑布开发模式下的计划有什么区别。

   我们知道敏捷宣言中提倡“响应变化高于遵循计划。”，它与传统瀑布开发模式或者说计划驱动的顺序开发模式的计划**不同点就是一个是偏向响应变化，一个是偏向遵循计划**。

   顺序开发过程中，是计划驱动，计划是工作如何开展、何时进行的权威信息源。因此，计划是需要遵循的。相比之下，在敏捷中，根据实时信息关注适应重新制定计划胜于遵循计划。我们认为盲目的相信计划往往会让我们忽视“计划可能有错”这个事实。

   敏捷开发过程中，我们的目标不是满足某个计划或者某个事先认为事情如何进展的预言。相反，我们的目标是快速地重新制定计划并根据开发过程中不断出现的、具有重要经济价值的信息进行调整。因此，通过梳理以后的工作项，可以对应的调整计划。原计划准备做的工作项可能被移入到下一个迭代中实现，这里体现的是“等价交换原则”，意思是用优先级高的突发性工作项，替掉同等工作量的其它工作项，这也是为了确保开发团队按照一个稳定的节奏交付。因为固定的开发团队一个迭代中容量是不变的，新加入突发工作项而不移除其它工作项势必会给开发团队交付带来压力，破坏开发团队的交付节奏。

图 1-21 等价交换示意图



具体在CodeArts中的体现，如图1-22和图1-23所示。

图 1-22 等价交换图一

图 1-23 等价交换图二

最后，从迭代中选取差不多工作量的低优先级工作项移回到Backlog中，即在CodeArts中完成了工作项的等价交换。

4. **迭代回顾，识别改善点**

   迭代回顾是一个会议，目标是持续改进流程，根据开发团队的需要改进和制定流程，以提高士气，提高效率，提高工作产出速率。几个迭代下来，需要对这类突发工作进行度量分析，识别改善点，持续改善。虽然我们提倡响应变化高于遵循计划，但同时执行迭代的时候也需要开发团队在不受干扰的情况下全力以赴的迭代。因此，就应该思考为什么每个迭代中都有外界的干扰（突发工作）存在，开发团队共同学会分析和找到解决办法才是真正的解决问题之道。

   这里给出一个CodeArts产品很好的小实践，可以提供客观数据帮助回顾。新纳入迭代待办列表的突发性工作项可以在CodeArts的“模块”下进行管理。也就是

说，在新建User Story之前建立一个“突发任务”模块，然后将User Story中的“模块”属性选中它。这是为了在后续几个迭代中对这类突发性工作进行度量分析（可以按模块排序，方便度量分析）以便持续改善做准备。如图1-24和图1-25所示。

图 1-24 新建突发模块图

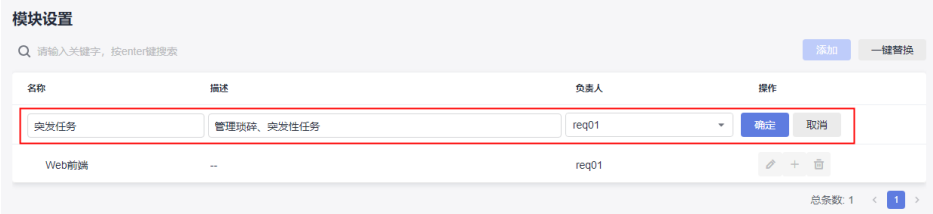


图 1-25 新建 User Story

选择“统计 > 新建报表 > 自定义报表 > 按迭代维度和实际工时汇总项 > 增加筛选条件（模块） > 保存”，对统计出来的数据进行分析。以图和表两种方式展示：

图 1-26 按图展示

图 1-27 按表展示

5. **同步进行人才培养**

   同步需要进行的是人才的培养，向跨职能型开发团队努力。不仅仅是应对处理突发性工作，而是让开发团队更高效。

   每个任务应该由谁来做，或者说突发性工作应该由谁来做？答案很明显，应该是能够最快且正确完成这个工作的人来做。如果这个人正在做其他工作，抽不出时间，但这个任务需要马上完成。开发团队成员都有责任考虑各种不确定因素，做出好的选择。如果开发团队成员都是T型技能的时候，每个任务都有好几个人可以做，那么开发团队就能够在迭代执行期间几个人全力完成制约工作项流程的任务，更灵活地平衡资源，使开发团队更高效。

了解更多：迭代待办列表

迭代待办列表在CodeArts中称“迭代”，它是一组为当前迭代选出的产品待办列表项，同时加上交付产品增量和实现迭代目标的计划。

当新工作出现时，开发团队需要将其加入到迭代待办列表中去。随着工作的执行或完成，剩余的工作量被估算并更新。当计划中的某个部分失去开发意义，就可以将其移除。在迭代期间，只有开发团队可以改变迭代待办列表。迭代待办列表是高度可见的，是对开发团队计划在当前迭代内工作完成情况的实时反映，该列表由开发团队全权负责。

迭代待办列表在迭代计划会议中形成，其中开发团队不会被动分配任务而是由开发团队成员主动认领自己擅长和喜爱的任务。任务被分解为以小时为单位，建议任务不要超过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。每项任务信息包括其负责人、工作量、承诺的完成时间及其在迭代中任意一天的剩余工作量，且仅开发团队有权改变其内容。

1.4.2 如何解决开发团队中的任务没人领取的问题

背景

在传统开发模式下，开发任务是由项目经理指派给个人的，而在敏捷开发模式中，开发任务是团队领取的。

很多企业在转型中遇到过这样的问题：“计划会议认领开发任务的时候，有几个任务没人认领怎么办？”

问题分析

首先，相对于传统开发模式的指派开发任务，我们需要知道为什么在敏捷开发中是领取任务。在敏捷中，不管是**敏捷宣言**还是**Scrum指南**，都没有指派（assign）一词，而是使用了一个术语**自组织**，如下：

- 最佳的架构、需求和设计出自于自组织的团队（敏捷宣言12项原则）。
- 自组织团队自己选择以何种方式来完成工作，而不是由团队之外的人来指导（Scrum指南）。
- 开发团队是自组织的。没有人（即使是Scrum Master）有权告诉开发团队应该如何把产品待办列表变成潜在可发布的功能增量（Scrum指南）。

那么“自组织”是什么呢?

从字面的意思来理解，**自组织**就是：安排分散的人或事物使其具有一定系统性或组成一个整体，而安排的人就是安排者自己。在敏捷开发中，自组织团队就是具备自我管理、自我驱动、自我学习等能力的敏捷开发团队本身，这样的团队一般具备如下特点：

- 团队成员自己“拉”工作，不是被动等待领导分配工作。
- 团队作为一个整体管理工作。
- 团队仍然需要辅导和指导，但不需要指挥和控制。
- 团队成员彼此沟通紧密，互通有无。
- 团队主动发现和提出问题并共同解决。
- 团队不断提高自己的技能，鼓励探索和创新。

更多关于**自组织**的相关内容不在本文的范围内，如感兴趣请参阅**参考文档**。

从**敏捷宣言**和**Scrum指南**关于任务的工作方式上来看，在我们践行敏捷的时候，主要发挥的是开发团队自身的主观能动性，开发团队由原来的控制性转变成了自组织性，而开发任务也就由原来的指派变为了领取。这样的好处是，领取任务就是发挥了人的主动性，而自主性是人们从事创造性和解决问题的动力之一，良好的自我组织能给团队和个人带来高绩效、出色的工作成果以及喜欢的工作环境。另外，每个人都是最了解自己的，也擅长为自己分配任务，相对于传统的指派开发任务所带来的易主观臆断、分配不当等更具有合理性。

然后，回到“计划会议认领任务的时候，有几个任务没人认领怎么办？”这个问题上。

在此之前需要先澄清的一个观点：在计划会议中，不一定非要全部领取完开发任务。在**Scrum指南**中指出“领取工作在Sprint计划会议和Sprint期间按需进行。”可以理解为，在每日Scrum站会上基于目标领取任务。另外，Mike Cohn也表示过，不建议在计划会议中领取开发任务，这样可能会导致目标由团队变为了个人，进而违背了敏捷的本意，降低了灵活性。

一般来说，开发任务没人认领的原因主要有：

- **开发任务的难度大**：当开发任务比较难以解决，超出了团队大部分成员的能力时，团队成员可能会存在担心加班加点而不愿意认领。
- **开发任务超范围**：当开发任务的内容超出团队成员所掌握的范围时，如开发不会测试，就可能会出现“我是想认领的，但能力有限”的情况。
- **担心受到他人指责**：工作内容存在一定的挑战性，担心由于自己没有做好，导致团队目标没有达成而受到指责。

那么应该如何解决呢?

解决方案

在一个敏捷Scrum团队中，Scrum Master扮演着重要的角色，该角色一部分的作用就是要帮助团队成为自组织型团队，以便让团队能以积极的心态去面对冲刺的开发任务。此外，当出现任务没有人愿意认领的情况时，首先Scrum Master应该帮助团队弄清楚没有人认领的原因是什么再对症下药，下面基于分析中的三种情况分别给出解决措施。

- **开发任务难度大**

  对于开发任务难度大的情况，Scrum Master应该组织团队进行有效的任务分解，使用探针Spike技术，探索出解决措施以降低任务的难度，再由团队去认领；或者鼓励技术能力较一般的成员和技术大牛通过结对编程的方式来一同认领任务。

  在华为云CodeArts中，可以对该类难度大的用户故事通过子工作项的方式进行拆分，同时在基本信息中通过设置处理人和抄送人的方式以记录结对编程的人员配对情况：

  图 1-28 Story 内容介绍

除此以外，在每日Scrum站会的时候要留意和了解该开发任务的情况，进行风险评估，如有问题及时帮助协调解决。在回顾会议中，应对该类情况进行分析并能输出基于团队的一套标准工作方式方法，然后将解决方案记录在团队知识库中，华为云CodeArts提供了知识库的功能，可以为团队很好的整理和记录工作方式，如图1-29所示。

图 1-29 任务认领说明

- **开发任务超范围**

  敏捷提倡的团队是跨职能团队，但是团队的跨职能并不意味着一个人能做所有的事情，我们希望的跨职能团队往往是由掌握多项技能的T型人才（每个成员在一个专业领域具有深度，而在其他领域具有广度）所组成的。首先需要Scrum Master能够和团队整理和维护成员技术矩阵，把个人技能掌握情况对团队公开（知道团队欠缺什么、知道可以和谁学等），然后定期组织技术分享等活动以帮助团队成员学习（主要以学习一项新的技术后的分享方式），这样可以在一定程度上提升成员在冲刺中愿意领取其他任务的热情。另外，还可以由专长成员和意愿成员组队，采用结对编程的方式领取任务，以实现个人技术的扩充。团队成员的T型能力建设，不仅仅能让团队领取任务的时候有更多的选择，也提供了成员的backup能力，减少无人认领的情况发生。此外，同样也需要Scrum Master留意日常评估风险和引导团队回顾该事项并维护团队知识库。

- **担心受到他人指责**

  工作内容存在一定的挑战性，担心由于自己没有做好，导致团队目标没有达成而受到指责。Scrum Master应该对团队贯彻以团队为整体的思想，并指导和强调Scrum的价值观，尊重团队的每一个成员的背景、经验，也包括开发任务的选择，还要鼓励成员能有勇气去选择和尝试。在实际的工作中，我们可以通过在墙上、白板等贴上标语（如“尊重他人”、“只有团队没有个人”等）的方式，让团队从思想意识方面发生转变，慢慢敢于去领取有挑战性的任务。此外，Scrum Master要充分保护好成员对有挑战工作认领的热情。如防止在回顾会议上出现指责和批斗的情况，回顾和总结永远应该聚焦的是做事的方式方法而不是对人的苛刻和指责。

总结

以上三种没有人认领任务的情况，是比较常见的。但在真正的实际项目中，每个公司或团队的情况都不尽相同，无法穷举所有，应具体情况具体分析。比如，一个刚刚转型的敏捷团队，在开发任务的领取上可能会更偏向于半指派半领取的方式。这就好比中国经济一样 “以市场经济为导向，适当进行宏观调控”。但不管这个“调控”的力度如何，我们都应该鼓励团队成员能积极主动地领取任务，并随着任务的进展情况灵活调整，及时做好风险把控，必要的时候需要其他渠道的协调帮助或相关领导的介入，以保证迭代的目标不受影响。

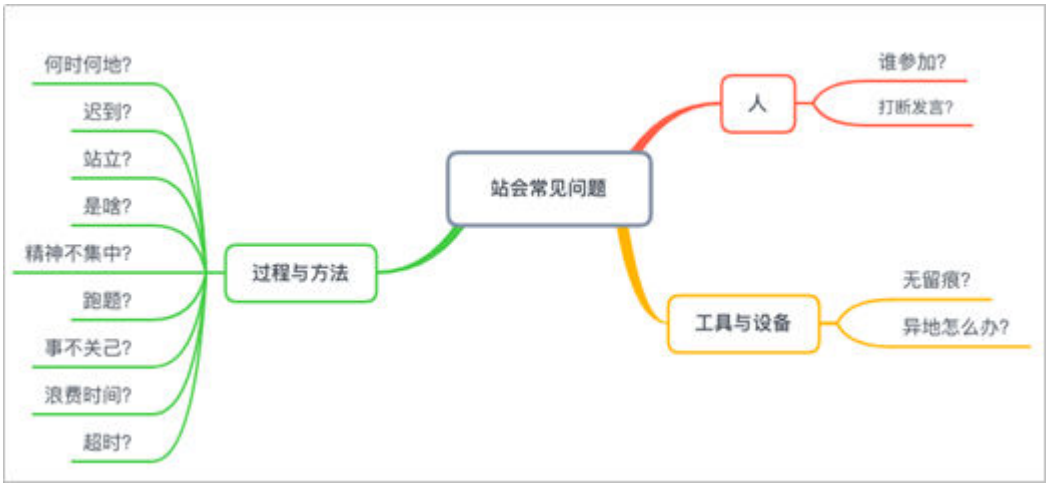
1.5 敏捷回顾

1.5.1 如何玩转每日站会

背景

企业的一些项目团队每天都开站会，但是效果不理想，好像是形式化的内容，并没有起到什么实质的作用。比如，开完站会后，成员继续做着手头的工作，成员依然只关心自己的工作，其他人员的工作完全不了解，好像站会并没有带来什么效果。再比如，开站会本身也有很多问题：会议超时、成员迟到、成员注意力不集中等，具体常见问题如图1-30所示。

图 1-30 站会常见问题



如何正确的开站会？站会的意义在哪里？可以不开站会吗？这些问题一直困惑着不少的团队。

问题分析

关于站会的问题大致分为两种场景：

- **场景一：**团队非常清楚应该开站会，认识到站会确实有一些价值，但是对于目前的站会状况不是很满意，如何玩转站会是团队关心的。对于这类的团队，问题的根源在于不是非常清楚站会的核心价值，以及不知道怎么样实践，团队更需要一些具体的措施来帮助他们更好的开站立会议。
- **场景二：**团队在试着开站立会议，不知道站立会议有什么价值，好像开和没开没有什么区别。针对这种情况，是因为团队没有尝到站会的价值带来的好处，团队没有概念，同样缺乏最佳实践。

综上，不管是第一种还是第二种情况，都需要对站会的价值进一步理解，也就是为什么要开站会，它的意义是什么？然后，都需要明确正确的站会应该怎么样开？最后，都需要一些最佳实践和关键点来帮助团队开好站会。

