CodeArts Req

Best Practices

Issue 01

Date 2024-05-31

HUAWEI TECHNOLOGIES CO., LTD.

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process.* For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

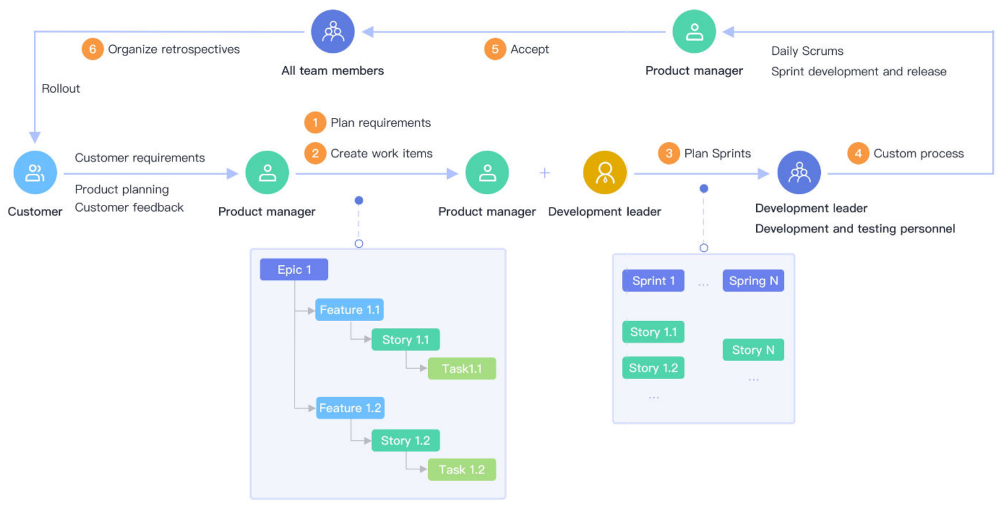
Contents

1 Best Practices of Scrum Projects

1.1 Practice Overview

Scrum applies to R&D management platforms in agile development mode. It enables short-term continuous delivery and quick response to requirements and market changes. A complete Scrum sprint process involves four phases: requirement planning, sprint planning, sprint development, and agile review. **Figure 1-1** shows the overall process.

Figure 1-1 Scrum sprint process



Based on the common issues identified through communication with a large number of customers, this chapter presents the best practices for the main phases of the Scrum process.

1.2 Requirement Management

1.2.1 Understanding the Four Keywords of Agile Requirement Management

Background

What is the relationship between concepts related to agile development, such as epic, feature, story, and task? How can we flexibly use these concepts to make agile requirement management more efficient? This section analyzes these four keywords in detail.

What Are Epic, Feature, Story, and Task?

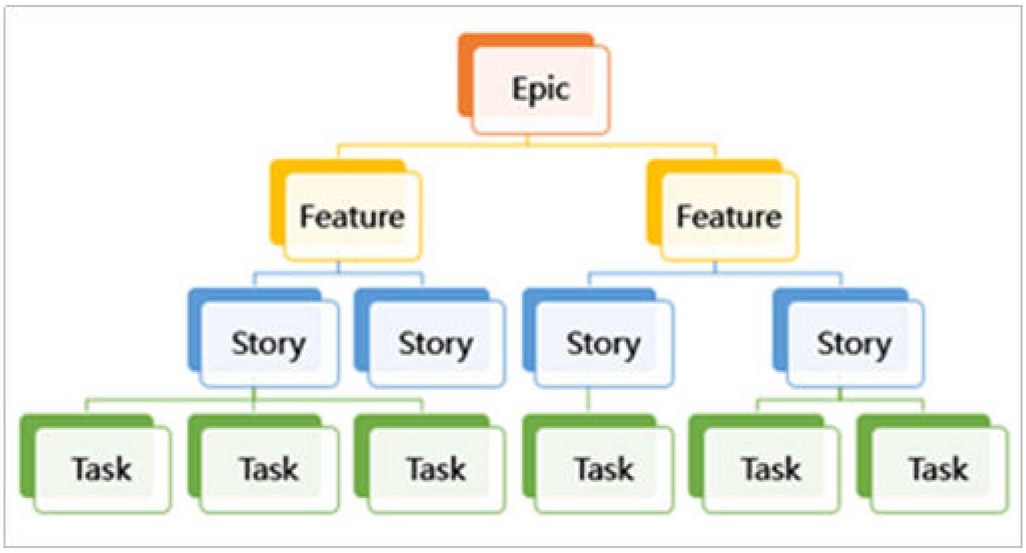
Epic, **Feature**, **Story**, and **Task** indicate requirement granularities in descending order, which are used to divide requirements. They can be regarded as requirement placeholders. Their meanings can be used as a reference for requirement division.

- **Epic**: project vision and goal. It holds significant strategic value for an enterprise, because its implementation, usually lasting several months, brings about the corresponding market position and returns.
- **Feature**: a product function or feature that delivers benefits. Features are more specific and intuitive than epics in customers experience, possessing business value. It usually takes weeks to complete sprints.
- **Story**: a user story. A story is a user's detailed description of product functions. It puts features into a product backlog for continuous planning and adjustment. Stories with high priorities are prioritized for delivery and hold significant user value. Stories must comply with the INVEST principle (**I**ndependent, **N**egotiable, **V**aluable, **E**stimable, **S**mall, and **T**estable). It takes several days to complete stories, usually in a sprint.
- **Task**: a specific task to be completed by team members. At the Sprint planning meeting, stories are assigned to members, and members break down the stories into tasks and estimate the workload. The whole process is usually completed within one day.

What Is the Relationship Between Epic, Feature, Story, and Task?

Epic-Feature-Story-Task manages requirements in a structured manner. It breaks down requirements layer by layer to build bottom-up dependencies, as shown in **Figure 1-2**.

Figure 1-2 Relationship between epic, feature, story, and task



Requirements change in development. We need to continuously adjust requirements to keep them and our team on the right track towards the goal. To this end, we must align requirements with the epic and ensure that stories and tasks which requirements belong to are associated with their upper layers.

For more information about structured requirement management, see **Managing Requirements in a Structured Manner**.

How Can We Flexibly Use These Concepts to Make Requirement Management More Efficient?

This section splits and displays the requirements of a case based on CodeArts Req to provide a detailed description of epic, feature, story, and task.

Case:

The turnover of a large store dropped sharply due to the impact of the Internet.

To keep its consumers and maintain the market position and share, the market decides to build its own online store in six months.

- **Step 1: Determine and create an epic.**

  As described above, granularity and meaning must be taken into consideration before requirement confirmation.

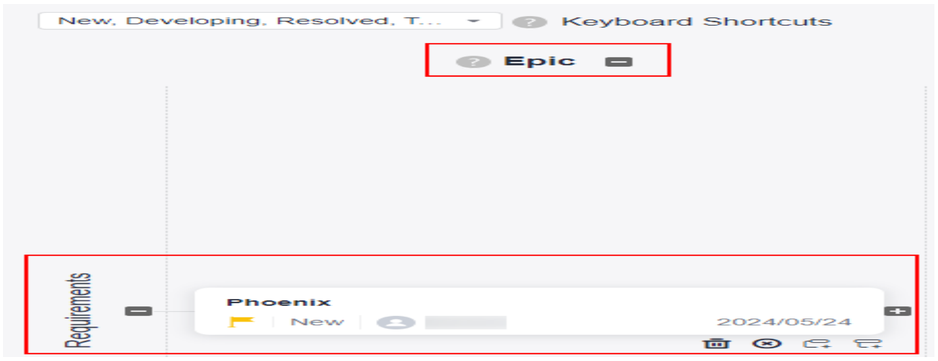
  Is a product an **epic**? Is each service module of a product an **epic** or **feature**? These questions need to be answered.

  – A product is usually of strategic significance. Therefore, it is suitable to take a product as an epic. However, it does not apply to all products. The product itself and its granularity matter. In this case, the online store cycle is 6 months, and the goal is to maintain market share. In view of granularity and strategic sense, it is suitable to take the online store as an epic.

  – Whether each service module is an epic or feature depends on the actual situation. For example, building a smart city is a vision, which includes large service modules, such as smart transportation, smart government, and smart community. Therefore, it is better to use epic as a placeholder.

Create a Scrum project in CodeArts and name it "Large online store". On the page for requirement planning page, create an epic.

Figure 1-3 Creating an epic

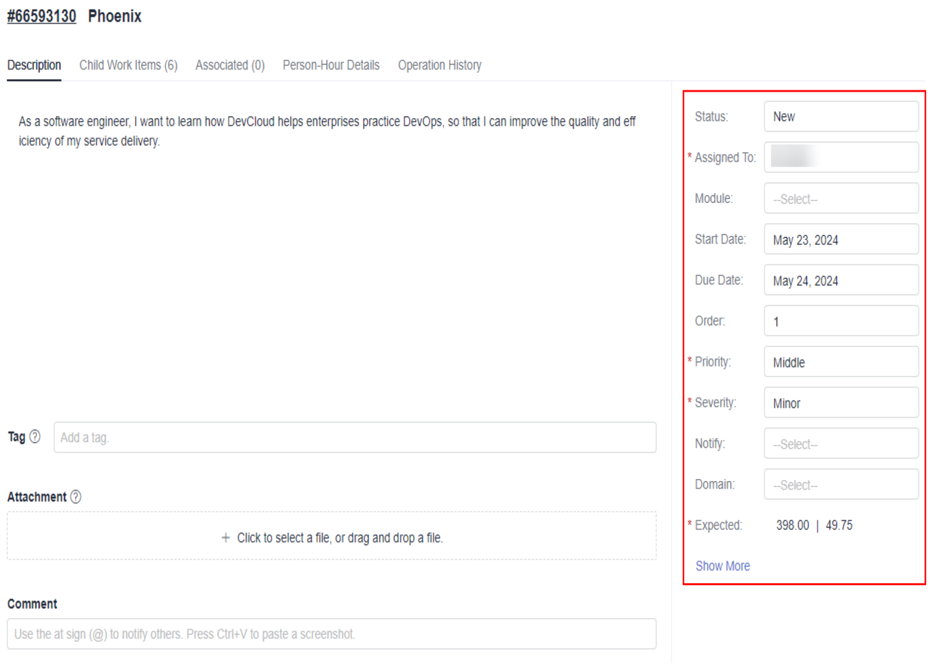


After the creation, click the store to go to the editing page. Fill in the description. You can use the template provided by CodeArts.

- **As**: As for this epic, the user is the entire company.
- **I want**: The desired result is to build an online store.
- **So that**: The goal is to keep consumers and maintain market position and share.

  Set the start time, end time, priority, importance, and estimated workload of this epic. The information is essential to understanding the product and project. Therefore, you need to fill in the information in detail.

Figure 1-4 Writing an epic


Description
Child Work Items (6)
Associated (0)
Person-Hour Details
Operation History
Tag
Add a tag.
Attachment
Click to select a file, or drag and drop a file.
Comment
Use the at sign (@) to notify others. Press Ctrl+V to paste a screenshot.
Show More

#66593130 Phoenix

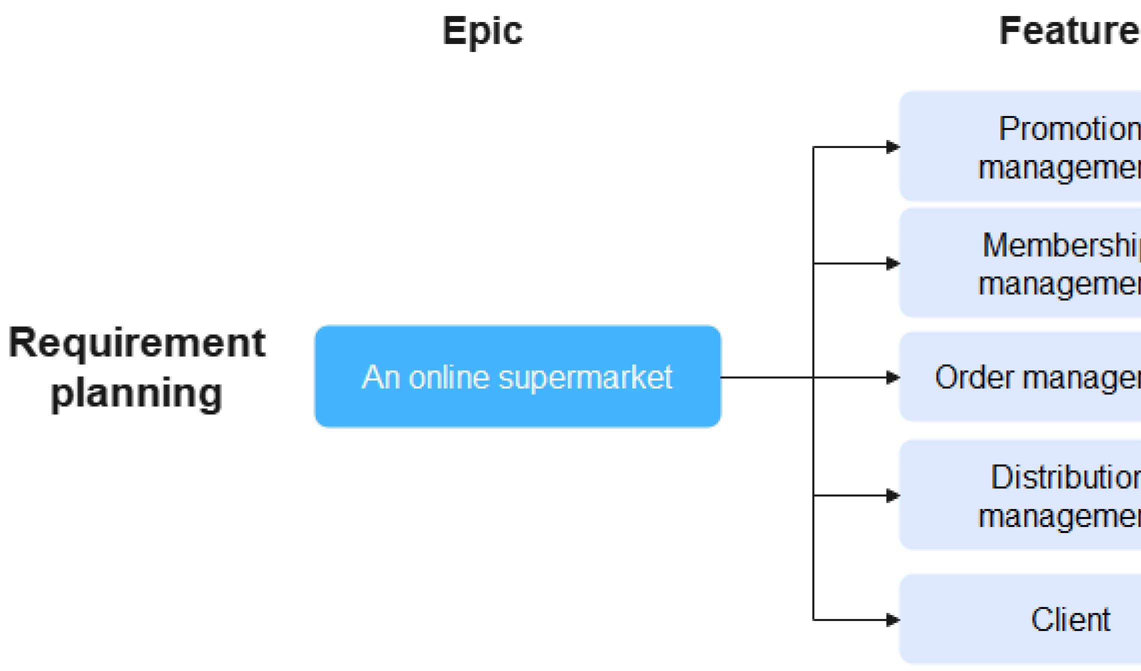
As a software engineer, I want to learn how DevCloud helps enterprises practice DevOps, so that I can improve the quality and efficiency of my service delivery.

Status:	New
* Assigned To:	
Module:	--Select--
Start Date:	May 23, 2024
Due Date:	May 24, 2024
Order:	1
* Priority:	Middle
* Severity:	Minor
Notify:	--Select--
Domain:	--Select--
* Expected:	398.00 | 49.75

- **Step 2: Break down the epic into features.**

  The customer requires that five functional modules (promotion management, member management, order management, delivery management, and client) be delivered within six months. A team sprint is two weeks. Each module requires two or three sprints. In view of granularity and meaning, it is suitable to take five modules as features.

Figure 1-5 Splitting the epic into features



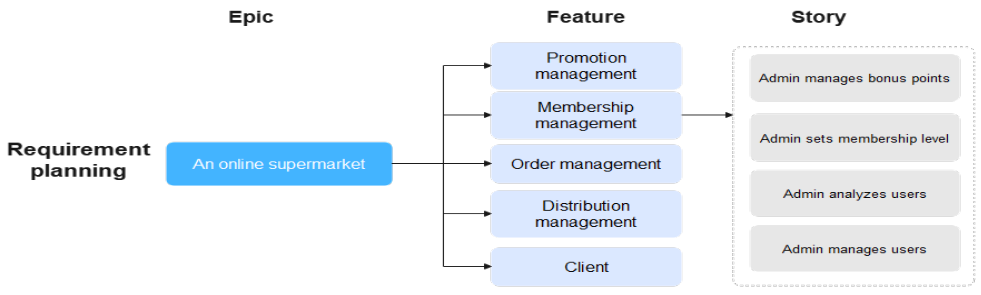
After the creation, you can edit the information on the details page. The information items on the page are the same as those on the epic page.

- **Step 3: Break down features into stories.**

  Agile development is gradually detailed. It is not required that all requirements be detailed at the same time. Only the stories of the current sprint and one or two future sprints should be detailed. The story of a future sprint can be general. Stories in the current Sprint must comply with the INVEST principle. The development team needs to complete delivery at the end of the Sprint.

  Member management, as a feature, has a higher customer priority. Therefore, member management needs to be broken down into stories and added to the product backlog in the requirement review. After the breakdown, the following functions related to the administrator need to be included: bonus point management, member level management, user analysis, and user management. These specific functions can be taken as stories. Note that it is better to deliver stories in a sprint. If the delivery fails, further break down the stories. Only delivered stories are valuable. Stories that cannot be delivered are a waste for the current Sprint. The stories after the breakdown are shown in **Figure 1-6**.

Figure 1-6 Breaking down features into stories



- **Step 4: Break down stories into tasks**

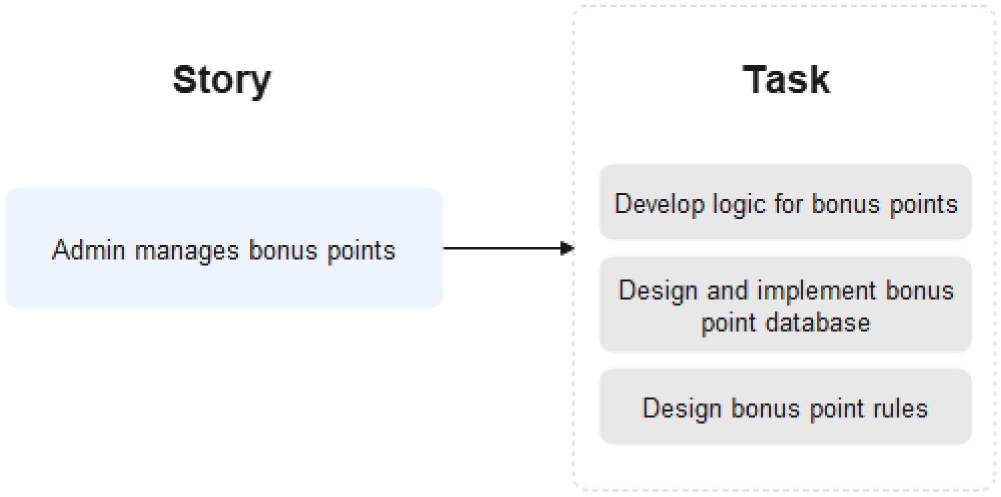
  At the Sprint planning meeting, the team and PO need to jointly select the stories to be completed in the Sprint from the product backlog according to the priority and put them into the sprint backlog. After claiming the story, team members break down the story into tasks for further estimation.

  So how can we **distinguish stories from tasks**?

  – **Story** focuses on value and needs to be completed in a sprint. The completion must comply with the INVEST principle and usually takes several days. Story is more of a noun. For example, bonus point management as a story can be described as follows: As an administrator, I can manage bonus points of members to provide different value-added services based on consumption levels.

  – **Task** focuses on value realization. Story functions are implemented based on tasks. Task completion usually takes 1 to 8 hours. Task is more of an action. For example, bonus point management as a story needs to be implemented through service logic development, bonus point rule design, and bonus point database design. These are tasks, as shown in **Figure 1-7**.

  Figure 1-7 Breaking down stories into tasks


Story
Task
Admin manages bonus points
Develop logic for bonus points
Design and implement bonus point database
Design bonus point rules

In this way, the online store requirements are broken down into epics, features, stories, and tasks.

Summary

When using "Epic-Feature-Story-Task" to manage requirements, pay attention to the following aspects:

1. In agile development, requirements are gradually refined and broken down in a top-down manner.
2. Epics and features are large-granularity requirements. They are users' expectations for products and descriptions of functions and features.
3. You need to break down features into smaller stories. Future requirements (possibly in three or more sprints) do not need to be further divided. In some sprints, break down stories into smaller ones.
4. In a sprint, stories are broken down into tasks.
5. All the rough and detailed stories are placed in the product backlog. The entire list must be detailed appropriately, emergent, estimated, and prioritized (DEEP). The list must be sorted out and prioritized periodically to ensure that high-priority requirements are implemented and delivered first.
6. Keep contact with the customer during the whole process to ensure that the functions to be implemented are what the customer really wants.

This document uses a user scenario to help understand the meanings and usage of "Epic-Feature-Story-Task". Every project is unique. Therefore, you need to focus on product and service features.

1.2.2 Making Effective Requirement Management and Planning When Software Project Requirements Change Frequently

Overview

This section describes how to make effective management and planning for frequent project requirement changes in terms of background, problem analysis, and solution.

Background

Frequent requirement changes are the biggest constraint for both project-based software and product-based software development. According to *China DevOps Status Report 2019*, more than half of enterprises believe that frequent requirement changes are the main culprit behind delayed software delivery. Therefore, it is imperative to solve or mitigate the impact of frequent requirement changes.

Problem Analysis

Customer collaboration models, personnel capabilities, and R&D processes vary with enterprises. Such variation causes different symptoms and root causes of frequent requirement changes. As a result, measures should be taken case by case. Based on our observation and communication with enterprises, we summarize the following scenarios:

- Requirements are disordered and frequently changed, making it difficult to manage.
- Requirement priorities are continuously adjusted, disrupting the development plan.
- Requirements are missing.

The following is the analysis of some scenarios:

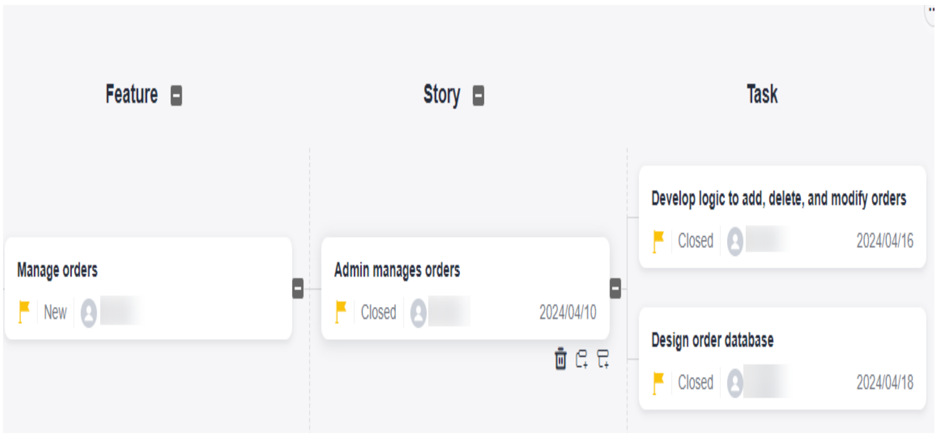
- **Scenario 1: A software project has a clear structure at its early stage, but the requirements change frequently and in detail at the later stage. How can the requirements be effectively managed and planned?**

  Project team members are troubled by unclear and incomplete requirements at the early stage. As a result, the requirements need to be modified at the later stage.

  In this case, requirements are not correctly planned. Requirement levels are not clearly divided and there is no standard mechanism. For example, a customer plans a user login function, as shown in the following figure. The customer plans to release the task of administrator login in the first version. Later, a mobile number login requirement is added. This task is set to be released in the second version. As a result, a story contains multiple tasks released in different versions (or sprints), which is inconvenient for management. In this case, the key to this problem is **managing requirements in a structured manner** and root causes are:

  a. Failed to distinguish fragmented requirements that are continuously generated during project implementation and iterative system/product functions and features
  b. Misunderstood and misused the "Epic-Feature-Story" requirement structure provided by CodeArts.

  Figure 1-8 Requirement structure



- **Scenario 2: Some leader wants to upgrade requirements in a software project. How can we deal with it in the agile R&D mode?**

  Upgrading requirements means raising their priorities in implementation. Therefore, this problem is about **requirement priority management**. To solve this problem, we need to understand:

  – Why does the leader need to upgrade requirements? If it is reasonable, the team needs to optimize the work arrangement process to minimize the impact on R&D and improve the response speed and capability.

  - How often does this happen? Frequent requirement upgrades are a bad norm. The team needs to consider the causes and how to adjust the process, system, collaboration model, or personnel capabilities. For occasional requirement upgrades, find a solution accordingly. There is no need to adjust processes and regulations.
  - Is there anything in common of the requirements to be upgraded? For example, are they related to a customer or a function domain (such as refund)? We can figure out solutions based on commonalities.
- **Scenario 3: Some urgent requirements are often added due to external reasons.**

  These requirements are unexpected and definitely important. Therefore, it is about managing requirement priorities.

  It is a problem of **missing important requirements**, which is caused by the failure to predict requirements. To deal with this problem, the team needs to know:
  - What are the external causes? Check whether these causes have something in common and take corresponding solutions.
  - Is there anything in common of the added requirements? If yes, handle them accordingly.
  - How temporary is the addition? If the team can adjust and respond more quickly so that the temporary requirements do not affect the team, this problem is no longer a problem.
  - Why does not the team adjust the process or work style? Is it unuseful, or does not the team know how to adjust the process or work style?

Solution

Based on the root causes obtained in the preceding analysis, the problems to be solved can be classified as follows:

1. **How can we manage requirements in a structured manner?**
2. **How can we manage requirement priorities?**
3. **How can we avoid missing important requirements?**

- **How can we manage requirements in a structured manner?**

  First, we do not need structured requirement management in all circumstances. Structured requirement management is required only when there are a large number of interrelated requirements and implemented requirements need to be managed and maintained. In this case, the team needs to use the Scrum project template in CodeArts, whose "Epic-Feature-Story" structure and requirement planning function are helpful for structured management. So what is this structure based on? The answer is **product or system functions and features**. Information such as versions, customers, and modules that software project management focuses on can be implemented through different attributes or even tags of requirements.

  Simply speaking, perform the following steps:

  a. Establish a CodeArts project for products or systems.
  b. Establish an "Epic-Feature-Story" requirement structure.
  c. Manage different modules and versions using the attributes of work items.

For details, see **Managing Requirements in a Structured Manner**.

- **How can we manage requirement priorities?**

  Requirement priority management helps determine the sequence of requirement fulfillment, thereby maximizing the return and minimizing the risk or investment. To properly manage priorities, we need to:

  a. Determine the priority model, such as **Kano Model**. The model includes factors to be considered and comprehensive judgment principle of the factors.
  b. Prioritize requirements based on specific factor quantification and sorting criteria. For example, cost-benefit approach determines the requirement sequence by revenue and profit.
  c. Adjust requirement priorities.
  d. Improve the priority model. Adjust the model or the implementation details based on the feedback.

  For details, see **Managing Requirement Priorities**.

- **How can we avoid missing important requirements?**

  We can take measures based on different time points, that is, before, during, and after important requirement missing. According to the Pareto principle, we need to ensure that: (1) For general problems, there are solutions for software project members to follow. (2) In special cases, there is an emergency mechanism that guides on-site handling, and a troubleshooting review.

  a. In-event: Solve the problem using a common or special solution.
  b. Post-event: Perform a review based on the model or approach to form a new common practice or special handling method.
  c. Pre-event: Specify how to distinguish between common and special situations and develop corresponding handling methods or emergency mechanisms.

  For details, see **Avoiding Missing Important Requirements**.

1.2.3 Managing Requirements in a Structured Manner

Why Do We Need to Manage Requirements in a Structured Manner?

We do not need structured management of software project requirements in all circumstances. If it is a transaction management requirement on the recording and status tracking during implementation, which does not need continuous tracking and requirement association maintenance afterwards, structured requirement management is not required. In this case, both the Scrum and Kanban project templates of CodeArts can be used to manage requirements and software projects. Structured requirement management is required only when there are a large number of interrelated requirements and implemented requirements need to be managed and maintained. In this case, we need to use the Scrum project template in CodeArts, whose "Epic-Feature-Story" structure and requirement planning function are helpful for structured management.

What Is Structured Requirement Management Based on?

Software R&D is either project-based or product-based. In general, the former is temporary or short-term, while the latter is long-term or requires continuous maintenance and update of functions and features. We can achieve project objectives and deliver the required functions and features by continuously improving the software product or system. This means that our structured requirement management needs to be based on product or system functions and features. Information such as versions, customers, and modules that software project management focuses on can be implemented through different attributes or even tags of requirements.

Using CodeArts for Structured Requirement Management

The recommended procedure is as follows:

- **Step 1: Create a CodeArts project.**

  Create a CodeArts project for a product or system. All requirements of the product or system are managed in the CodeArts project.
- **Step 2: Establish an "Epic-Feature-Story" requirement structure.**
  a. Epic needs to carry business value, that is, an epic needs to be meaningful to the enterprise itself.

     Service modules of a product or system, such as the user center, shopping cart, and delivery management, can be an epic. For example, it is suitable for a cloud-based freight company to take the fuel card service as an epic so that functions related to the fuel card can be used as features.
  b. Feature needs to bring benefits to users. That is, users can understand and recognize the value of a feature. Generally, a feature can be directly perceived and operated by users.

     Features are what service modules are broken down into. Simply speaking, a feature is a service or user process. If the user center is taken as an epic, user information, order management, or address management can be a feature. If the fuel card service is taken as an epic, purchasing a fuel card or managing fuel cards can be a feature.
  c. A feature is usually large and complex. Therefore, it needs to be broken down to small-granularity stories to carry a specific user operation.

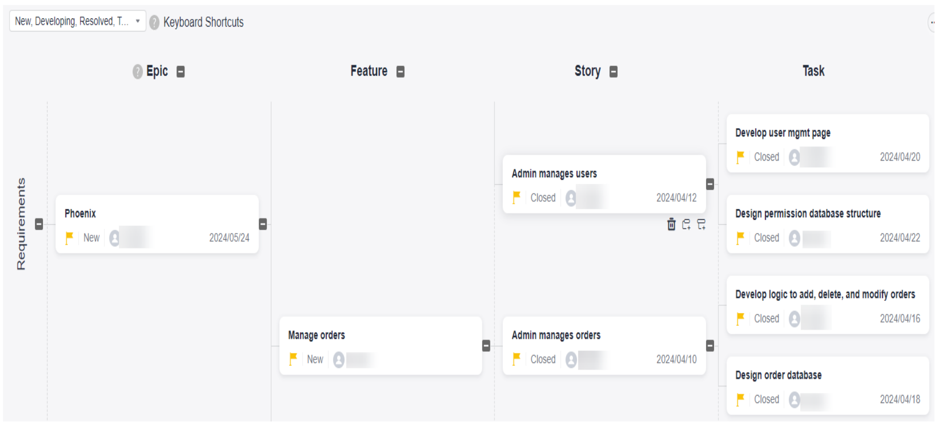
     For example, viewing all orders, filtering orders, changing user aliases, and customize profile pictures are stories.
  d. Tasks, at one level lower than stories, are mainly used for work division and collaboration. That is, stories that can be assigned to specific people do not need to be broken down.

