谈谈Bug的生命周期和Bug状态变更原则

2025-05-07

## 前言

在软件产品研发中，Bug既是测试工作最为重要的产出，也是开发人员修复问题的直接输入，更是产品质量改进的主要抓手。

在前文【】中，我们从测试人员的角度，分析了在提交Bug时应如何帮助团队更高效地去进行质量改进。

但除了Bug提交环节，在我们工作中，Bug从发现到被修复，会经历一个完整的生命周期，对应到我们提交的问题单，会呈现不同的状态。而Bug在这些不同状态间的迁移，其实反映了团队围绕Bug的协作沟通过程。实际工作中，因为Bug认定或状态设定上产生的分歧屡见不鲜，特别是在很多将Bug作为重要KPI数据的团队，测试和开发之间因为Bug产生激烈争论，时有发生。比如：

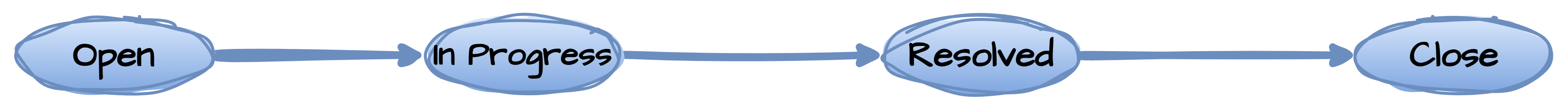
* 开发和测试对Bug认定有分歧，测试觉得是Bug，开发觉得不是问题，怎么处理？
* Bug归属产生分歧，是前端问题还是后端问题？
* Bug无法复现，应不应该关闭？
* …..等等

本文，我们就来详细梳理一下，Bug的完整生命周期，以及它在不同阶段的状态处理原则。

## Bug的生命周期及不同状态

Bug并不是凭空产生，是在测试过程中暴露出来的质量问题，从被发现到完成修复并确认无误，会经历一个过程，这个过程就是Bug的生命周期。在软件研发过程中，针对这个生命周期的管理，通常会由Bug管理系统（常用的比如Jira、禅道、Bugfree、HP QC等）来跟踪和同步每个Bug的状态，并在开发和测试人员之间起到协作桥梁的作用。

Bug生命周期的主要过程大致如下：



### 已提交（Open）

**Open** 状态，是Bug被发现以后的初始状态。通常由发现Bug的测试人员录入Bug管理系统，形成问题单，此时Bug的状态处于 **Open** 状态，也表示该Bug待处理。

另外，如果有已关闭的历史Bug，后来发现其实并未解决，也可以重新将Bug激活，此时Bug也会处于 **Open** 状态。

在该状态下，提交Bug的测试人员应该指定处理Bug的开发人员进行下一步处理，通常会根据Bug的现象，直接指定到能修复Bug的开发这里。不过一般Bug管理系统，也会设置默认处理人，对于**Open** 状态的Bug，默认处理人通常会设置为开发负责人，会做进一步更准确的判断以便重新指派。

| 状态 | 说明 | 当前负责人 | 移交处理人 |
| --- | --- | --- | --- |
| Open | 初始状态，待处理 | 测试人员 | 开发负责人或Bug归属人 |

### 处理中（In Progress）

开发人员被指派Bug后，会进入Bug的分析阶段，此时Bug不能确定是否能被修复，所以会进入**处理中**状态，经过分析后，可以对Bug进行修复操作，或发现并不是自己可以完成修复的问题，再将bug重新指派给实际应该修复bug的开发人员。

这里如果存在Bug在开发人员间的移交，Bug状态会保持在 **In Progress**, 仅仅当前处理人会发生变化。

| 状态 | 说明 | 当前负责人 | 移交处理人 |
| --- | --- | --- | --- |
| In Progress | Bug进入分析，通常针对较复杂Bug | 被指派开发人员 | 实际Bug修复人 |

### 已解决（Resolved）

Bug在经过开发人员的修复后，会标记为**已解决**。该状态其实代表测试人员可以对Bug进行验证。

这里需要注意，很多开发人员会有的一个误区，就是我把代码进行修改或者做过自测，就是**Resolved**。其实并不是，**Resolved** 状态的重点是被移交的处理人，也就是**测试人员是可以进行验证**的。