解决措施

如何玩转站会？我们按照如下思路学习下。

图 1-31 站会学习思路

1. **理解站会价值**

   团队每天站着召开的短时间会议称之为每日站会。每日站会是团队对每天工作检视和调整，提前进行自组织。

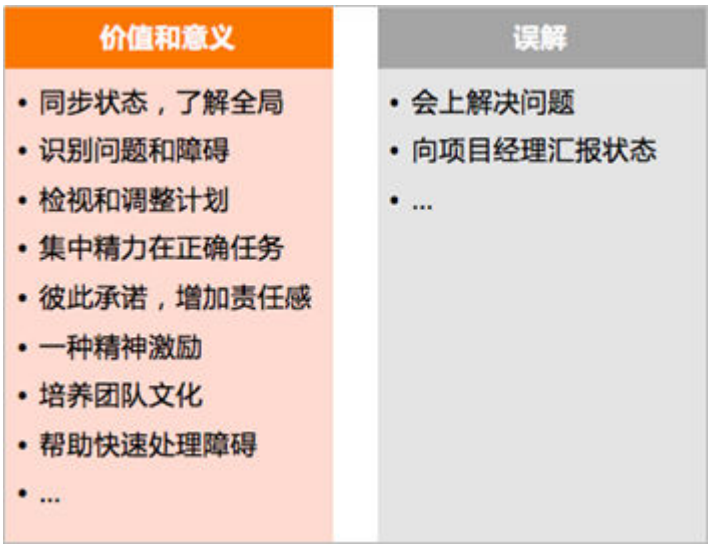
   通过站会团队每个人可以了解全局，知道发生了什么事情，实现冲刺目标的进展如何，对当天的工作是否需要修改计划，有什么问题或者障碍需要处理。每日站会是一个检视、同步、适应性制定每日计划的活动，以帮助自组织团队更好地完成工作。

   有些团队认为每日站会是解决问题的，是传统意义上向项目经理汇报状态的会议，其实都是不准确的，或者说误解了它的核心意义和价值。

   每日站会对于让团队成员每天集中精力在正确的任务上是十分有效的。因为团队成员在同伴面前当众作出承诺，所以一般不会推脱责任。给团队成员一种精神激励，要对每日的工作目标信守承诺。每日站会还可以保证Scrum Master和团队成员可以快速处理障碍，培养团队文化，让每个人意识到我们是“整个团队在一同战斗”，一些没有使用敏捷的组织有时候也同样采用每日站会。

   归纳站会的价值和意义，以及误解：

   图 1-32 站会的价值、意义和误解

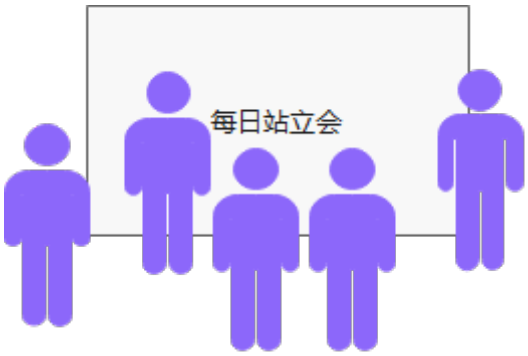
   

2. **明确正确站会**

正确的站会应该怎么开呢？我们一起学习下。

对于每天的工作，为了提前进行自组织，团队成员准时围绕着白板前站立（增加仪式感）。

图 1-33 站会示意图

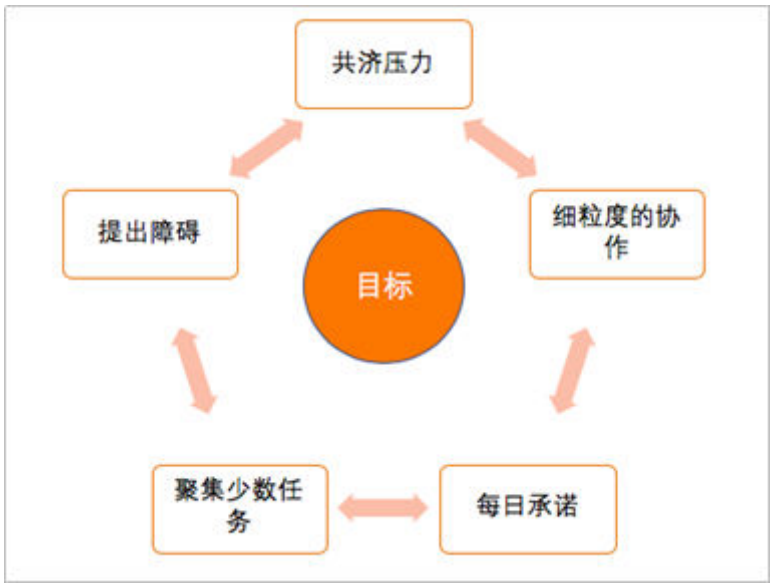


团队成员在站会上需要轮流发言，回答如下三个问题：

a. 我昨天做了什么？（从上次站会到现在，我做了什么？）
b. 今天计划做什么？（在下次站会之前，我会做什么？）
c. 我遇到了哪些问题和障碍？（哪些问题和障碍阻止了我的工作或使我的工作放缓？）

这简单的三个问题可以促使团队成员每天都要检视自己的工作、制定自己的工作计划、获得清除障碍的帮助以及对团队做出承诺。如果团队按正确的方式开站会，进行得好的话，可以达到如下效果：

图 1-34 站会目标



- **共济压力**

  健康的敏捷团队都会有共济压力。所有的团队成员都承诺要一起完成冲刺的工作。这就使得团队成员之间相互依赖并且对彼此负责。如果一个团队成员连续几天都做相同的事情，并且没有进展，显然缺乏前进的动力，而其它团队成员不能视而不见。因为他未完成的工作会变成其他成员的障碍。

- **细粒度的协作**

  在站立会中，团队成员的交流应该快速而且有重点。举例，当一个成员说完今天计划做什么后，另外一个成员可以会说：“哦，原来你今天计划做这个啊，这就意味着我要调整我的工作优先级，没关系，你按照你的计划做吧，我可以调整。很高兴你说了这些。”这种细粒度的协作使得团队成员知道他们之间如何及何时仰仗对方。一个敏捷团队应该追求高效、零等待，避免等待浪费。

- **聚集少数任务**

  在站会期间，团队中的每位成员都可以知道哪些工作正在进行，哪些工作已经完成。健康的团队应该关注事情的完成，也就是说任务不能一直处在进行中。在站会中，团队需要确认哪几个少数任务是当前的焦点，这样团队就可以尽快把焦点任务做完。换句话说，做完10件事，远比正在做100件事儿更有意义。

- **每日承诺**

  在站会上，团队成员需要对团队做出承诺。这样团队成员就知道敏捷交付什么成果并如何保证彼此负责。

- **提出障碍**

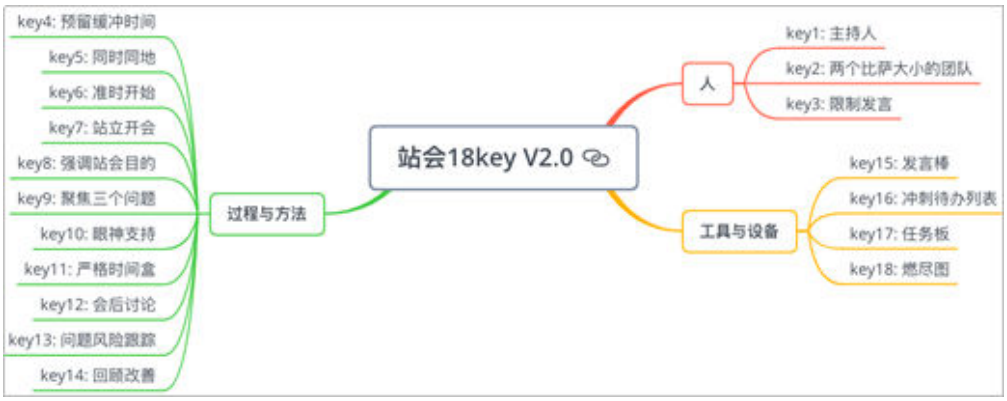
  其实在敏捷中任何时间都可以提出障碍，但是站会是一个黄金时刻，团队成员可以停下来认真思考“有什么事情阻碍了我或让我的工作放缓了”。

3. **最佳实践和关键点**

   通过大量实践总结出一些能够帮助开好站会的关键点，也许这些关键点并不是全适用，所以还是要根据现实情况做出合适自己团队的选择和裁剪。这些关键点，我们称之为“站会18 key”。

   站会18key按照人（People）、过程与方法（Procedures and methods）、工具与设备（Tools and equipment）划分，帮助大家记忆和学习。

   图 1-35 站会 18key

   - **Key 1: 主持人**

     会议主持人（比如Scrum Master，也可以团队成员轮班，轮流感受下站会的节奏）确保会议的举行，并控制会议时间，团队成员进行简短有效的沟通。

   - **Key 2: 两个比萨大小的团队**

     在《Scrum敏捷软件开发》一书中，作者麦克·科思提出了一个简单的方法用来辨别什么是合适的团队规模，那就是，如果两个比萨够整个团队成员吃的话，那么这个团队的规模比较适合。

     因为两个比萨大小的团队跟家庭的规模相似，站立会的目标可以轻松达成。当团队是家庭规模大小时，人们头脑中就很容易追踪到团队中发生的事情。

人们可以很容易地记住每个人每天的承诺，以及每个人对于其他成员或团队成果的责任。Scrum中也建议团队规模不要太大，一般为7-9人左右。

最后再强调一点，并不是要求团队一定要按以上18key进行开站会，18key只是在很多实践中总结出来的一些经验，曾经解决过很多问题。对于每个具体团队需要结合实际情况进行选择应用18key。

- **Key 3: 限制发言**

  团队外成员也可以参与，但没有发言权。在每日站会中，只有为了实现冲刺目标面全力投入的人应该发言，对于参与者，应该作为旁观者。

- **Key 4: 预留缓冲时间**

  建议开发团队在上班时间后的半小时或者1小时后开每日站会。这样可以给例行工作（如查看邮件）提供一些缓冲时间。晚点开会还可以给开发团队一点时间来检查前一天的工作（比如，前一天晚上开始运行的自动化测试工具所生成的缺陷报告）。

- **Key 5: 同时同地**

  每日站立会议应尽可能在同一时间、同一地点召开，建议在团队的可视化的任务板前面召开。同一时间和地点也可以有效帮助团队成员形成固有的节奏，不用浪费时间再找地点和确认当天的开会时间。

- **Key 6: 准时开始**

  所有的团队成员需自觉按时到场，会议主持人要按照预定的时间按时开始会议，而不管是否有人还没到。对于迟到的人员要有一些惩罚措施，比如缴纳罚金或做俯卧撑等。惩罚措施和数量由团队成员事先共同商定，如果是罚金，如何支配也由团队共同决定。如果团队成员就是不自觉按时到场怎么办？关于更多这方面的解决方案请参考下面的了解更多中的“**了解更多：成员迟到的解决方案**”。

- **Key 7: 站立开会**

  团队成员一定要站着开会，这也是会议的名字叫站立会议的原因。站着开会确实比坐着开会简明扼要，让人更想快一点结束会议，开始一天的工作。坐下容易使人放松，精神不集中，不易控制时间。

- **Key 8: 强调站会目的**

  经常强调站会目的，特别适合刚刚启用站会的团队。可以由Scrum Master来强调，如果没有Scrum Master也可以由其它Leader（轮职的主持人也可以）来强调。然后询问团队成员对站会的意见及得到的成果，几次以后，团队成员可能选择目标声明作为每天的度量，在每次站会之后，团队成员对自己的表现做出相应的评价，是一种强有力的自我管理工具。

- **Key 9: 聚焦三个问题**

  站会期间，团队成员就说那典型的三个问题（昨天…今天…障碍是…），其它事情不说。只讨论已完工和即将开始的工作，或者在这些工作中碰到的问题和障碍。目的不是向领导汇报工作，而是团队成员之间相互交流，以便共同了解项目情况和共同解决问题。

- **Key 10: 眼神支持**

  这是一个好玩的游戏：当一个人站在前面发言时，要求其他团队成员都直视发言人，并进行眼神交流。别让发言人抓到你在看别处。这个游戏帮助发言人的发言简洁，同时可以加强成员对发言人所讲内容的理解。这样可以帮助团队加速完善每日计划。

- **Key 11：严格时间盒**

  站会是开发团队的一个时间盒限定为 15 分钟的事件。 时间建议不要太久，对于5-9人的团队来讲15分钟的会议时间足够。

- **Key 12: 会后讨论**

  某位团队成员在发言期间，其他人员应认真倾听，如有疑问可简短确认，但不应做过多讨论。如果对某位成员的报告内容感兴趣或需要其他成员的帮助，任何人都可以在每日站立会议结束后即刻召集相关感兴趣的人员进行进一步的讨论。

- **Key 13: 问题风险跟踪**

  将站会成员遇到的问题和风险做概要的记录（不必详细，只要说明重点即可，不需要在记录上花费更多的时间），然后保留到知识库，或者方便大家跟踪的地方。此目的是确保这些问题和风险得到了闭环（例如，问题和风险可以会后按排专题讨论、跟踪，直到关闭）。

- **Key 14: 回顾改善**

  本身每日站会就是最小化的戴明环（PDCA），另外团队在回顾会议上时也可以对站会开的效果进行回顾，哪些地方做得好，哪些地方做得不好，有哪些改善点可以在下一轮迭代中改善等（站会只是回顾会议中一个回顾点，如果没有问题不用做专题回顾）。

- **Key 15: 发言棒**

  站会时可以利用一些小道具来保证会议不会超时。我会找一只笔或者一个娃娃（女生多的团队）作为发言棒仍给一位成员，让他拿着发言棒陈述完“三个问题”，然后将其交给下一位。我会找一只笔或者一个小物件作为发言棒仍给一位成员，让他拿着发言棒陈述完“三个问题”，然后将其交给下一位。没有拿发言棒的成员不允许发言。如果有人用时过长，就及时打断，并要求将发言棒传给下一个人。

- **Key 16: 冲刺待办列表**

  站会中，成员在发言时可以利用冲刺待办列表来检视当前工作项的完成状态。冲刺待办列表记录了团队成员工作的进展，需要每天更新并跟踪。电子化的冲刺待办列表更能很好的解决异地团队开站会思路不聚集的问题。发言人在讲那“三个问题”时，同步可以展示冲刺待办列表给团队。

  CodeArts冲刺代办列表演示如下：

  图 1-36 CodeArts 冲刺代办列表

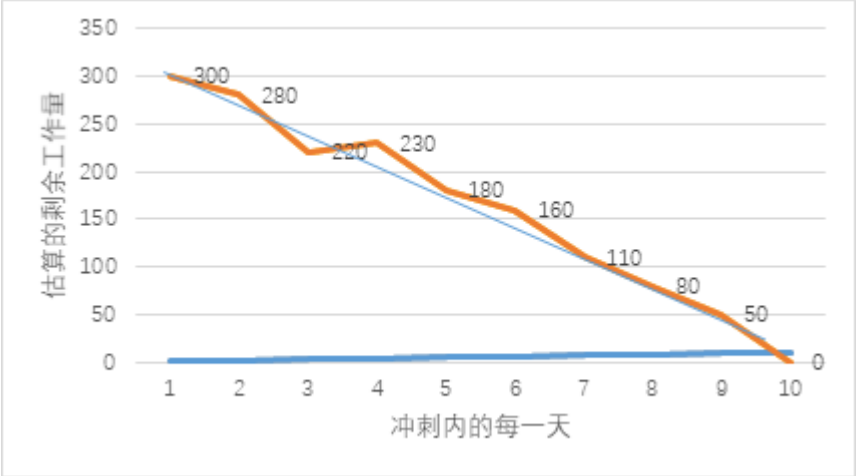
- **Key17: 任务看板**

  在站会期间，通过任务板，团队中每一个人都可以知道哪些工作正在进行，哪些工作已经完成。团队关注事情的完成，一直处于进行中的任务为被发现，成为当前的焦点，这样团队就可以尽快把这些焦点问题解决掉。

- **Key 18: 燃尽图**

  燃尽图是将进展和剩余工作情况可视化的有力工具。一般竖轴表示剩余工作量（小时、故事点或工作项个数），横轴表示冲刺时间（一般单位为天）。

图 1-37 燃尽图



开站会时，发言人可以利用燃尽图来做进展讲解。燃尽图让所有团队成员一眼就可以看出冲刺的状态，进展情况非常清楚，看出工作是否在按计划进行，状态是否良好。这些信息可以帮助团队确定是否可以完成预定数量的工作项，并在冲刺早期做出明智的决定。使用燃图易达成如下效果：

i. 高可视性，直观展示进度情况和剩余工作。
ii. 快速识别风险。
iii. 帮助团队建立信息，了解自己的能力。
iv. 了解团队成员工作步调。
v. 了解团队冲刺计划。
vi. 和任务墙能非常高效地匹配使用。