     Tasks are specific work that engineers need to do, which are irrelevant to the business value, user value, and single operations of users. Generally, stories are broken down into tasks based on specific components and modules (such as front end, background, and database) or work processes (such as UCD, development, test, and deployment).

  The following figure is an example:
  - Epic: user center
  - Feature: address management

- Story: address creation
- Task: [Web client] page entry and address editing form or [Database] user address data table design and implementation

Figure 1-9 Requirement planning

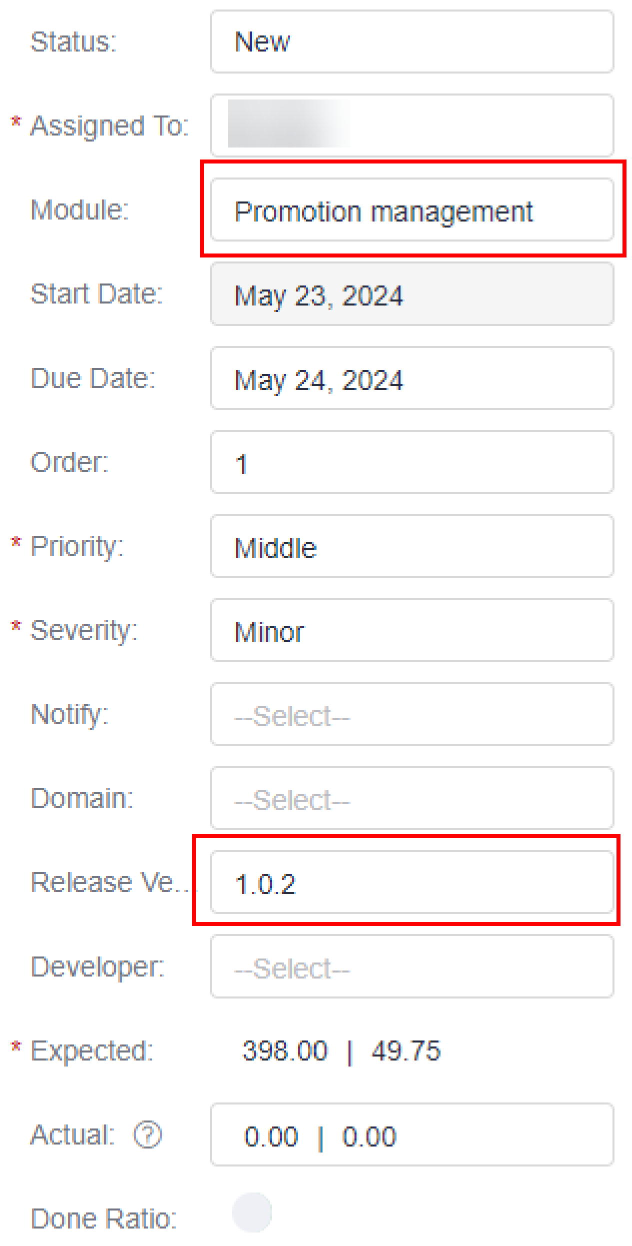


- **Step 3: Manage different modules and versions based on work item attributes.**

  Figure 1-10 shows the attributes on the work item details page.

  - **Module**: web client.
  - **Release Version**: 1.0.1.2.

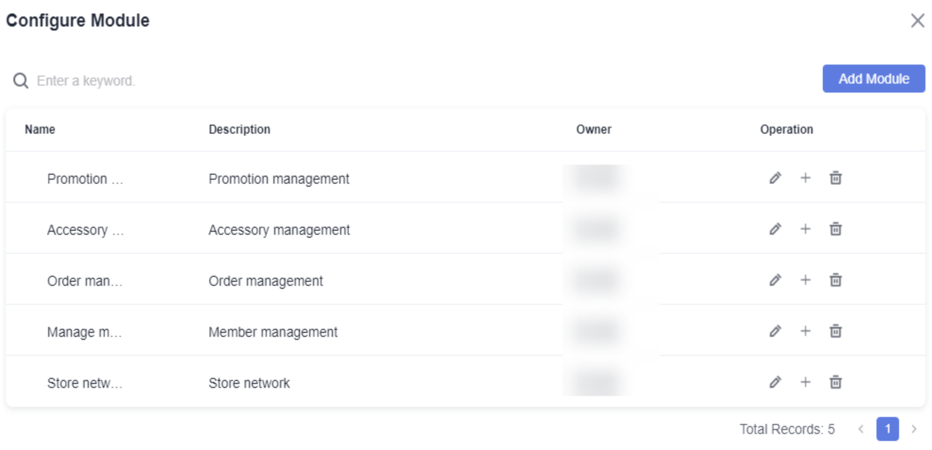
Figure 1-10 Attribute example



Status:	New
* Assigned To:	
Module:	Promotion management
Start Date:	May 23, 2024
Due Date:	May 24, 2024
Order:	1
* Priority:	Middle
* Severity:	Minor
Notify:	--Select--
Domain:	--Select--
Release Ve...	1.0.2
Developer:	--Select--
* Expected:	398.00 | 49.75
Actual: ?	0.00 | 0.00
Done Ratio:	

To maintain a module list, click ⚙ next to the **Module** field on the editing page of the work item. In the displayed window, you can add, modify, or delete a module.

Figure 1-11 Editing a module



Name	Description	Owner	Operation
Promotion ...	Promotion management		
Accessory ...	Accessory management		
Order man...	Order management		
Manage m...	Member management		
Store netw...	Store network		

Total Records: 5

In the backlog view of work item management, after adding the **Module** field in the **Select Field** window, you can easily view the modules related to the work item and filter the modules.

1.2.4 Managing Requirement Priorities

Four-Step Management

Requirement priority management helps determine the sequence of requirement fulfillment, thereby maximizing the return and minimizing the risk or investment. To properly manage priorities, we need to:

1. Determine the priority model. Seemingly simple, priority is actually a value based on comprehensive judgment of multiple factors. These factors and judgment principles constitute the priority model.
2. Prioritize requirements. Calculate requirement priorities using the priority model.
3. Adjust requirement priorities.
4. Improve the priority model. If requirement priorities are adjusted frequently, analyze and improve the priority model as needed. In cases that the requirement has been delivered or released, but its actual usage or value of the function does not meet the expectation, review the requirement analysis to identify whether the analysis or the priority model is incorrect, and make corresponding adjustments.

Determining the Priority Model

Cost-benefit analysis (CBA) is the simplest priority model. The Eisenhower Matrix and Kano model are also priority models. Simple or complex, all these models can help determine requirement priorities. Select one as needed.

Note that the priority model should not be too complex at the beginning. Otherwise, high costs of requirement management exert an adverse impact on requirement development and delivery. A simple, useful model should always be preferred. It is recommended that enterprises start with a simple model and improve the model continuously.

- A simple model means a few factors to be considered. For example, CBA contains only two factors, which is simpler than other models with more factors.
- A simple model also means a narrower value range or a lower precision requirement. For example, it is simpler to predict the profit as high, medium, or low than in the unit of CNY10,000.

Archive the selected model and make it accessible to all or related personnel. For example, it is a good practice to record the model in CodeArts Wiki.

Prioritizing Requirements

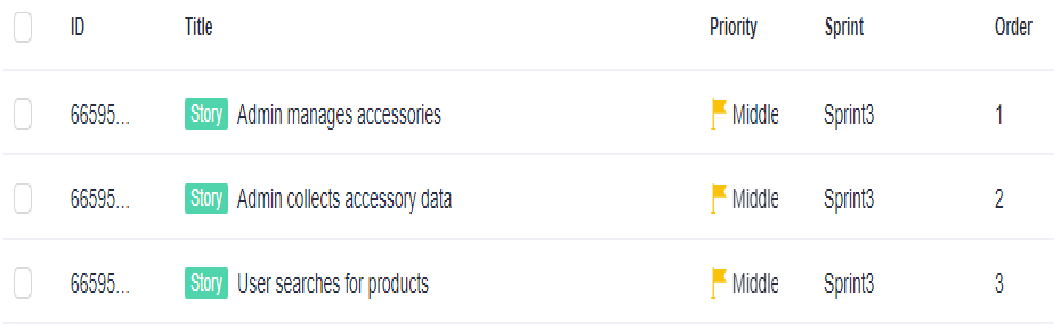
In CBA, the expected market revenue and R&D investment are used as the benefit and cost respectively. Calculate the difference or ROI. For example, there are two requirements to be prioritized. For requirement A, the estimated profit and R&D investment are CNY100,000 and CNY30,000 RMB respectively, so the estimated profit is CNY70,000 and ROI is 233%. For requirement B, the estimated profit and R&D investment are CNY50,000 and CNY40,000 RMB respectively, so the estimated profit is CNY10,000 and ROI is 25%. It is easy to see the priority of requirement A higher than that of requirement B. Assume that there are another two requirements. For requirement C, the estimated profit and ROI are CNY70,000 RMB and 50%, and for requirement D, the estimated profit and ROI are CNY10,000 RMB and 500%. How can we determine the priorities of these four requirements?

In this case, we need to introduce weights to calculate a comprehensive value, as shown in the following table:

Requirement	Estimated Revenue (CNY10,000)	Estimated Cost (CNY10,000)	Estimated Profit (CNY10,000)	Profit Weight	Profit Weighted Value	ROI	ROI Weight	ROI Weighted Value	Comprehensive Value	Priority Sequence
Requirement A	10	3	7	0.1	0.7	233%	1	2.33	3.03	2
Requirement B	5	4	1	0.1	0.1	25%	1	0.25	0.35	4
Requirement C	21	14	7	0.1	0.7	50%	1	0.5	1.2	3
Requirement D	2	1	1	0.1	0.1	500%	1	5.0	5.1	1

According to the results in the preceding table, we need to add requirements D, A, C, and B to the development plan in sequence. In CodeArts, the **Order** field (1–100) can be used to display priority.

Figure 1-12 Story priorities



ID	Title	Priority	Sprint	Order
66595...	Story Admin manages accessories	Middle	Sprint3	1
66595...	Story Admin collects accessory data	Middle	Sprint3	2
66595...	Story User searches for products	Middle	Sprint3	3

Adjusting Requirement Priorities

Adjusting requirement priorities is simple. Just reset the value of the **Order** field in CodeArts. More importantly, however, you need to record the priority adjustment in CodeArts Wiki, including why the adjustment is required, how the adjustment is performed, and any specific considerations. It can be used as a reference in retrospective meetings after each sprint for review.

Improving the Priority Model

The priority model must change to the evolving market, users, and products. We can review and analyze the requirement priority model to find out what can be enhanced. The review can be conducted regularly (for example, once a month) to go over all the requirements involved, or requirements with adjusted priorities or problems. Also, the review can be conducted irregularly and problem-driven. For example, if a large number of requirement priorities are adjusted in a day, a review meeting can be scheduled for that day or the next day to analyze the cause.

To achieve a good review effect, we must restore the situation when the problem occurs as much as possible. Therefore, the records in CodeArts Wiki are very important. The retrospective meeting should provide as much information as possible to participants for a better discussion.

During the review, we need to identify the root cause. We need to find out whether the model design (such as factors and dimensions) is incorrect, whether the value or weight is incorrect (perhaps estimating the revenue for a specific requirement is difficult), or whether process management is improper (such as incorrect priorities due to premature estimation without sufficient information), and make targeted improvements.

1.2.5 Avoiding Missing Important Requirements

Guidelines

We need to find out why these urgent and important requirements cannot be predicted earlier. Also, we need to know:

- What are the external causes? Check whether these causes have something in common and take corresponding solutions.
- Is there anything in common of the added requirements? If yes, handle them accordingly.
- How temporary is the addition? If we can adjust and respond more quickly so that the temporary requirements do not affect the team, this problem is no longer a problem.
- Why do not we adjust the process or work style? Is it unuseful, or do not we know how to adjust the process or work style?

Procedure

We can take measures based on different time points, that is, before, during, and after the event.

1. **In-event handling**

   Take the following measures as required:

   – For important, non-urgent requirements, add them as usual. If they are frequently missed, consider whether the practice of requirement analysis and planning is correct. If yes, strengthen structured requirement management and make a global plan.

   – For important, urgent requirements, adjust the current development sequence and prioritize these requirements. Then, conduct a review on two aspects: requirement priority and structured requirement management, and make improvements to avoid similar situations.

   – For unimportant requirements, handle them as usual. Determine whether to prioritize them based on their urgency and impact. If this situation occurs repeatedly, it is recommended to perform an analysis on structured requirement management and discuss improvement solutions.

2. **Post-event handling**

   Post-event handling is actually a review. The key to the review is to perform deduction and analysis based on the formulated model and specifications. We need to find out whether there is a problem with the implementation, the model is too simple to cover the special requirements, or the judgment deviation is huge due to inadequate skills. Only when the correct root cause is found can the problem be solved. Therefore, we need to conduct a careful review.

   So how do we conduct a review? There are books in the industry for our reference. For example, the MOI model proposed by Gerald Weinberg in *Becoming a Technical Leader* is a good review approach.

   – M: Motivation. Are people not motivated to do it?

   – O: Organization. Is it so disorganized or undisciplined that people do not know what they should do?

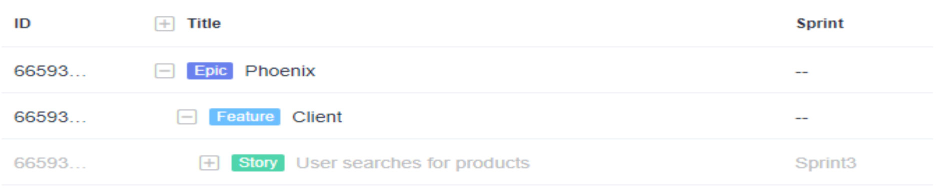
– I: Idea/Innovation. Is there a lack of ideas or ideas on how to solve these problems?

Due to limited capabilities, experience, and number of problems, we may not be able to obtain an accurate conclusion or an effective solution. On the one hand, we can take measures that have been clearly defined and observe the effect for continuous improvement. On the other hand, we can take some temporary measures.

a. Reserve time: If it is difficult to analyze why requirements are always missed and cannot be handled in a targeted manner, we might as well adopt a vague approach. For example, we can obtain the work records of a past period of time, evaluate the workload consumed by the unexpected requirements of each sprint in this period, and calculate the average value. Reserve a certain amount of time based on the value for unexpected requirements during subsequent sprint arrangement.
b. Break down requirements: When we need to adjust the work sequence due to unexpected requirements, it is very likely that the impacts cannot be avoided due to the large granularity of requirements. Therefore, we should break down the requirements as much as possible, making it more flexible to adjust the requirement work sequence.

To calculate the reservation time, use the "Epic-Feature-Story" structure of CodeArts to aggregate unexpected requirements. For example, create a special epic named "Unexpected Requirements" and features to carry the unexpected requirements (shown as stories) in each sprint. Record the workload. After the sprint, calculate the number of unexpected requirements and the workload.

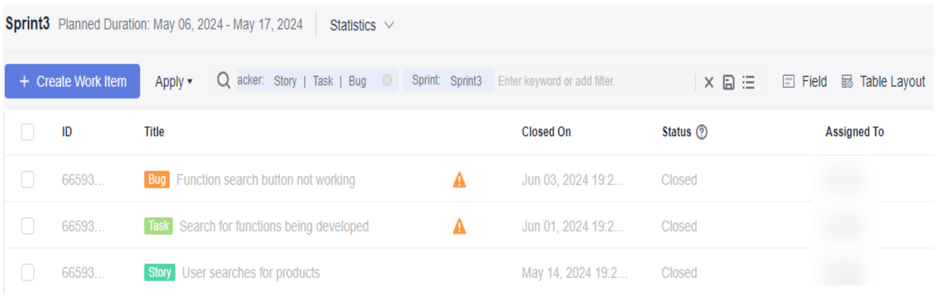
Figure 1-13 Sprint planning



ID	Title	Sprint
66593...	Epic Phoenix	--
66593...	Feature Client	--
66593...	Story User searches for products	Sprint3

The **Module** field can also be used to record and collect statistics on unexpected requirements. For example, create a module named "Unexpected Requirements" to mark all unexpected requirements, and filter or view reports based on the module to calculate the workload of unexpected requirements.

Figure 1-14 Sprint content



Sprint3 Planned Duration: May 06, 2024 - May 17, 2024 Statistics

acker: Story | Task | Bug Sprint: Sprint3

ID	Title	Closed On	Status	Assigned To
66593...	Bug Function search button not working	Jun 03, 2024 19:2...	Closed	
66593...	Task Search for functions being developed	Jun 01, 2024 19:2...	Closed	
66593...	Story User searches for products	May 14, 2024 19:2...	Closed	

3. **Pre-event handling**

After the in-event handling and post-event review, we need to think of preventive measures. We need to perform structured requirement management and requirement priority management, publicize related regulations, allocate personnel, and develop capabilities. In this way, the impact of unexpected requirements can be effectively avoided or reduced.

1.3 Sprint Plan

1.3.1 Properly Planning the Sprint Timebox

Background

A team of around seven members uses the Scrum framework. The sprint timebox that the team currently uses is one week. The sprint goals are not always achieved at the end of a sprint. Many work items need to be completed across sprints.

Problem Analysis

Sprint goals are not well achieved, which is currently a problem for sprints. There are several following causes:

Team members face many exploratory work items that require learning costs for those unfamiliar with the work content. As a result, it takes longer to complete work items than usual. The workload of each user story is also heavy, mostly more than 24 hours. The Product Owner (PO) has very high standards on the work item completion, and the review is strict. Unqualified work items are often redone in the sprint. The current team sprint is one week, and the four major events are based on the Scrum framework. The average duration of the sprint planning and review meetings is about 2 hours.

The main factors that affect sprint goal achievement are listed in **Table 1-1**.

Table 1-1 Factors affecting sprint goal achievement (1)

No.	Factor	Analysis
1	Many exploratory work items	Since the status quo is difficult to change, learning costs are mandatory. Team members can share knowledge. As their capabilities improve, learning costs will gradually be reduced.
2	Members unfamiliar with the work domain, requiring learning costs	
3	Large user stories	Due to the particularity of work items, user stories are generally large. Split them into smaller stories or create tasks under each user story. Achieve sprint goals at the task level.

No.	Factor	Analysis
4	High completion standards by PO	Further understand and clarify the completion standard by PO, including Acceptance Criteria (AC) and Definition of Done (DoD), in the sprint planning.
5	Short sprint	The team sprint is initially set to one week. If the goal is not achieved as expected, adjust the sprint.
6	Prolonged sprint planning and retrospective meetings	Generally, the sprint planning of a one-month sprint lasts eight hours. Therefore, it is recommended that the planning meeting of a one-week sprint last two hours. Similarly, the retrospective meeting of a one-month sprint normally does not exceed three hours. Therefore, it is a severe timeout for a retrospective meeting of a one-week sprint to take two hours.

Solution

On the one hand, a short sprint leaves little time for work item completion, with team member busy preparing for planning, review, and retrospective meetings. It is difficult to develop emergency handling capabilities and build a stable team. On the other hand, a long sprint goes against the concept of timebox. Therefore, if the team often fails to achieve the sprint goal in a one-week sprint, try to extend the timebox to two weeks. At the same time, adjust the size of user stories to improve the efficiency of the four events.

The sprint timebox extension has many favorable influences, as shown in **Table 1-2**.

Table 1-2 Factors affecting sprint goal achievement (2)

No.	Factor	Influence of Sprint Extension From One to Two Weeks
1	Many exploratory work items	There is sufficient time reserved for learning, exploratory work, and emergencies.
2	Members unfamiliar with the work domain, requiring learning costs	
3	Large user stories	The planning meeting is more comprehensive and detailed. User stories are broken down into smaller ones, and tasks are created under each user story. Sprint goals are achieved at the task level.

No.	Factor	Influence of Sprint Extension From One to Two Weeks
4	High completion standards by PO	The completion standard by PO, including AC and DoD, are fully understood and clarified in the planning meeting.
5	Short sprint	Team members feel more confident in an empowering atmosphere, with an increase in the sprint goal completion rate.
6	Prolonged sprint planning and retrospective meetings	Meetings are held more efficiently.

Sprint timebox recommendations:

- One to two weeks

  R&D teams that are committed to time-sensitive products and flexible services require prompt adjustment and regular self-check.
- Two weeks

  Teams with a relatively stable pace and large stories, which require more time for review and rectification.
- Three to four weeks

  Teams with a stable pace and requirements.

More about Timebox

In Scrum, the development process is known as a sprint, which is an iteration or cycle usually lasting a month. Sprint is set in a timebox, that is, the sprint has fixed start and end time.

- **Timebox advantages:**
  a. Timeboxing is used to control the number of works in progress (WIPs). WIP is a list of work items that have been started but are not yet completed. The development team only focuses on work items that they think can be completed within a sprint, so timeboxing restricts the number of WIPs for each sprint.
  b. Timeboxing requires the team to prioritize the tasks so that they can dedicate their time to quickly completing high-value work.
  c. Timeboxing displays the progress by showing how many timeboxes are required to finish the development of a complex feature. This also allows a team to learn exactly how much remains to be done to deliver the entire feature.
  d. Timeboxing has an end date to prevent endless work.
  e. A sprint has to be finished within a timebox, which means the deadline is definite and cannot be changed. This can motivate a Scrum team to go all out to complete the work of a sprint on time. Without this time pressure, the Scrum team would not have a sense of urgency.

f. Timeboxing can improve predictability. The team can predict the amount of work that can be completed in the next sprint.

- A short sprint duration features easier planning, quick feedback, limited errors, and high return on investment (ROI). It helps maintain the enthusiasm of team members and provides more effective checkpoints. The following describes more details about its benefits:

  a. It is easier to plan the workload for a short work period than a long one. Plans are also more accurate and executable.
  b. Fast feedback can help abandon inappropriate product paths and development methods to avoid more cost of poor quality (COPQ). Most importantly, it allows teams to quickly identify and leverage business opportunities that can easily slip by.
  c. Mistakes made during one to two weeks are limited. Even though an entire sprint is a failure, the incurred loss is acceptable. Repeated short-term sprints are good chances for frequent trial-and-error, feedback, and adjustments.
  d. Shorter duration means earlier and more frequent delivery and more opportunities for quick production to generate more ROI.
  e. Short duration helps maintain a high level of enthusiasm. The motivations and interest of a team decline over time. If a project lasts for too long without presenting any results, team members will gradually lose interest. The frequent and quick delivery of runnable products can fill team members with a sense of satisfaction and fuel them to keep working till reaching the sprint goal.
  f. In traditional waterfall development, milestones such as analysis, design, coding, testing, and operation are often less accurate metrics. Scrum provides more meaningful checkpoints (sprint reviews) at the end of each sprint, where team members make judgments and decisions based on the workable features that are demonstrated. With more checkpoints to check and correct, teams can better cope with complex projects.

1.3.2 Moving Task Cards in the Kanban After Requirements Change in Sprints

Background

This section describes how to move task cards in the Kanban in normal cases and after requirements change in sprints.

Moving Task Cards in Normal Cases

One of the main purposes of using the Kanban is to control the number of WIPs. Pull movement is required to effectively control the number of WIPs and prevent overstock of work items.

Note that a set of moving rules should be customized before the move. The rules are adjusted to satisfy the team.

Moving Task Cards After Requirements Change

- **Rejecting the change**

  To keep the team going all out for the sprint goal after the sprint backlog and goal are confirmed, requirement changes proposed by the PO are generally not accepted. The PO is responsible for sorting out the sprint backlog. To be specific, the PO should prepare the product backlogs for the next one to three sprints and then develop the sprint backlog based on priority. Therefore, it is recommended that the team reject frequent requirement changes. Requirement rejection also strengthens the PO's control over requirements. Therefore, it is better for the team to move cards in the Kanban normally in this case.

- **Embracing the change**

  It is unrealistic to completely reject requirement changes. Sometimes, high-priority requirements are subject to changes. For example, if time-sensitive requirements cannot be completed in this sprint, we will fail to get in on the ground floor. However, we must comply with the "NO CHANGE" principle. That is, after receiving a requirement change, analyze the requirement and the impact on the current sprint instead of directly accepting or rejecting the change. In this case, check the following items:

  - Requirements with no value

    Reject the requirements with no value and do not move cards on the Kanban after negotiating with the PO. Here we will spare you the details about where these requirements come from.

  - Requirements with few changes and little impact

    Accept the change of high-priority requirements with little impact on the sprint, but evaluate workload and perform an exchange. To put it simply, replace low-priority requirements that have not been implemented from the Kanban and move them to the product backlog. This is a process of product backlog refinement. Then, add cards of high-priority requirements to the Kanban. When replacing requirements that have been partially done, move them back to the **Product Backlog** column if we seek to meet feature requirements, or move them to the **Done** column if the team needs to collect statistics on the workloads on the physical Kanban. In the second situation, if the electronic Kanban also works, move the requirements to the **Product Backlog** column.

  - Requirements with many changes and great impact

    For changes in high-priority requirements that have great impact on the sprint, stop the current sprint and plan a new sprint. For example, if the requirement in the current sprint is of no value or the changed requirement requires a large amount of workload, you should stop the sprint. Then arrange the card on the Kanban based on the latest sprint backlog.

1.4 Sprint Development

1.4.1 Managing Unexpected Tasks in Software Development Teams

Background

How does the development team manage unexpected tasks?

Some software development teams often do not know how to manage unexpected tasks, such as providing training support.

Unexpected tasks impact the progress and efficiency of the development team, making a difference on whether the sprint goal can be achieved and even the success of the entire project. Therefore, it is important for the team to reduce or resolve unexpected tasks.

Problem Analysis

We can classify the scenarios into the following two types.

Scenario	Analysis
Mixing support and development tasks, with unexpected tasks occurring as a norm	Support tasks are mixed with regular development tasks, causing developers to often switch work content. This makes management more difficult and wastes time as developers have to constantly switch tasks back and forth. Therefore, the root cause is the development team mode. In this mode, problems and risks need to be handled as they arise.
Primary development work, with unexpected emergencies or normal requirement changes	Development team members lack clear and transparent rules on handling emergencies, leading to unpreparedness. The root cause is backlog management. To be specific, the team needs to figure out how to update the backlog for unexpected work items, that is, how to deal with requirement changes and temporary task addition.

- **In the first scenario**, development and support tasks need to be handled simultaneously without clear priorities, complicating task management.
- **In the second scenario**, customers often submit unexpected, urgent tasks to the development team. The extra workload adds to the risk of failing to achieve the sprint goal.

The development team needs to find a way to handle trivial, unexpected support work and improve work efficiency.

Solution

More and more development teams are adopting Agile, from which CodeArts is inspired. The following solutions, including Scrum, are based on Agile:

- For the first scenario, divide developers into two groups, one for development and the other for emergencies. The development group focuses on development while the other group stands ready to unexpected work (also responsible for easy, loosely coupled development). Both groups shall comply with rules of the development team to manage work items.
- The second scenario is more common. The following describes how to deal with work priority and requirement changes systematically.

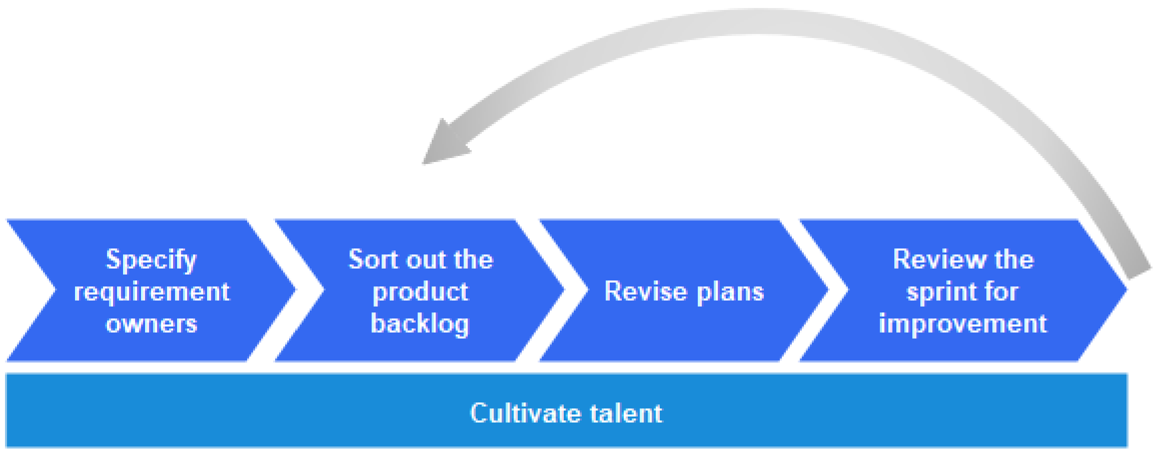
Solutions for the second scenario:

Managing work items is essential for the development team. Therefore, before forming work items, we need to determine who is responsible for work items (that is, the source of work items), and make a plan for work item implementation.