因为实际工作中，测试人员对产品进行测试，会有测试轮次的概念，并不是随时都可以测试。 所以只有当**修复的代码已经进入下一轮的提交测试**，才应该将状态置为 **Resolved**。否则当前测试环境中，被修复的代码尚未部署，当然实际并无法完成验证。

（*当然更完善的Bug管理系统，会再增加一个****待验证****的状态，这时已解决就只代表开发完成了代码的修复，而待验证才是重新提测。本文重在说明常用必要状态，不再过多扩展*）

| 状态 | 说明 | 当前负责人 | 移交处理人 |
| --- | --- | --- | --- |
| Resolved | Bug已经完成修复，标记为已解决，即待验证 | Bug修复人 | 测试人员 |

### 已关闭（Closed）

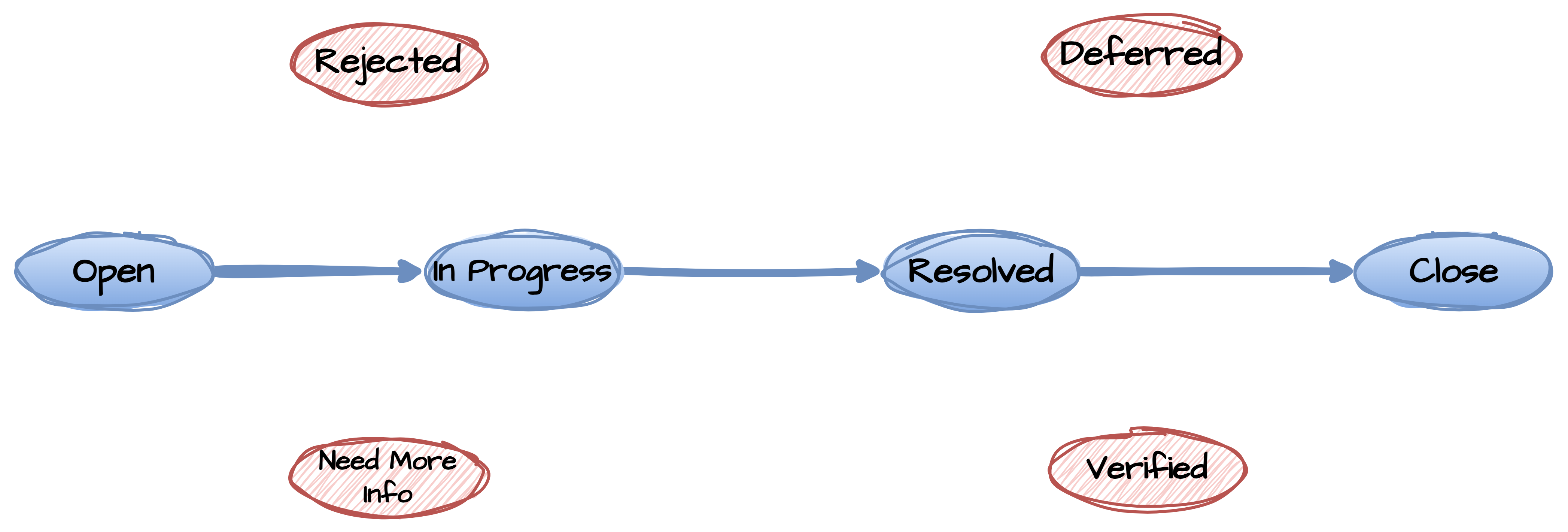
经过验证，确认Bug已被修复后，Bug可以置为 **已关闭** 状态。该状态代表Bug所反映的质量问题在当前产品版本中已不复存在。