关于**18key**，并不是站立会议时要把所有**18key**都要执行一遍，这里的**18key**只是提供了一些参考实践和关键点，**18key**来源于大量的实践，也解决过团队站会的问题，所以大家在站会遇到了问题时，可以先想到这个**18key**，然后选择适合自己团队的key。举一个例子，这里有四个key是关于工具的，这些工具我们都要使用吗？不一定都要使用。敏捷宣言里提到“个体和互动高于流程和工具”，工具是为团队服务的，不是团队的负担，更不能被工具所绑架。所以团队一起选择适合的，才是正确的做法。

了解更多：成员迟到的解决方案

对于经常有人迟到的现象，团队成员在回顾会议上可以认真分析原因，重新征求团队成员意见，为什么每日站会的开始时间一定是早晨九点，其它时间是否可以，是否有什么困难，团队成员共同找出问题原因并做出决定。

促进团队成员需要自觉按时到场的意识，尊重别人的意识。会议主持人要按照预先定的时间、地点开始会议，而不管是否还有人没到场。有人迟到不要重新讲解他们错过的开会内容，否则会传达可以迟到的信号。

对于迟到的人员要有一些惩罚措施，比如红包、做俯卧撑、全体下午茶等。惩罚措施和数量由团队成员事先共同商定。如果是红包，如何支配由团队共同决定。相比别人给你的规则，大家更愿意执行自己提出的规则，守自己的承诺。

如果说惩罚措施毫无约束力，那就把惩罚做到可视化。比如在白板中规划出一个特定区域，每迟到一次就把照片贴上去，次数累加。这个特定迟到的区域是迟到信息的扩大器，让更多的人看到。

对于经常迟到的人需要谈话，试着理解他有哪些问题，是否有真正的困难，关心团队成员，大家一起帮助解决困难。

如果迟到现象严重，可能不是团队能解决的问题了，可以试着从公司政策方面施压，严格执行公司的考勤制度，但其实不符合敏捷的自管理思想，不是真正解决问题的方法。

总结下解决迟到现象应该关注以下因素：

- 分析原因，关心成员，共同决定。
- 同一时间，同一地点，准时开会。
- 有人迟到，不重复同步信息。
- 建议小惩罚机制。
- 理解迟到原因，是否有困难。

1.6 成员管理

1.6.1 如何在项目团队人员变动频繁时对新人进行有效培养和管理

背景

企业随着业务的扩张，需要新员工不断加入，经常会遇到这样的问题，其开发组长要对每一位新人交代相关的知识点、工作方式以及团队信息等，工作量在短期内激增。在一个项目中，随着时间推移、业务的扩张，项目中的核心成员，如项目经理、开发组长等往往都会面临如下几种情况和挑战：

- 新员工加入，需要诸多方面的培养（培训），以便能快速进入工作状态。
- 老员工的离职，导致项目中缺少能了解和掌握关键技术和业务的人员。
- 员工在工作了一段时间后，对自己的规划有了新的想法，从而想要转换工作方向。

那么，项目负责人应该如何应对这些事件呢?

问题分析

一个项目在从小到大的过程中，项目团队也势必扩张，面临新员工的加入。新员工对刚接触的项目不够熟悉，所以针对新员工的培养（培训）是非常有必要投入人员配置的。

项目发展过渡到平稳期的时候，可能会因为公司体制、薪资待遇、员工身心疲惫等原因，出现老员工的离职等情况，如果离职的老员工是核心骨干，就可能导致某些如业务无人知晓、关键技术断层等风险，在此期间，也包括老员工随着项目的延续，慢慢产生了对原来工作厌倦，或者因认知的提升以想要寻求一些新的工作内容，进而做了转型的打算。所以上述问题，如果没有得到较好的解决，将会影响到项目的进度和造成不必要的开销，甚至对于团队内部成长、自组织能力的发展建设也是不利的。

所以问题的关键，仍旧是新人的培养。

解决措施

一般来说，在针对新员工加入所带来的内耗、关键核心人员的离职风险、个人发展转型等情况的应对，可以从**团队信息**、**工作方式**以及**知识管理**三方面来通过建立信息管理库进行解决。

CodeArts是集华为研发实践、前沿研发理念、先进研发工具为一体的研发云平台（更多了解请见**参考文档**）。

1.6.2 如何有效管理项目成员的权限

背景

本文主要是解决客服管理成员权限分配、成员添加的问题。

问题分析

- 如何使用IAM来控制权限?

  通常一个IAM主账号下，可以创建多个软件开发项目。默认情况下，只有IAM主账号默认可以设置是否能允许子账号创建项目，只有IAM主账号能查看所有项目和成员等。在某些企业场景下，IAM主账号可以通过IAM的细粒度权限管理，设置部分子账号可以代替IAM主账号的设置权限。

- 项目成员的角色有哪些?

  通过项目管理创建的所有项目都支持基于本项目的权限设置，且每一个项目的权限设置相互独立。

  在项目管理中，默认项目角色包含以下几大类：

  - 项目管理员：项目的创建者。
  - 项目经理：项目开发管理员。
  - 测试经理：项目测试管理员。
  - 产品经理：项目的需求分析管理者。
  - 系统工程师：项目的架构分析管理者。
  - Committer：参与项目开发的人员。
  - 开发人员：参与项目开发的人员。
  - 测试人员：参与项目测试的人员。
  - 参与者：参与项目指定工作处理的人员。
  - 浏览者：关注或浏览项目内容的成员。
  - 运维经理：参与项目维护工作的成员。

- 如何将成员添加到项目并分配角色?

解决措施

需求管理服务使用统一身份认证服务（Identity and Access Management，简称IAM）管理整个租户下多项目的统一权限。在单个项目内，基于具体项目设置进行项目内的权限管理。需求管理的权限管理分为两种，分别是云服务级和项目级。

- 云服务级：服务级的权限通过统一身份认证服务设置。IAM是华为云提供权限管理的基础服务，无需付费即可使用，您只需要为您账号中的资源进行付费。
  - 关于IAM的详细介绍，请参见**IAM产品介绍**。

  - 云服务级权限详情请参见**需求管理云服务级权限介绍**。
- 项目级：项目级的权限通过需求管理服务设置。项目级权限详情请参见**项目级权限**。

更多关于成员的相关内容请参见**需求管理用户指南-成员**。

1.7 附录

1.7.1 参考文档

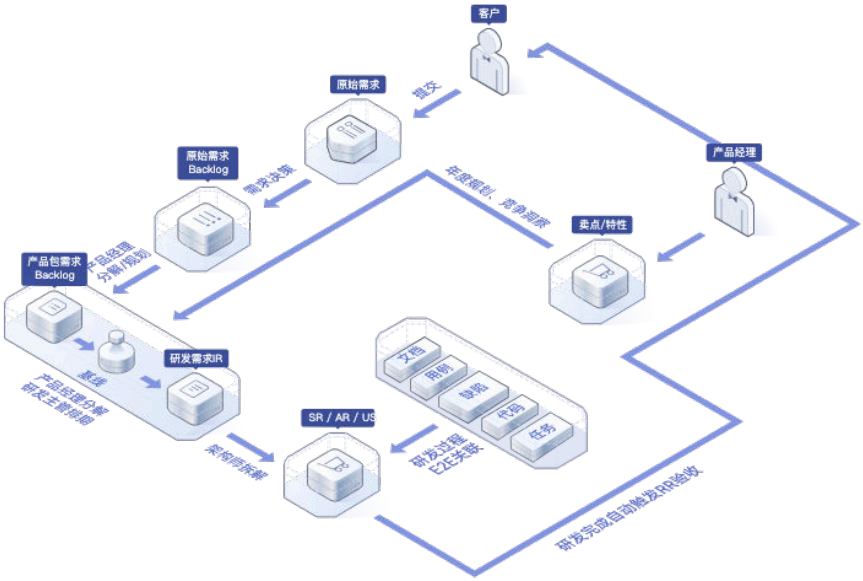
- 《Scrum精髓》，Kenneth S. Rubin
- 《用户故事与敏捷方法》，Mike Cohn
- 2019年中国DevOps行业现状报告：中国信息通信研究院、华为云DevCloud、南京大学联合发布
- 《用户故事实战》，Mike Cohn
- 《成为技术领导者》，杰拉尔德·温伯格
- 《复盘+：把经验转化为能力》，邱昭良
- Scrum指南（2017-Scrum-Guide-Chinese-Simplified），2017年11月版
- Kenneth S. Rubin. Scrum精髓[M].北京：清华大学出版社
- Scrum指南2007版
- Mark C. Layton. 敏捷项目管理[M].北京：人民邮电出版社
- **Should Team Member Sign Up for Tasks During Sprint Planning?**
- Lyssa Adkins. 如何构建敏捷项目管理团队[M].北京：电子工业出版社

2 IPD 系统设备类项目最佳实践

2.1 IPD 系统设备类项目实践概述

IPD系统设备类预置了华为集成产品管理开发的最佳实践，针对系统设备类提供结构化研发流程，具备客户原始需求管理、产品特性树管理、研发需求分解分配、基线、变更、跨项目协作等关键特性，整体流程如图2-1所示。

图 2-1 IPD 系统设备类项目管理流程



2.2 如何理解 IPD 系统设备类的需求模型

首先我们需要先明确，RR、SF、IR、SR、AR这几个需求模型中的工作对象的含义。

表 2-1 IPD 系统设备类项目工作对象的含义

功能	说明
原始需求（RR）	来自公司内、外部客户的，以客户视角描述的原始问题或者原始诉求。客户需求属于原始需求的一种类别。此类需求需要RMT/RAT分析评审后作出决定。
系统特性（SF）	是版本支撑“客户问题（PB）”所具备的重大能力。原则上系统特性是产品包的主要卖点（销售亮点）集合，每条特性都是满足客户特定商业价值诉求的端到端解决方案。一部分特性是可以通过License控制单独销售。
IR（初始需求）	站在内外部客户/市场角度，以准确的语言（完整的背景、标准的格式）重新描述的需求。IR来自于两部分： 1. 可以从RR/SF分解。 2. 直接由产品规划产生。
SR（系统需求）	站在研发视角，描述系统对外呈现的、可测试的全部功能需求和非功能需求。其中功能需求是对系统提供的功能的场景化的具体要求，非功能需求是对系统的成本、全局质量属性（主要是DFX）、技术限制等非功能性方面的具体要求。
AR（分配需求）	根据基层组织分工不同，以可交付的视角，由SR进一步分解分配到子系统/模块的功能或非功能需求。

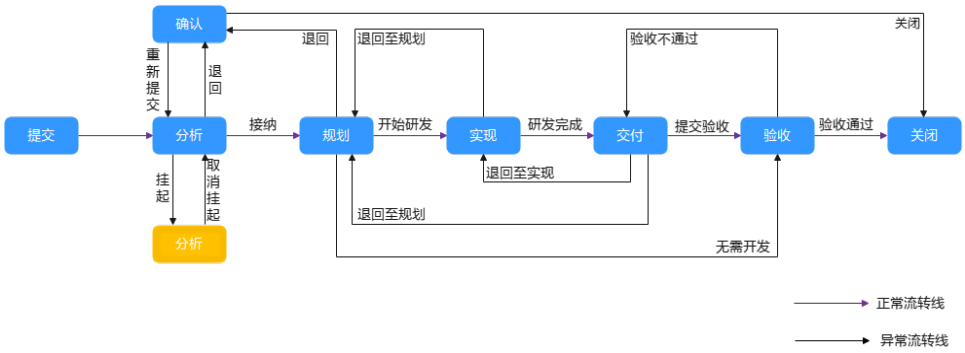
2.3 原始需求管理实践

2.3.1 原始需求管理介绍

成功产品的核心特征是满足客户需求，客户需求是企业发展的原动力。CodeArts Req打破了传统需求管理工具仅在研发阶段发挥作用的限制，将客户与市场需求也同步覆盖，提供了完整的客户需求采集、价值需求决策、交付与验收流程，让需求进展和动态客户实时透明，市场需求流动提速70%。

RR客户原始需求来自公司内部和外部的客户诉求，以客户视角描述的原始问题或者原始诉求，客户需求属于原始需求的一种类别。此类需求需要对应承接人分析后作出决定。RR客户原始需求的主要流程分为：提交、分析、规划、实现、交付、验收、关闭。

图 2-2 原始需求状态转换流程图



2.3.2 名词解释

- **归属内部**

  由内部与外部提出的所有原始需求，对属于本项目需求的所有汇总收集。
- **提给外部**

  需要由跨项目或跨组织共同协作所提出的外部原始需求。
- **分解子需求/关联子需求**

  该条原始需求分析评估并接纳过后，需要通过层层分解，由分解子需求或关联子需求，细分其他颗粒度的研发工作。
- **承接人**

  是指该原始需求得到关键的共同决策，作为该条需求的主要承接人。

2.3.3 模拟案例

2.3.3.1 案例概述

某公司计划推出一款智能手表，研发周期较长，研发过程涉及多个部门、多团队的协作，如何保证原始需求在多个组织间的流转，最终达到有效闭环呢？下面请跟随我们，一起遍历整个原始需求生命周期管理实践。

2.3.3.2 创建原始需求

根据来自公司内、外部客户的原始诉求，按照“归属内部”和“提给外部”进行分类并将其录入创建原始需求。

前提条件

- 管理员已注册账号并开通CodeArts。详细操作指导请参见**注册华为账号并开通华为云**。
- 管理员已给项目成员完成IAM用户的创建。详细操作指导请参见**创建IAM用户**。
- 已创建**IPD系统设备类项目**。

操作步骤

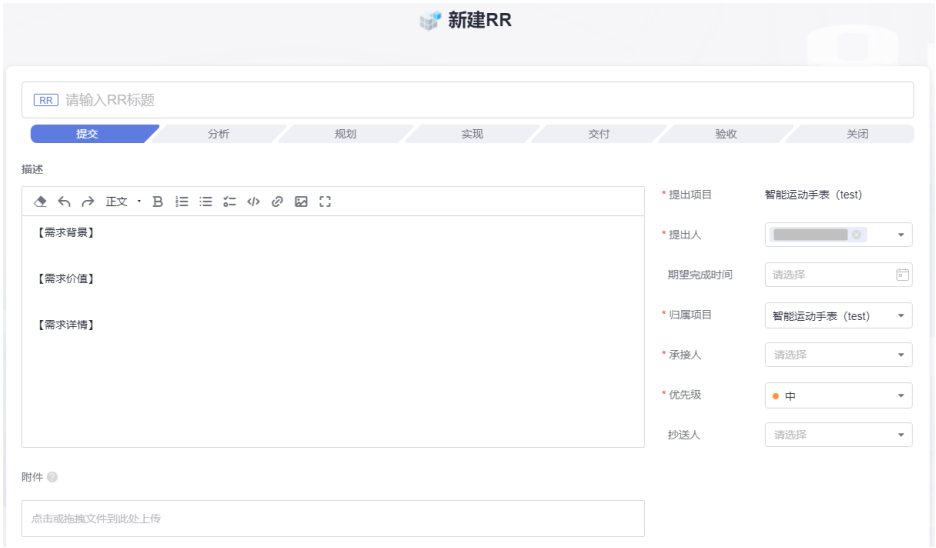
1. 在原始需求列表中，单击“新建RR”，进入新建页面。

图 2-3 原始需求列表

2. 填写原始需求信息。

   新建原始需求时，需填写标题、描述、提出人、归属项目、承接人等必填字段，其中“描述”字段尤为重要。创建人需要详细阐述需求背景、需求价值和需求详情，描述越详尽，后续责任人分析就越准确。