1. Specify the product manager as the unique requirement source.
2. Sort out the product backlog and prioritize work items.
3. Revise the plan based on the capabilities of the development team and update the sprint goal.
4. Review and find where enhancements are needed.
5. Analyze sprints for continuous improvement and cultivate talent to build a cross-functional team.

The following diagram displays the solution.

Figure 1-15 Solution diagram


Specify requirement owners
Sort out the product backlog
Revise plans
Review the sprint for improvement
Cultivate talent

1. **Specify the requirement owner.**

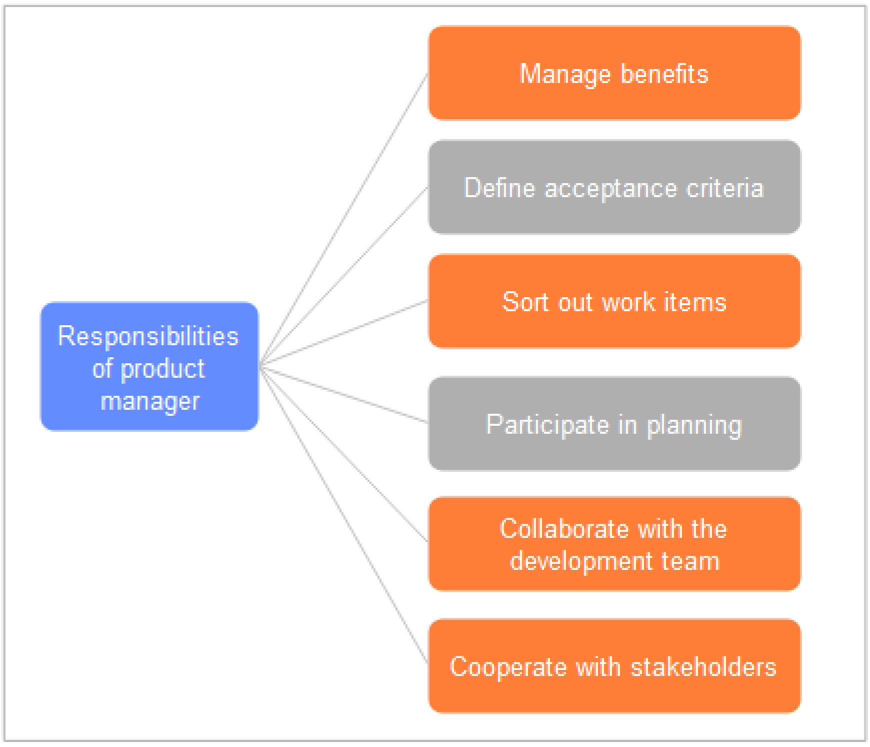
   Specify the requirement owner as the unique requirement source. In CodeArts, the **project manager** is the requirement owner. The requirement owner needs to:

   - Have a good understanding of the requirements of stakeholders, customers, and users in the project (including the unexpected work items mentioned above) and their priorities. From this perspective, the product manager acts as the requirement owner.
   - Communicate with the development team about the features to be developed and the development sequence. The requirement owner also must ensure that there are clear feature acceptance criteria. From this perspective, service analysts and testers act as the requirement owner.

Moreover, the requirement owner must be able to make economical decisions for more benefits at the version, sprint, and backlog levels. In the following, the product manager (like product owner in Scrum) is used for requirement owner.

The following figure shows the main responsibilities of a product manager.

Figure 1-16 Main responsibilities of the product manager


Responsibilities of product manager
Manage benefits
Define acceptance criteria
Sort out work items
Participate in planning
Collaborate with the development team
Cooperate with stakeholders

The highlighted responsibilities in **Figure 1-16** are related to the unexpected work in a sprint. The product manager needs to clearly communicate with stakeholders to determine the priority of each unexpected task and whether it must be completed in this sprint based on business value and management benefits. Once it is determined that these unexpected tasks are of high priority and must be completed in this sprint, the backlog needs to be sorted out.

2. **Sort out the backlog.**

   Sorting out the backlog means placing high-priority tasks in the sprint backlog. For details about the sprint backlog, see **More Details About Sprint Backlogs**.

   – **Backlog** is a prioritized list of expected product functions.
   – **Work items** are the to-dos in a backlog. For users and customers, most work items are valuable features and functions. Work items also include defect rectification, technical improvement, and knowledge acquisition, and any work that the product owner considers valuable (even unexpected valuable work).
   – **Sorting** refers to three important activities: establishing and refining, evaluating, and prioritizing work items.

Change the backlog structure by sorting out activities.

Figure 1-17 Backlog structure

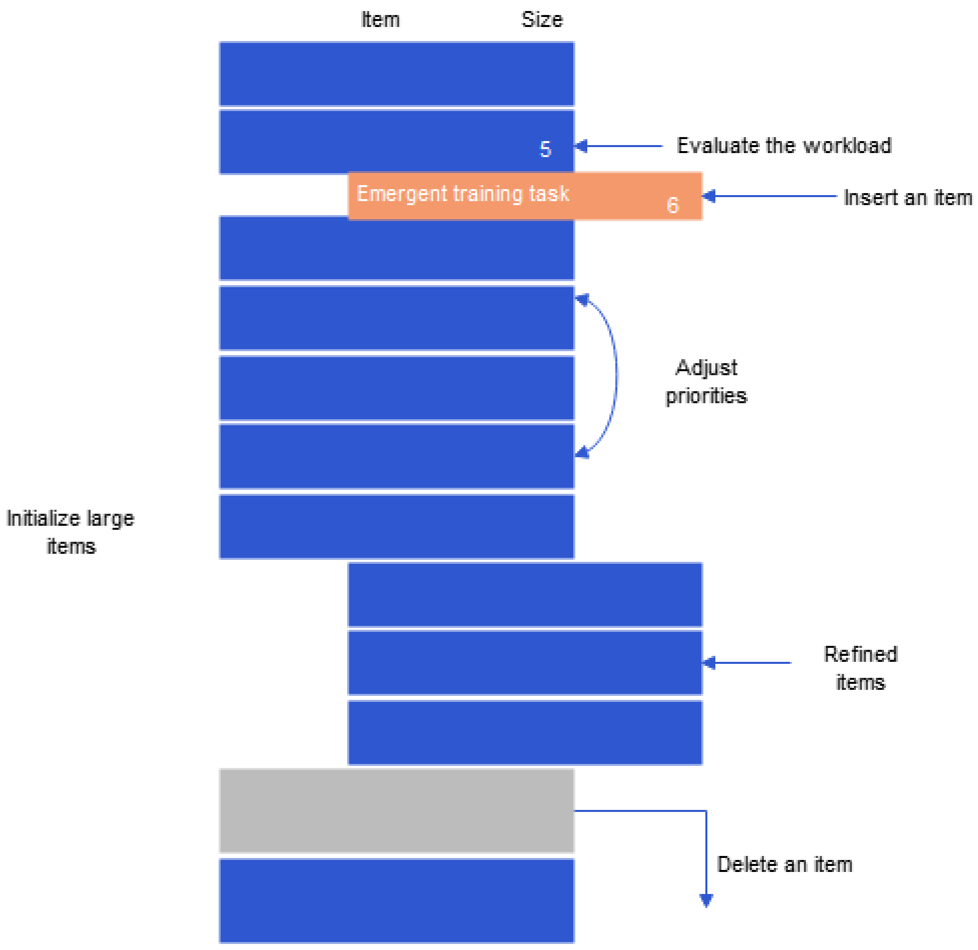
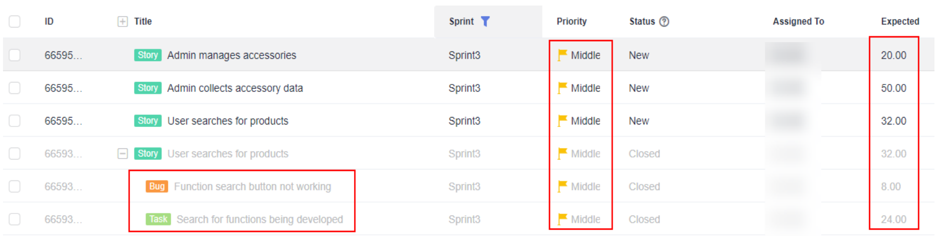


Figure 1-18 shows the sorting result in CodeArts.

Figure 1-18 Requirement sorting



ID	Title	Sprint	Priority	Status	Assigned To	Expected
66595...	Story Admin manages accessories	Sprint3	Middle	New	[redacted]	20.00
66595...	Story Admin collects accessory data	Sprint3	Middle	New	[redacted]	50.00
66595...	Story User searches for products	Sprint3	Middle	New	[redacted]	32.00
66593...	Story User searches for products	Sprint3	Middle	Closed	[redacted]	32.00
66593...	Bug Function search button not working	Sprint3	Middle	Closed	[redacted]	8.00
66593...	Task Search for functions being developed	Sprint3	Middle	Closed	[redacted]	24.00

The red boxes in **Figure 1-18** show the sorting by the product manager. The unexpected training task is properly prioritized and the workload is evaluated. In addition, the task of querying product names is broken down. As the final decision-maker of sorting, the product manager is responsible for sorting out unexpected work. A good product manager can coordinate stakeholders to arrange enough time for sorting based on the characteristics of the development team and project type. In addition, the development team needs to estimate the workload of unexpected work and help the product manager prioritize work items based on technology dependencies and resource constraints. If an unexpected training task has a high priority, it will be added to the backlog of the current sprint.

3. **Revise the plan.**

   Revise the plan based on the capabilities of the development team and update the sprint goal. Before revising the plan, we need to know what is a plan in Agile development and what are the differences between plans in Agile and traditional development modes.

   As the Agile Manifesto advocates responding to changes instead of always following the plan, **the Agile development focuses on responding to changes while the task-driven development tends to follow the plan**.

   Sequential development is plan-driven. The plan is the authoritative information source of how and when work is done. Therefore, the plan is always followed. In contrast, Agile prefers revising plans based on real-time information, rather than sticking to a potentially incorrect plan.

   Agile development is not about following a predetermined plan or prediction, but revising the plan based on the valuable economic information that emerges in the process. Therefore, you can adjust the plan after sorting out work items. Work items in the original plan may be moved to the next sprint. The capacity of a fixed development team remains unchanged in a sprint. Adding unexpected work items without removing others will definitely bring pressure to the team and disrupt the delivery pace. Therefore, the equivalent exchange principle shall be followed, which means that the replaced work has the same workload as the unexpected high-priority work.

   Figure 1-19 Equivalent exchange principle

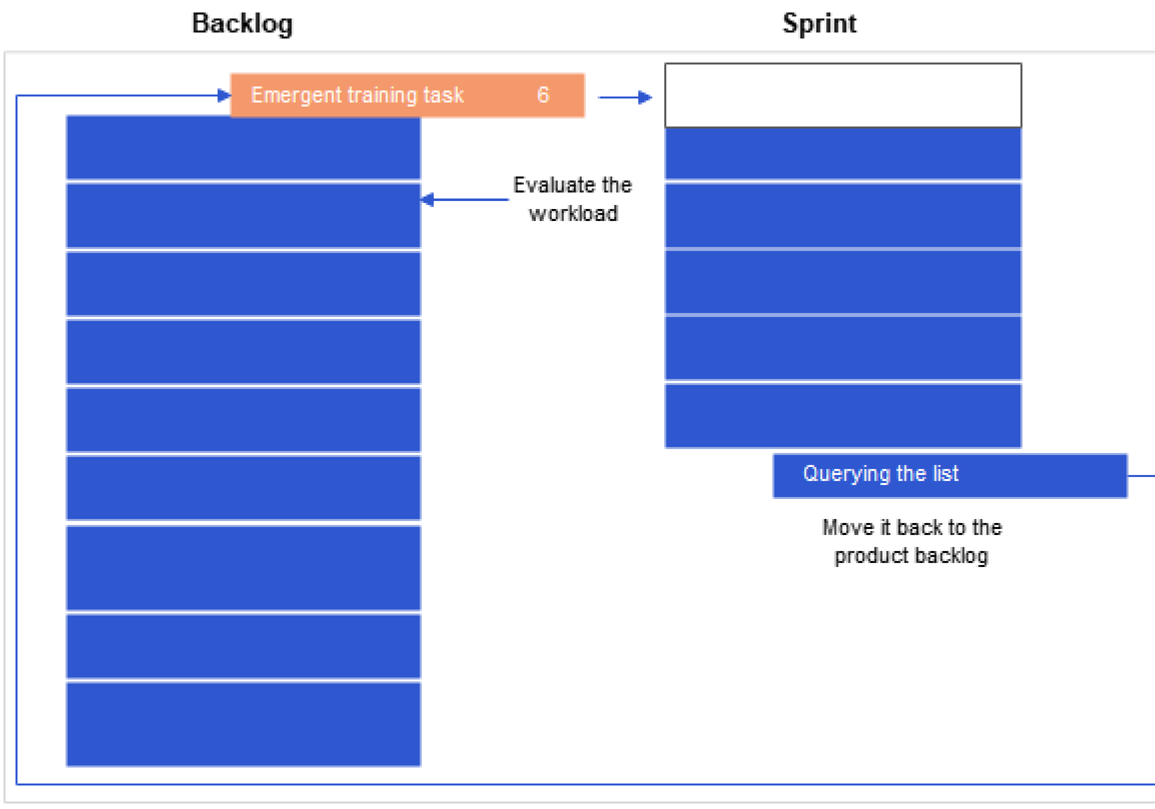
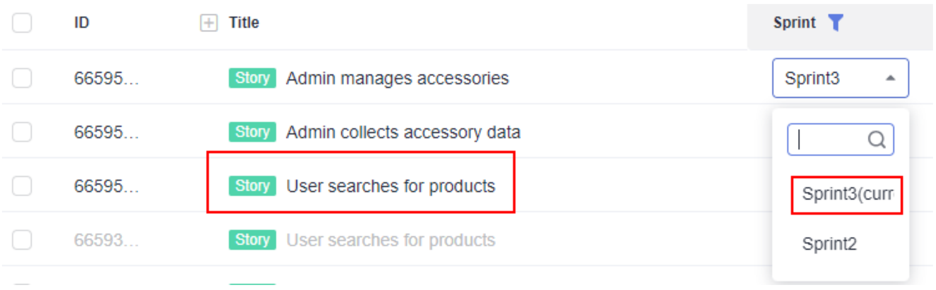
   

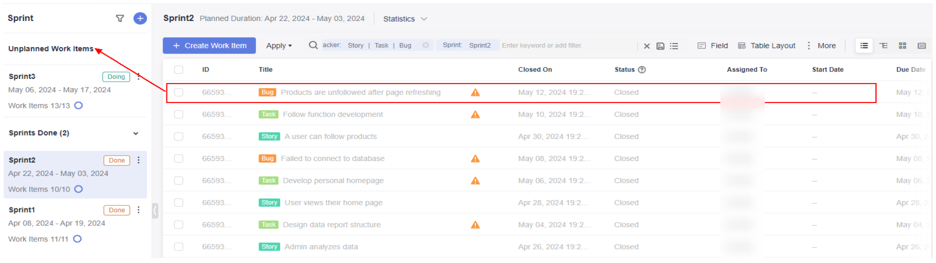
   Figure 1-20 and **Figure 1-21** show the exchange details in CodeArts.

Figure 1-20 Equivalent exchange 1



	ID	Title	Sprint
	66595...	Story Admin manages accessories	Sprint3
	66595...	Story Admin collects accessory data	
	66595...	Story User searches for products	
	66593...	Story User searches for products	

Figure 1-21 Equivalent exchange 2



	ID	Title	Closed On	Status	Assigned To	Start Date	Due Date
	66593...	Bug Products are unfollowed after page refreshing	May 12, 2024 19:2...	Closed	[redacted]	--	May 12,
	66593...	Task Follow function development	May 10, 2024 19:2...	Closed	[redacted]	--	May 10,
	66593...	Story A user can follow products	Apr 30, 2024 19:2...	Closed	[redacted]	--	Apr 30,
	66593...	Bug Failed to connect to database	May 08, 2024 19:2...	Closed	[redacted]	--	May 08,
	66593...	Task Develop personal homepage	May 06, 2024 19:2...	Closed	[redacted]	--	May 06,
	66593...	Story User views their home page	Apr 28, 2024 19:2...	Closed	[redacted]	--	Apr 28,
	66593...	Task Design data report structure	May 04, 2024 19:2...	Closed	[redacted]	--	May 04,
	66593...	Story Admin analyzes data	Apr 26, 2024 19:2...	Closed	[redacted]	--	Apr 26,

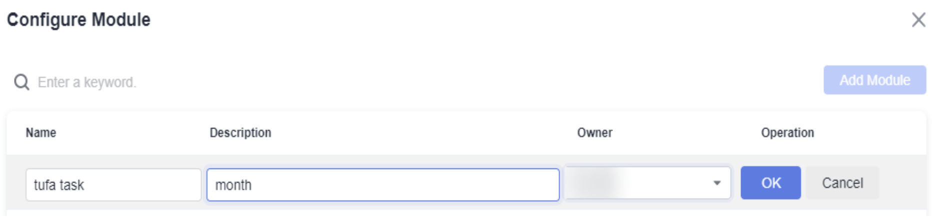
Finally, move the low-priority work item with a similar workload back to the backlog. The equivalent exchange of work items is completed in CodeArts.

4. **Review the sprint for improvement.**

   Conduct a sprint retrospective to improve the process based on the needs of the development team, thereby boosting morale for higher output efficiency. Analyze the unexpected work in sprints for continuous improvement. In Agile development, prioritizing changes over following the plan is important, but the development team needs to go out for the sprint when there are no interference (unexpected work). Therefore, we should think about why there would be interference in each sprint and find a solution.

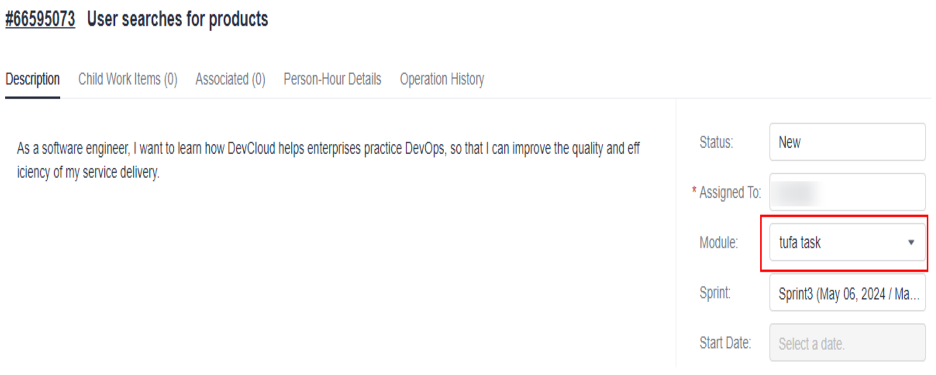
   Figure 1-22 and **Figure 1-23** are a good practice of CodeArts, which can provide objective data for review. Unexpected work items added to the sprint backlog can be managed in **Modules** of CodeArts. That is, create a module for unexpected tasks before creating a user story, and then select this module in the user story. This is to measure and analyze such unexpected work in subsequent sprints (which can be sorted by module to facilitate measurement and analysis) for continuous improvement.

Figure 1-22 Creating a module for unexpected work



Name	Description	Owner	Operation
tufa task	month	[redacted]	OK Cancel

Figure 1-23 Creating a user story



#66595073 User searches for products

Description Child Work Items (0) Associated (0) Person-Hour Details Operation History

As a software engineer, I want to learn how DevCloud helps enterprises practice DevOps, so that I can improve the quality and eff iciency of my service delivery.

Status:	New
* Assigned To:	
Module:	tufa task
Sprint:	Sprint3 (May 06, 2024 / Ma...
Start Date:	Select a date.

Choose **Statistics** > **Create Report** > **Custom Report**. On the page that is displayed, set **Analysis Dimension (X-axis/Table row values)** to **Sprint** and **Measure (Y-axis/Table cell values)** to **Actual person-hour**. Select **Module** on the page displayed after clicking **Add Filtering Criterion**. Click **Save** to analyze the statistics in diagrams and tables.

Figure 1-24 Diagram display

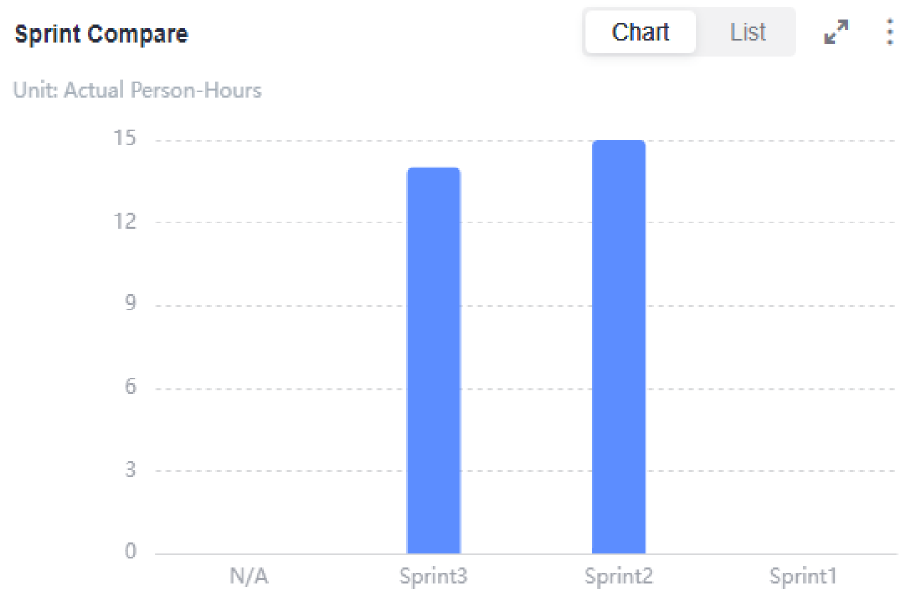
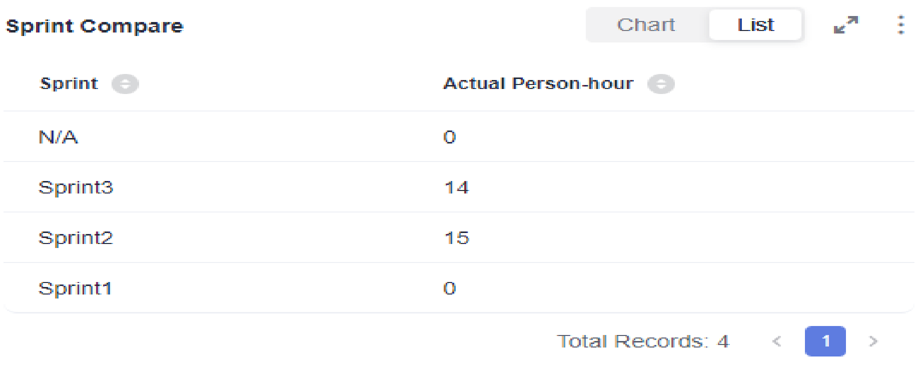


Figure 1-25 Table display



Sprint Compare

Chart List

Sprint	Actual Person-hour
N/A	0
Sprint3	14
Sprint2	15
Sprint1	0

Total Records: 4 < 1 >

5. **Cultivate talent.**

   At the same time, cultivate talent to build a cross-functional team. We not only seek to handle emergencies, but achieve higher efficiency.

   It is obvious that we should find the right person for each task (including unexpected tasks). If the person is unavailable, find another to complete the task in time. Each team member is responsible for making wise decisions when faced with uncertainties. A development team composed of T-shaped talent can always find someone to tackle tasks, achieving a higher efficiency.

More Details About Sprint Backlogs

The sprint backlog (called "sprint" in CodeArts) is a set of product backlog items selected for the Sprint. Plans to deliver the product increment and achieve the sprint goal are also included.

When a new work item emerges, the development team needs to add it to the sprint backlog. As the work is performed or completed, the remaining workload is estimated and updated. When a part of the plan loses development significance, it can be removed. During a sprint, only the development team can modify the sprint backlog. The sprint backlog shows the status of works planned by the team during the sprint in a visible and real-time way. The development team has full ownership of the backlog.

The sprint backlog is created in the sprint planning, where development team members proactively claim the tasks that suit and interest them. Task completion duration is measured in the unit of hour. It is recommended that tasks be completed within 16 hours. A task requiring 16 hours should be further broken down. Task information includes the owner, workload, committed completion time, and remaining workload on any day of the sprint. Only the development team can modify the task information.

1.4.2 How Can I Deal With Unclaimed Tasks in My R&D Team?

Background

In traditional development mode, project managers assign tasks to individuals. In agile development, however, teams claim tasks.

During transformation, many enterprises have always found some tasks unclaimed during sprint planning.

Problem Analysis

Compared with the traditional development mode, we need to know why tasks are claimed in agile development. In Agile, neither the *Agile Manifesto* nor the *Scrum Guide* has the word "assign", but uses a term "self-organizing", as follows:

- The best architectures, requirements, and designs come from self-organizing teams (12 principles of the *Agile Manifesto*)
- Self-organizing teams choose how to do the work themselves, rather than being directed by someone outside the team (*Scrum Guide*).
- The development team is self-organized. No one (not even the Scrum master) tells the development team how to turn product backlog into increments of potentially releasable functionality (*Scrum Guide*).

So what is self-organizing?

Literally, "self-organizing" is the act of organizing scattered individuals or objects to create a systematic whole. The person responsible for arranging them is the organizer themselves. In agile development, a self-organizing team is a team with self-management, self-driving, and self-learning capabilities. Such a team has the following characteristics:

- Team members proactively take on tasks instead of waiting for leaders to assign tasks.
- The team is managed as a whole.
- The team still needs coaching and guidance, but not command and control.
- Team members communicate closely with one another.
- The team actively finds and addresses problems, working together to solve them.
- The team continuously improves their skills and encourages exploration and innovation.

For more details about "self-organizing", see **Reference Documents**.

According to the *Agile Manifesto* and *Scrum Guide*, when we practice agile development, the development team takes the initiative. The team changes from being controlled to being self-organizing, and the development task shifts from being assigned to being claimed. Claiming tasks allows people to take initiative, and autonomy motivates them to be creative and solve problems. Strong self-organizing brings high performance, great results, and a positive work environment for both teams and individuals. In addition, individuals know themselves the best and are good at assigning tasks to themselves. Compared with the conventional approach of assigning development tasks, this method is more reasonable.

Then, go back to how can l deal with unclaimed tasks in the team?

Before that, it is important to clarify one thing: during the sprint planning, it is not necessary to claim all development tasks. In the *Scrum Guide*, it is noted that "tasks are claimed on demand during Sprint Planning and Sprints." A task is claimed based on your goals during the daily scrum. In addition, Mike Cohn advised against claiming development tasks during the sprint planning. This can lead to a shift in focus from team collaboration to individual work, which violates the intention of agile development and hampers flexibility.

Generally, the reasons why no one claims a development task are as follows:

- **Difficult development tasks**: If a task is too difficult for most team members to handle, they may worry about working overtime and are unwilling to claim the task.
- **Development tasks beyond scope**: When the content of a development task is beyond the scope of the team members, for example, the development members do not know how to do testing, they might feel limited in their ability to claim the task.
- **Afraid of criticizing and judging**: The tasks can be challenging. Team members are afraid to fail the task and let the whole team down.

So how should we solve this problem?

Solution

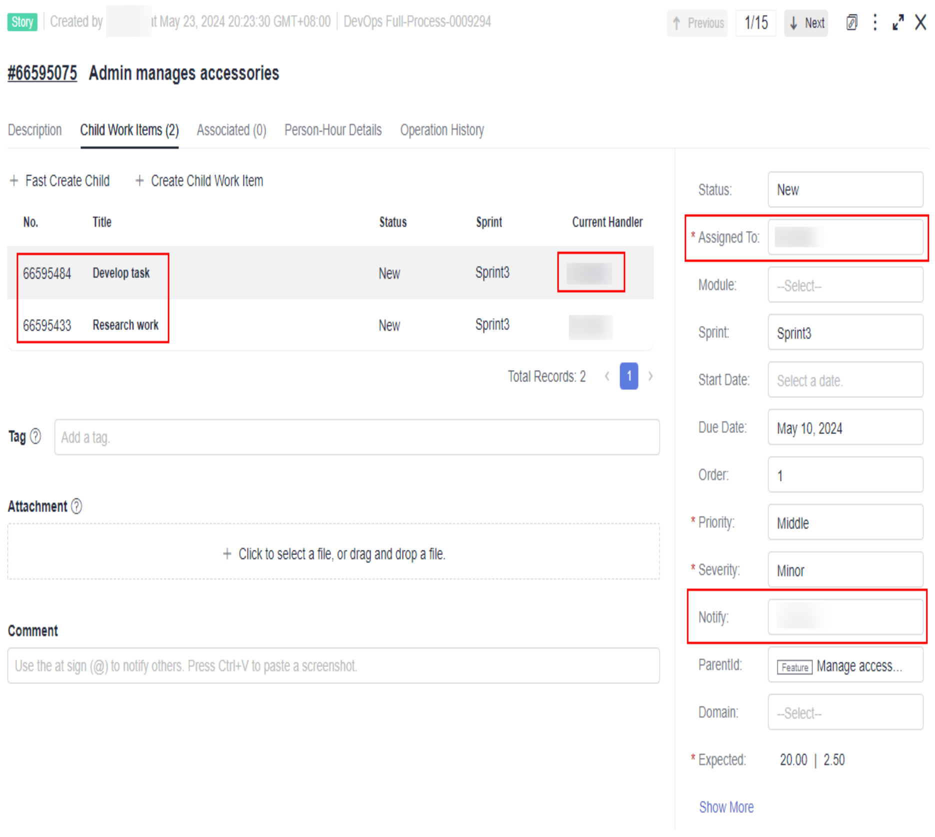
The Scrum master is crucial in an agile scrum team. One of their responsibilities is to help the team become self-organizing, enabling them to approach sprint development tasks with a positive mindset. In addition, when no one is willing to claim a task, the Scrum master should help the team figure out the reasons and then take measures accordingly. The following provides solutions based on the three situations in the analysis.