关闭Bug应该是一个很严肃的事情，通常应该由Bug的提交人进行确认后才可关闭。当然特殊情况下，产品的决策团队（CCB）也可决定Bug是否可以进行关闭。

另外，关闭的Bug需要重新激活的情况也时有发生，比如验证时场景考虑不够完整，环境问题导致误关或者是提测版本切换后，已修复的Bug又被重新改错，或是错误代码又被重新合入等等，这时就会牵涉Bug重开（ReOpen）操作

| 状态 | 说明 | 当前负责人 | 移交处理人 |
| --- | --- | --- | --- |
| Closed | Bug已确认修复，版本中已不存在Bug对应的质量问题 | Bug提交人 | 无 |

除了上面几个最主要的Bug生命周期状态外，实际工作中，还会存在一些特殊情况，也会对应到不同的Bug状态



### 已拒绝（Rejected）

测试过程中，测试人员出现误判，或者环境配置有误时，还是比较容易出现测试提交的Bug实际并不是质量问题的情况，开发人员在经过分析后，认为不是bug，此时就可以将Bug置为 **已拒绝** 状态。这时Bug会回到测试人员手中，进行确认，若认可误报，可由提交人进行关闭操作。

| 状态 | 说明 | 当前负责人 | 移交处理人 |
| --- | --- | --- | --- |
| Rejected | 经分析Bug属于误报，不反映质量问题 | Bug分析人 | Bug提交者 |

但实际中，针对被拒绝Bug，很多时候是测试和开发人员对需求或是否质量问题的分歧导致，在分歧无法自行弥合的情况下，正常的操作应该是： 测试人员重新提交Bug，但Bug不在指派到开发人员，而是指派到产品决策团队CCB（其实主要是BA或PO），由CCB进行仲裁，根据仲裁结果来判断是否需要进行修复或关闭。

### 已验证（Verified）

测试人员验证Bug无误后，通常可以进行关闭操作将状态置为**Closed**，但更完整的生命周期，其实还又一个**已验证**的状态。这个状态的使用，通常出现在验证Bug的测试人员和提交Bug的提交人并不一致，验证人对Bug实施验证后，不能代表提交人，确认Bug可关闭。

另外一种情况，就是更严谨的大型软件研发流程，测试环节可能也包括多个。子系统级别的测试团队测试完成后，还会有大系统级别的系统测试或全面验收，在子系统级别的测试进入后续测试后才会将状态置为**Closed**

| 状态 | 说明 | 当前负责人 | 移交处理人 |
| --- | --- | --- | --- |
| Verified | Bug经过验证已被修复 | Bug验证人 | Bug提交者 |

### 待补充信息（Need More Info）

这个状态是针对测试人员提交的Bug信息不完整或不足以对Bug完成分析，开发人员难以理解具体的Bug现象等情况。这时处理人将问题单置为 **Need More Info** 状态，返回给Bug提交人补充相关信息。

| 状态 | 说明 | 当前负责人 | 移交处理人 |
| --- | --- | --- | --- |
| Need More Info | Bug单信息不充分，需要补充更详细的信息 | Bug分析人 | Bug提交者 |

### 已延期（Deferred）

延期一般针对下面几种情况： 1. Bug难以复现，较难分析出产生原因 2. Bug修改难度大，需要对产品进行较大改动，且存在规避方案 3. Bug优先级较低，不影响主要功能，在版本发布前还有其他更重要问题需要解决

针对以上情况，经过产品CCB团队决策，可以将Bug延期处理。同时Bug的负责分析人，还是需要继续针对Bug做分析或后续进行修复。

但这里需要注意，**Deferred** 状态不代表Bug不是问题，在测试团队的质量分析时，还是应该将该状态的Bug视作有效Bug看待。也就是这类Bug还是会影响到版本最终的质量评估结论。

| 状态 | 说明 | 当前负责人 | 移交处理人 |
| --- | --- | --- | --- |
| Deferred | Bug因特定原因无法在当前版本解决，经项目CCB团队决策认可后，可将Bug保留在版本中，留待后续解决 | CCB决策团队 | Bug分析人 |