图 2-4 新建 RR



3. 填写完毕后，单击“提交”，该需求将流转到分析环节。

2.3.3.3 处理原始需求

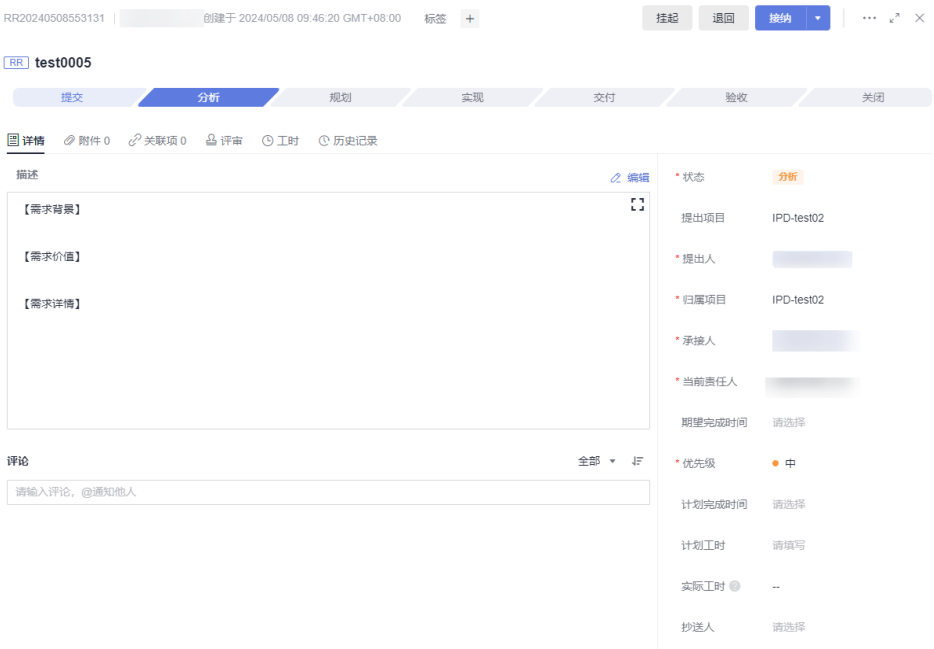
原始需求创建成功后，承接人需要处理该需求，处理过程包括分析、规划、实现、交付、验收、关闭。

- **分析原始需求**
- **规划原始需求**
- **实现原始需求**
- **交付原始需求**
- **验收原始需求**
- 关闭原始需求：需求验收通过后即可变为“关闭”状态。

分析原始需求

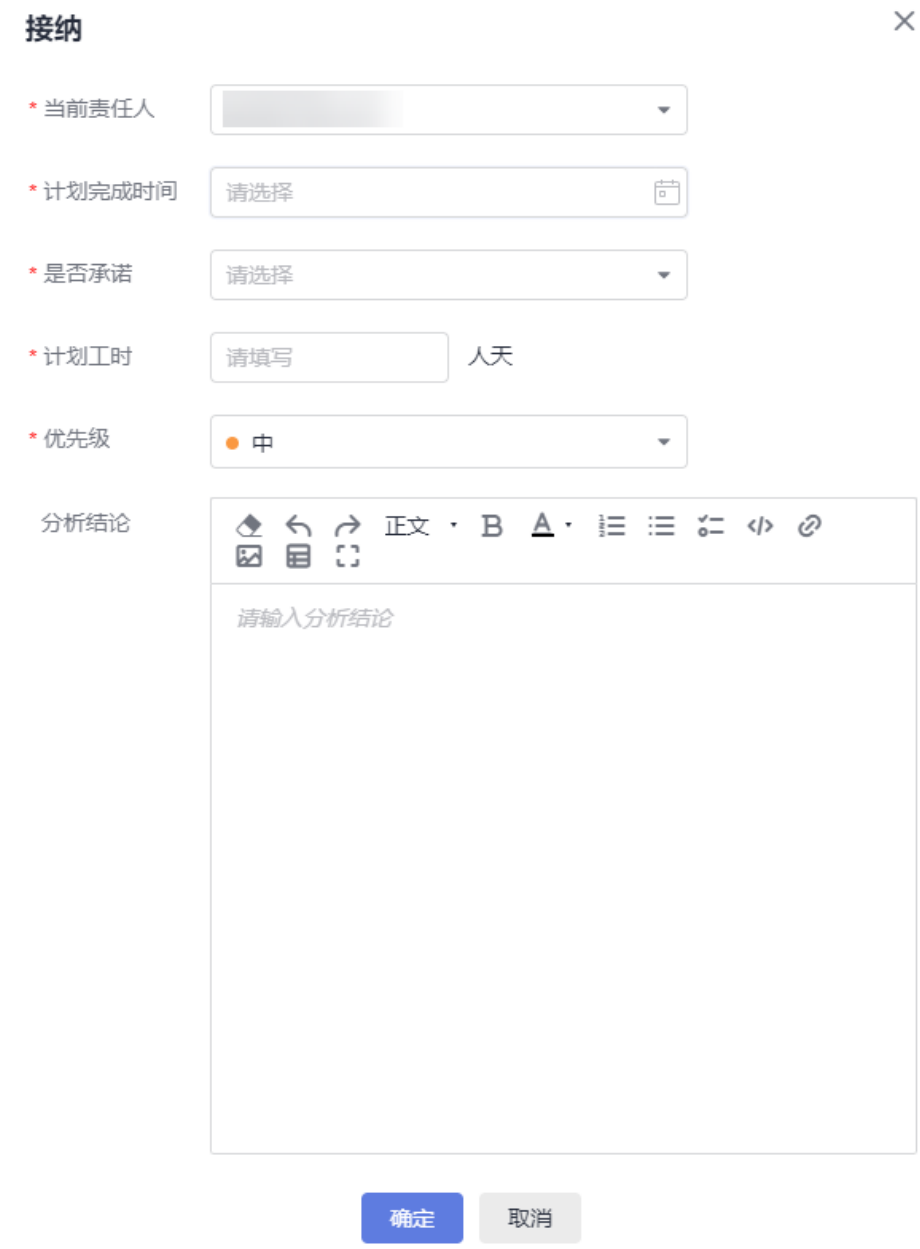
责任人在原始需求“分析”阶段定位诉求价值，分析决策后可选择“接纳”、“退回”或者“挂起”需求。

图 2-5 分析原始需求



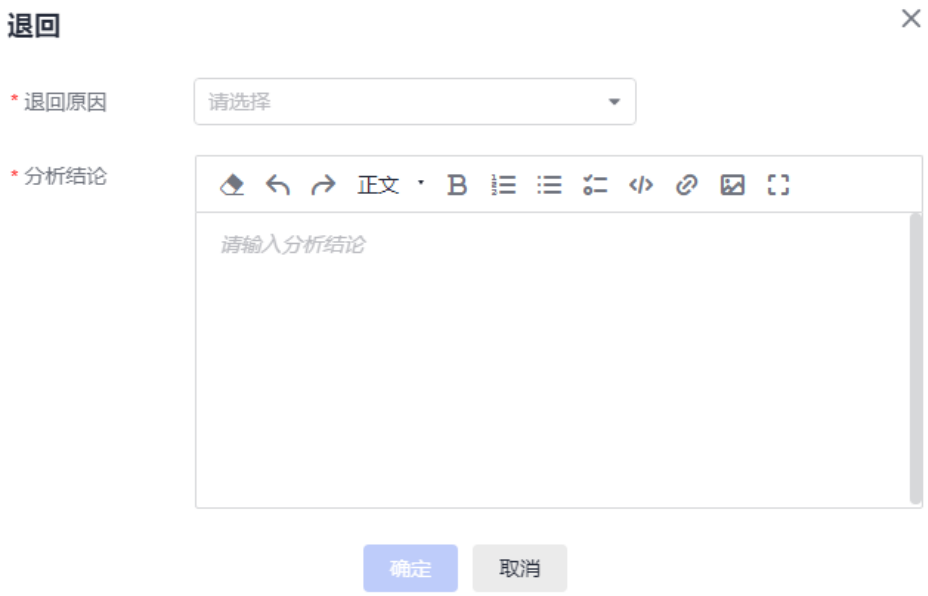
- 如果责任人分析后选择“接纳”需求，则必填对应的字段，明确需求后续的工作计划。

图 2-6 接纳需求



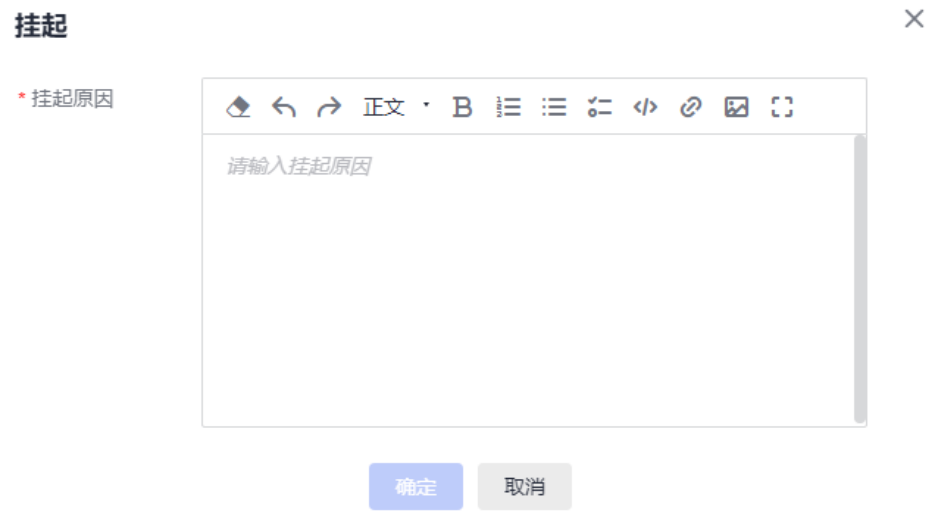
- 如果责任人分析后选择“退回”需求，则必填对应的字段，分析不合理的因素。

图 2-7 退回需求



- 当分析决策认为该需求处于非紧急状态，可选择“挂起”需求，并填写挂起原因，使需求进入挂起状态，暂时停止作业。

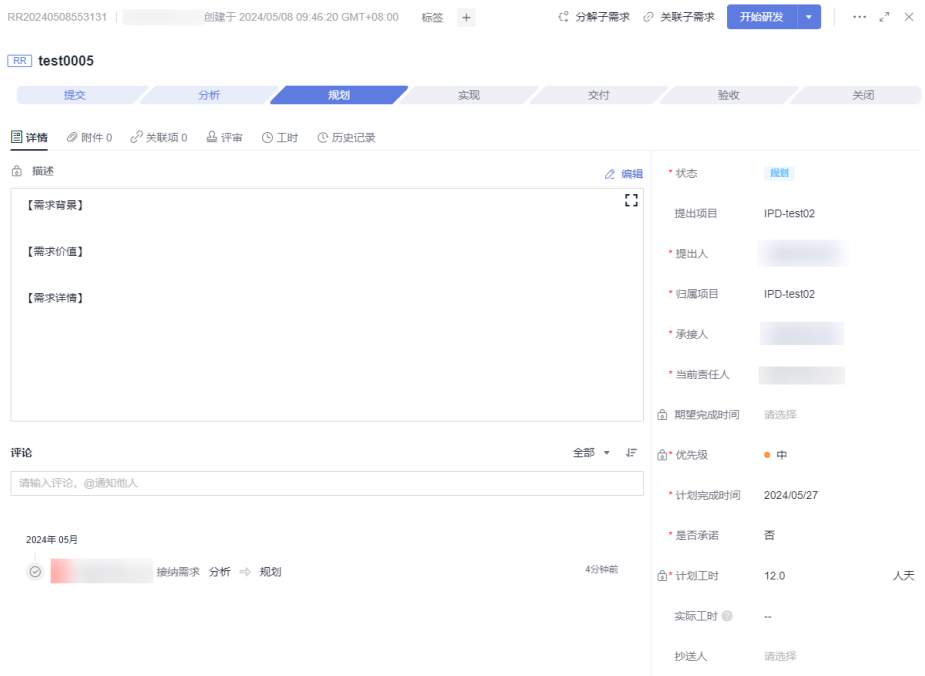
图 2-8 挂起需求



规划原始需求

RR分析接纳后，则进入需求开发规划阶段，根据工作评估是否需要“分解子需求”、“关联子需求”或直接“开始研发”进行需求拆解的协作环节。

图 2-9 规划原始需求



如分解子需求成功后，此RR自动流转到“实现”状态。

图 2-10 分解子需求

实现原始需求

根据研发工作过程，随时再次“分解子需求”和“关联子需求”。

图 2-11 实现原始需求

拆解的所有子IR需求或所关联的子工作项，都处于“完成”状态后，RR自动流转到“交付”状态。

交付原始需求

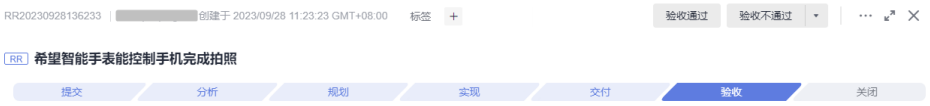
责任人通过各维度评估交付质量，决策是否直接“提交验收”或退回到对应环节重新作业修改。

图 2-12 交付原始需求

验收原始需求

当“交付”流转到“验收”环节时，由提出方责任人在“提给外部”里进行评估该条需求的实现验收。实现完全满足诉求，则验收通过需求单，需求单进入“关闭”。

图 2-13 验收原始需求



当然，由RR所拆解的子需求内容，可通过列表页一目了然，全局跟踪。

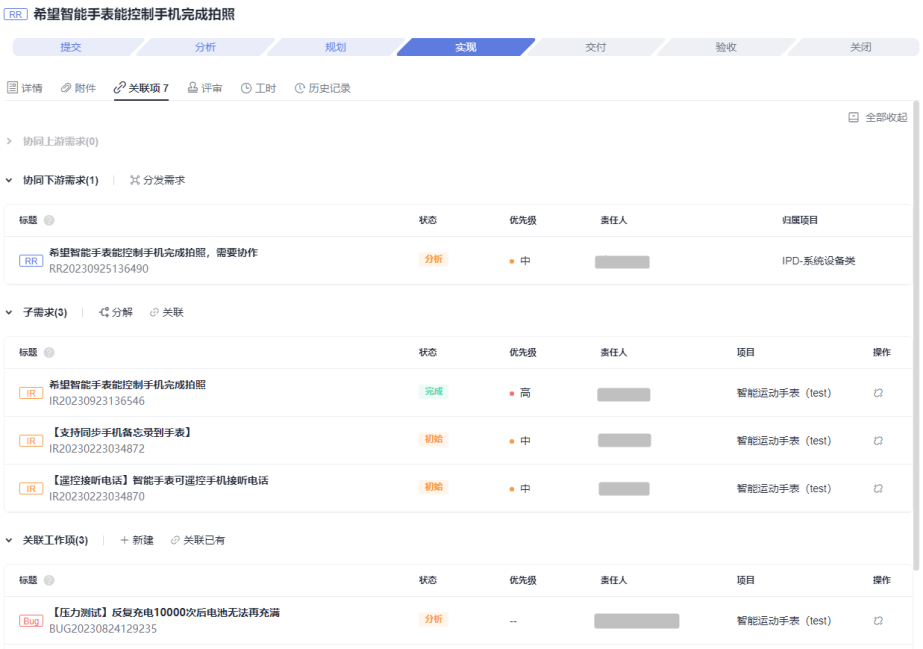
图 2-14 查看原始需求列表

2.3.3.4 编辑原始需求详情

2.3.3.4.1 关联项

原始需求一旦提交进入作业流程，将会与各种工作项产生关联关系。需求从“收集-分析-决策-实现-验收”一条完成的作业流程，每个环节的过程中涉及跨组织、跨团队的多方人员协作，关联项则是以达到如实记录和追溯的目的。这时我们可以在“关联项”页签中对这些工作项进行关联、取消关联、新建子需求、分解子需求以及分发需求的相关操作。