- **Difficult development tasks**

  If the development task is too difficult, the Scrum master should organize a team to effectively break down the task and use the probe Spike technology to explore solutions for making the task easier. Then a team claims the task. Or encourage members with average technical skills and expertise to work together through pair programming.

  In CodeArts, you can split difficult user stories into child work items. In addition, you can set **Assigned To** and **Notify** in the basic information to keep track of pair programming partnerships.

  Figure 1-26 Story content introduction



Story | Created by [redacted] at May 23, 2024 20:23:30 GMT+08:00 | DevOps Full-Process-0009294

Previous 1/15 Next

#66595075 Admin manages accessories

Description | Child Work Items (2) | Associated (0) | Person-Hour Details | Operation History

+ Fast Create Child + Create Child Work Item

No.	Title	Status	Sprint	Current Handler
66595484	Develop task	New	Sprint3	[redacted]
66595433	Research work	New	Sprint3	[redacted]

Total Records: 2 < 1 >

Tag: Add a tag.

Attachment: + Click to select a file, or drag and drop a file.

Comment: Use the at sign (@) to notify others. Press Ctrl+V to paste a screenshot.

Status:	New
* Assigned To:	[redacted]
Module:	--Select--
Sprint:	Sprint3
Start Date:	Select a date.
Due Date:	May 10, 2024
Order:	1
* Priority:	Middle
* Severity:	Minor
Notify:	[redacted]
ParentId:	Feature Manage access...
Domain:	--Select--
* Expected:	20.00 | 2.50

Show More

  In addition, pay attention to and update the development task during the daily scrum, assess potential risks, and help the team resolve problems promptly. In the Sprint retrospective, analyze this type of situation, output a list of team-based standard procedures, and record the solution in the team Wiki. Huawei Cloud CodeArts provides the Wiki feature to sort out and record the working methods for the team.

- **Development tasks beyond scope**

  Agile promotes cross-functional teams, but cross-functional teams do not mean that one person can do everything. The cross-functional teams here usually are T-shaped talent with multiple skills (Each team member has expertise in one area and knowledge in other areas). The Scrum master should organize and manage the member technology matrix with the team. They should also identify the areas where the team lacks expertise and facilitate knowledge sharing among team members. Regular activities like technology sharing sessions can help team members learn new skills. This encourages team members to do other tasks during the sprint. Additionally, members with specific expertise and those who are interested can collaborate as a team to do tasks using pair programming. This allows for the enhancement of individual skills and technology knowledge. By developing T-shaped capabilities among team members, they can have more choices when claiming tasks, back up the team, and reduce the occurrence of unclaimed tasks. In addition, the Scrum master also needs to focus on daily risk assessment and guide the team in reviewing tasks and updating the team Wiki.

- **Afraid of criticizing and judging**

  The tasks can be challenging. Team members are afraid to fail the task and let the whole team down. The Scrum master should express the idea of teamwork, guide and highlight Scrum values, respect the background and experience of each team member, choose development tasks, and give credits to team members who have the courage to try. In our workplace, we can put slogans (such as "respect others" and "team over individual") on walls and whiteboards. This helps shift the team's mindset and encourages them to take on more challenging tasks over time. And the Scrum master should fully support those who are willing to challenge difficult tasks. During Sprint retrospective, make sure the focus is to discuss ways to improve works rather than criticizing and judging others.

Summary

The preceding three situations where no one claims the task are common. However, every company or team faces different circumstances in their projects. Therefore, it is impossible to list all the situations. You need to analyze your actual situation. For example, an agile team that has just transformed may prefer semi-assignment and semi-claiming for their development tasks. This is like China's economy aiming to be market-oriented with proper macro control. No matter which strategy we choose, team members should be motivated to actively claim tasks, arrange them as necessary, and promptly manage any risks. If needed, we may need to seek support from other sources or leaders to ensure the iteration goal is not affected.

1.5 Agile Review

1.5.1 How to Have Daily Scrums

Background

Some project teams of the enterprise meet daily, yet the outcome is not up to par. Daily scrums are much of a formality. Usually after a daily scrum, members go back to their individual tasks. They care only about their own work, and other team members do not understand their work at all. It seems that the daily scrum makes no difference.

How can we have a proper daily scrum? What is the point of it? Do we have to do it? Many teams have been struggling with these questions.

Problem Analysis

There are two scenarios for daily scrum questions.

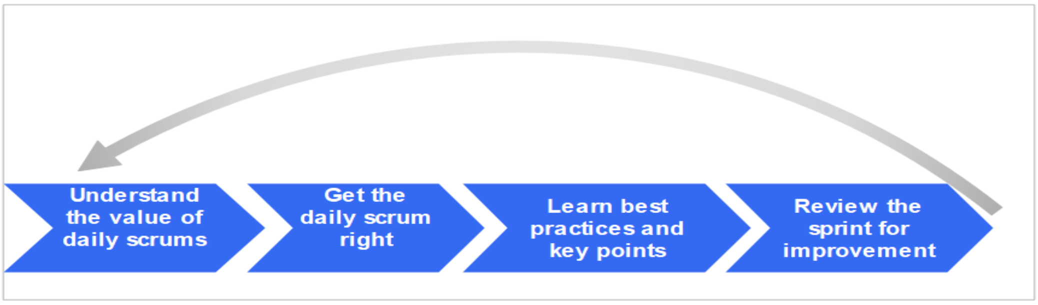
- **Scenario 1:** The team understands the importance of the daily scrum but is unhappy with its current status. They want to make the most of the daily scrum. Teams like these struggle because they are not very clear about the core value of daily scrums and how to effectively conduct them. To have better daily scrums, these teams need some specific measures.
- **Scenario 2**: The team is trying to hold a daily scrum and does not know its value. It seems that there is no difference between having a daily scrum and not having one. In this case, the team has no understanding of daily scrums, not to mention any best practices and benefits.

To sum up, a deeper understanding of the value of the daily scrum, that is, why it is necessary and what it signifies, is needed in both scenarios. Then, you need to know how to have a good daily scrum. Finally, some best practices and key points are needed to help teams with a good daily scrum.

Solution

How can we have a good daily scrum? Take the following steps.

Figure 1-27 Daily scrum flowchart


Understand the value of daily scrums
Get the daily scrum right
Learn best practices and key points
Review the sprint for improvement

1. **Understanding the value of daily scrums**

   Daily scrum is a short meeting held on each day by a team that remains standing. It reviews daily work and make adjustment, and self-organizing in advance.

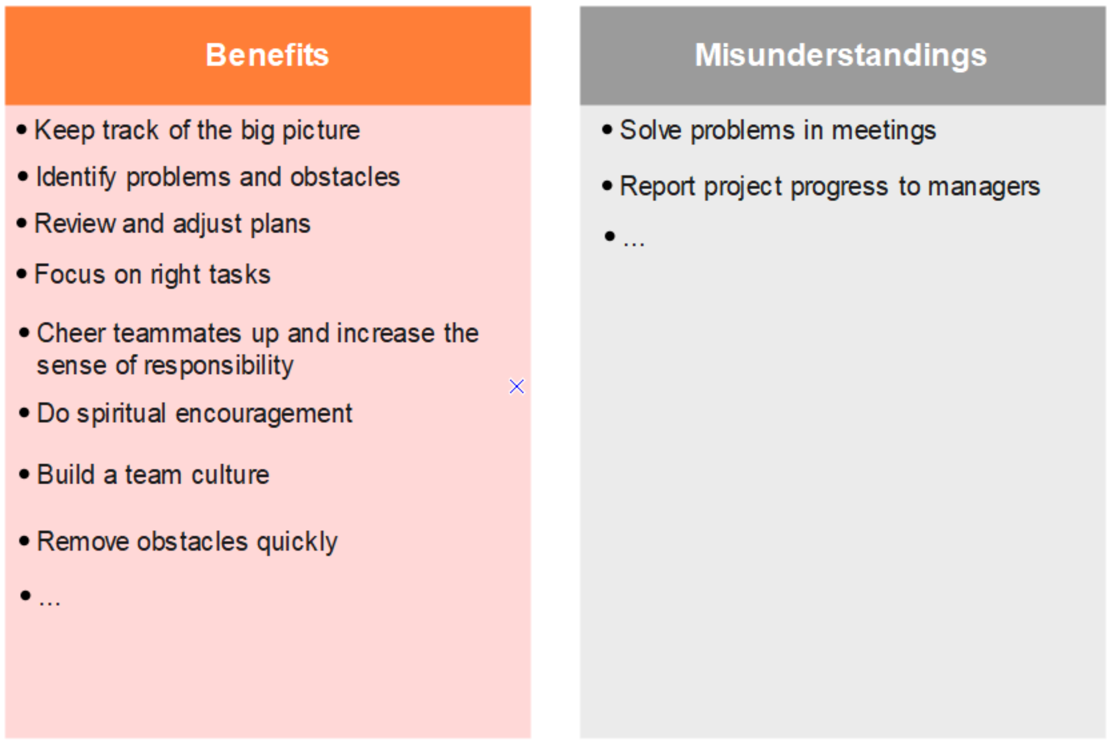
During the daily scrum, every team member can grasp the big picture, learn about what has happened recently, track the progress towards the sprint goal, assess if any adjustments are required for the day's tasks, and identify challenges or obstacles. A daily scrum is an activity that reviews, synchronizes, and adapts to daily plans to help self-organizing teams better do their work.

Some teams think that daily scrums are used to solve problems or report project progress to managers. However, their core significance and value are often misinterpreted.

Daily scrums help team members stay focused on their right tasks every day. Team members usually do not shirk responsibility because they make commitments in front of their coworkers. Team members are motivated to ensure they meet their daily goals. The daily scrum also ensures that the Scrum master and team members can promptly address problems, cultivate team culture, and make everyone feel the team spirit. Some organizations that do not use agile methodologies sometimes do daily scrums.

The value, significance, and misunderstandings of daily scrums are summarized as follows.

Figure 1-28 Benefits and misunderstandings



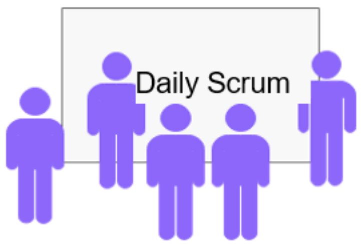
Benefits	Misunderstandings
• Keep track of the big picture	• Solve problems in meetings
• Identify problems and obstacles	• Report project progress to managers
• Review and adjust plans	• ...
• Focus on right tasks	
• Cheer teammates up and increase the sense of responsibility	
• Do spiritual encouragement	
• Build a team culture	
• Remove obstacles quickly	
• ...	

2. **Getting the daily scrum right**

   Let us learn how to do a right daily scrum.

   Team members gather around the whiteboard on time (to increase the sense of ceremony) to plan their daily tasks.

Figure 1-29 Daily scrum illustration

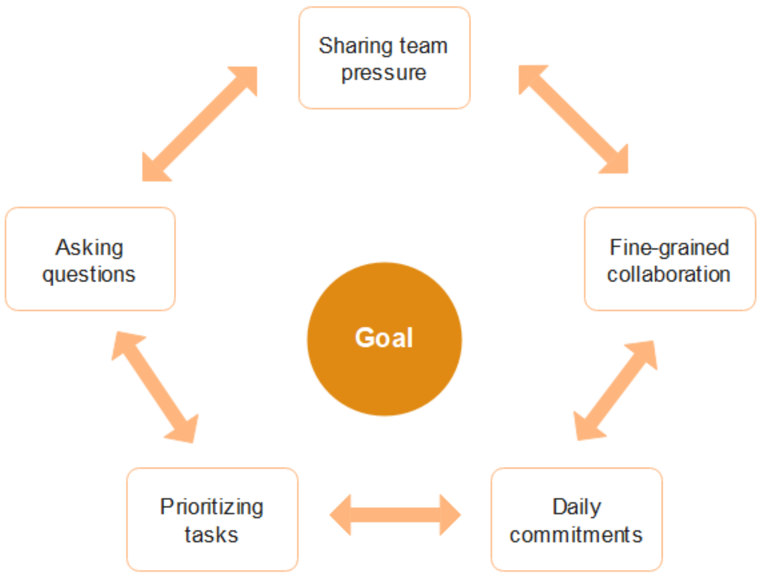


Team members need to take turns speaking and answer the following three questions during the daily scrum:

a. What did I do yesterday? (What have I done since the last daily scrum?)
b. What do l plan to do today? (What will I do before the next daily scrum?)
c. What problems and obstacles have I encountered? (What problems and obstacles have prevented or slowed my work?)

These three simple questions can prompt team members to review their work daily, create their own work plans, seek assistance in overcoming obstacles, and make commitments to the team. If the team does the daily scrum properly, they can achieve the following outcomes.

Figure 1-30 Daily scrum goals



– **Sharing team pressure**

Healthy agile teams are under pressure to work together. All team members promised to work together to finish the sprint work. Therefore, team members are interdependent and accountable to each other. Team members cannot ignore some other member doing the same thing for many days without any progress and motivation to move forward. Their unfinished work will be an obstacle for other team members.

– **Fine-grained collaboration**

  In daily scrums, team members should communicate efficiently with clear purposes. For example, if a member shares their plan for the day, another member can say, "Oh, so you are planning to do that today. I will need to rearrange my work priority. That is fine. Go ahead with your plan, I will make the necessary adjustments. I am glad you said that." This fine-grained collaboration allows team members to know how and when to collaborate with each other. An agile team should strive for higher efficiency and zero waiting time.

– **Prioritizing tasks**

  In the daily scrum, every team member can stay updated on the ongoing and completed tasks. A healthy team needs to prioritize finishing tasks instead of constantly having tasks in progress. In the daily scrum, the team needs to prioritize their tasks so that they can focus on finishing these tasks as soon as possible. In other words, getting 10 things done is far more meaningful than doing 100 things.

– **Daily commitments**

  At the daily scrum, team members need to make commitments to the team. In this way, team members know what will be delivered agilely and how to be accountable to one another.

– **Asking questions**

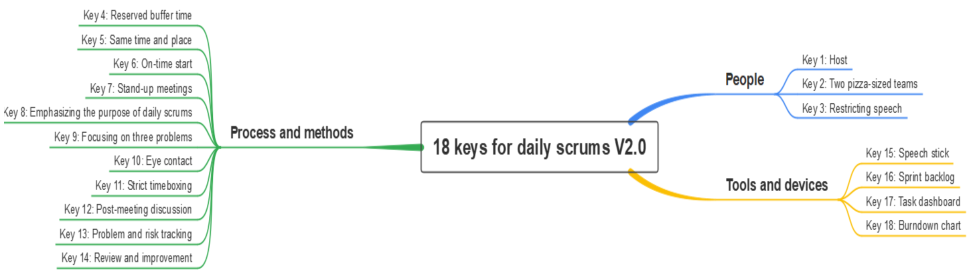
  Problems can be raised at any time in Agile, but daily scrum is a golden time for team members to reflect on "what hinders me or slows down my work".

3. **Learning best practices and key points**

   Summarize some key points for successful daily scrums through a large number of practices. Maybe these key points are not fully applicable. Therefore, it is essential to customize these points to suit the specific circumstances. These key points are called "18 keys for daily scrums".

   The "18 keys for daily scrums" are divided into people, procedures and methods, and tools and equipment to help you learn and memorize.

Figure 1-31 18 keys for daily scrums


18 keys for daily scrums V2.0
People
Key 1: Host
Key 2: Two pizza-sized teams
Key 3: Restricting speech
Process and methods
Key 4: Reserved buffer time
Key 5: Same time and place
Key 6: On-time start
Key 7: Stand-up meetings
Key 8: Emphasizing the purpose of daily scrums
Key 9: Focusing on three problems
Key 10: Eye contact
Key 11: Strict timeboxing
Key 12: Post-meeting discussion
Key 13: Problem and risk tracking
Key 14: Review and improvement
Tools and devices
Key 15: Speech stick
Key 16: Sprint backlog
Key 17: Task dashboard
Key 18: Burndown chart

– **Key 1: Host**

  The host (such as the Scrum master) holds a meeting on time and manages the meeting duration. Team members can talk to each other concisely. They can take turns hosting the meeting to feel the pace of the meeting.

– **Key 2: Two pizza-sized teams**

  In *Succeeding with Agile: Software Development Using Scrum* by Mike Cohn, a good way to determine the team size is to see if two pizzas can feed everyone on the team.

  Because two pizza–sized teams are similar to the size of the family, the goal of the daily scrum can be easily achieved. When the team is a family size, it is easy to track what happens in the team. People can easily remember everyone's daily commitment and their responsibility for the team. It is also recommended to keep the team size small in Scrum, usually 7 to 9 people.

  Finally, the team does not have to do daily scrums based on the above 18 keys. These keys are simply a collection of experiences and solutions to various problems. Each team may need to choose 18 keys according to their own needs.

– **Key 3: Restricting speech**

  Non-team members are welcome to join, but they are not allowed to speak. During the daily scrum, only team members who are fully committed to reaching the sprint goal should talk.

– **Key 4: Reserved buffer time**

  The development team should consider having a daily scrum either 30 minutes or 1 hour after their work hours start. This leaves some buffer time for regular tasks, such as checking emails. It can also give the development team extra time to review the work from the day before, such as defect reports generated by automated testing tools that were initiated overnight.

– **Key 5: Same time and place**

  Daily scrums ideally take place at the same time and place, with a visualized task board in front of team members. By meeting regularly at a fixed time and place, team members can develop the habit, saving time and efforts to find a meeting spot or confirm schedules.

– **Key 6: On-time start**

  All team members should arrive on time. The host should start the meeting on time, even if someone has not arrived yet. Latecomers should be punished, such as paying fines or doing push-ups. Team members decide on the penalties beforehand. If the penalty is a fine, the team also decides on how to use it. What if team members keep being late? For more solutions, see **Learn More: Solutions for Members Being Late**.

– **Key 7: Stand-up meetings**

  Team members must stand for a meeting, which is why the meeting is called a daily scrum. Standing at a meeting is more efficient than sitting, encouraging people to wrap up the meeting sooner and get started on their work for the day. Sitting down is easy to relax, but difficult to stay focused and manage time effectively.

– **Key 8: Emphasizing the purpose of daily scrums**

  The purpose of the daily scrum is often emphasized, especially for teams new to daily scrums. The Scrum master or other leaders (or meeting hosts) can emphasize the purpose of daily scrums. They gather feedback from team members about the daily scrum and the progress they have made. Over time, team members might decide to use the goal statement as their daily benchmark. Following each daily scrum, team members assess their own performance. It is an effective self-management tool.

– **Key 9: Focusing on three problems**

  During the daily scrum, team members only focus on three topics (yesterday... today... the problems are ...) and do not talk about other things. Only discuss work that has been finished and is about to start, and any problems and challenges faced during that work. The purpose is not to report work to leaders, but to communicate with each other to understand the project progress and solve problems together.

– **Key 10: Eye contact**

  It is a fun game in which someone is asked to speak in front of the group, and other members are asked to maintain eye contact with the speaker. The speaker will see who is looking away as they speak. This game keeps speeches concise and improves members' understanding of what speakers say, thereby accelerating the improvement of the daily schedule.

– **Key 11: Strict timeboxing**

  The daily scrum is a 15-minute timebox for the development team. It is recommended that the meeting not be too long. For a team of 5 to 9 members, a meeting of 15 minutes is enough.

– **Key 12: Post-meeting discussion**

  When someone is speaking, other members should listen carefully. They can ask concise questions for clarification, but they should not have lengthy discussions. To learn more about a member's report or seek assistance from other members, they are free to gather interested individuals for a discussion right after the daily scrum.

– **Key 13: Problem and risk tracking**

  A brief record of the problems and risks encountered by the meeting members shall be made (avoid too many details), and saved to the knowledge base or somewhere appropriate. The purpose is to ensure that these problems and risks are closed (For example, problems and risks can be discussed and tracked after the meeting until they are closed).

– **Key 14: Review and improvement**

  The daily scrum is a basic Plan-Do-Check-Act (PDCA). Also, the team can get feedback of the daily scrum during the retrospective meeting, including its pros and cons for the next sprint. (The daily scrum serves as a review point during the review meeting. If everything is going smoothly, no additional review is necessary.)

– **Key 15: Speech stick**

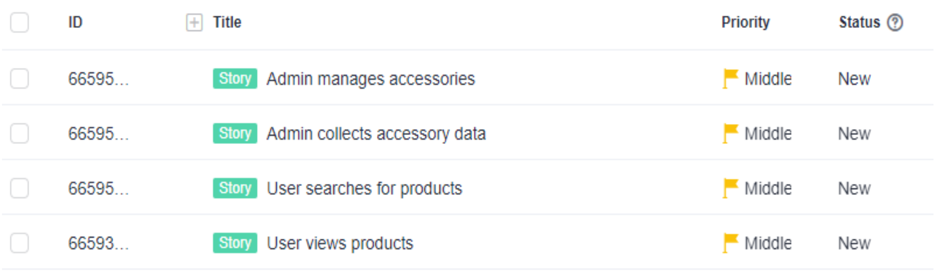
  Some small props can be used to ensure that the meeting does not time out. I will find either a pen or a small item as a speech stick, ask a member to answer the three questions, and then pass it on to the next member. Only members with the speech stick can talk. If someone takes too long, interrupt in time and ask for the speech stick to be passed on to the next person.

– **Key 16: Sprint backlog**

In a daily scrum, team members can refer to the sprint backlog to check the task progress. The sprint backlog records the progress of team members' work and needs to be updated and tracked daily. The electronic sprint backlog can keep local and remote teams on the same page during daily scrums. When the speaker talks about the three topics, the sprint backlog can be displayed to the team.

The CodeArts sprint backlog is as follows.

Figure 1-32 CodeArts sprint backlog



ID	Title	Priority	Status
66595...	Story Admin manages accessories	Middle	New
66595...	Story Admin collects accessory data	Middle	New
66595...	Story User searches for products	Middle	New
66593...	Story User views products	Middle	New

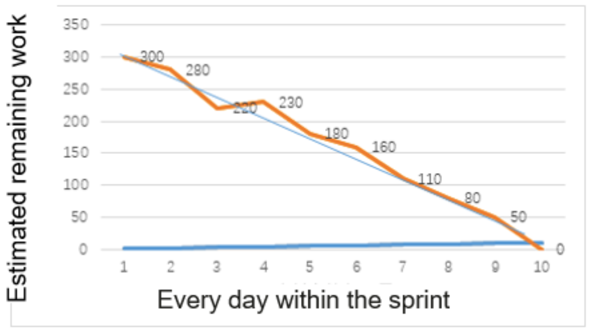
– **Key17: Task dashboard**

In the daily scrum, the task board helps the team members to stay updated on the ongoing and completed work. The team prioritizes finishing tasks that are ongoing. By identifying these tasks, the team can resolve them promptly.

– **Key 18: Burndown chart**

The burndown chart is a powerful tool for visualizing progress and remaining work. Generally, the vertical axis indicates the remaining workload (hours, story points, or number of work items), and the horizontal axis indicates the sprint duration (usually in days).

Figure 1-33 Burndown chart



Speakers can use a burndown chart to explain the progress during the daily scrum. The burndown chart enables all team members to see the sprint progress at a glance, including whether the work is on track and in good condition. The progress assists the team in figuring out if they can finish a set number of tasks and decide early in the sprint. The following effects can be achieved by using the burndown chart:

i. High visibility, displaying progress and remaining work intuitively
ii. Rapid risk identification
iii. Helping the team develop knowledge and understand their strengths
iv. Knowing the work pace of team members
v. Understanding the team sprint plan
vi. Using together with task walls for better efficiency

You do not have to use all **18 keys** in a daily scrum. These keys are just suggestions based on successful practices. If you face problems during the daily scrum, consider the **18 keys** and choose keys that fit your team. For example, there are four keys about tools. Do we need to use all these tools? No. As mentioned in the Agile Manifesto, individuals and interactions are higher than processes and tools. Tools serve the team, instead of weighing them down or even holding them back. Therefore, it is important for the team to choose the right keys.

Learn More: Solutions for Members Being Late

Team members can discuss reasons for lateness at the retrospective meeting, gather opinions, or change the start time of the daily scrum. For example, why does the meeting need to start at 9:00 a.m? Can we do another time? What are the difficulties? They can work together to find a solution.

Emphasize the significance of team members arriving on time. The host should start the meeting at the scheduled time and place, even if some attendees are missing. Do not convey the meeting content again to latecomers. Otherwise, someone would think it acceptable to be late.

There should be some punishments for those who are late. The team members decide on the punishments beforehand. Compared with the rules given by others, you are more willing to follow your own rules and keep your commitments.

If the punishment is not enforceable, visualize the punishment. For example, make a specific spot on the whiteboard and pin a photo each time you are late. This particular spot attracts people to see.

Talk to those who are often late, try to understand what problems and difficulties they have, care about them, and help them solve the problems together.

If tardiness is prevalent, the team may not be able to solve the problem. You can attempt to enforce the company's policies and strictly follow the attendance system. However, this goes against the concept of agile self-management and cannot really solve the problem.

To sum up, consider the following factors when addressing the problem of tardiness:

- Analyze the cause, care about the members, and decide together.
- Do the daily scrum at a fixed time and place, and on time.
- If someone is late, do not convey meeting content again.
- Establish a mechanism for light punishment.

- Understand the reason for being late and whether there are difficulties.

1.6 Member Management

1.6.1 How Do I Cultivate and Manage New Members of My Project Team When the Team Members Change Frequently?

Background

As the business grows, an enterprise often has many new employees joined. The development team leader needs to explain knowledge, working modes, and team information to each new employee. The workload increases sharply in a short period of time. The core members of a project, such as the project manager and development team leader, often face the following challenges:

- New employees need many trainings to get started with their work.
- The resignation of senior employees leads to the lack of personnel who understand key technologies and services.
- After working for a period of time, employees have new plans for their career and want to change their work.

So, how should the project owner address these challenges?

Problem Analysis

As a project develops, the project team continuously expands with new employees. Since new employees are unfamiliar with the project to which they have been just assigned, trainings are necessary for new employees.

When the project becomes stable, some senior employees resign due to various reasons such as the business structure, salary, and physical and mental fatigue. Some senior employees gradually become tired of their work, or they want to seek opportunities for transformation. This results in the lack of personnel who knows certain services and key technologies. If these problems are not solved, the project progress will be affected and unnecessary costs will be incurred. It will also be unfavorable to the internal growth and self-organization of the team.

The key to these problems is still the cultivation of new employees.

Solution

Generally, the joining of new employees, key employee resignation, and personal development can be addressed by establishing an information library for managing **team information**, **working modes**, and **knowledge**.

CodeArts is an R&D cloud platform built with Huawei's development practices, cutting-edge R&D concepts, and advanced R&D tools (see **Reference Documents**).

1.6.2 How Do I Manage the Permissions of Project Members?

Background

This section describes how to add customer service members and grant them permissions.

Problem Analysis

- How do I use IAM to control permissions?

  An account can create multiple software development projects. By default, only accounts can view all projects and members, and determine whether their Identity and Access Management (IAM) users can create projects. In some enterprise scenarios, an account can authorize an IAM user through fine-grained permissions management so that the user can configure settings on behalf of the account.
- What roles can project members have?

  All projects created on the project management page support independent permission settings. This means you can configure different permissions in each project.

  Project management involves the following default roles:
  - Project administrator: the creator of a project
  - Project manager: project development administrator
  - Test manager: project test administrator
  - Product manager: requirement analysis manager of a project
  - System engineer: architecture analysis manager of a project
  - Committer: participant in project development
  - Developer: participant in project development
  - Tester: participant in project testing
  - Participant: participates in specified tasks of a project
  - Viewer: a member who follows or browses a project
  - O&M manager: a member responsible for O&M of a project
- How do I add members to a project and assign them roles?

Solution

CodeArts Req uses IAM to manage permissions for multiple projects of a tenant. Each project can have different permissions. There are two types of permissions managed in CodeArts Req: cloud service and project permissions.

- Cloud service permissions are configured using IAM. IAM is a basic service for permissions management in Huawei Cloud. It can be used free of charge. You pay only for the resources in your account.
  - For details about IAM, see **IAM Service Overview**.
  - For details about cloud service permissions, see **Cloud-Service-Level Permissions**.
- Project permissions are configured in CodeArts Req. For details about project permissions, see **Project-Level Permissions**.

For more information about members, see **CodeArts Req User Guide > "Managing Members"**.