图 2-15 设置关联项

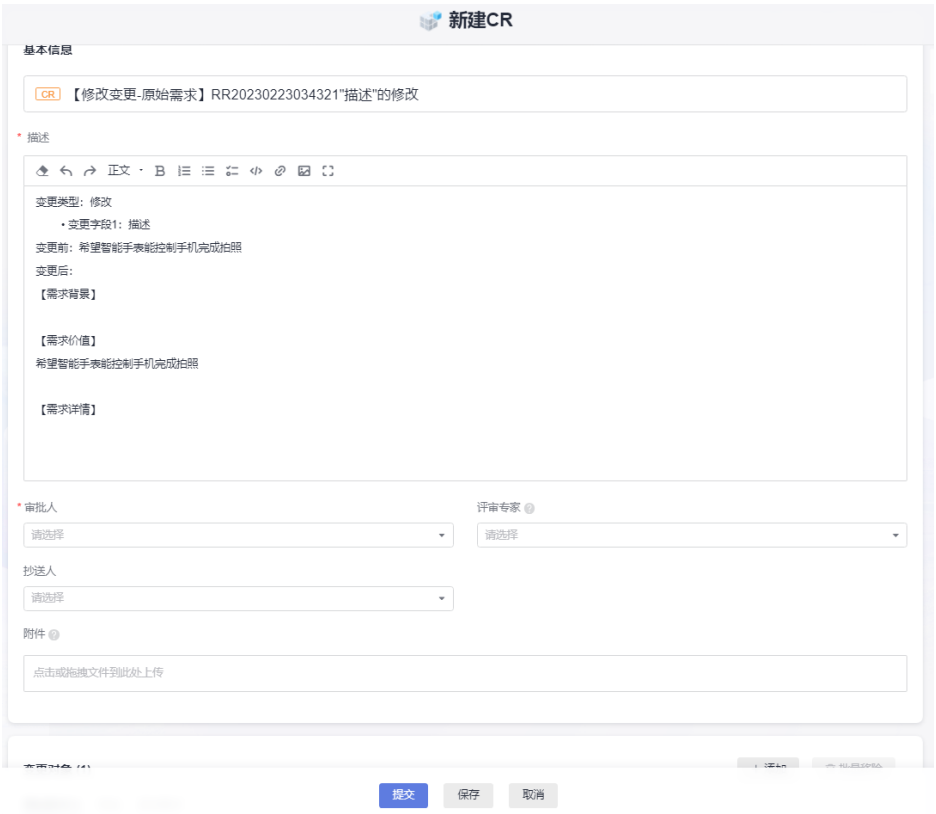


2.3.3.4.2 评审

原始需求的标题、描述、优先级、计划完成时间属于受控字段，即需求一旦提交，这些字段不可随意修改。如果确实需要修改字段值，则须走“评审”流程，待评审流程完成后，修改的字段值会自动变更为修改后的值。

图 2-16 修改受控字段值

图 2-17 新建 CR



2.3.3.4.3 工时

任何工作项在作业过程中都会产生一定的人力投入，这部分信息需要被记录，为效能洞察提供数据基础。您可以在“工时”页签下新增、编辑、删除当前原始需求的工时信息。“工时”页签下“卷积工时之和”为该需求下所有子需求计划工时或实际工时之和。

图 2-18 设置工时

3 缺陷管理最佳实践

3.1 缺陷管理介绍

在整个产品生命周期管理中，缺陷管理是非常关键的一环。无论是硬件系统还是软件开发，都难免遇到不计其数的缺陷，如果缺陷管理不善，产品质量势必大打折扣。华为基于多年沉淀的质量运营管理经验，打造出一套行之有效的缺陷管理优秀实践，为团队提供统一、高效、风险可视的缺陷跟踪平台，确保每一个缺陷都被高质高效闭环。

3.2 名词解释

- **严重程度**

  对缺陷严重等级的评估值，从高到低分别是致命、严重、一般、提示。
- **发现发布/发现迭代**

  是指缺陷的发现版本，或者缺陷是在哪个发布计划/迭代计划中产生的。
- **发现环境**

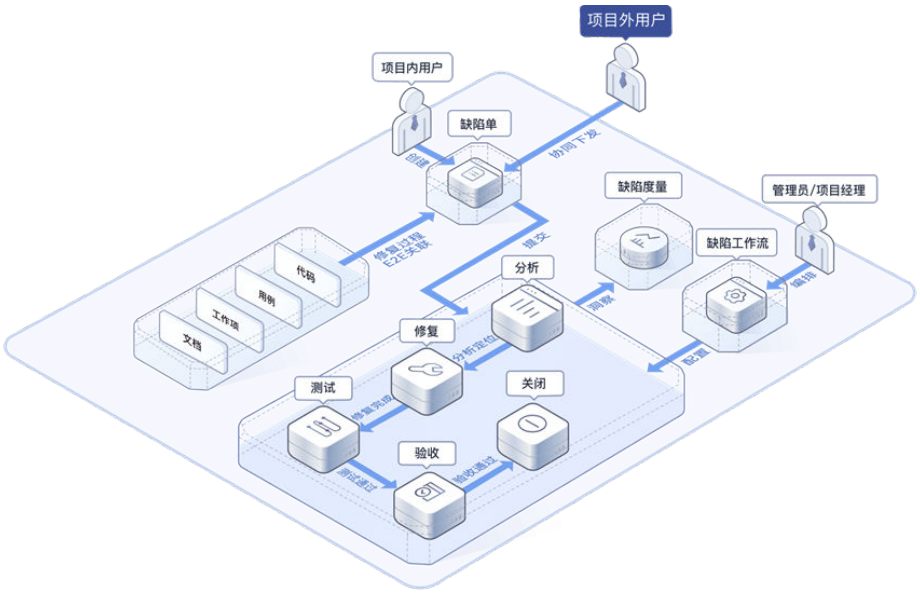
  是指该缺陷存在于哪个环境，常见的有生产环境、alpha环境、Beta环境等。
- **修复发布/修复迭代**

  是指该缺陷计划于哪个发布计划/迭代计划进行修复。

3.3 缺陷管理实践概述

在产品研发过程中，往往存在各团队、各项目各自为战，产品质量难管控、缺陷修复进度难追踪的问题，严重影响产品交付效率。缺陷管理严格把控缺陷提出、分析、修复、测试、验收、关闭的完整流程，提供跨项目的缺陷作业跟踪追溯能力，实时识别产品缺陷风险，为组织的产品交付质量提供保障。

图 3-1 缺陷管理流程



本文将基于系统设备类项目，介绍缺陷管理流程中涉及的主要阶段的最佳实践。

3.4 模拟案例

3.4.1 案例概述

某公司计划推出一款智能手表，研发周期较长，研发过程涉及多个部门、多团队的协作，如何保证缺陷在多个组织间的流转、最终达到有效闭环呢？下面请跟随我们，一起遍历整个缺陷生命周期管理实践。

3.4.2 创建缺陷

缺陷被检测到之后，首先要做的就是将其录入缺陷管理系统，也就是创建缺陷。

前提条件

- 管理员已注册账号并开通CodeArts。详细操作指导请参见**注册华为账号并开通华为云**。
- 管理员已给项目成员完成IAM用户的创建。详细操作指导请参见**创建IAM用户**。
- 已创建**IPD系统设备类项目**或**IPD独立软件类项目**。

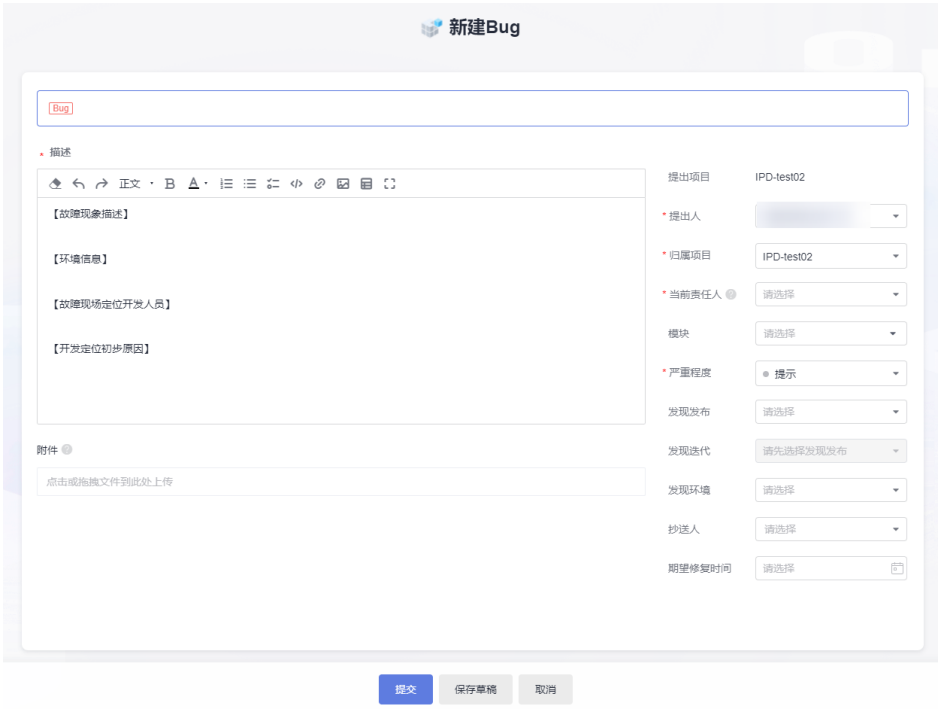
操作步骤

1. 在缺陷管理列表中，单击“新建Bug”，进入新建Bug页面。

2. 填写缺陷信息。

   新建缺陷时，需要填写标题、描述、当前责任人、严重程度等字段，其中“描述”字段尤为重要。创建人需要详细阐述缺陷的现象、期望达到的效果，以及可能存在的报错信息等，描述越详尽，后续分析与定位就越精准。

   图 3-2 新建 Bug



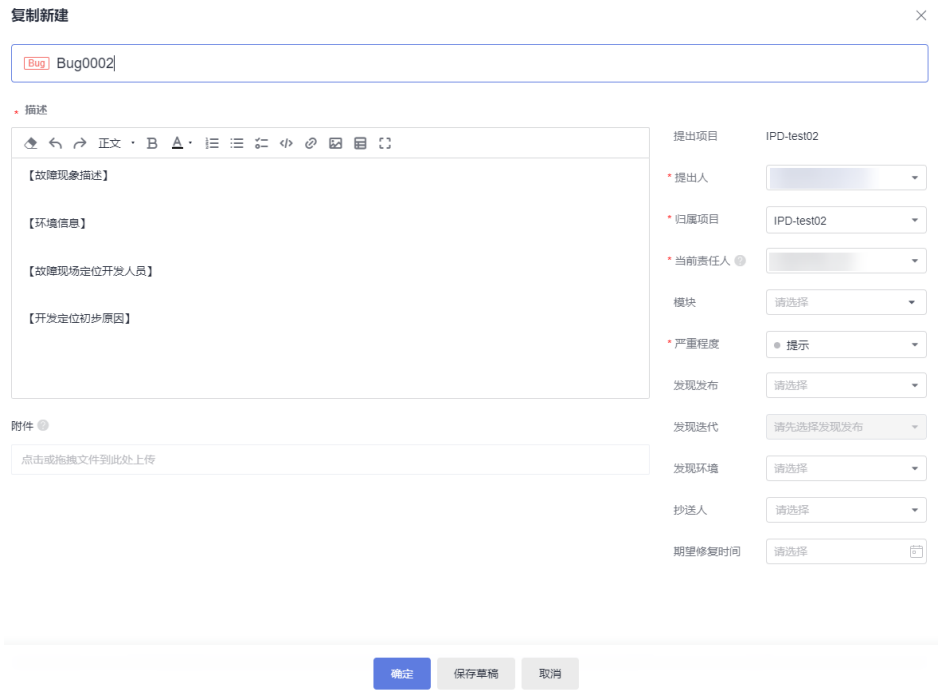
3. 填写完毕后，单击“提交”，缺陷将流转到分析定位环节。

更多操作

如果您在新建Bug时，发现已有同类型的缺陷，或者信息相近的缺陷，可以单击该缺陷的“复制新建”按钮，即可将当前缺陷的信息自动填入，避免填写大量重复信息，提高作业效率。

图 3-3 复制缺陷

图 3-4 修改缺陷信息

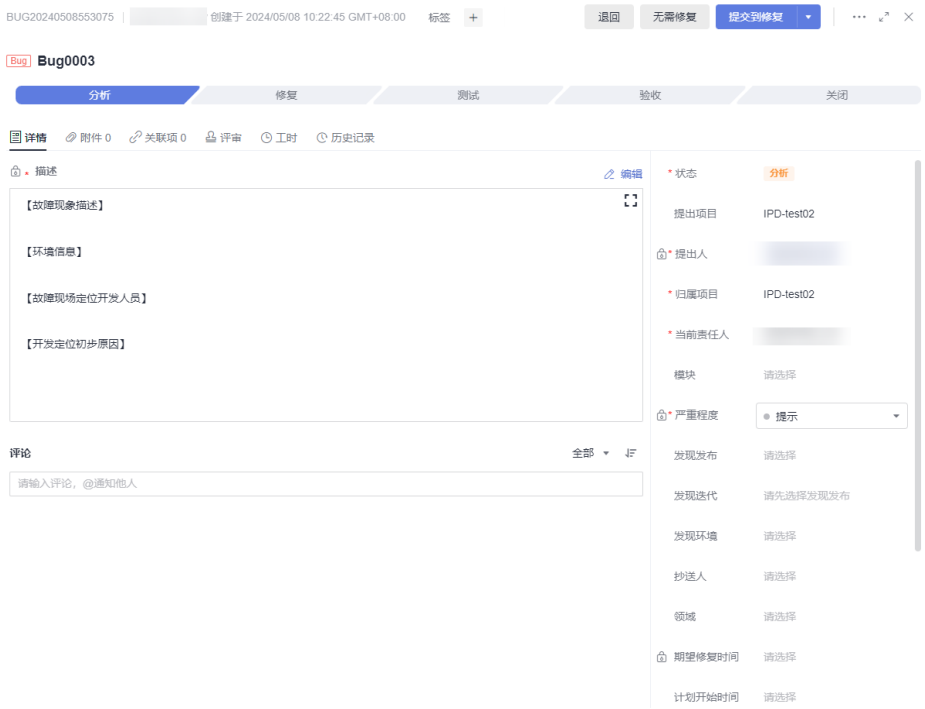


此外，在复制新建时，也可以选择缺陷的归属项目，换言之，我们可以将当前项目的缺陷复制一份到其他项目，减轻重复提单的工作量。但需要注意的是，复制新建的缺陷是一个完全独立的数据对象，在当前项目中将无法追溯到该缺陷。如果您需要建立一个内容相同，但可通过当前缺陷追溯查看的缺陷，建议您使用**协同下发缺陷**功能。

3.4.3 分析缺陷

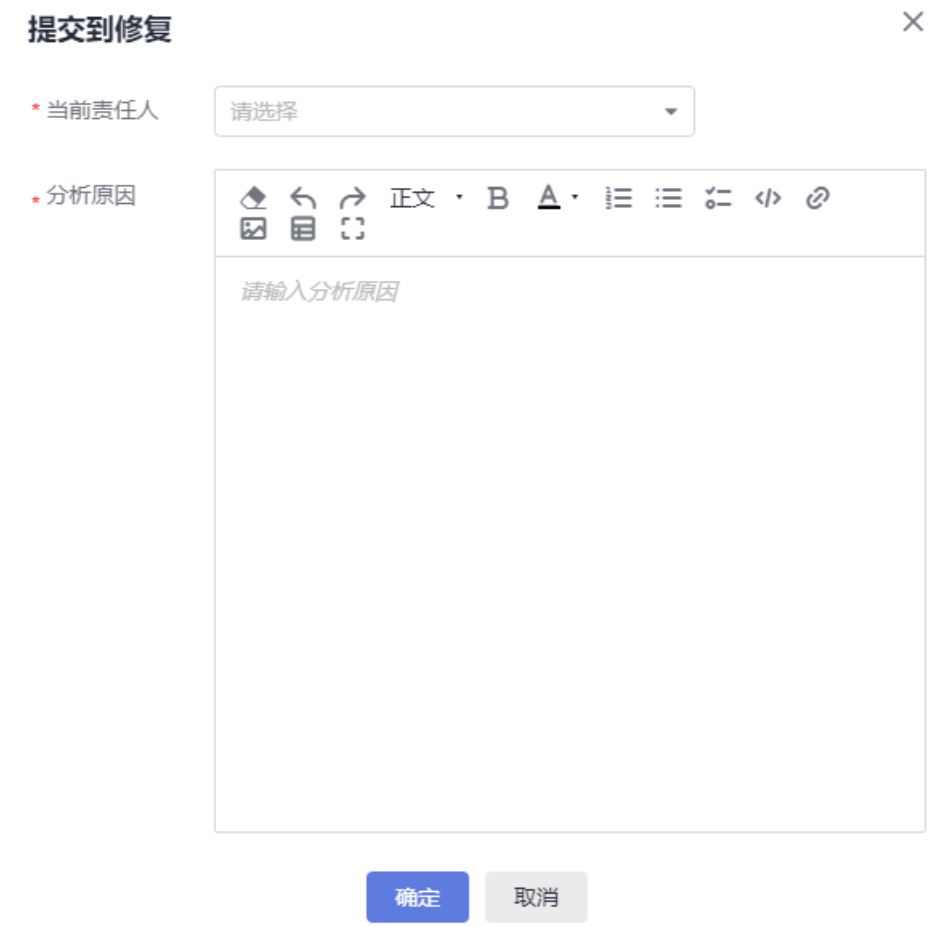
缺陷流转到分析环节后，将由上一步指定的责任人进行分析定位。责任人查看缺陷信息，并根据缺陷所处的环境、故障现象等，定位出缺陷根因。

图 3-5 分析缺陷



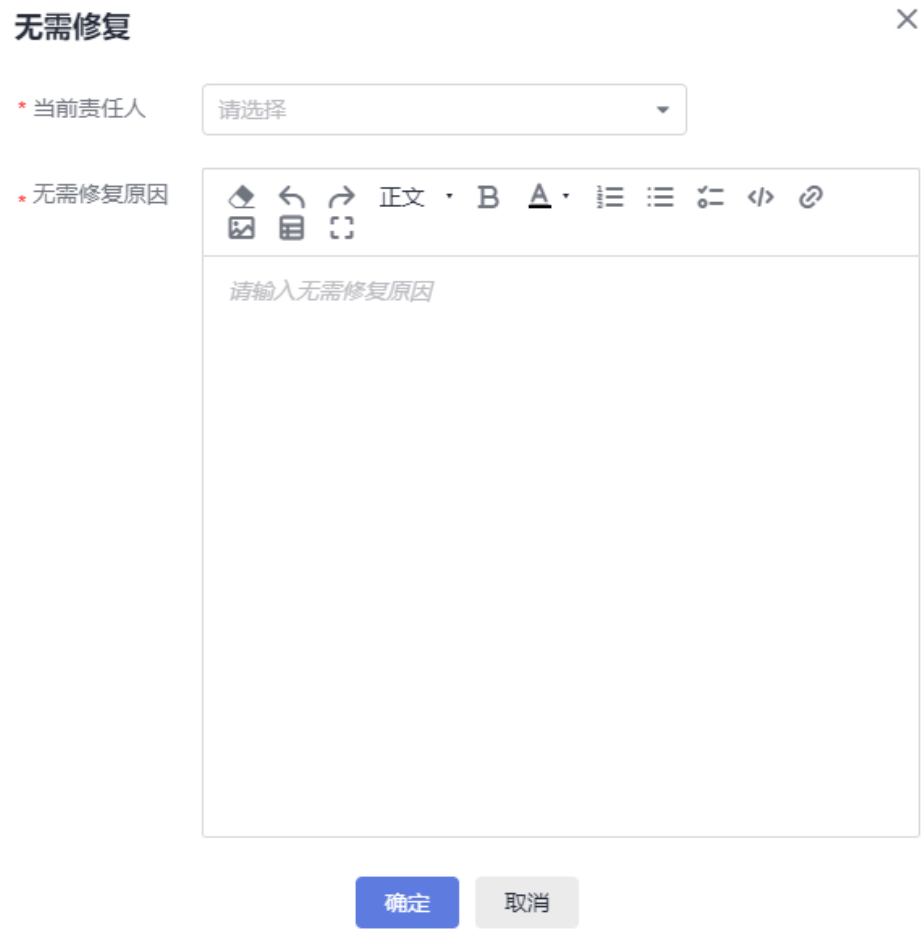
- 如果当前责任人认为该缺陷需要修复，则单击“提交到开发修复”并填入分析原因，缺陷将流转到修复环节，由对应的修复责任人进行修复。

图 3-6 提交到开发修复



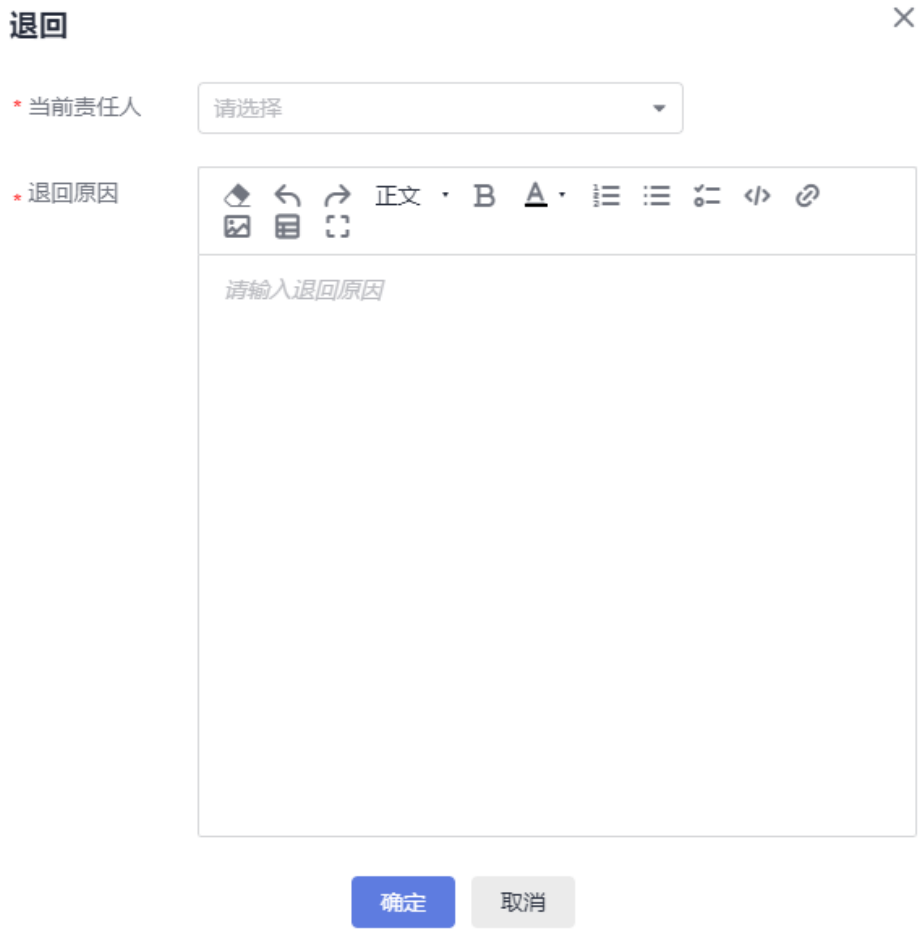
- 如果当前责任人认为该缺陷不需要修复，则单击“无需修复”并填写原因，缺陷将流转到测试环节，由测试责任人进行确认。

图 3-7 填写无需修复缺陷原因



- 如果当前责任人认为缺陷描述不清晰，无法定位，也可以单击“退回”并填写退回原因，缺陷将退回到“确认”环节，由创建人进行再次确认。

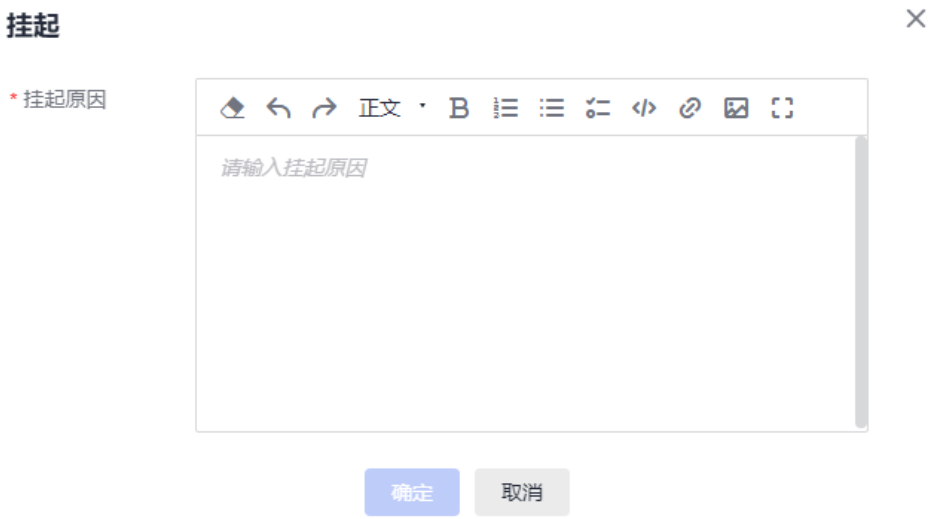
图 3-8 填写退回缺陷原因



- 如果当前缺陷出于某种客观因素，暂时无法分析定位，也可以单击“挂起”并填写挂起原因，使缺陷进入挂起状态，暂时停止作业。

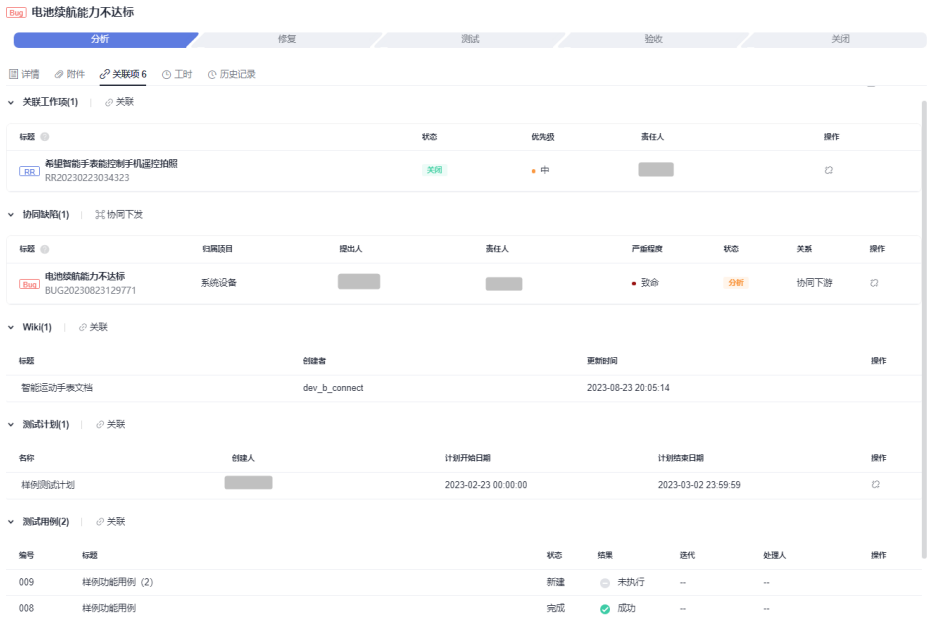
图 3-9 挂起缺陷

图 3-10 填写挂起缺陷原因



缺陷一旦提交进入作业流程，将会与各种工作项产生关联关系，例如缺陷可能是由某个需求引入的，则需要关联该需求以达到如实记录和追溯的目的。这时我们可以在“关联项”页签中进行关联和取消关联工作项。

图 3-11 缺陷关联项



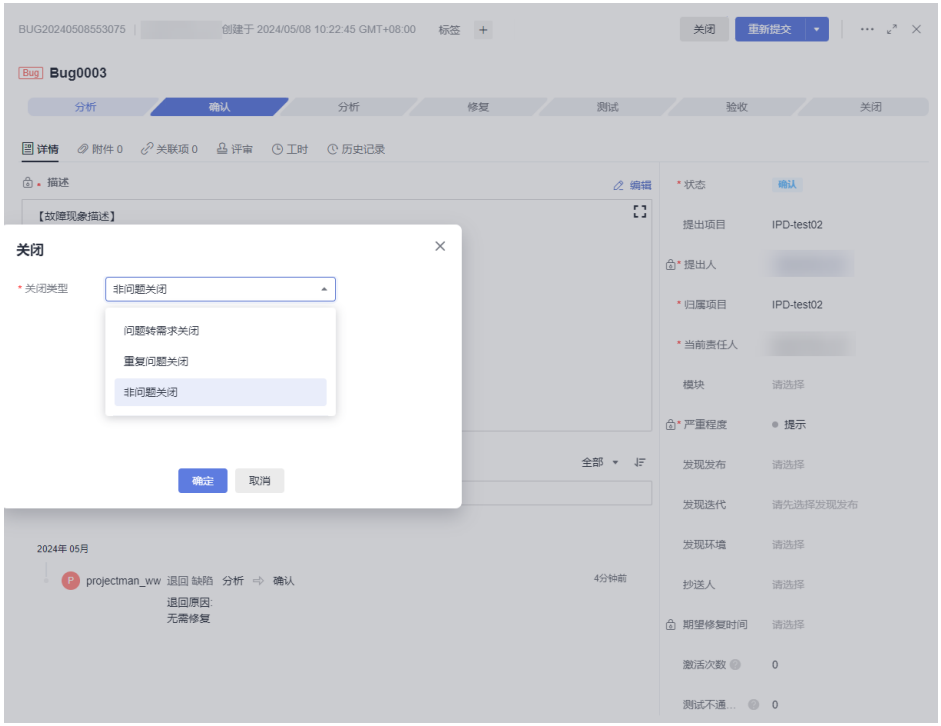
任何工作项在作业过程中都会产生一定的人力投入，需要记录这部分信息，为效能洞察提供数据基础。您可以在“工时”页签下新增、编辑、删除当前缺陷的工时信息。

图 3-12 缺陷工时

3.4.4 确认缺陷

分析环节退回的缺陷，将进入“确认”环节，此时创建人可以再次编辑缺陷信息，并重新提交，如果创建人发现该缺陷并不是一个问题，或者该缺陷已经修复，或者无法作为缺陷进行管理，则可以单击“关闭”，将缺陷关单。