1.7 Appendix

1.7.1 Reference Documents

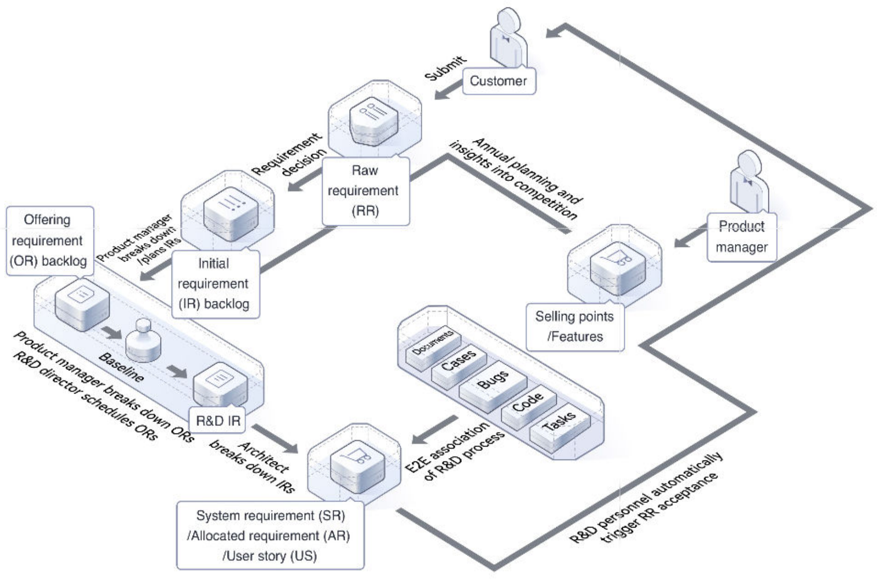
- *Essential Scrum* by Kenneth S. Rubin
- *User Stories for Agile Approach* (A translation of *User Stories Applied: For Agile Software Development*) by Mike Cohn
- *2019 State of DevOps Report of China* by China Academy of Information and Communications Technology (CAICT), Huawei Cloud CodeArts (formerly "DevCloud"), and Nanjing University
- *User Stories Applied: For Agile Software Development* by Mike Cohn
- *Becoming a Technical Leader* by Gerald M. Weinberg
- *Turn Your Experience into Capabilities* by Qiu Zhaoliang
- *The Scrum Guide*. November 2017
- *Essential Scrum* by Kenneth S. Rubin. [M]. Beijing: Tsinghua University Press
- *Scrum Guide*. 2007
- *Agile Project Management For Dummies* by Mark C. Layton. [M]. Beijing: Posts & Telecom Press
- **Should Team Member Sign Up for Tasks During Sprint Planning?**
- *Coaching Agile Teams* by Lyssa Adkins. [M]. Beijing: Publishing House of Electronics Industry

2 Best Practices of IPD-System Device Projects

2.1 Overview

The IPD-system device template presets Huawei's best practices in integrated product development. It offers a structured R&D process (see **Figure 2-1**) for system devices, involving raw requirement management, product feature tree management, R&D requirement breakdown and allocation, baseline management, change, and cross-project collaboration.

Figure 2-1 Process of IPD-system device projects


Customer
Submit
Raw requirement (RR)
Requirement decision
Annual planning and insights into competition
Product manager
Selling points /Features
Initial requirement (IR) backlog
Product manager breaks down /plans IRs
Offering requirement (OR) backlog
Baseline
R&D IR
Product manager breaks down ORs
R&D director schedules ORs
Architect breaks down IRs
Documents
Cases
Bugs
Code
Tasks
E2E association of R&D process
System requirement (SR) /Allocated requirement (AR) /User story (US)
R&D personnel automatically trigger RR acceptance

2.2 Requirement Model

First, let's look at the work objects in this requirement model: RR, SF, IR, SR, and AR.

Table 2-1 Work objects of IPD-system device projects

Work Object	Description
Raw requirement (RR)	An original problem or demand described from the perspective of internal and external customers. Customer requirements are a type of RRs. This type of requirements must be analyzed and reviewed by the Requirement Management Team (RMT) and Requirement Analysis Team (RAT).
System feature (SF)	A major capability of a version that supports problems (PBs). In principle, a system feature is a set of key selling points (highlights) of a product package. Each feature is an E2E solution that meets customers' specific business requirements. Some features can be sold independently via license control.
Initial requirement (IR)	A requirement re-described accurately (with a complete background and standard format) from the perspective of internal and external customers and markets. IRs can be produced: ● By breaking down RRs or SFs. ● From product plans.
System requirement (SR)	A functional or non-functional requirement described from the R&D perspective. This type of requirements is visible to the outside and testable. Functional requirements are scenario-specific, while non-functional requirements are about system costs, global quality attributes (mainly DFX), and technical limitations.
Allocated requirement (AR)	A functional or non-functional requirement that an SR is broken down into for subsystems/modules to deliver, based on the division of responsibilities of basic-level organizations.

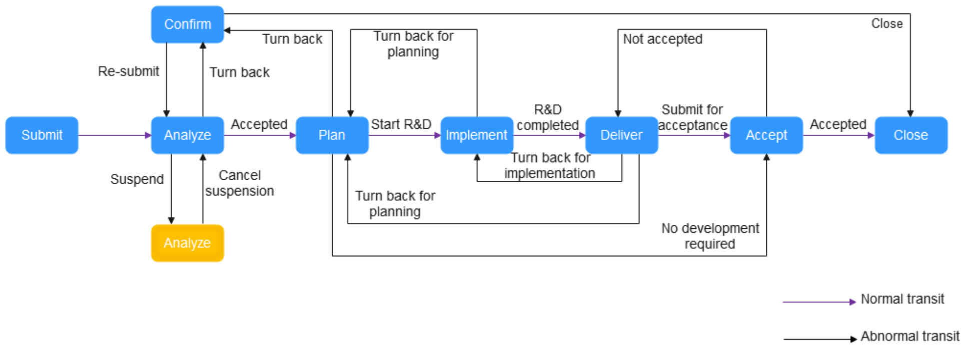
2.3 RR Management

2.3.1 Introduction

A successful product must meet customer needs, which are the driving force of enterprise development. Unlike traditional requirement management tools that only play a role in the R&D phase, CodeArts Req covers both customer and market requirements by providing a complete process from customer requirement collection, high-value requirement decision-making, to delivery and acceptance. It makes real-time requirement progress transparent and improves market requirement flow by 70%.

RRs are the original problems or demands of both internal and external customers. They are described in the perspective of customers. Customer requirements are a type of RRs. These requirements must be analyzed by the corresponding recipient. The process of RRs is as follows: submit, analyze, plan, implement, deliver, accept, and close.

Figure 2-2 RR status transition flowchart



2.3.2 Glossary

- **This Project**

  A collection of RRs that are proposed to a project from either inside or outside the project.
- **Other Projects**

  A collection of external RRs proposed across projects or organizations.
- **Create/Associate Child Requirement**

  An RR is analyzed, evaluated, and accepted, and then broken down into child requirements or associated with other child requirements for smaller granularities of R&D work.
- *__Recipient__

  A member who undertakes an RR after key joint decision-making.

2.3.3 Examples

2.3.3.1 Introduction

A company plans to launch a smart watch. However, the R&D will take a long period, and requires collaboration of multiple departments and teams. How can the company transfer RRs among organizations and ensure that the RRs can be finally closed? Now, let's look at some RR lifecycle management practices.

2.3.3.2 Creating RRs

Classify the original demands from both internal and external customers into "This Project" and "Other Projects", and convert the demands into RRs.

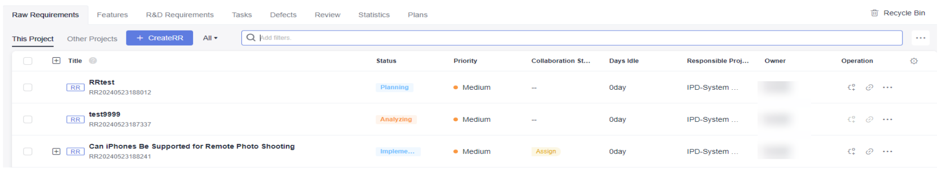
Prerequisites

- The administrator has registered a HUAWEI ID and enabled CodeArts. For details, see **Registering a HUAWEI ID and Enabling Huawei Cloud Services**.
- The administrator has created IAM users for the project members. For details, see **Creating an IAM User**.
- An **IPD-system device project** has been created.

Procedure

1. On the **Raw Requirements** tab, click **CreateRR**.

Figure 2-3 Raw Requirements tab



Raw Requirements | Features | R&D Requirements | Tasks | Defects | Review | Statistics | Plans | Recycle Bin

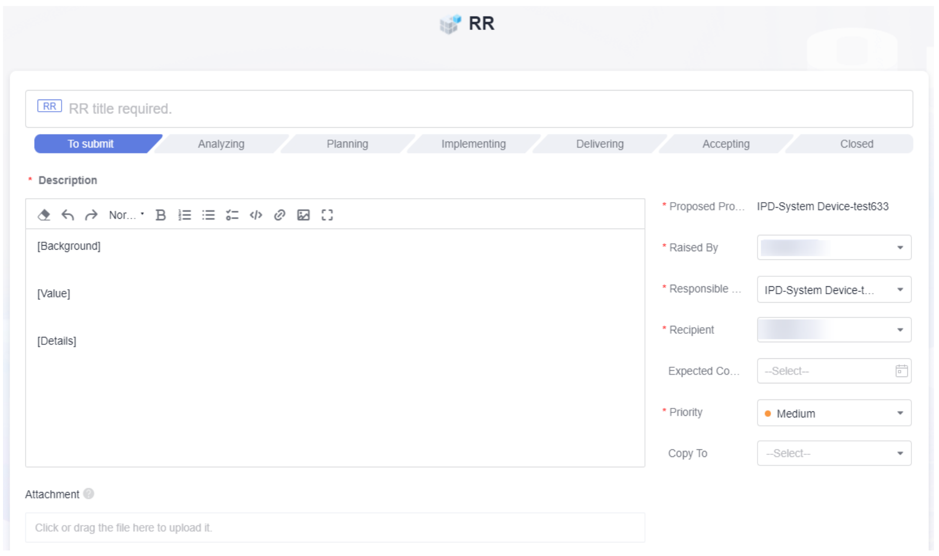
This Project | Other Projects | + CreateRR | All | Add filters.

Title	Status	Priority	Collaboration St...	Days Idle	Responsible Proj...	Owner	Operation
RR RRtest RR20240523188012	Planning	● Medium	--	0day	IPD-System ...	[redacted]	
RR test9999 RR20240523187337	Analyzing	● Medium	--	0day	IPD-System ...	[redacted]	
RR Can iPhones Be Supported for Remote Photo Shooting RR20240523188241	Impleme...	● Medium	Assign	0day	IPD-System ...	[redacted]	

2. Set the RR information.

   When creating an RR, fill in the mandatory fields such as **Title**, **Description**, **Raised By**, **Responsible Project**, and **Recipient**. The **Description** field is especially important. Describe the background, value, and details so that the RR can be analyzed accurately by owners in subsequent phases.

Figure 2-4 Creating an RR



RR

RR RR title required.

To submit | Analyzing | Planning | Implementing | Delivering | Accepting | Closed

* Description

[Background]

[Value]

[Details]

* Proposed Pro...	IPD-System Device-test633
* Raised By	[redacted]
* Responsible ...	IPD-System Device-t...
* Recipient	[redacted]
Expected Co...	--Select--
* Priority	● Medium
Copy To	--Select--

Attachment

Click or drag the file here to upload it.

3. Click **Submit**. The requirement transits to the analyzing phase.

2.3.3.3 Handling RRs

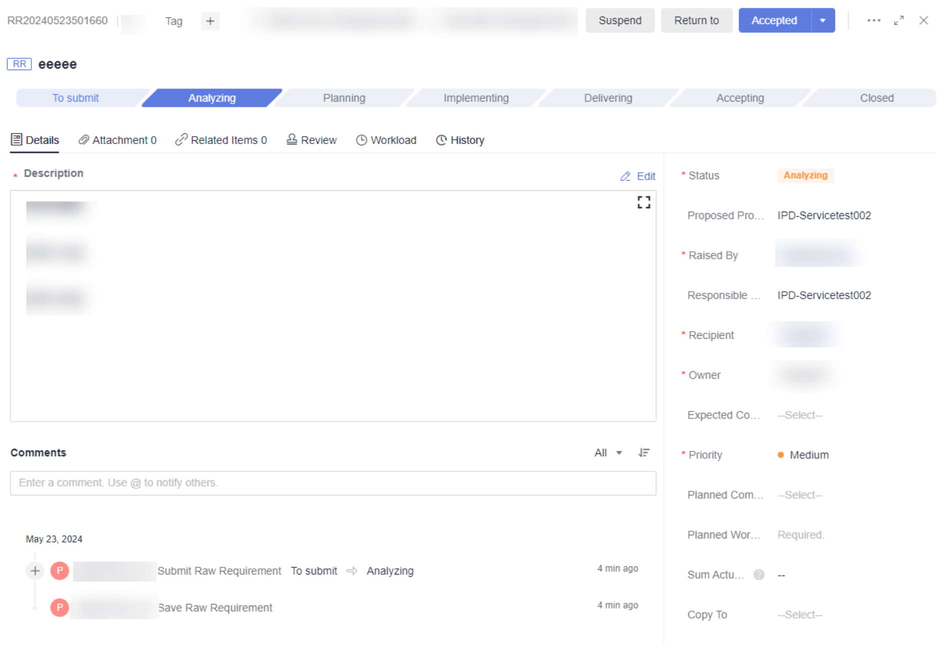
After an RR is created, the recipient needs to handle it for analysis, planning, implementation, delivery, acceptance, and closure.

- **Analyzing an RR**
- **Planning an RR**
- **Implementing an RR**
- **Delivering an RR**
- **Accepting an RR**
- Closing an RR: Once accepted, an RR changes to the **Closed** state.

Analyzing an RR

The owner assesses the RR's value in the **Analyzing** phase, and then chooses to accept, turn back, or suspend the RR after analysis and decision-making.

Figure 2-5 Analyzing an RR

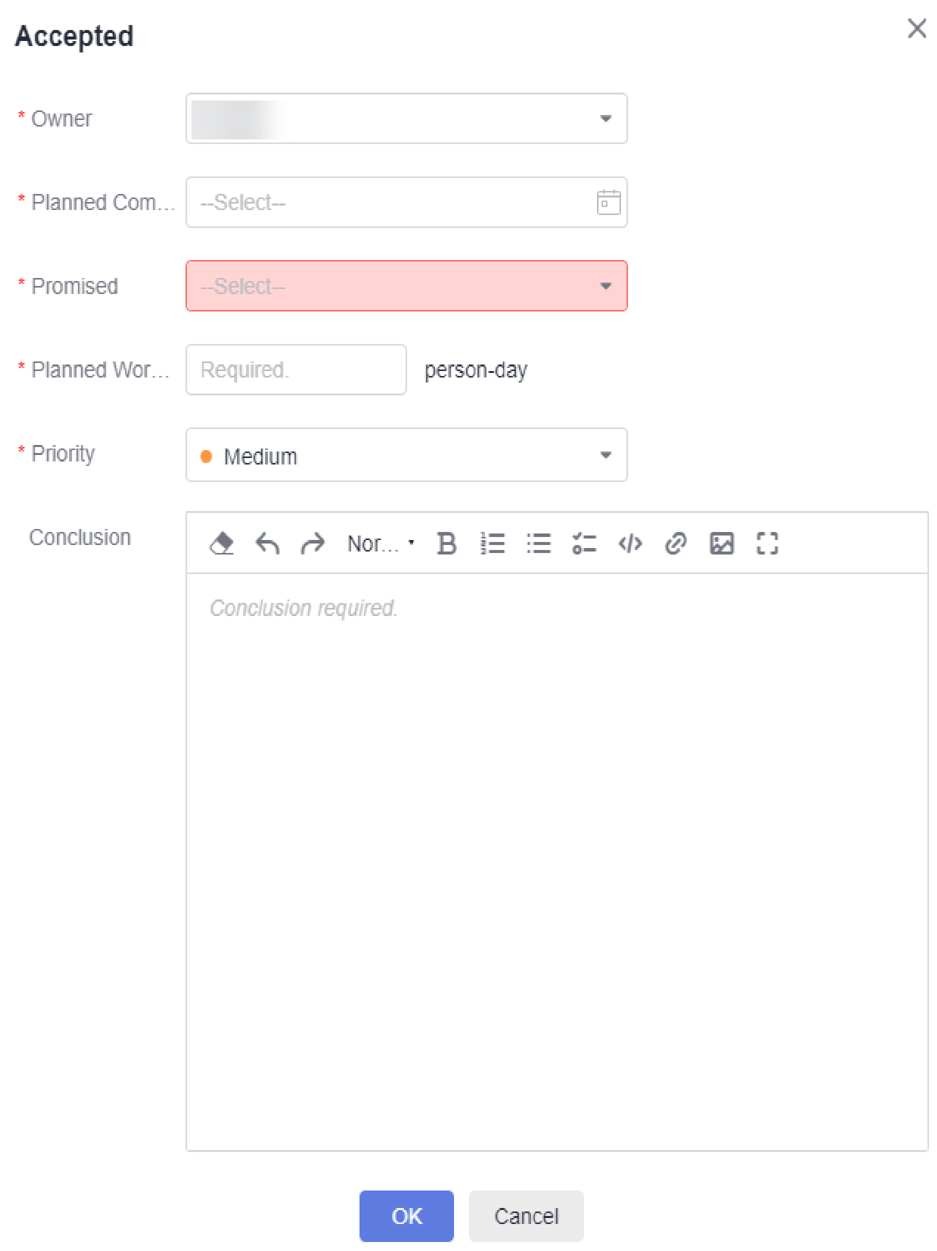


May 23, 2024
Submit Raw Requirement To submit → Analyzing 4 min ago
Save Raw Requirement 4 min ago

* Status	Analyzing
Proposed Pro...	IPD-Servicetest002
* Raised By	
Responsible ...	IPD-Servicetest002
* Recipient	
* Owner	
Expected Co...	--Select--
* Priority	● Medium
Planned Com...	--Select--
Planned Wor...	Required.
Sum Actu...	--
Copy To	--Select--

- If the owner accepts the requirement after analysis, they must fill in the mandatory fields and specify the subsequent work plan.

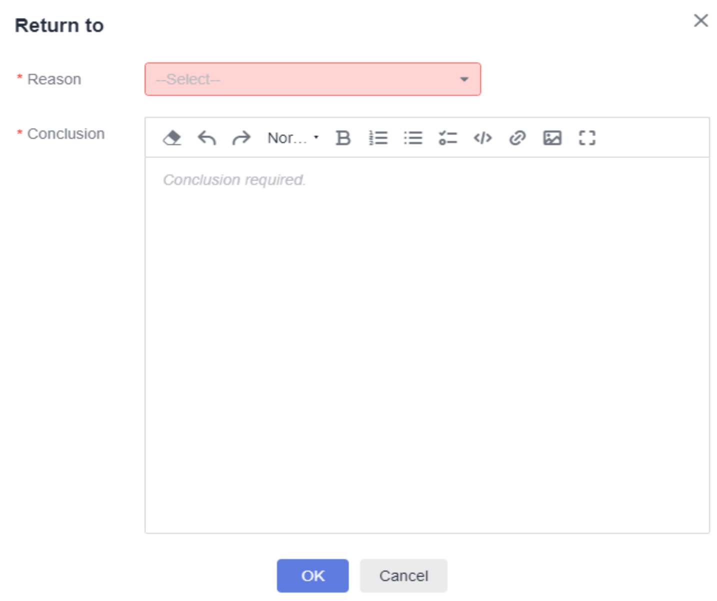
Figure 2-6 Accepting a requirement


Accepted

* Owner	
* Planned Com...	--Select--
* Promised	--Select--
* Planned Wor...	Required. person-day
* Priority	● Medium
Conclusion	Conclusion required.

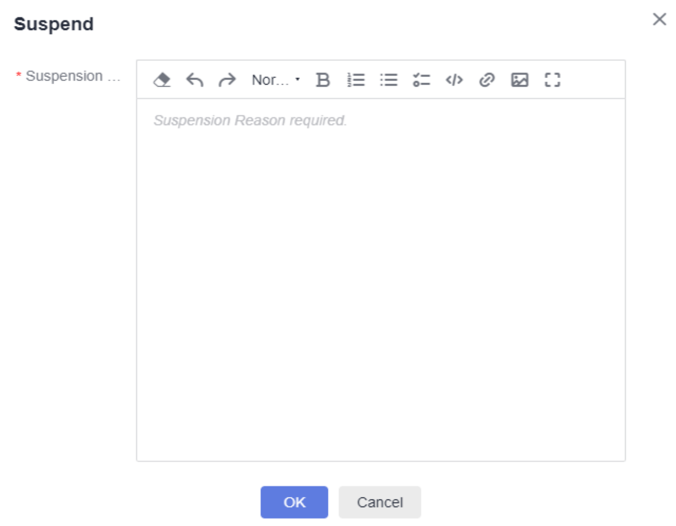
- If the owner turns back the requirement after analysis, they must also fill in the mandatory fields and reasons.

Figure 2-7 Turning back a requirement



- If the requirement is considered not urgent after analysis and decision-making, you can suspend it and provide your reasons.

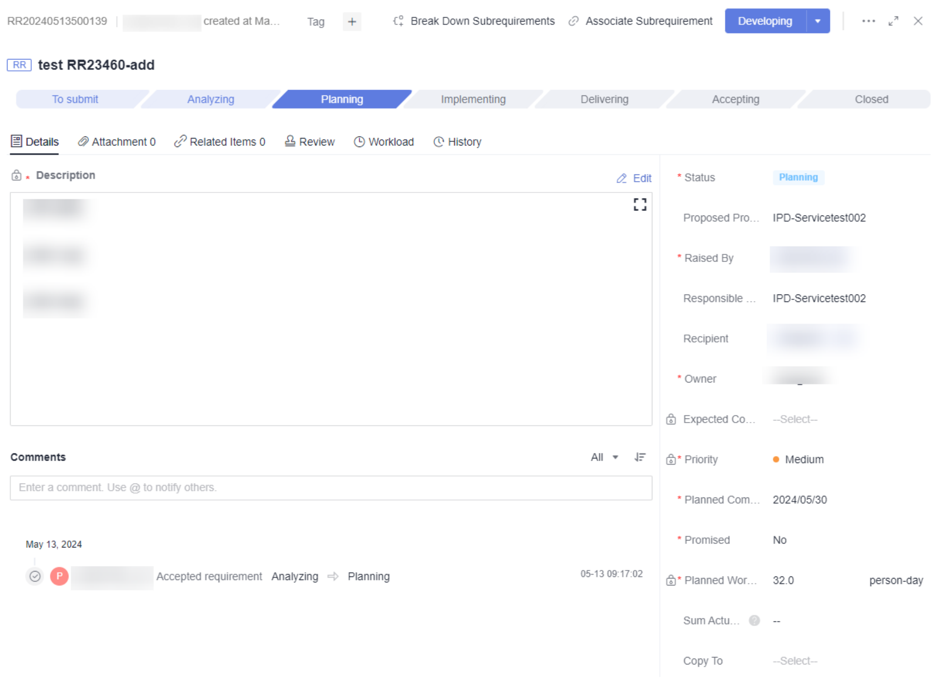
Figure 2-8 Suspending a requirement



Planning an RR

After an RR is analyzed and accepted, it moves into the planning phase. Determine whether to break down the RR, associate child requirements, or directly start the development.

Figure 2-9 Planning an RR


RR20240513500139 | created at Ma... Tag + Break Down Subrequirements Associate Subrequirement Developing

RR test RR23460-add

To submit | Analyzing | Planning | Implementing | Delivering | Accepting | Closed

Details | Attachment 0 | Related Items 0 | Review | Workload | History

Description — Edit

Comments — All

Enter a comment. Use @ to notify others.

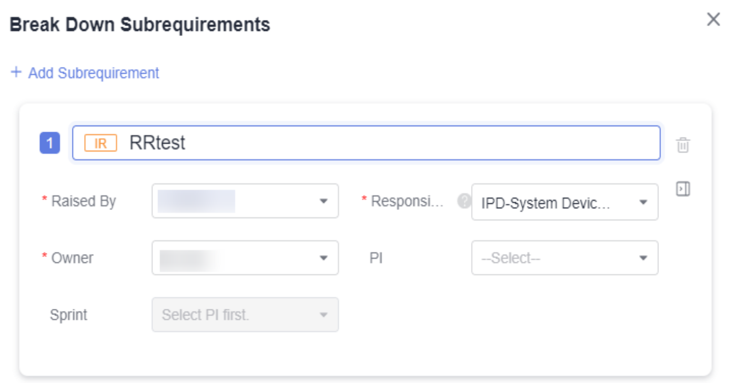
May 13, 2024

Accepted requirement Analyzing ⇨ Planning 05-13 09:17:02

* Status	Planning
Proposed Pro...	IPD-Servicetest002
* Raised By	
Responsible ...	IPD-Servicetest002
Recipient	
* Owner	
Expected Co...	--Select--
* Priority	● Medium
* Planned Com...	2024/05/30
* Promised	No
* Planned Wor...	32.0 person-day
Sum Actu...	--
Copy To	--Select--

After the RR is broken down, it automatically transits to the **Implementing** state.

Figure 2-10 Breaking down a requirement


Break Down Subrequirements

+ Add Subrequirement

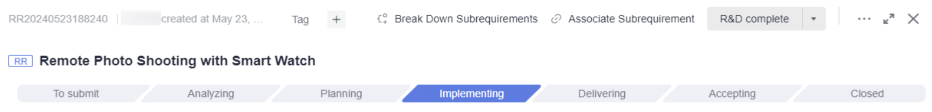
1 IR RRtest

* Raised By		* Responsi...	IPD-System Devic...
* Owner		PI	--Select--
Sprint	Select PI first.		

Implementing an RR

If needed, you can break down the RR or associate child requirements with it anytime during the R&D process.

Figure 2-11 Implementing an RR


RR20240523188240 | created at May 23, ... Tag + Break Down Subrequirements Associate Subrequirement R&D complete

RR Remote Photo Shooting with Smart Watch

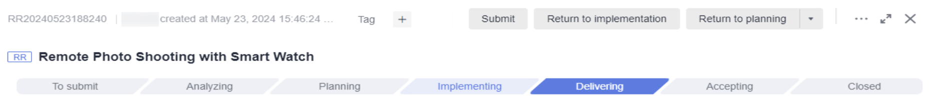
To submit | Analyzing | Planning | Implementing | Delivering | Accepting | Closed

When all created IRs and associated child requirements are completed, the RR automatically transits to the **Delivering** state.

Delivering an RR

The owner evaluates the delivery quality from various dimensions, and decides whether to directly submit the requirement for acceptance or turn it back to the corresponding phase for modification.

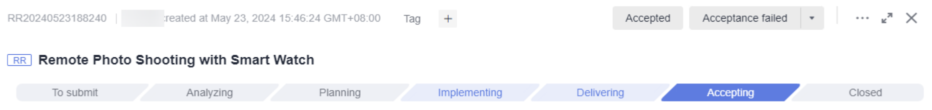
Figure 2-12 Delivering an RR



Accepting an RR

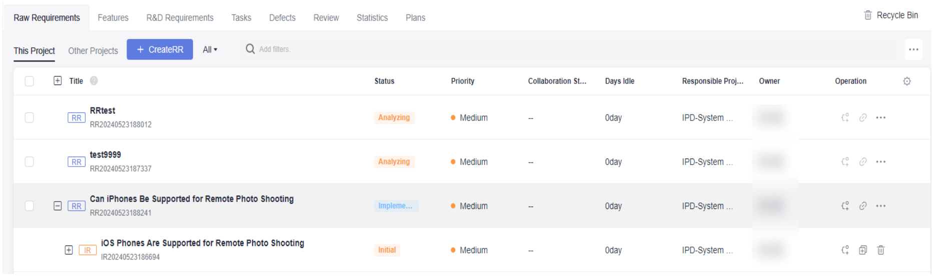
After the RR transits from the **Delivering** state to the **Accepting** state, the proposer evaluates whether the requirement can be accepted on the **Other Projects** tab. If the requirement is fully met, the proposer accepts it, and it then changes to the **Closed** state.

Figure 2-13 Accepting an RR


RR20240523188240 | [redacted] created at May 23, 2024 15:46:24 GMT+08:00
Tag
Accepted
Acceptance failed
RR Remote Photo Shooting with Smart Watch
To submit
Analyzing
Planning
Implementing
Delivering
Accepting
Closed

You can check all the child requirements created from the RR in the RR list for global tracing.