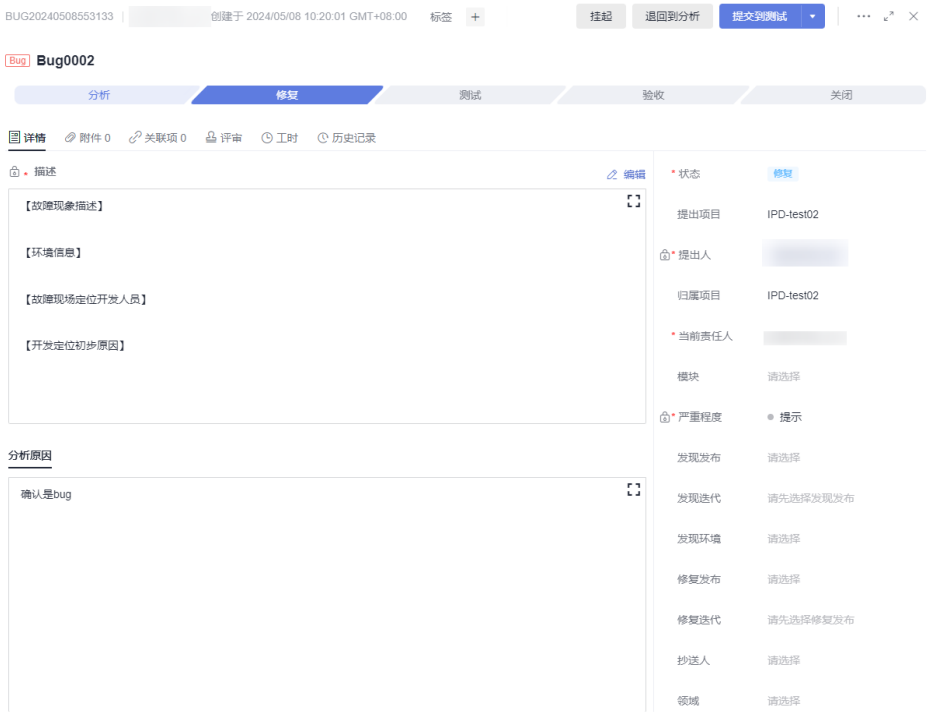
图 3-13 确认缺陷



3.4.5 修复缺陷

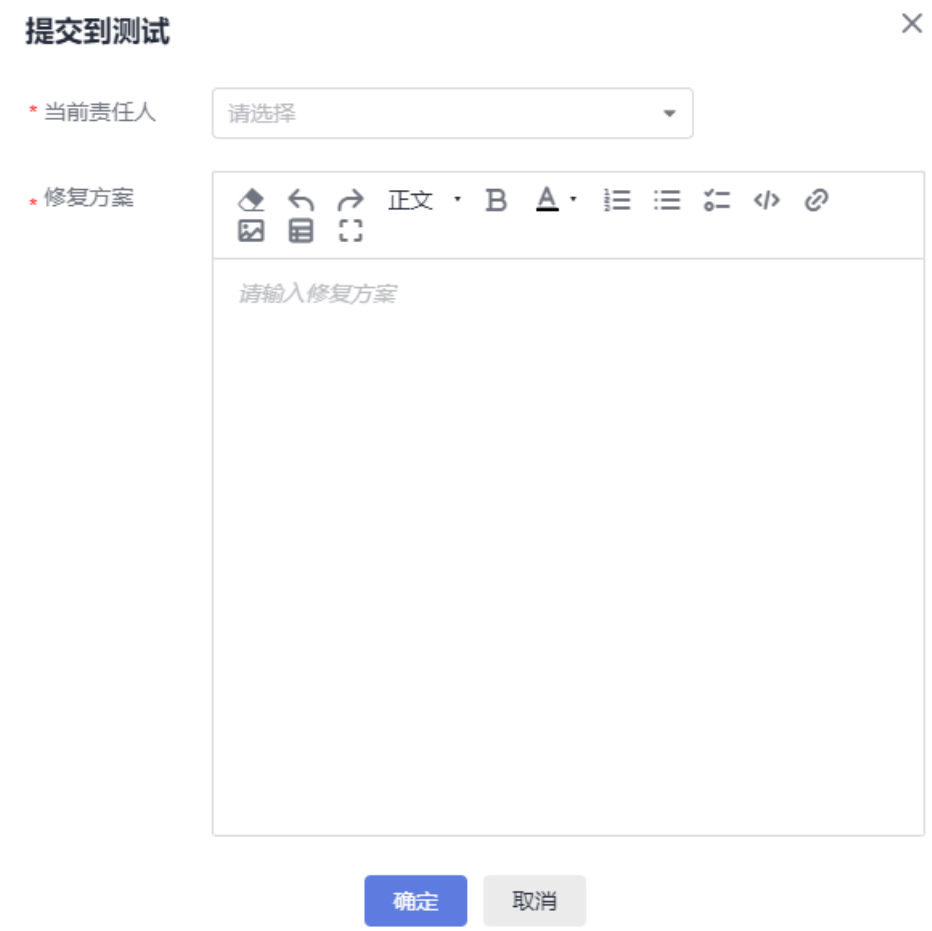
缺陷流转到修复阶段后，将由上一步指定的责任人进行修复。修复责任人将根据缺陷的描述、分析责任人提供的问题根因等信息，对缺陷进行排期修复。这里的排期就是前面所说的“修复发布/修复迭代”，修复责任人可以根据缺陷的严重程度、工作量大小等关键信息，将当前缺陷安排进修复计划中。

图 3-14 修复缺陷



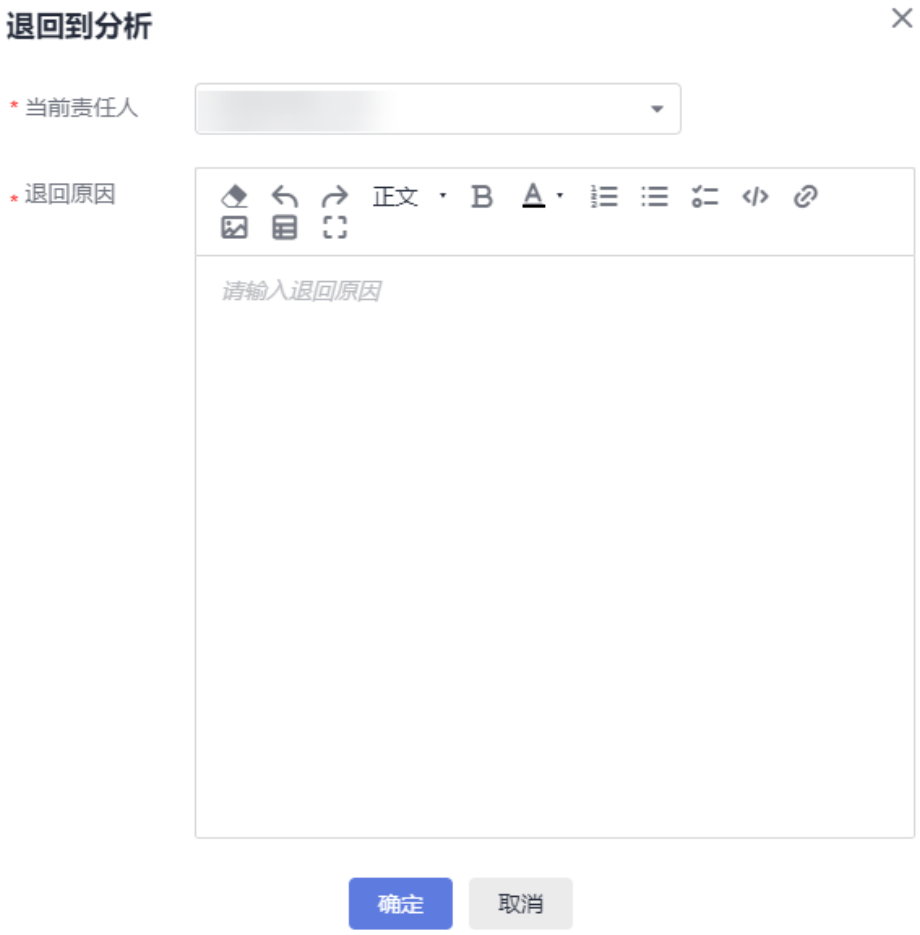
- 修复责任人完成修复作业并自测通过后，即可单击“提交到测试”并附上自己的修复方案，缺陷将流转到测试环节，由测试人员进行回归测试。

图 3-15 填写缺陷修复方案



- 如果修复责任人认为问题根因定位不合理，也可以单击“退回到分析”并填写退回原因，将缺陷退回给上一步的分析责任人进行重新分析。

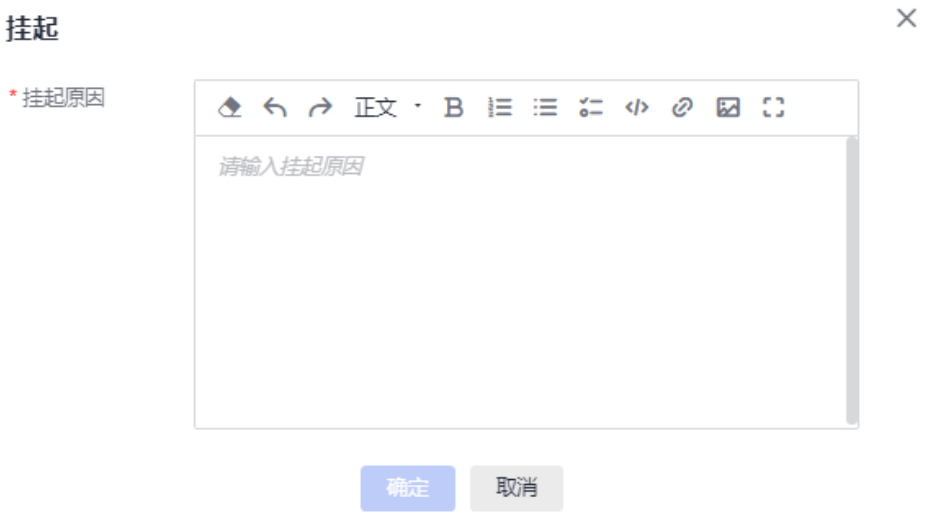
图 3-16 填写缺陷退回到分析的原因



- 如果修复责任人认为该缺陷暂时无法修复，也可以单击“挂起”并填写挂起原因，暂时中止作业。

图 3-17 挂起缺陷

图 3-18 填写挂起原因

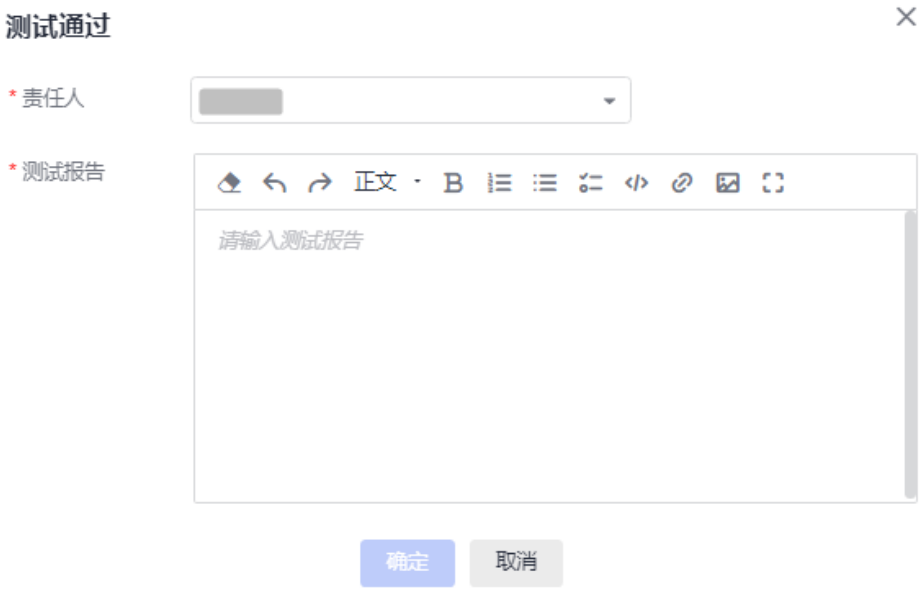


3.4.6 测试缺陷

缺陷修复后，还需要测试人员进行测试，因此缺陷流转到测试环节后，测试人员将根据缺陷的描述、分析原因、修复方案，在缺陷发现环境中进行回归测试并输出测试报告。

测试人员回归验证缺陷确实已经修复的，可单击“测试通过”并填写测试报告。

图 3-19 填写测试报告

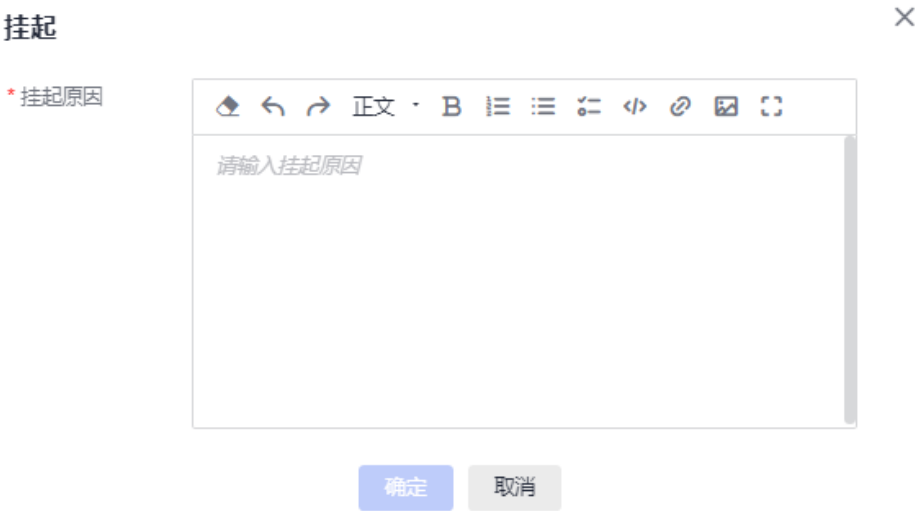


回归测试不通过的，可单击“退回到修复”并填写测试报告，让修复责任人继续修复。

图 3-20 填写缺陷退回到修复的原因

如果当前缺陷暂时无法继续回归验证，也可以单击“挂起”，终止作业。

图 3-21 填写缺陷挂起原因



3.4.7 验收缺陷

测试通过后，如果测试人员和缺陷的创建人不是同一人，则缺陷还需要走回给创建人进行验收。

创建人按照此前触发该缺陷的操作步骤进行验收，发现缺陷确实已修复的，可单击“验收通过”，将缺陷闭环。

3.4.8 关闭缺陷

验收通过后，缺陷关闭，此时缺陷处于闭环状态，整个缺陷生命周期到此结束。

3.4.9 激活缺陷

在产品的不断演进中，有些缺陷可能会重复出现，这时可以再次激活原先已经闭环的缺陷，重启缺陷作业流程，并在原有的分析、修复、测试基础上，进行再次修复。缺陷激活后会进入到分析环节，重新开始作业。

图 3-22 激活缺陷

3.4.10 协同下发缺陷

前面说到，对于系统设备类的项目，通常研发周期较长，涉及部门、团队较多，各个组织互相依赖，互为前提，因此常常出现一个缺陷涉及多个部门的情况。这时可以通过协同下发功能，将同一个缺陷下发给各相关方，下游项目收到该缺陷后，将启动各自的缺陷修复作业流程，各部门互相协作、跟进，共同推动缺陷的修复。同时，还可以在关联项中对自己的协同上游缺陷、协同下游缺陷进行跟踪，随时查看修复进度。

图 3-23 协同下发缺陷

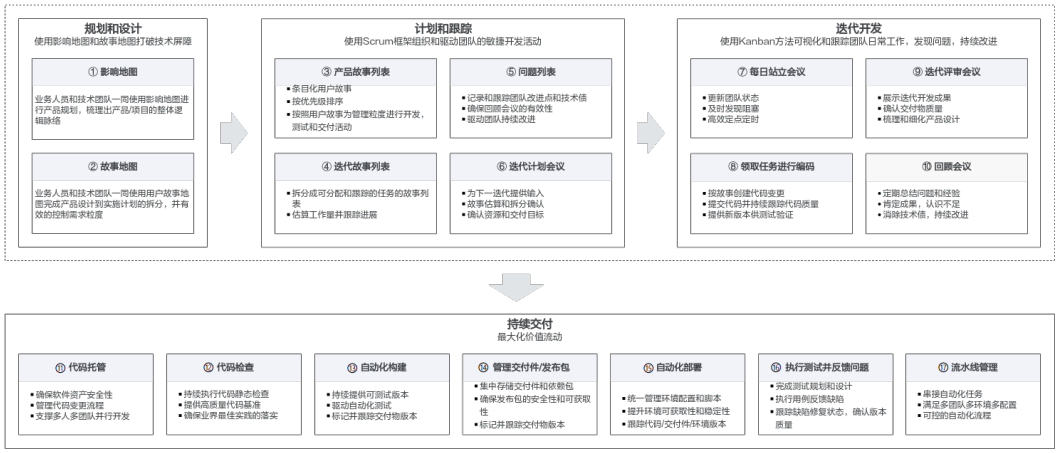
4 HE2E DevOps 实践：管理需求

4.1 方案概述

背景信息

HE2E（HE2E即华为端到端） DevOps实施框架，是结合了多年研发经验并集合了业界先进的实践所形成的一套可操作可落地的敏捷开发方法论，如图4-1所示。

图 4-1 HE2E DevOps 实施框架



- 规划和设计

  步骤①和②是业务（或者是客户）与技术之间进行产品规划，梳理产品整体脉络，以及进行产品规划实施设计，并控制需求粒度与拆分的过程。

  - 软件开发的本质是为了解决问题，提供用户价值的，而不仅是为了提供功能。影响地图就是用来鉴别用户需求是什么，深层的根因是什么。
  - 用户故事就是目标和需求的载体，以用户的场景来讲故事，便于在客户、业务与开发之间进行信息的传递。在这个过程中，独立的需求条目的堆积，很容易导致只能看到各个需求条目，不能从整个解决方案思考需求。用户故事以用户使用的场景为主线，将大的阶段点，及其细分的活动，以树状的结构进行梳理和展现，既可以看到独立的需求条目，又能够看到整体需求场景。

- 计划和跟踪、迭代开发

  步骤③~⑩是Scrum框架过程，是主要的管理实践。

  - Scrum定义了一个相对完整的敏捷过程管理的框架。在CodeArts中，将Scrum的框架与团队日常的开发活动，很好的融合起来。主要的过程产物包括产品故事列表、迭代故事列表、潜在可交付的产品增量、以及过程中产生的问题列表；核心的团队活动包括Sprint计划会议、团队每日站会、Sprint演示会议、Sprint回顾会议等会议、以及团队的日常更新。
- 持续交付