Figure 2-14 Viewing the RR list



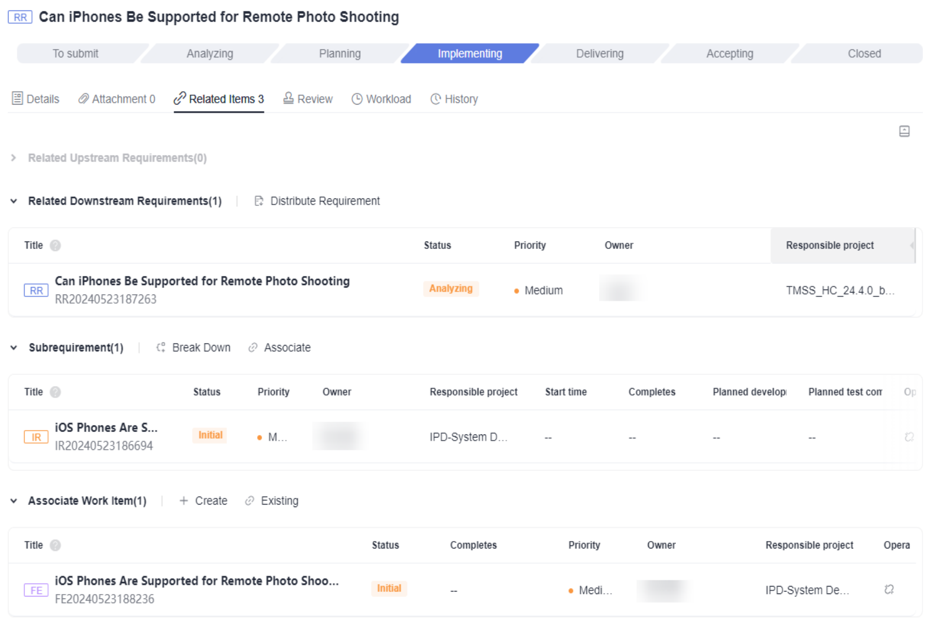
Title	Status	Priority	Collaboration St...	Days Idle	Responsible Proj...	Owner	Operation
RR RRtest RR20240523188012	Analyzing	● Medium	--	0day	IPD-System ...	[redacted]	
RR test9999 RR20240523187337	Analyzing	● Medium	--	0day	IPD-System ...	[redacted]	
RR Can iPhones Be Supported for Remote Photo Shooting RR20240523188241	Impleme...	● Medium	--	0day	IPD-System ...	[redacted]	
IR iOS Phones Are Supported for Remote Photo Shooting IR20240523186694	Initial	● Medium	--	0day	IPD-System ...	[redacted]	

2.3.3.4 Editing RR Details

2.3.3.4.1 Related Items

Once started, an RR is related to various work items. It goes through a complete process from collection, analysis, decision-making, implementation, to acceptance. Different parties from various organizations and teams work together in each phase. Related items are used to record and trace operations. On the **Related Items** tab, you can associate/disassociate work items, create child requirements, and break down/allocate requirements.

Figure 2-15 Setting related items


RR Can iPhones Be Supported for Remote Photo Shooting

To submit | Analyzing | Planning | Implementing | Delivering | Accepting | Closed

Details | Attachment 0 | Related Items 3 | Review | Workload | History

Related Upstream Requirements(0)

Related Downstream Requirements(1) | Distribute Requirement

Title	Status	Priority	Owner	Responsible project
RR Can iPhones Be Supported for Remote Photo Shooting RR20240523187263	Analyzing	Medium		TMSS_HC_24.4.0_b...

Subrequirement(1) | Break Down | Associate

Title	Status	Priority	Owner	Responsible project	Start time	Completes	Planned develop	Planned test com	Op
IR iOS Phones Are S... IR20240523186694	Initial	M...		IPD-System D...	--	--	--	--	

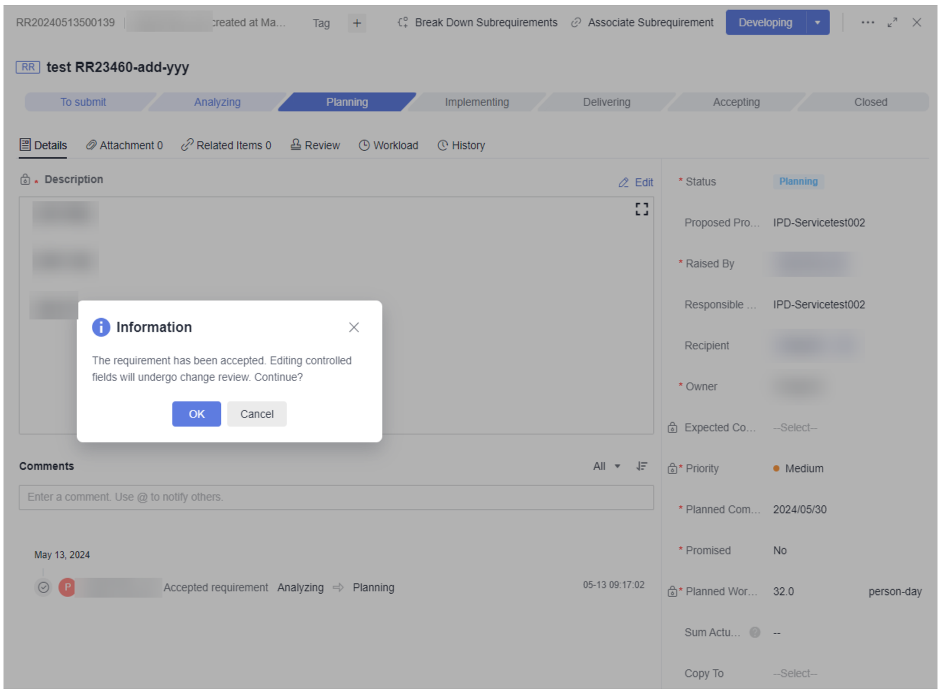
Associate Work Item(1) | Create | Existing

Title	Status	Completes	Priority	Owner	Responsible project	Opera
FE iOS Phones Are Supported for Remote Photo Shoo... FE20240523188236	Initial	--	Medi...		IPD-System De...	

2.3.3.4.2 Review

The title, description, priority, and planned completion time of an RR are controlled fields. They cannot be modified without permission once the RR is submitted. If you have to do so, you need to start a review process. After the review is complete, the relevant fields will be updated with the changed values.

Figure 2-16 Modifying controlled fields


RR20240513500139 | created at Ma... Tag + | Break Down Subrequirements | Associate Subrequirement | Developing

RR test RR23460-add-yyy

To submit | Analyzing | Planning | Implementing | Delivering | Accepting | Closed

Details | Attachment 0 | Related Items 0 | Review | Workload | History

Description Edit

Information

The requirement has been accepted. Editing controlled fields will undergo change review. Continue?

OK Cancel

* Status	Planning
Proposed Pro...	IPD-Servicetest002
* Raised By	
Responsible ...	IPD-Servicetest002
Recipient	
* Owner	
Expected Co...	--Select--
* Priority	Medium
* Planned Com...	2024/05/30
* Promised	No
* Planned Wor...	32.0 person-day
Sum Actu...	--
Copy To	--Select--

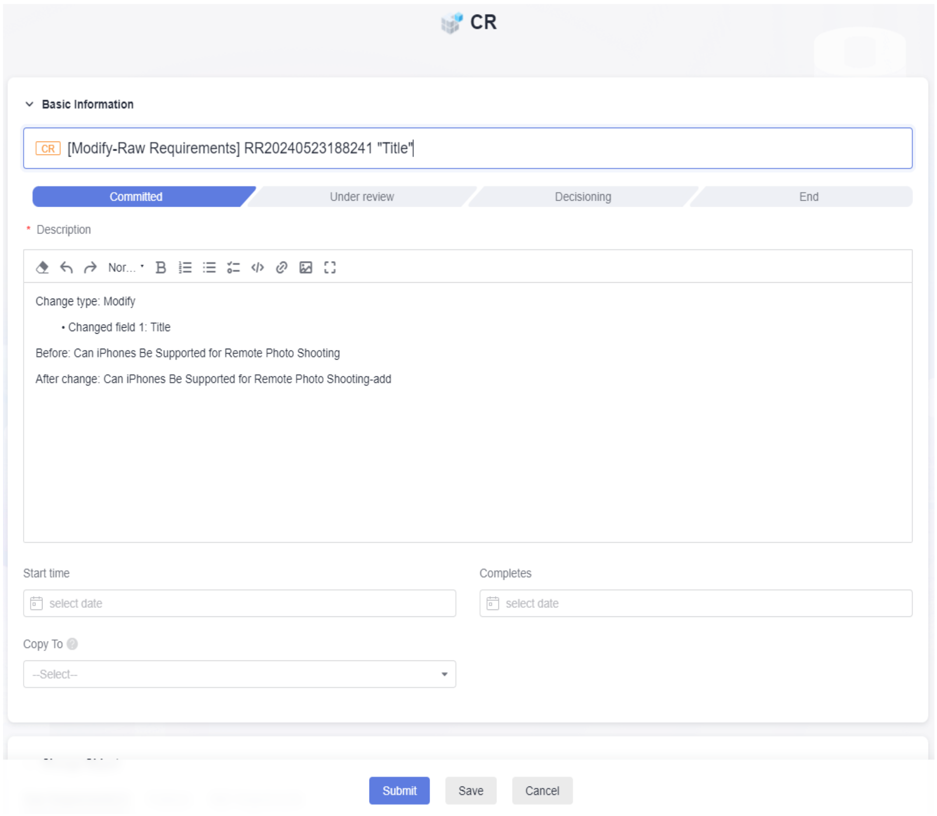
Comments All

Enter a comment. Use @ to notify others.

May 13, 2024

Accepted requirement Analyzing → Planning 05-13 09:17:02

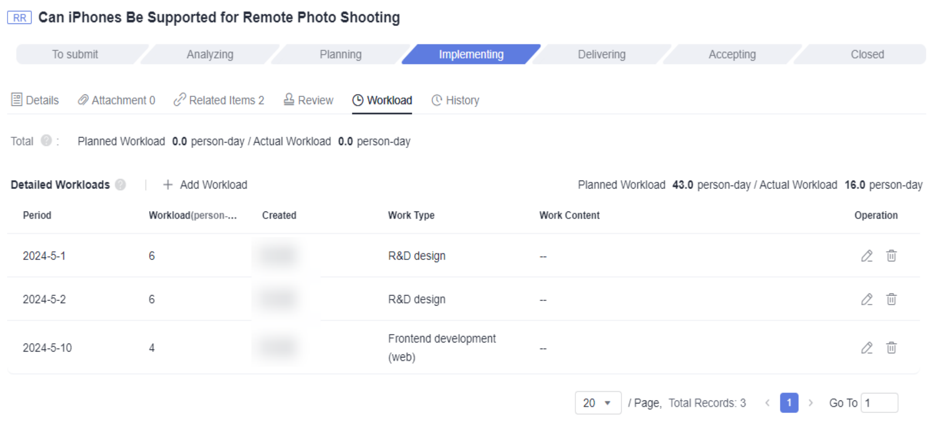
Figure 2-17 Creating a CR



2.3.3.4.3 Workloads

All work items require labor input to complete. The workloads need to be recorded to provide data basis for efficiency insights. You can add, edit, and delete the workloads of an RR on the **Workload** tab, where **Total** means the sum of the planned or actual person-days of all child requirements.

Figure 2-18 Setting workloads


RR Can iPhones Be Supported for Remote Photo Shooting
To submit
Analyzing
Planning
Implementing
Delivering
Accepting
Closed
Details
Attachment 0
Related Items 2
Review
Workload
History
Total : Planned Workload 0.0 person-day / Actual Workload 0.0 person-day
Detailed Workloads
+ Add Workload
Planned Workload 43.0 person-day / Actual Workload 16.0 person-day
20 / Page, Total Records: 3
1
Go To 1

Period	Workload(person-...	Created	Work Type	Work Content	Operation
2024-5-1	6		R&D design	--	
2024-5-2	6		R&D design	--	
2024-5-10	4		Frontend development (web)	--	

3 Best Practices of Defect Management

3.1 About Defect Management

Defect management is a critical part in product lifecycle management. Both hardware and software development will encounter countless defects. Poor defect management will affect the product quality. Based on years of experience in quality and operation management, Huawei has developed a set of effective defect best practices. It provides a unified, efficient, and visualized defect tracking platform to ensure that each defect can be closed with high quality and efficiency.

3.2 Glossary

- **Severity**

  There are four severities in descending order: critical, major, minor, and info.

- **Responsible Release/Sprint**

  The version in which a defect is found, or the release plan or sprint in which a defect is produced.

- **Environment**

  The environment where a defect exists. There are production, alpha, and beta environments.

- **Fix Release/Sprint**

  The release or sprint in which a defect is planned to be fixed.

3.3 Practice Overview

During product R&D, teams and projects often work independently. It is difficult to control product quality and track defect rectification progress, which severely affects product delivery efficiency. Defect management means to control the complete process from defect submission, analysis, rectification, testing, acceptance, to closure. It enables cross-project defect tracking and real-time identification, assuring product delivery quality of your organization.

This chapter uses a system device project as an example to describe the best practices in defect management.

3.4 Examples

3.4.1 Introduction

A company plans to launch a smart watch. However, the R&D will take a long period, and requires collaboration of multiple departments and teams. How can the company transfer defects among organizations and ensure that the defects can be finally closed? Now, let's look at some defect lifecycle management practices.

3.4.2 Creating a Bug

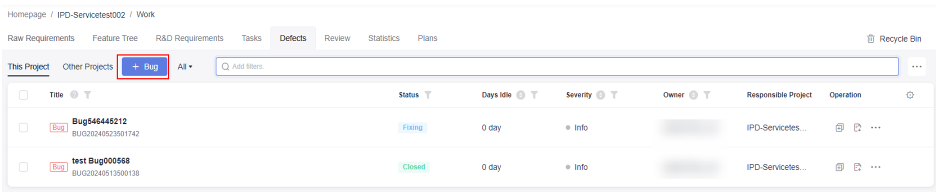
When a defect is detected, the first thing to do is record it in the defect management system by creating a bug.

Prerequisites

- The administrator has registered a HUAWEI ID and enabled CodeArts. For details, see **Registering a HUAWEI ID and Enabling Huawei Cloud Services**.
- The administrator has created IAM users for the project members. For details, see **Creating an IAM User**.
- An **IPD-system device project** or **IPD-standalone software project** has been created.

Procedure

1. On the **Defects** tab, click **Bug**.

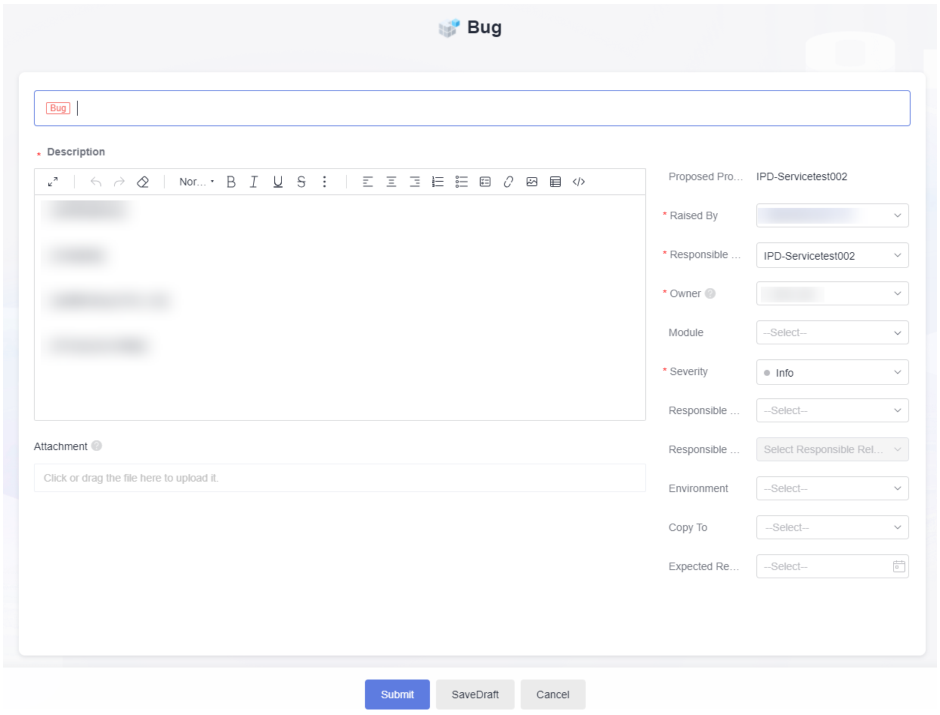


Title	Status	Days Idle	Severity	Owner	Responsible Project	Operation
Bug Bug546445212 BUG20240523501742	Fixing	0 day	Info	[redacted]	IPD-Servicetes...	
Bug test Bug000568 BUG20240513500138	Closed	0 day	Info	[redacted]	IPD-Servicetes...	

2. Enter the bug information.

   When creating a bug, fill in the mandatory fields such as **Title**, **Owner**, and **Severity**. The **Description** field is especially important. Describe the bug symptoms, expected fixing effect, and possible error messages in detail, so that the bug can be analyzed and located accurately in subsequent phases.

Figure 3-1 Creating a bug



Bug

Bug

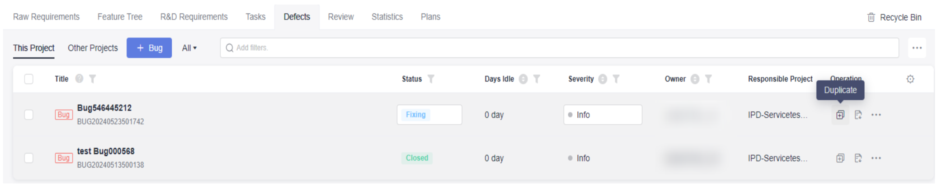
Proposed Pro...	IPD-Servicetest002
* Raised By	[redacted]
* Responsible ...	IPD-Servicetest002
* Owner	[redacted]
Module	--Select--
* Severity	● Info
Responsible ...	--Select--
Responsible ...	Select Responsible Rel...
Environment	--Select--
Copy To	--Select--
Expected Re...	--Select--

3. Click **Submit**. The bug goes to the analyzing phase.

More Operations

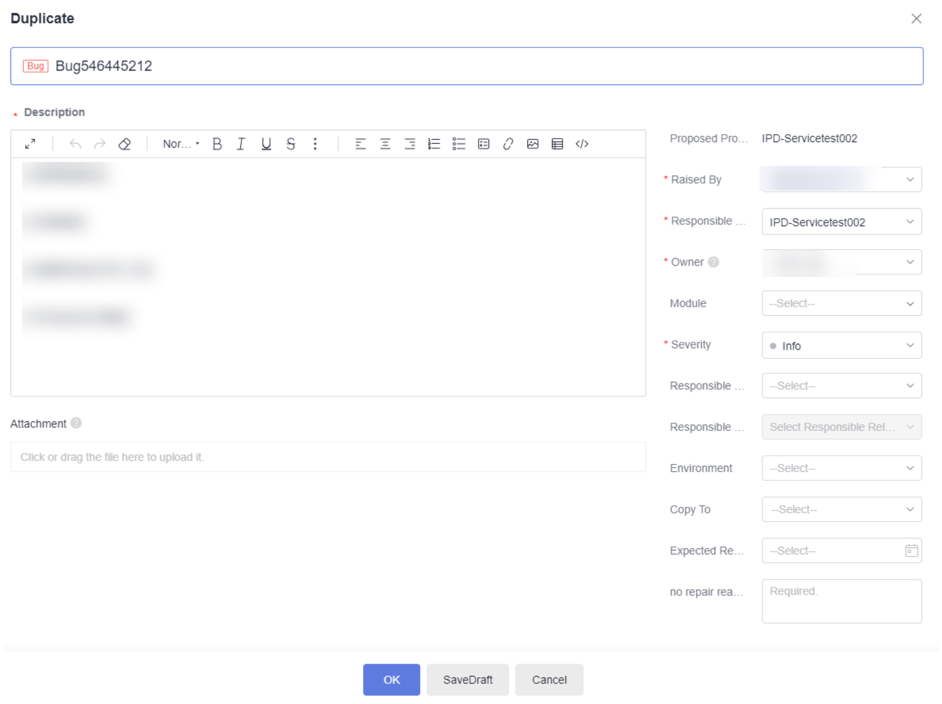
If you find another bug of the same type or with similar information when creating a bug, you can click **Duplicate** to auto fill the bug information.

Figure 3-2 Duplicating a bug



Title	Status	Days Idle	Severity	Owner	Responsible Project	Operation
Bug Bug546445212 BUG20240523501742	Fixing	0 day	● Info	[redacted]	IPD-Servicetes...	
Bug test Bug000568 BUG20240513500138	Closed	0 day	● Info	[redacted]	IPD-Servicetes...	

Figure 3-3 Modifying a bug



Duplicate

Bug Bug546445212

* Description

Attachment

Click or drag the file here to upload it.

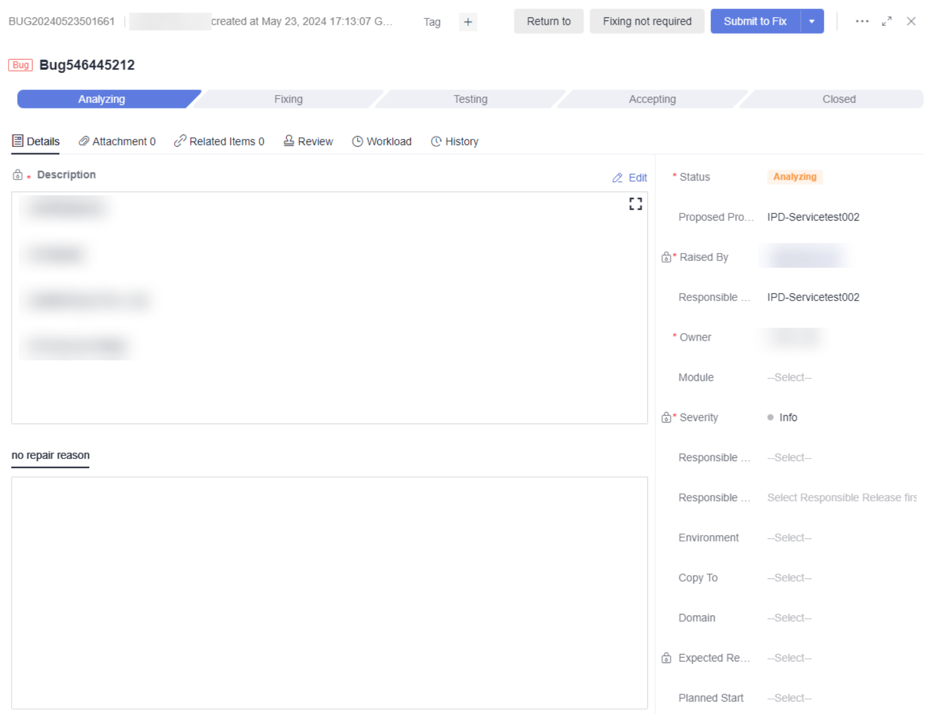
Proposed Pro...	IPD-Servicetest002
* Raised By	
* Responsible ...	IPD-Servicetest002
* Owner	
Module	--Select--
* Severity	Info
Responsible ...	--Select--
Responsible ...	Select Responsible Rel...
Environment	--Select--
Copy To	--Select--
Expected Re...	--Select--
no repair rea...	Required.

In addition, you can select a new responsible project for the duplicated bug. This means you can copy the bugs of the current project to other projects to reduce the workload of creating bugs. Please note that a duplicated bug is a completely independent data object and cannot be traced back in the current project. If you want to create an identical bug that can be traced through the current bug, you are advised to use the function described in **Collaborating on a Bug**.

3.4.3 Analyzing a Bug

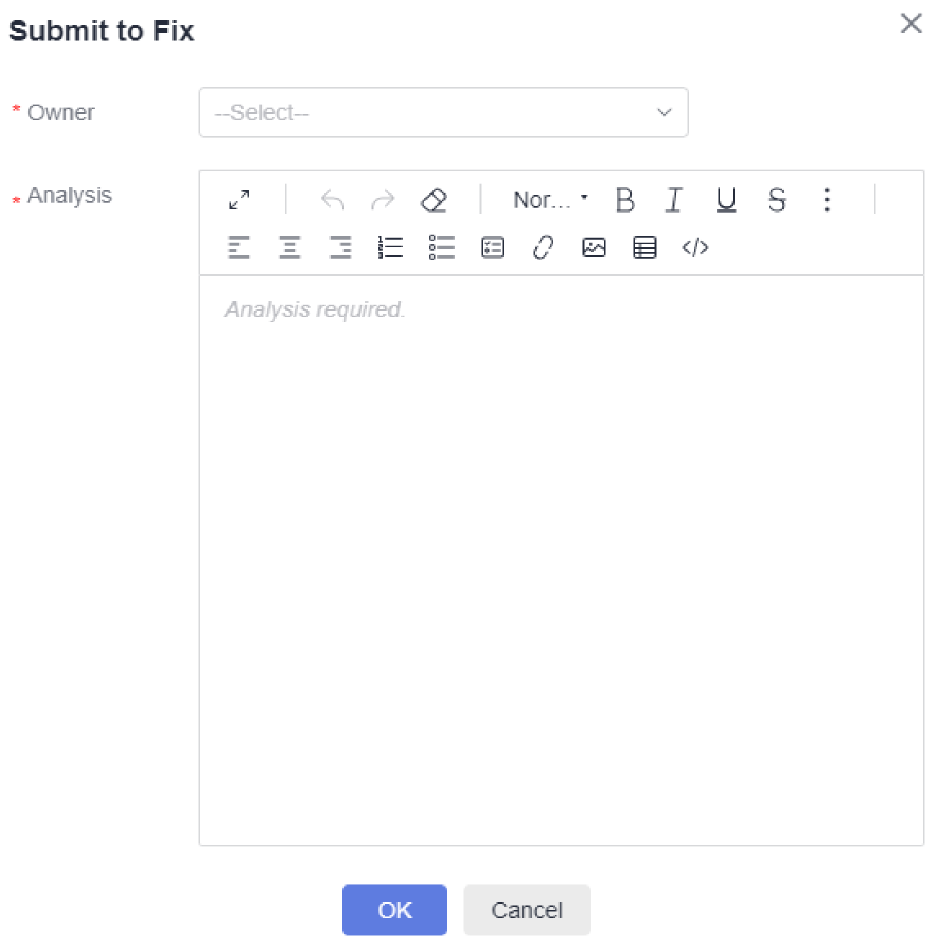
After a bug transits to the analyzing phase, the owner specified in the previous step analyzes and locates the bug. They check the bug information and locate the root cause based on the bug's environment and symptoms.

Figure 3-4 Analyzing a bug



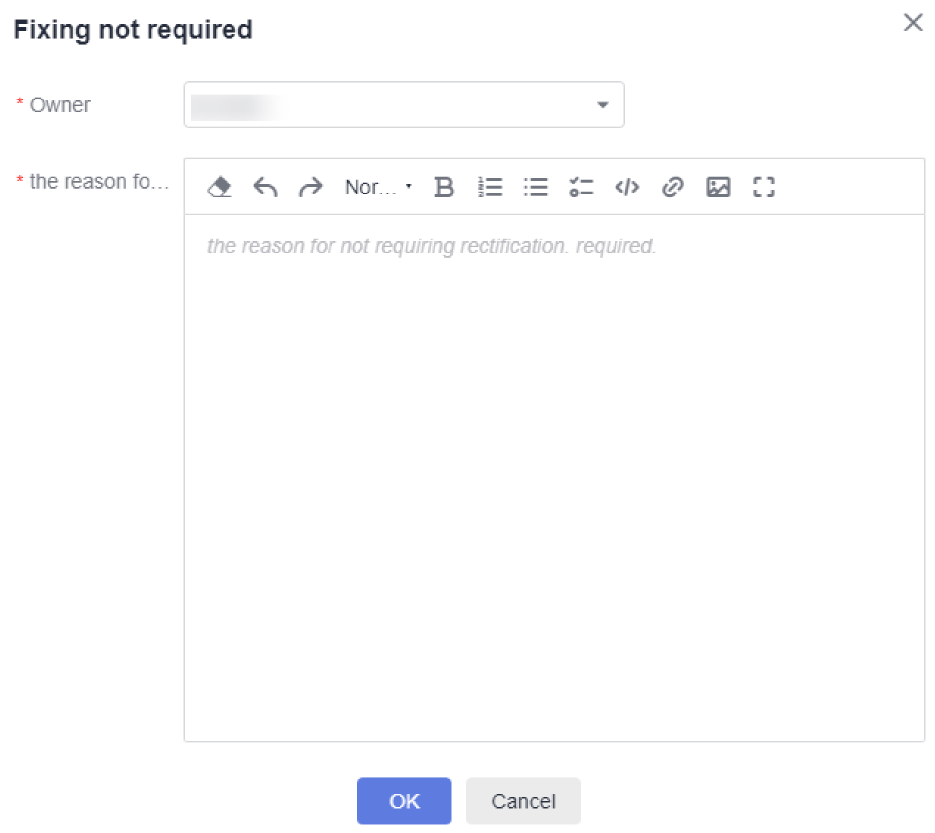
- If the owner thinks that the bug needs fixing, click **Submit to Fix** and enter the cause. Then the bug transits to the fixing phase, waiting for the corresponding owner to fix it.

Figure 3-5 Submitting a bug to R&D for fixing



- If the owner thinks that the bug does not need fixing, click **Fixing not required** and enter the reason. Then the bug transits to the testing phase, waiting for the test owner to confirm it.

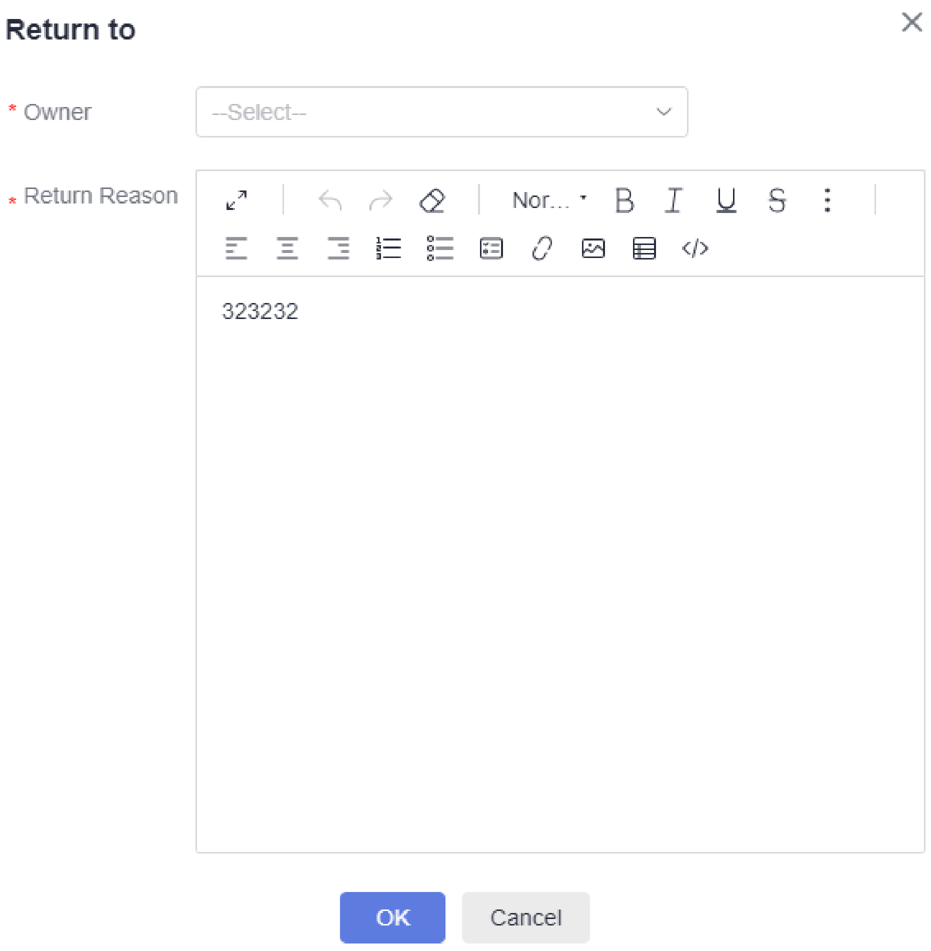
Figure 3-6 Entering the reason for needing no fixing



the reason for not requiring rectification. required.

- If the owner thinks that the bug is unclear and cannot be located, click **Return To** and enter the reason. Then the bug transits to the confirming phase, waiting for the creator to confirm it again.

Figure 3-7 Entering the reason for turning back a bug



- If the bug cannot be analyzed and located due to some objective factors, click **Suspend** and enter the reason. Then the bug is suspended.

Figure 3-8 Suspending a bug

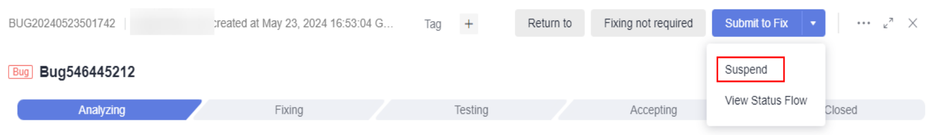
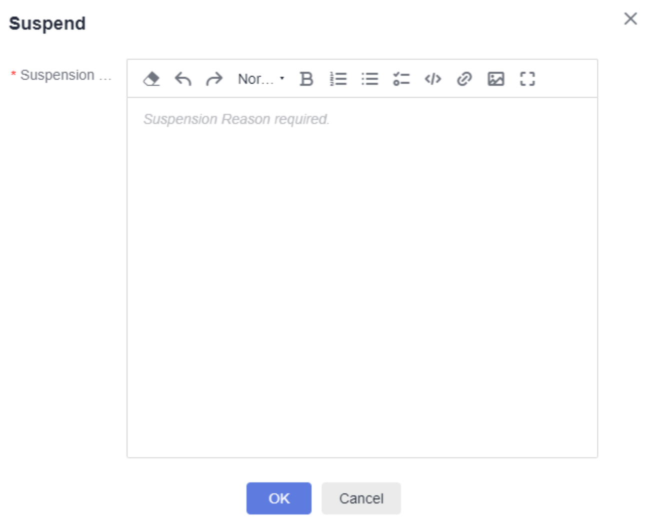
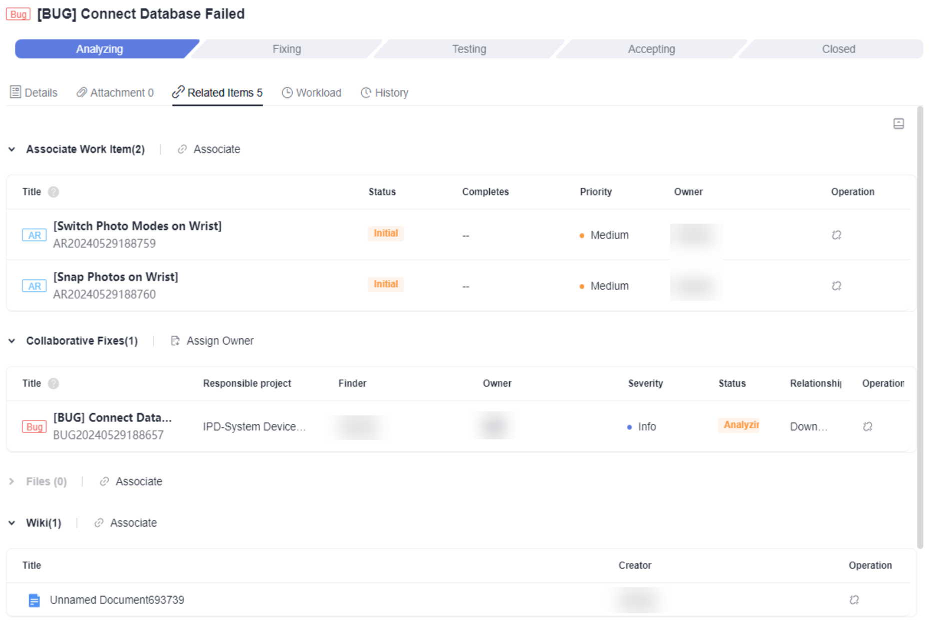


Figure 3-9 Entering the reason for suspending a bug



Once started, the bug is related to various work items. For example, if the bug may be introduced by a requirement, associate them to facilitate recording and tracing. In this case, you can associate or disassociate work items on the **Related Items** tab.

Figure 3-10 Related items of a bug



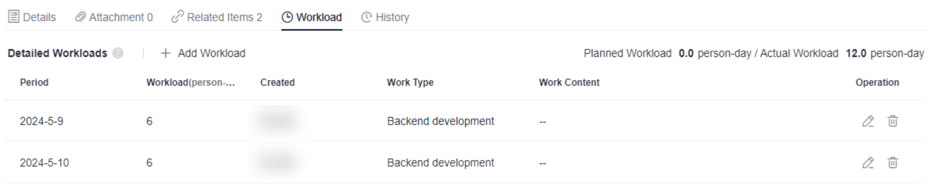
Title	Status	Completes	Priority	Owner	Operation
AR [Switch Photo Modes on Wrist] AR20240529188759	Initial	--	Medium	[redacted]	
AR [Snap Photos on Wrist] AR20240529188760	Initial	--	Medium	[redacted]	

Title	Responsible project	Finder	Owner	Severity	Status	Relationshi	Operation
Bug [BUG] Connect Data... BUG20240529188657	IPD-System Device...	[redacted]	[redacted]	Info	Analyzir	Down...	

Title	Creator	Operation
Unnamed Document693739	[redacted]	

All work items require manpower input during the operation. The information needs to be recorded to provide data basis for efficiency insights. You can add, edit, and delete the workloads of a bug on the **Workload** tab.

Figure 3-11 Workloads of a bug



Detailed Workloads

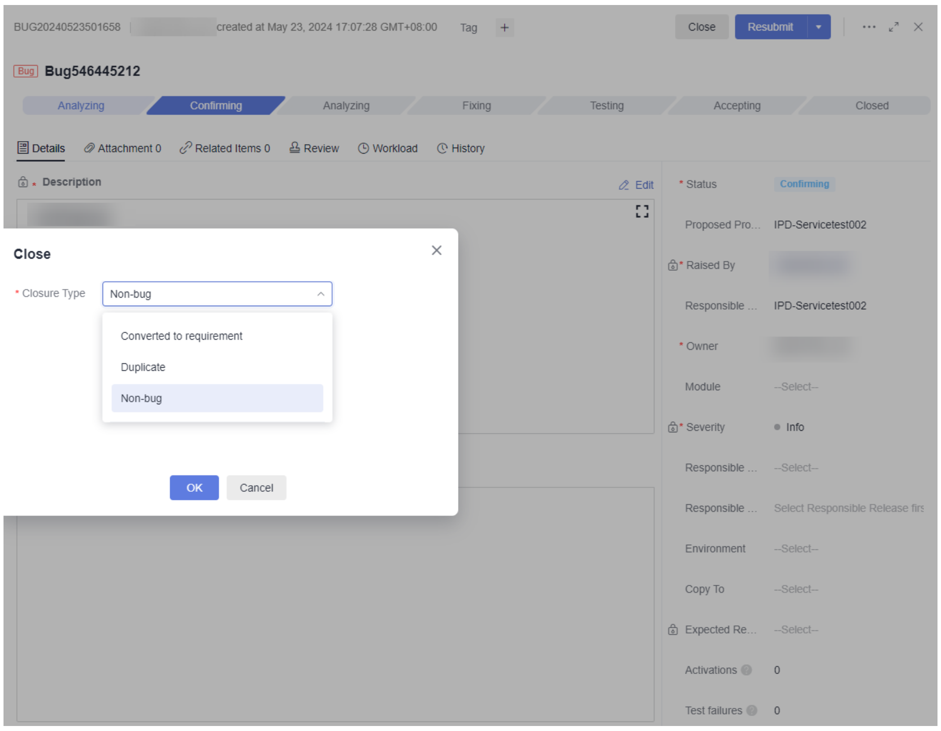
Planned Workload 0.0 person-day / Actual Workload 12.0 person-day

Period	Workload(person-...	Created	Work Type	Work Content	Operation
2024-5-9	6	[redacted]	Backend development	--	
2024-5-10	6	[redacted]	Backend development	--	

3.4.4 Confirming a Bug

If a bug is turned back in the analyzing phase, it enters the **Confirming** phase. In this case, the creator can edit the bug and submit it again. If the creator finds that the bug is not a problem, has been fixed, or cannot be managed as a bug, the creator can click **Close** to close the bug.

Figure 3-12 Confirming a bug



BUG20240523501658 [redacted] created at May 23, 2024 17:07:28 GMT+08:00

Bug546445212

Description

Close

* Closure Type: Non-bug

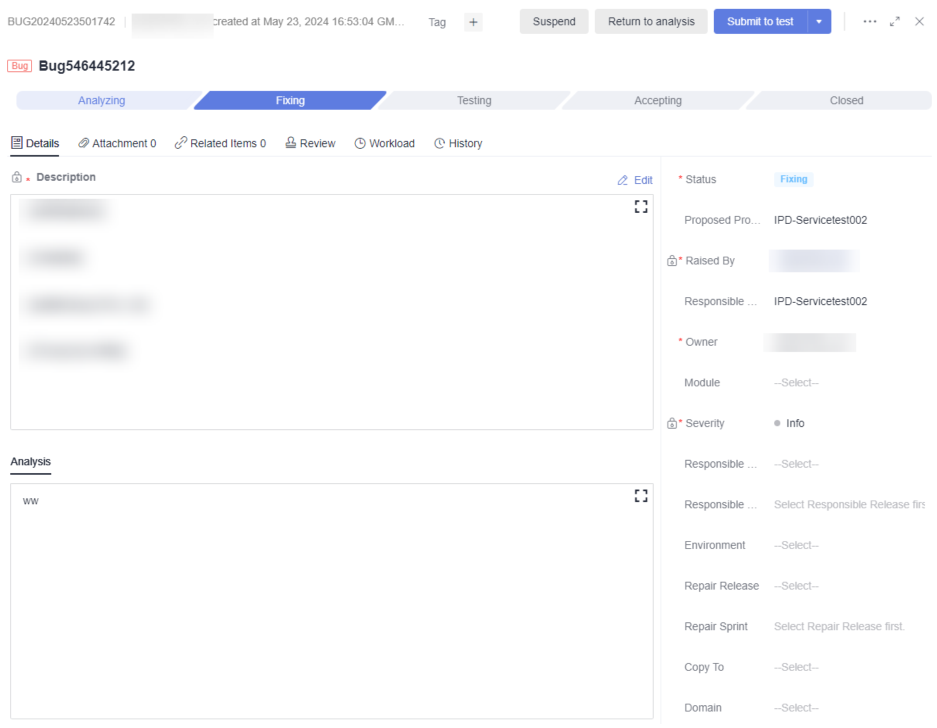
Converted to requirement
Duplicate
Non-bug

* Status	Confirming
Proposed Pro...	IPD-Servicetest002
* Raised By	[redacted]
Responsible ...	IPD-Servicetest002
* Owner	[redacted]
Module	--Select--
* Severity	Info
Responsible ...	--Select--
Responsible ...	Select Responsible Release firs
Environment	--Select--
Copy To	--Select--
Expected Re...	--Select--
Activations	0
Test failures	0

3.4.5 Fixing a Bug

After a bug transits to the fixing phase, the owner specified in the previous step fixes the bug. The fix owner schedules the bug based on its description and the root cause provided by the analysis owner. Scheduling means to plan a fix release/ sprint. The fix owner can arrange the bug in a fix plan based on key information such as the bug severity and workloads.

Figure 3-13 Fixing a bug


BUG20240523501742 | [redacted] created at May 23, 2024 16:53:04 GM... Tag +

Suspend | Return to analysis | Submit to test

Bug Bug546445212

Analyzing | Fixing | Testing | Accepting | Closed

Details | Attachment 0 | Related Items 0 | Review | Workload | History

Description Edit

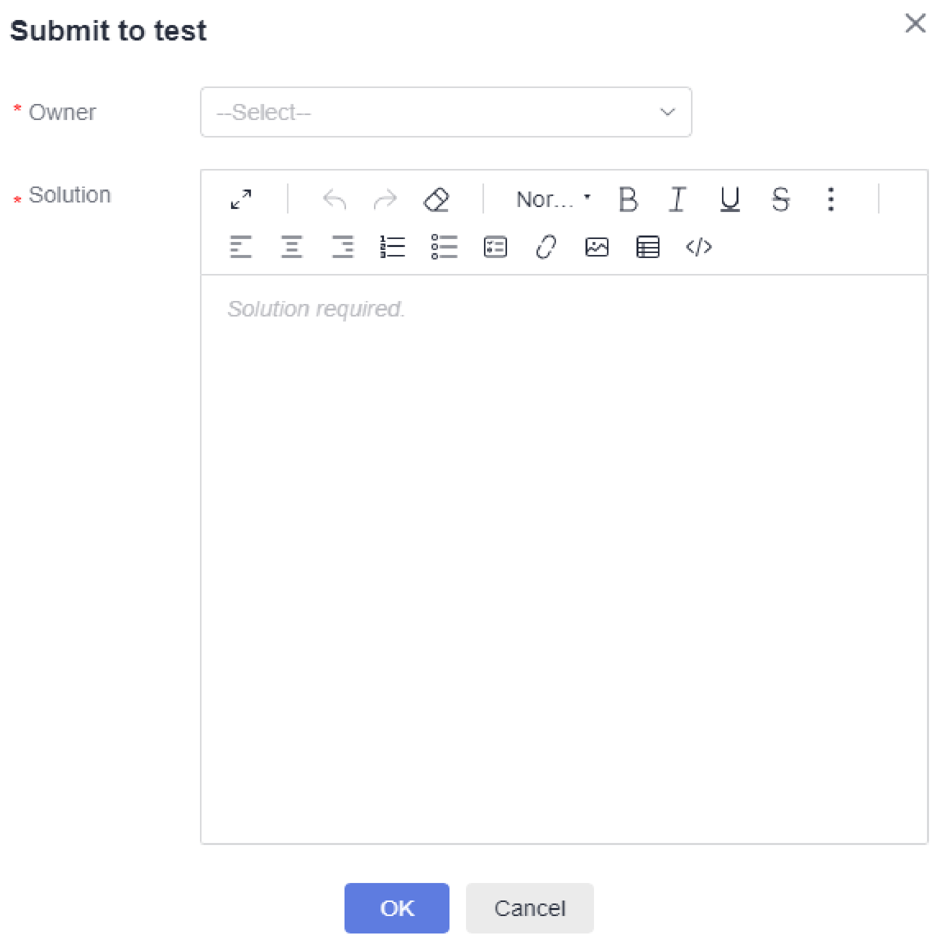
Analysis

ww

* Status	Fixing
Proposed Pro...	IPD-Servicetest002
* Raised By	[redacted]
Responsible ...	IPD-Servicetest002
* Owner	[redacted]
Module	--Select--
* Severity	Info
Responsible ...	--Select--
Responsible ...	Select Responsible Release firs
Environment	--Select--
Repair Release	--Select--
Repair Sprint	Select Repair Release first.
Copy To	--Select--
Domain	--Select--

- After the fix owner completes the fixing and self-test, click **Submit to test** and attach the fix solution. Then the bug transits to the testing phase, waiting for the tester to perform a regression test.

Figure 3-14 Filling in a fix solution

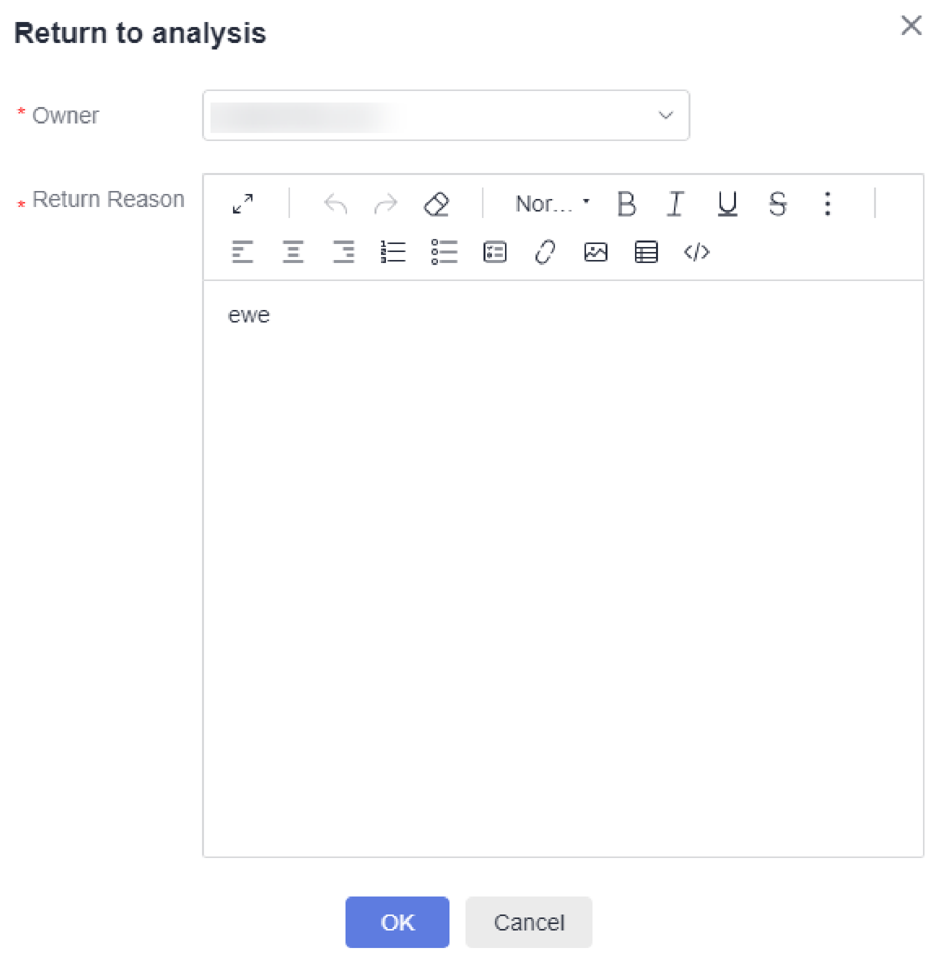

Submit to test

* Owner: --Select--

* Solution: Solution required.

- If the fix owner thinks that the root cause is unreasonable, they can click **Return to analysis** and enter the reason to turn back the bug to the analysis owner in the previous step for re-analysis.

Figure 3-15 Entering the reason for turning back a bug for analysis



- If the fix owner thinks that the bug cannot be fixed at present, they can click **Suspend** and enter the reason to suspend the bug.

Figure 3-16 Suspending a bug

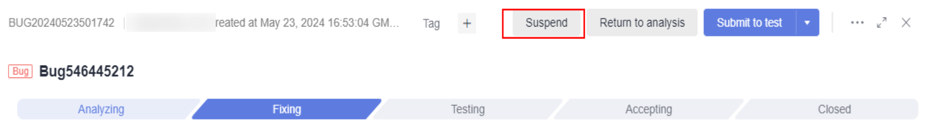
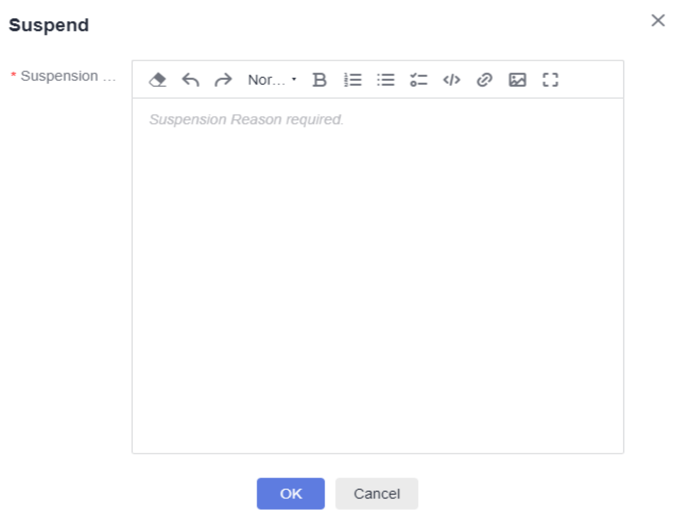


Figure 3-17 Entering the suspension reason

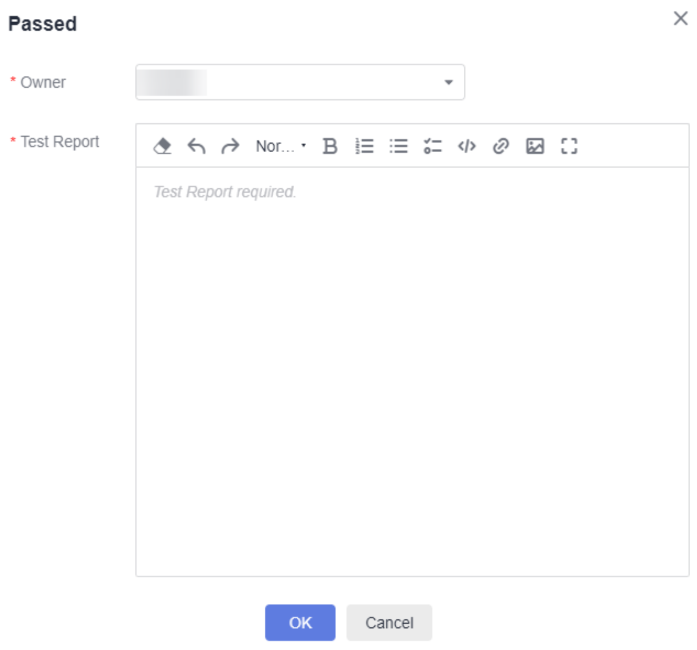


3.4.6 Testing a Bug

When fixed, a bug needs to be tested by a tester. After the bug transits to the testing phase, the tester performs a regression test based on the bug description, cause, and fix solution in the environment where the bug was discovered. Then the tester outputs a report.

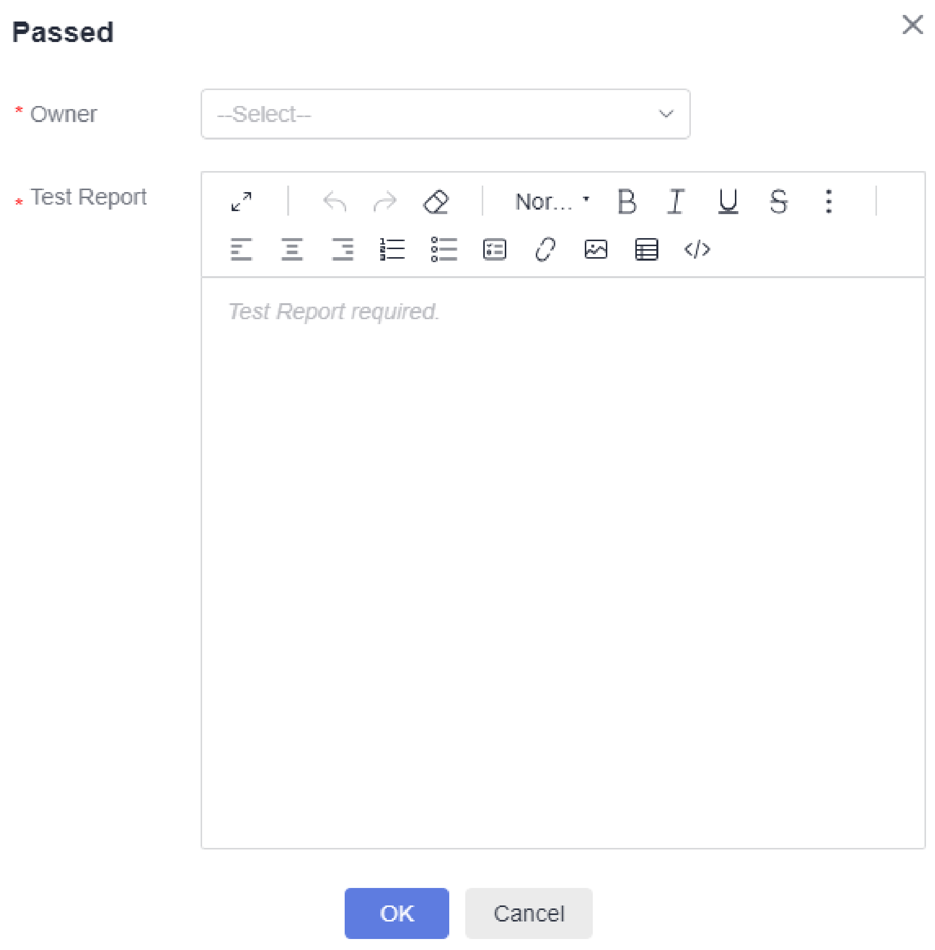
If the tester verifies that the bug has been rectified, they can click **Passed** and fill in a test report.

Figure 3-18 Filling in a test report



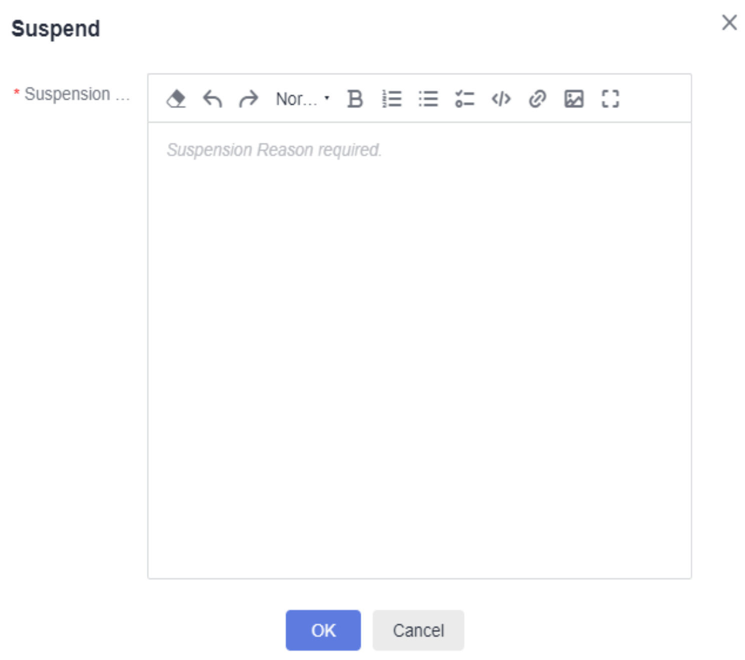
If the regression test fails, click **Return to fixing** and fill in a test report. Then the fix owner continues to fix the bug.

Figure 3-19 Entering the reason for turning back a bug for fixing



If the regression test cannot be performed for the bug at present, click **Suspend** to suspend the bug.

Figure 3-20 Entering the reason for suspending a bug



3.4.7 Accepting a Bug

After the test is passed, if the tester and the bug creator are not the same person, the bug needs to be transferred back to the creator for acceptance.