  从步骤⑪开始，进入到工程实践，也就是通常说的CI/CD过程。

  - 持续交付以代码配置管理为基础，除了传统意义的代码资产安全与管控、多人并行开发、版本与基线管理外，也体现了团队的协作与沟通。
  - 代码检查（即静态扫描）、自动化的构建、各阶段的自动化测试、以及相应的自动化部署过程，都被有机的串联在流水线上。
  - 除了代码检查、构建、测试、部署等动态的阶段与活动，还有制品管理，以及各级的环境管理，包括开发环境、测试环境、准生产环境，以及生产环境。
  - 持续交付流水线就是将整个持续交付中，都有哪些阶段，分别运行在什么环境，每个阶段执行什么活动，准入与准出的质量门禁，以及每个阶段的输入与输出的制品进行管理。

应用场景

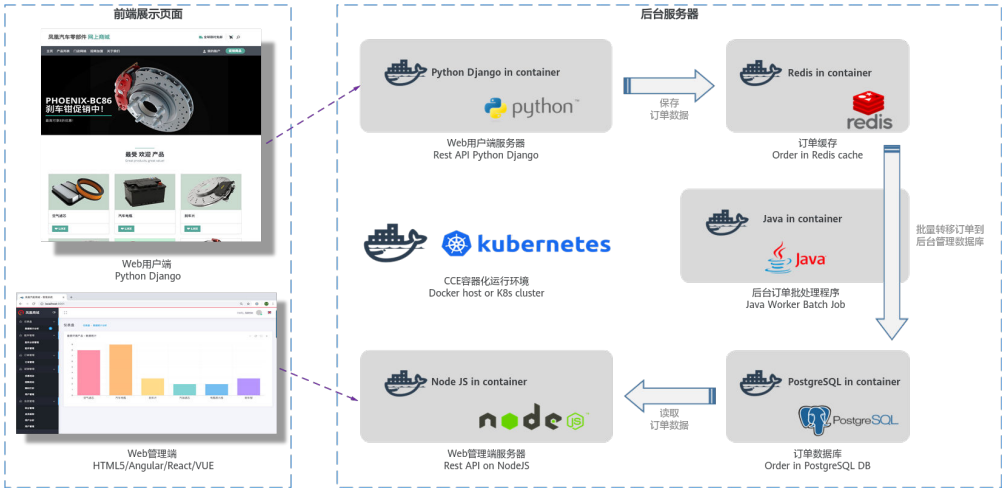
通过一套汽车零部件配件电子商城示例代码“凤凰商城”，以及“DevOps全流程示例项目”，介绍如何使用CodeArts实现HE2E DevOps框架。该方案适用于敏捷/Scrum研发项目。

方案架构

- “凤凰商城”示例程序架构

  “凤凰商城”示例程序的架构图如图4-2所示。

图 4-2 凤凰商城技术架构图



示例程序由表4-1中的5个可以独立开发、测试和部署的微服务组件构成。

表 4-1 凤凰商城微服务组件表

微服务组件	说明
Web用户端服务器（对应样例代码中的“Vote”功能）	● 业务逻辑：用户可以通过浏览器访问此服务的WebUI。当用户在特定商品上单击“Like”时，服务将用户所选择物品的记录保存在Redis缓存中。 ● 技术栈：Python、Flask框架。 ● 应用服务器：Gunicorn。
Web管理端服务器（对应样例代码中的“Result”功能）	● 业务逻辑：用户可以通过浏览器访问此服务的WebUI，会动态显示用户端UI上用户单击“Like”的统计数据，此数据来自PostgreSQL数据库。 ● 技术栈：Node.js、express框架。 ● 应用服务器：server.js。
后台订单批处理程序（对应样例代码中的“Worker”功能）	● 业务逻辑：此服务为后台进程，会监控Redis缓存中物品记录，并将新纪录取出并保存在PostgreSQL数据库中，以便管理端UI可以抽取数据进行统计显示。 ● 技术栈：.net core或者Java（此服务提供两种技术栈实现了同样的功能，可根据需要修改配置选择其中一个作为运行时进程）。
订单缓存	● 业务逻辑：此服务作为用户端UI服务的数据持久化服务存在。 ● 技术栈：Redis
订单数据库	● 业务逻辑：此服务作为管理端UI服务的数据源。 ● 技术栈：PostgreSQL

- “DevOps全流程样例项目”构成

  “DevOps全流程样例项目”是一个Scrum类型的模板项目，项目中预置了部分服务的使用模板。项目实践过程中涉及到的产品及服务如下表。

表 4-2 实践涉及产品/服务列表

服务		说明
CodeArts	需求管理	预置3个已规划并已完成的迭代、项目的模块设置、以及若干统计报表。
	代码托管	预置代码仓库“phoenix-sample”，存放项目示例代码。
	代码检查	预置4个任务
	编译构建	预置5个任务。
	制品仓库	用于存储通过构建任务生成的软件包。
	部署	预置3个应用。

服务		说明
	测试计划	功能测试用例库，预置十余个测试用例。
	流水线	预置5条流水线。
其它组件和服务	统一身份认证服务	用于管理账号。
	容器镜像服务	用于存放构建任务生成的Docker镜像。
	云容器引擎	用于软件包部署，与ECS部署属于两种不同的部署方式。
	弹性云服务器	用于软件包部署，与CCE部署属于两种不同的部署方式。

方案优势

- 针对需求变动频繁、开发测试环境复杂、多版本分支维护困难、无法有效监控进度和质量等研发痛点，提供一站式云端管理平台，管理软件开发全过程。
- 提供可视化、可定制的持续交付流水线服务，实现持续交付，让软件上线提速一倍。

4.2 准备工作

背景介绍

本文以“DevOps全流程示例项目”为例，介绍如何在项目中管理需求。

本样例项目采用Scrum模式进行迭代开发，每个迭代周期为两周，前3个迭代已经完成凤凰商城版本的开发，当前正在进行迭代4的规划。

按照项目规划，迭代4要完成的功能为：限时打折管理、团购活动管理。

由于业务与市场的变化，临时新增一个紧急需求：门店网络查询功能，因此迭代4的规划中增加此功能的开发。

项目中涉及以下四个成员角色：

表 4-3 项目角色列表

项目成员	项目角色	工作职责
Sarah	产品负责人（项目创建者）	负责产品整体规划与产品团队的组建。
Maggie	项目经理	负责管理项目交付计划。
Chris	开发人员	负责项目代码的开发、编译、部署及验证。

项目成员	项目角色	工作职责
Billy	测试人员	负责编写测试用例并执行。

前提条件

已**购买CodeArts**（需购买基础版套餐包与测试服务基础包）。

创建项目

在开展项目实践前，由产品负责人Sarah创建项目。

步骤1 **登录软件开发生产线控制台**。

步骤2 单击，选择区域。

步骤3 单击“立即使用”

步骤4 单击“新建项目”，选择“DevOps全流程示例项目”。

步骤5 输入项目名称“凤凰商城”，单击“确定”，完成项目创建。

----结束

添加项目成员

步骤1 进入“凤凰商城”项目，进入“设置 > 通用设置 > 服务权限管理 > 成员”页面。

步骤2 单击项目成员列表上方“添加成员 > 从本企业导入用户”。

步骤3 在弹框中单击“创建用户”，跳转至“用户”页面。

图 4-3 添加成员

步骤4 单击“创建用户”，依次创建三个用户“Maggie”、“Chris”、“Billy”。

步骤5 返回CodeArts，刷新浏览器，重新单击项目成员列表上方“添加成员 > 从本企业导入用户”，勾选成员“Maggie”、“Chris”、“Billy”，单击“下一步”。

步骤6 单击每一行的“项目角色”下拉列表，为成员Maggie选择角色“项目经理”、Chris选择角色“开发人员”、Billy选择角色“测试人员”，单击“保存”。

----结束

4.3 管理项目规划

管理需求规划

步骤1 为新需求创建工作项。

由于门店网络查询功能是新增的需求，因此产品负责人Sarah要将它加入需求规划视图中。

1. 进入“凤凰商城”项目，单击导航“工作”，在页面中选择“规划”页签。
2. 单击“凤凰商城思维导图”。

   说明

   如果“规划”页签中显示为空白，请创建思维导图。

   1. 单击 + 规划，在下拉列表中选择“思维导图规划”。
      在弹框中输入名称“需求规划”，单击“确定”，页面跳转至思维导图详情。
   2. 单击“添加Epic”，在弹框中勾选“凤凰商城”，单击“确定”。
3. 新建Feature“门店网络”。
   a. 在Epic“凤凰商城”下方单击图标。
   b. 输入标题“门店网络”，回车保存。

图 4-4 新建 Feature

4. 按照同样的方式，为Feature“门店网络”添加Story“作为用户应该可以查询所有门店网络”。

步骤2 编辑Story。

1. 单击Story“作为用户应该可以查询所有门店网络”，参照下表编辑Story信息。

表 4-4 Story 配置

配置项	配置建议
描述信息	输入“作为用户，我想要查询所有门店，以便于挑选合适的门店获取服务”。
优先级	选择“高”。
重要程度	选择“关键”。

2. 为了便于开发人员理解，在本地准备一个“门店网络列表”文件（本文档中为excel表，表格内容参照下表）。

表 4-5 门店网络列表

分店名称	分店地址
Branch A	123 meters to the departure floor, Terminal 1, Airport E
Branch B	No. 456, Street G, Area F
Branch C	No. 789, Street J, Area H
Branch D	West side of Building K, Avenue L, Area K

3. 返回Story编辑页面，单击“点击添加附件或拖拽文件到此处上传”，选择“本地上传”，将列表文件上传至工作项中作为附件。
4. 单击“保存”，完成Story详情的编辑。

----结束

管理迭代规划

步骤1 创建迭代。

1. 进入“凤凰商城”项目，单击导航“工作”，在页面中选择“迭代”页签。
2. 单击页面左上角“迭代”字样后的  ，参照表4-6在弹框中配置迭代信息，单击“确定”。

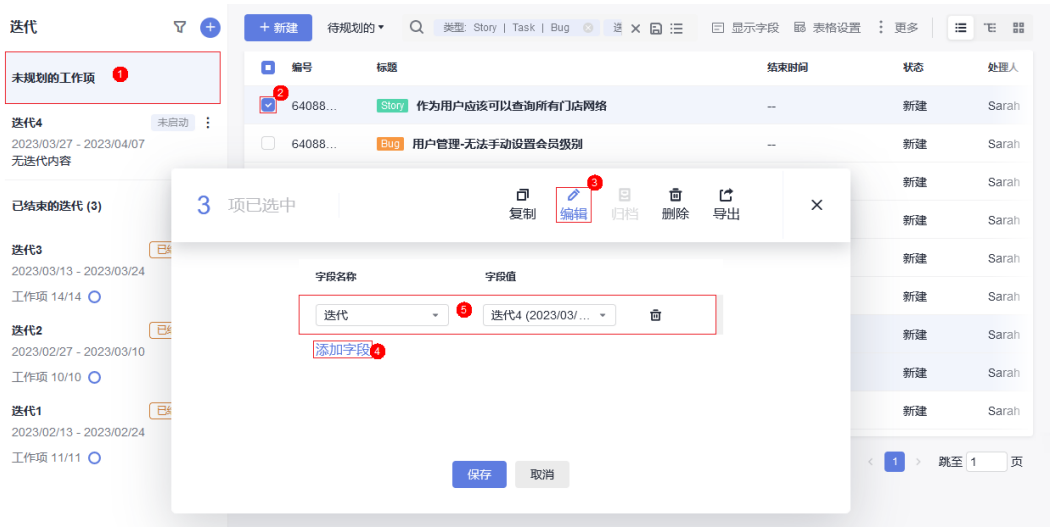
表 4-6 迭代信息配置

配置项	配置建议
迭代名称	输入“迭代4”。
计划时间	设置时长为2周。

步骤2 规划迭代。

1. 单击页面左侧导航“未归划的工作项”。
2. 根据规划，在列表中勾选以下三个Story。
   - 作为用户应该可以查询所有门店网络。
   - 作为管理员应该可以添加团购活动。
   - 作为管理员应该可以添加限时打折。
3. 在页面底部单击“编辑”。
4. 单击“添加字段”。
5. 在字段名称下拉列表中选择“迭代”，并在字段值下拉列表中选择“迭代4”，单击“保存”。

图 4-5 规划迭代



步骤3 分配Story。

1. 单击页面左侧导航“迭代4”。
2. 选择全部Story，参照**规划迭代**的方式，将处理人设置为“Chris”。

步骤4 分解Story。

1. 在列表中找到Story“作为用户应该可以查询所有门店网络”，单击Story名称。
2. 在页面右侧滑出窗口中选择“子工作项”页签。
3. 单击“快速新建子工作项”，输入标题“前端展示 - 添加门店网络菜单”，并选择处理人“Chris”，单击“确定”完成。
4. 按照同样的方式，添加Task“后台管理 - 添加门店网络管理维护模块”。

----结束

监控和跟踪项目状态

- 每日站立会议跟踪任务进度。

  迭代开始后，项目组通过每日站立会议沟通每个工作项的当前进展，并对工作项状态进行更新。

使用卡片模式能够简单直观的查看迭代中各工作项的当前状态。

进入“迭代”页面，单击图标，切换到卡片模式。页面中展示了处于每种状态下的工作项卡片，通过拖拽工作项卡片即可更新其状态。

- 迭代评审会议验收迭代成果。

  在到达迭代的预计结束时间前，项目组召开迭代评审会议，展示当前迭代的工作成果。

  “迭代”页面提供了迭代统计图表，团队可以方便的统计当前迭代的进度情况，包括需求完成情况、迭代燃尽图、工作量等。

  进入“迭代”页面，单击“统计”，即可展开迭代进度视图。

4.4 管理项目配置

管理项目通知

项目经理Maggie希望当任务（工作项）分配给团队成员时，该成员能够收到通知，以便及时处理。

步骤1 进入“凤凰商城”项目，单击导航“设置 > 工作配置 > 通知设置”。

步骤2 页面中显示样例项目中的默认配置。

由于默认配置可以满足需求，因此本文档中暂不做配置的修改。如果您有需要，直接勾选所需选项即可，服务将自动保存您的选择。

步骤3 验证配置结果。

当项目经理完成分解Story操作后，开发人员Chris将收到以下两类通知。

- 站内信：Chris登录首页后，页面右上角图标处将显示数字，单击该图标即可看到通知。
- 邮件：如果为项目成员创建用户时配置了邮箱，且项目成员在个人设置中开启了邮件通知，则将会收到服务发出的邮件。

  说明

  每个成员均可以设置是否接收邮件通知。开启邮件通知的方法为：

  1. 单击页面右上角的用户名，在下拉列表中选择“个人设置”，默认跳转至“消息通知”页面。
  2. 在页面中找到“邮件通知”，勾选“开启”。可以单击“更改设置”，修改邮箱地址。

----结束

定制项目工作流程

在迭代Review会议中，团队将向产品负责人做产品演示，并出示测试报告，由产品负责人确认Story是否完成。而当前的Story状态中没有能够显示测试已完成的状态，因此测试人员建议增加一个状态“验收中”。

项目经理Maggie通过以下操作为Story添加状态。

步骤1 进入“凤凰商城”项目，单击导航“设置 > 工作配置”。

步骤2 在页面左侧导航中选择“公共状态设置”，页面将显示样例项目默认的工作项状态列表。

步骤3 单击“添加状态”，参照表4-7在弹框中编辑状态信息，单击“添加”保存。

表 4-7 状态信息配置

配置项	配置建议
状态	输入“验收中”。
状态属性	选择“进行态”。

步骤4 在页面左侧导航中选择“Story设置 > 状态与流转”，页面将显示样例项目默认的Story状态列表。

步骤5 单击“添加已有状态”，在弹框中勾选“验收中”，单击“确定”保存。

图 4-6 添加 Story 状态

步骤6 通过拖拽将状态“验收中”的顺序至于“测试中”之后。

步骤7 验证配置结果。

1. 单击“工作”，在页面中选择“工作项”页签。
2. 在列表中单击任意Story名称，查看Story详情。
3. 单击“状态”项的值，在下拉列表中可以看到选项“验收中”。

----结束