The creator performs acceptance according to the procedure for triggering the bug. If the bug has been rectified, click **Accepted** to close the bug.

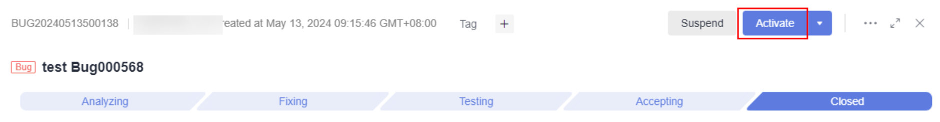
3.4.8 Closing a Bug

When a bug is accepted, it is closed. Now, the entire bug lifecycle ends.

3.4.9 Activating a Bug

As a product evolves, some bugs may occur again. If a bug occurs again, you can activate it, and fix it again based on the original analysis, fixing, and testing results. Once activated, the bug transits to the analyzing phase and starts its handling process.

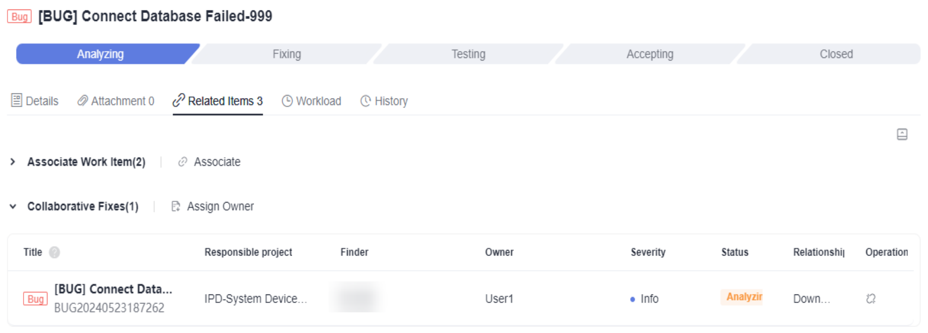
Figure 3-21 Activating a bug



3.4.10 Collaborating on a Bug

System device projects usually take a long time in R&D and involve many departments and teams, which depend on each other. Therefore, a bug of these projects often involves multiple departments. To fix the bug, you can assign it to these related parties. After receiving the bug, the downstream projects start their own bug fixing processes. All departments collaborate to promote the fixing. In addition, you can track your upstream and downstream bugs in related items at any time to check the fixing progress.

Figure 3-22 Collaborating on a bug



Bug [BUG] Connect Database Failed-999

Title	Responsible project	Finder	Owner	Severity	Status	Relationshi	Operation
Bug [BUG] Connect Data... BUG20240523187262	IPD-System Device...	[redacted]	User1	● Info	Analyzir	Down...	

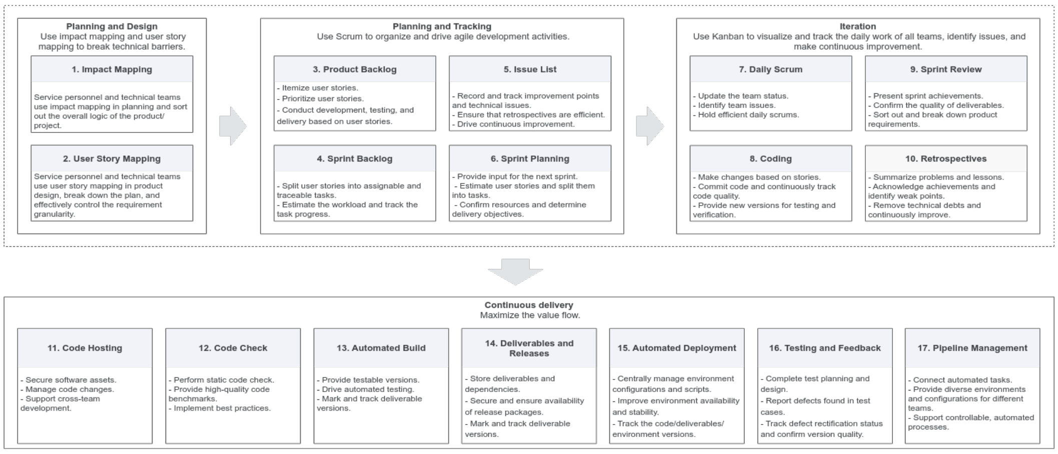
4 Huawei E2E DevOps Practice: Managing Requirements

4.1 Overview

Background

HE2E (short for "Huawei end-to-end" DevOps implementation framework is an operable, feasible, and agile development methodology developed based on years of R&D experience and industry-leading practices, as shown in **Figure 4-1**.

Figure 4-1 HE2E DevOps implementation framework



Planning and Design
Use impact mapping and user story mapping to break technical barriers.

1. Impact Mapping
Service personnel and technical teams use impact mapping in planning and sort out the overall logic of the product/project.

2. User Story Mapping
Service personnel and technical teams use user story mapping in product design, break down the plan, and effectively control the requirement granularity.

Planning and Tracking
Use Scrum to organize and drive agile development activities.

3. Product Backlog
- Itemize user stories.
- Prioritize user stories.
- Conduct development, testing, and delivery based on user stories.

4. Sprint Backlog
- Split user stories into assignable and traceable tasks.
- Estimate the workload and track the task progress.

5. Issue List
- Record and track improvement points and technical issues.
- Ensure that retrospectives are efficient.
- Drive continuous improvement.

6. Sprint Planning
- Provide input for the next sprint.
- Estimate user stories and split them into tasks.
- Confirm resources and determine delivery objectives.

Iteration
Use Kanban to visualize and track the daily work of all teams, identify issues, and make continuous improvement.

7. Daily Scrum
- Update the team status.
- Identify team issues.
- Hold efficient daily scrums.

8. Coding
- Make changes based on stories.
- Commit code and continuously track code quality.
- Provide new versions for testing and verification.

9. Sprint Review
- Present sprint achievements.
- Confirm the quality of deliverables.
- Sort out and break down product requirements.

10. Retrospectives
- Summarize problems and lessons.
- Acknowledge achievements and identify weak points.
- Remove technical debts and continuously improve.

Continuous delivery
Maximize the value flow.

11. Code Hosting
- Secure software assets.
- Manage code changes.
- Support cross-team development.

12. Code Check
- Perform static code check.
- Provide high-quality code benchmarks.
- Implement best practices.

13. Automated Build
- Provide testable versions.
- Drive automated testing.
- Mark and track deliverable versions.

14. Deliverables and Releases
- Store deliverables and dependencies.
- Secure and ensure availability of release packages.
- Mark and track deliverable versions.

15. Automated Deployment
- Centrally manage environment configurations and scripts.
- Improve environment availability and stability.
- Track the code/deliverables/environment versions.

16. Testing and Feedback
- Complete test planning and design.
- Report defects found in test cases.
- Track defect rectification status and confirm version quality.

17. Pipeline Management
- Connect automated tasks.
- Provide diverse environments and configurations for different teams.
- Support controllable, automated processes.

- Planning and Design

  Steps 1 and 2 in the diagram represent the process of product planning between service personnel (even customers) and technicians to sort out the overall product logic, implementing product planning and design, and controlling the requirement granularity and breakdown.

  – Software development solves issues and delivers value, not simply provides functions. The impact map is used to identify user requirements and root causes.

  - During microservice design, we put objectives and requirements in user stories to facilitate information exchanges among customers, service personnel, and developers. If you view only separate requirement items, you will not think from the entire solution's perspective. User stories focus on scenarios, sort out and display stages and activities in a tree structure. In this way, you will view both requirement items and overall requirement scenarios.
- Planning, Tracing, and Sprint

  Steps 3 to 10 are the main management practices in a Scrum framework process.

  - Scrum defines a relatively complete framework for agile process management. CodeArts well integrates the Scrum framework with daily activities of development teams. Main process deliverables include the product backlog, sprint backlog, potential deliverable product increments, and issue list. Core team activities include Sprint planning meetings, daily Scrums, Sprint reviews, Sprint retrospective meetings, and daily team updates.
- ContainerOps

  Start from step 11 and enter the engineering practice in a CI/CD process.

  - CI/CD is based on code configuration management. It covers not only traditional security control of code assets, concurrent development, and version and baseline management, but also reflects team collaboration and communication.
  - The pipeline connects code check (or static scanning), automated build, automated testing in all stages, and automated deployment.
  - CI/CD also covers continuous artifact management and environment management at different levels, including development, test, quasi-production, and production environments.
  - The CI/CD pipeline manages stages, environments, activities, entry and pass quality gates, and input and output artifacts in each stage.

Application Scenarios

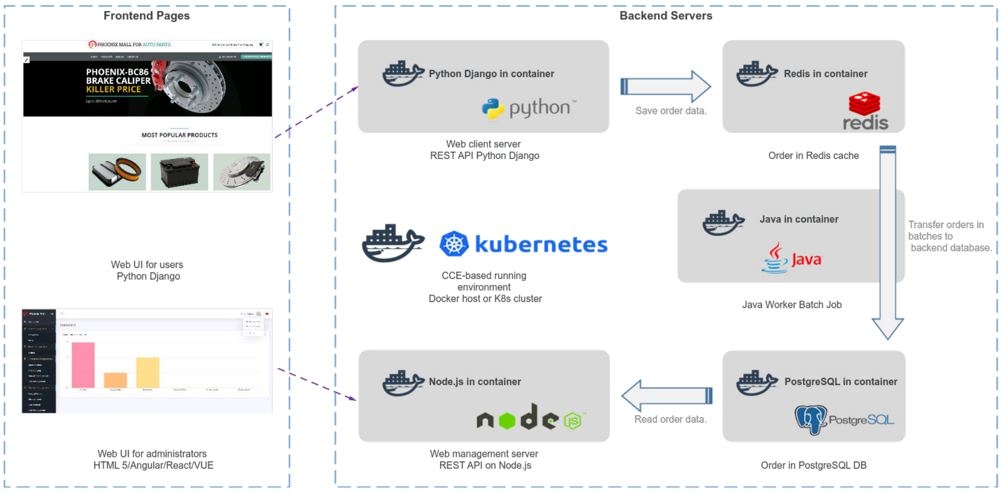
We will use the sample code for an auto part e-mall named Phoenix Mall and a DevOps full-process sample project to describe how to use CodeArts to implement the HE2E DevOps framework. This solution is applicable to agile/Scrum R&D projects.

Solution Architecture

- Phoenix Mall Architecture

  Figure 4-2 shows the architecture of the Phoenix Mall sample project.

Figure 4-2 Technical architecture of Phoenix Mall



The sample project consists of five microservice components that can be independently developed, tested, and deployed, as shown in **Table 4-1**.

Table 4-1 Microservice components of Phoenix Mall

Microservice Component	Description
Web client server (corresponding to the Vote function in the sample code)	• Service logic: Users can use a browser to access the web UI of this service. When a user clicks **Like** on a specific offering, the service saves the record of the selected offering in the Redis cache. • Technology stack: Python and Flask frameworks • Application server: Gunicorn
Web management server (corresponding to the Result function in the sample code)	• Service logic: Users can use a browser to access the web UI of this service. The statistics about **Like** clicked by users on the UI are dynamically displayed. The data is obtained from the PostgreSQL database. • Technology stack: Node.js and Express frameworks • Application server: server.js

Microservice Component	Description
Background order batch processing program (corresponding to the Worker function in the sample code)	• Service logic: This service is a background process. It monitors item records in the Redis cache, obtains new records, and saves them in the PostgreSQL database so that the management UI can extract data for statistics display. • Technology stack: .net core or Java (This service provides two technology stacks to implement the same function. You can modify the configuration and select one as the runtime process.)
Order cache	• Service logic: Persists data for the client UI. • Stack: Redis
Order database	• Service logic: Persists data for the management UI. • Stack: PostgreSQL

- Composition of the DevOps Full-Process Sample Project

  This project uses Scrum and presets some service templates. Products and services involved in this project.

Table 4-2 List of involved products/services

Service		Description
CodeArts	Req	Presets three planned and completed Sprints, project module settings, and several statistical reports.
	Repo	Presets the code repository **phoenix-sample** to store project sample code.
	Check	Presets four tasks.
	Build	Presets five tasks.
	Artifact	Stores software packages generated by build tasks.
	Deploy	Presets three applications.
	TestPlan	Presets more than 10 test cases in a function test case library.
	Pipeline	Presets five pipelines.
Other components and services	Identity and Access Management (IAM)	Manages accounts.

Service		Description
	SoftWare Repository for Container (SWR)	Stores Docker images generated by build tasks.
	Cloud Container Engine (CCE)	Deploys software packages, which is different from ECS-based deployment.
	Elastic Cloud Server (ECS)	Deploys software packages, which is different from CCE-based deployment.

Advantages

- We provide a one-stop cloud DevOps platform to manage the entire software development process, to address R&D pain points such as frequent requirement changes, complex development and test environments, difficult multi-version maintenance, and failure to effectively monitor the progress and quality.
- This service provides visualized and customizable pipeline services for CD, doubling the software rollout speed.

4.2 Preparations

Background

This section uses the DevOps Full-Process Sample Project as an example to describe how to manage requirements in a project.

This sample project uses the Scrum mode for iterative development. Each Sprint lasts for two weeks. The Phoenix Mall version has been developed in the first three Sprints, and Sprint 4 is being planned.

According to the project plan, time-limited discount and group buying activity management functions need to be implemented in Sprint 4.

Due to business and market changes, store network query is added as an urgent requirement. Therefore, this function will be developed in Sprint 4.

The following four roles are involved in the project.

Table 4-3 Project role list

Project Member	Project Role	Responsibility
Sarah	Product owner (project creator)	Be responsible for the overall product planning and product team setup.

Project Member	Project Role	Responsibility
Maggie	Project manager	Manage project delivery plans.
Chris	Developer	Develop, compile, deploy, and verify project code.
Billy	Tester	Write and execute test cases.

Prerequisites

You have **purchased CodeArts** (together with the basic edition package and CloudTest basic package).

Creating a Project

Before starting the practice, Sarah creates a project.

Step 1 **Log in to the CodeArts console**.

Step 2 Click and select a region.

Step 3 Clicking **Access Service**.

Step 4 Click **Create Project**, and select **DevOps Full-Process Sample Project**.

Step 5 Enter the project name **Phoenix Mall** and click **OK**. The project is created.

----End

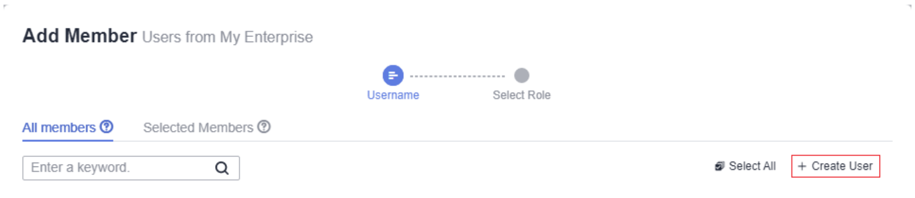
Adding Project Members

Step 1 Go to the **Phoenix** project, and choose **Settings** > **General** > **Service Permissions** > **Member**.

Step 2 Click **Add Members** above the project member list and choose **Import Users From Enterprise** from the drop-down list.

Step 3 In the dialog box that is displayed, click **Create User**. The **Users** page is displayed.

Figure 4-3 Adding members



Step 4 Click **Create User** and name them **Maggie**, **Chris**, and **Billy** in sequence.

Step 5 Return to the CodeArts page, refresh the browser, click **Add Members** above the member list, and choose **Import Users From Enterprise**. Select members **Maggie**, **Chris**, and **Billy**, and click **Next**.

Step 6 Click the **Role** drop-down list in each row, select **Project manager** for Maggie, **Developer** for Chris, and **Tester** for Billy, and then click **Save**.

----End

4.3 Managing Project Plans

Managing Requirements

Step 1 Create a work item for the new requirement.

The store network query function is a new requirement. Therefore, the product owner Sarah needs to add it to the requirement planning view.

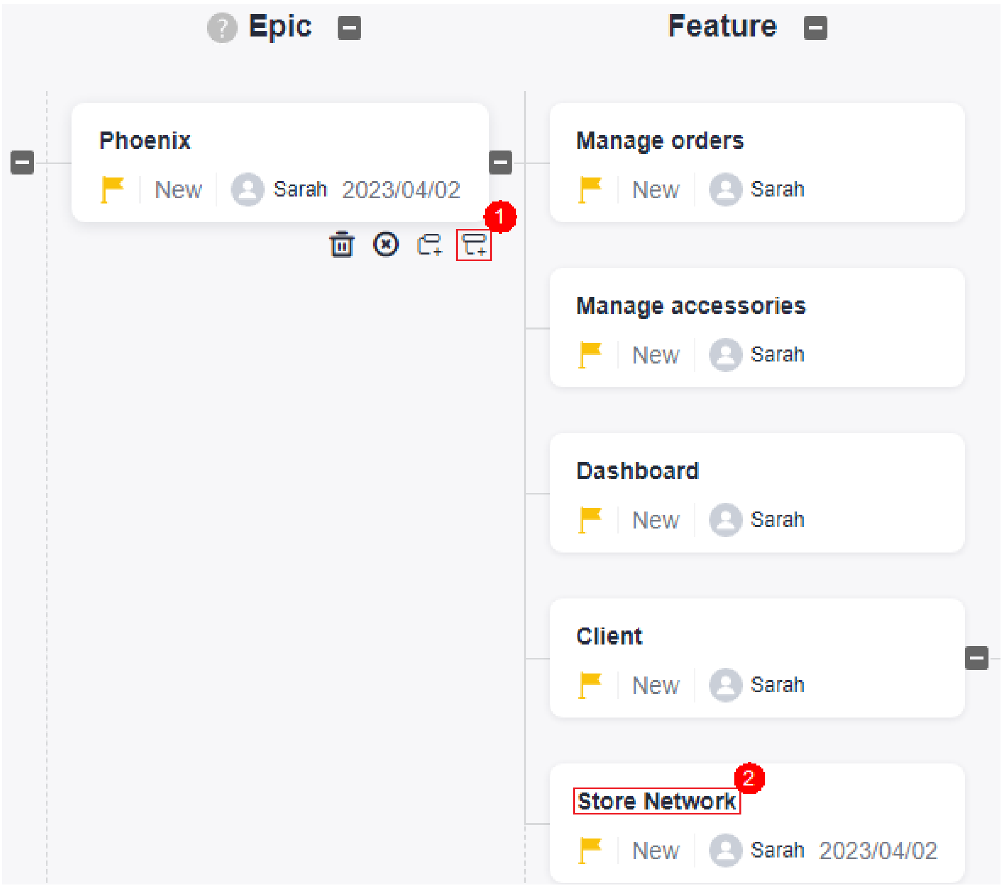
1. Go to the **Phoenix** project, choose **Work** from the left navigation pane, and click the **Plans** tab.
2. Go to the Phenix Mall mind map.

   NOTE

   If the **Plans** tab page is empty, create a mind map.

   1. Click + Create and choose **Mind Map** from the drop-down list.
      In the dialog box that is displayed, enter the name **Requirement_Planning**, and click **OK**. The mind map details page is displayed.
   2. Click **Add Epic**. In the dialog box that is displayed, select **Phoenix** and click **OK**.
3. Create a feature **Store Network**.
   a. Click under Epic **Phoenix**.
   b. Enter the name **Store Network** and press **Enter** to save the settings.

Figure 4-4 Creating a feature



4. Use the same method to add a story **User can query network of all stores** to the feature **Store Network**.

Step 2 Edit the story.

1. Click Story **to query all store network** and edit story information by referring to the following table.

Table 4-4 Story configurations

Configuration Item	Suggestion
Description	Enter **As a user, I want to query all stores so that I can select a proper store to obtain the service**.
Priority	Select **High**.
Severity	Select **Critical**.

2. Prepare a local Excel file for **Store Network**. For details about the file content, see the following table.

Table 4-5 Store network list

Branch Name	Branch Address
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. Return to the story editing page, find **Click to select a file, or drag and drop a file**, choose **Upload** from the drop-down list, and upload the list file to the work item as an attachment.
4. Click **Save**. The story details are edited.

----End

Managing Sprints

Step 1 Create a Sprint.

1. Go to the **Phoenix** project, choose **Work** from the left navigation pane, and click the **Sprints** tab.
2. Click ⊕ next to **Sprint** in the upper left corner of the page. In the dialog box that is displayed, configure Sprints by following **Table 4-6**. Click **OK**.

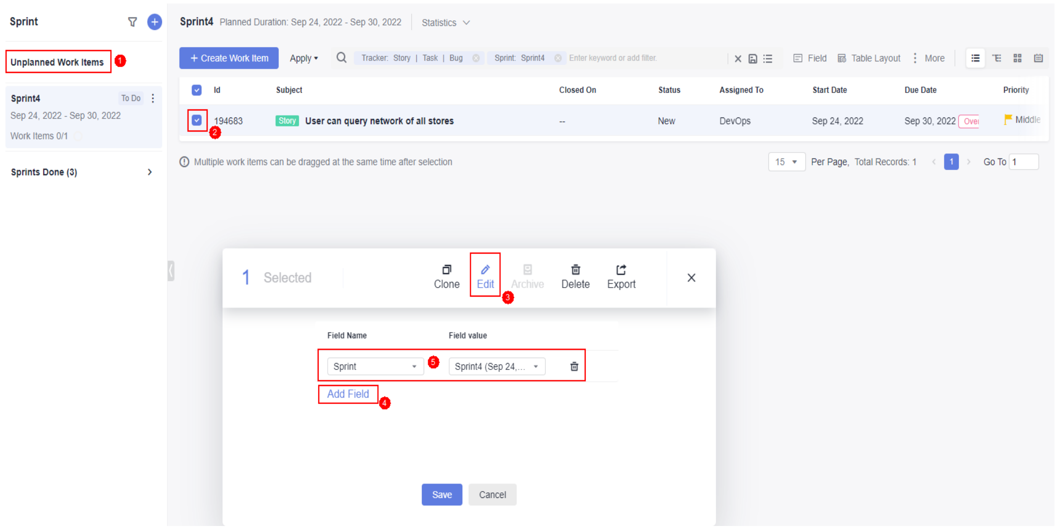
Table 4-6 Sprint information configurations

Configuration Item	Suggestion
Sprint Name	Enter **Sprint4**.
Planned Duration	Set the duration to 2 weeks.

Step 2 Plan the Sprint.

1. From the left navigation pane, choose **Unplanned Work Items**.
2. Select the following three stories in the list as planned:
   – User can query network of all stores
   – Admin can add group buying activities
   – Admin can add time-limited discounts
3. Click **Edit** at the bottom of the page.
4. Click **Add Field**.
5. Choose **Sprint** from the **Field Name** drop-down list box, select **Sprint4** from the **Field Value** drop-down list box, and click **Save**.

Figure 4-5 Planning the Sprint



Id	Subject	Closed On	Status	Assigned To	Start Date	Due Date	Priority
194683	Story User can query network of all stores	--	New	DevOps	Sep 24, 2022	Sep 30, 2022 Over	Middle

Step 3 Assign the stories.

1. Choose **Sprint4** from the left navigation pane.
2. Select all stories and set **Assign To** to **Chris** following **Planning Sprint**.

Step 4 Break down the stories.

1. Find the story **User can query network of all stores**. Click the story name.
2. In the right pane of the page, click the **Child Work Items** tab.
3. Click **Fast Create Child**. Enter the title **Frontend display-add store network menu**, assign it to **Chris**, and click **OK**.
4. Use the same method to add the task **Background management-Add store network management and maintenance module**.

----End

Monitoring and Tracking Project Status

- Daily Stand-ups to Track Task Progress

  After the Sprint starts, the project team communicates the current progress of each work item through daily stand-up meetings and updates the status.

  You can view the status of work items in a Sprint in the card mode.

  Go to the **Sprints** tab page and click ▦ to switch to the card mode. This page displays work item cards in each status. You can drag a work item card to update its status.

- Review Meeting to Accept Results

  Before the expected end time of the Sprint, the project team holds a review meeting to present work achievements of the current Sprint.

  The **Sprints** tab page provides Sprint statistics and charts. The team can easily collect statistics on the progress of the current Sprint, including the requirement completion status, Sprint burndown chart, and workload.

  Go to the **Sprints** tab page and click **Statistics** to display the progress view.

4.4 Managing Project Configurations

Managing Project Notifications

The project manager Maggie wants team members to be notified when assigned a task (work item) so that the members can handle the task (work item) in time.

Step 1 Go to the **Phoenix Mall** project, and choose **Settings** > **Work** > **Notifications** from the navigation pane.

Step 2 View default settings of the sample project displayed on the page.

We will keep default settings unchanged because they can meet requirements. If you need, modify the settings, which will be automatically saved.

Step 3 Verify the result.

When the project manager is done with breaking down the story, the developer Chris will receive the following two types of notifications.

- Direct messages: After Chris logs in to the homepage, he will view a number in the upper right corner. He can click 🔔 to view the notification.
- Email: The project member also receives an email if an email address has been configured for the corresponding user and the **Email Notifications** option has been enabled on the **This Account Settings** page.

NOTE

All members can set whether to receive email notifications. To enable email notification, perform the following steps:

1. Click the username in the upper right corner of the page and choose **This Account Settings** from the drop-down list. The **Notifications** page is displayed by default.
2. Find **Email Notifications** on the page and click **Enable**. You can click **Edit Settings** to change the email address.

----End

Customizing a Project Workflow

In the Sprint review meeting, the team demonstrates the product to the product owner and presents the test report. The product owner confirms whether the story is complete. However, the current story status does not show that the test is complete. Therefore, the tester suggests adding a status **Accepting**.

The project manager Maggie performs the following operations to add a status to a story.

Step 1 Go to the **Phoenix Mall** project, and choose **Settings** > **Work** from the navigation pane.

Step 2 Choose **Common Statuses** from the middle navigation pane. View the default work item statuses of the sample project.

Step 3 Click **Add Status**. In the dialog box that is displayed, edit the status information by referring to **Table 4-7**, and click **Add**.

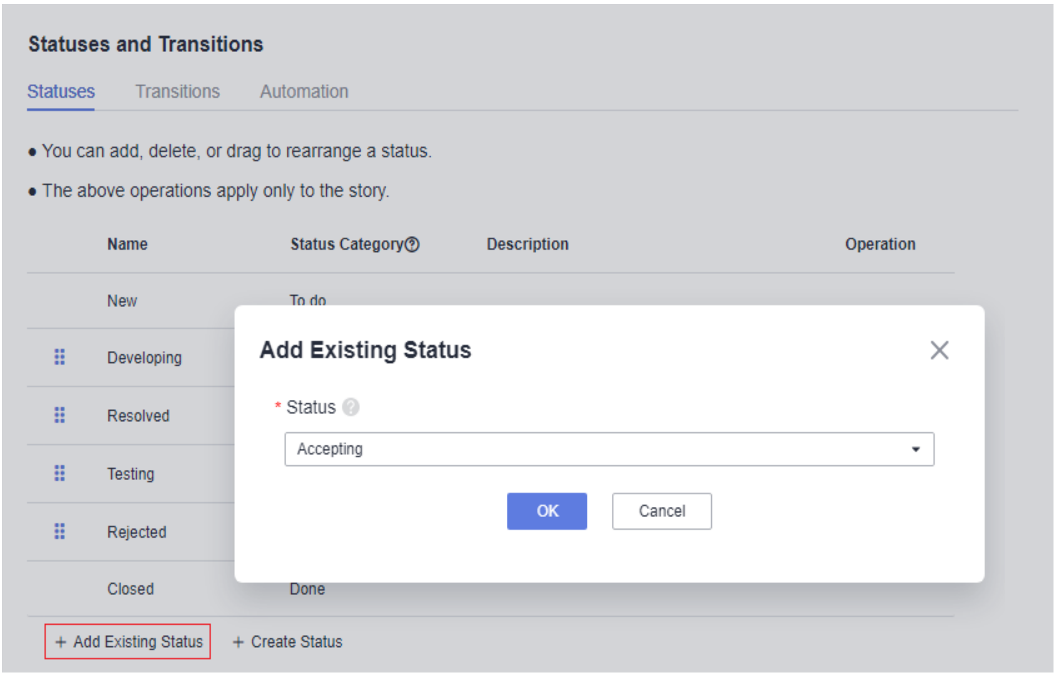
Table 4-7 Status configurations

Configuration Item	Suggestion
Status	Enter **Accepting**.
Status Category	Select **Doing**.

Step 4 Choose **Stories** > **Statuses and Transitions** from the middle navigation pane. View the default story statuses of the sample project.

Step 5 Click **Add Existing Status**. In the displayed dialog box, select **Accepting** and click **OK**.

Figure 4-6 Adding a story status


Statuses and Transitions

Statuses | Transitions | Automation

- You can add, delete, or drag to rearrange a status.
- The above operations apply only to the story.

Name	Status Category	Description	Operation
New	To do		
Developing			
Resolved			
Testing			
Rejected			
Closed	Done		

Add Existing Status

* Status: Accepting

OK | Cancel

Step 6 Drag and place **Accepting** below **Testing**.

Step 7 Verify the configuration result.

1. Choose **Work** from the left navigation pane, and click the **Work Items** tab.
2. Click any story name in the list to view story details.
3. Click the value in the **Status** column. In the drop-down list, you can see that the **Accepting** option is displayed.

----End