以上就是我们针对Bug生命周期中会出现的主要的一些状态定义及相关处理原则的介绍。除了上面列出的这些Bug状态外，更完整的Bug生命周期还包括有：

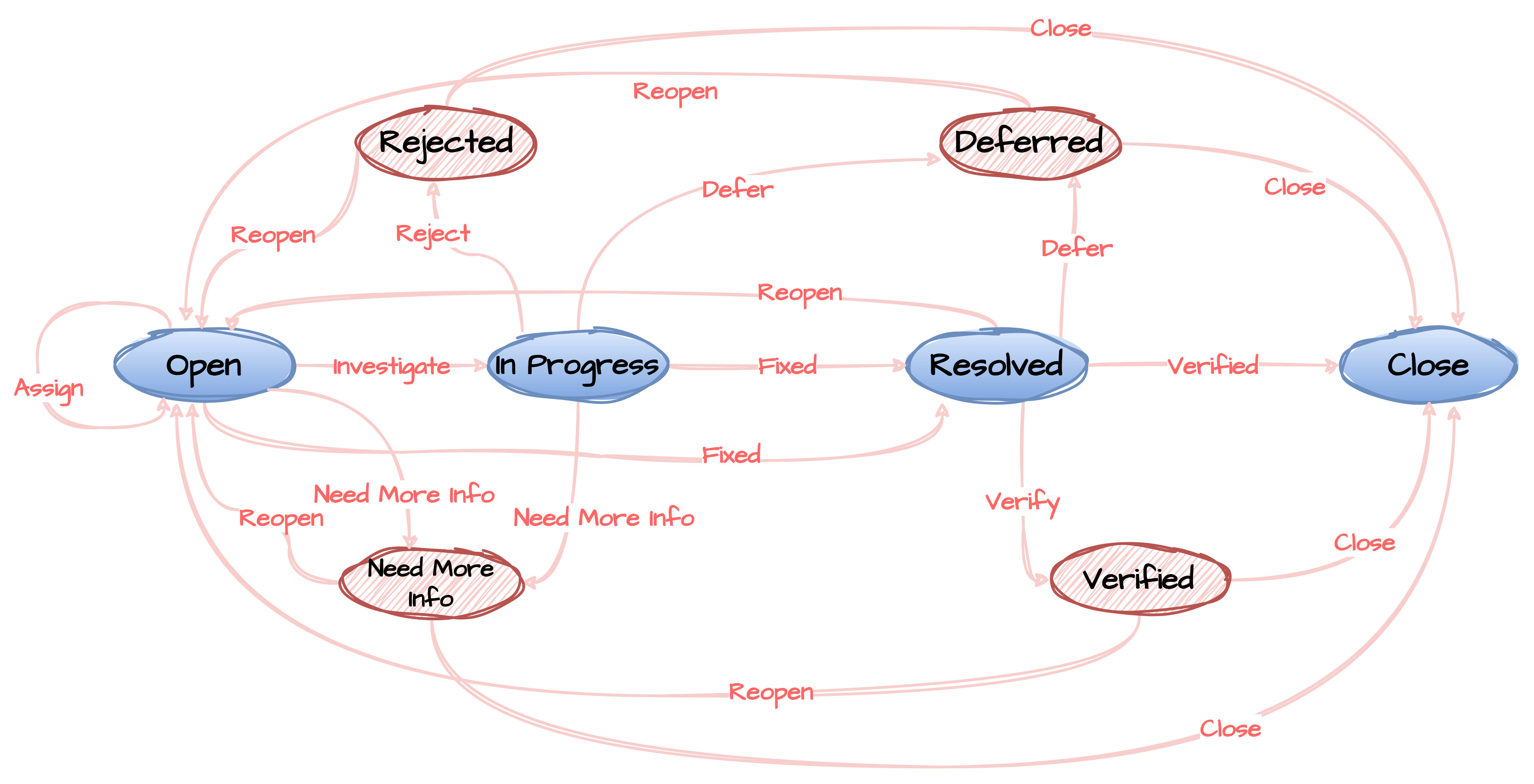
* **待验证：** 已修复，等待版本提测
* **调研中：** 针对比较复杂的Bug，牵涉技术选型、方案对照等较多研究工作时，可在处理Bug前定义该状态
* **已调研：** 针对上面进行了方案研究的Bug，但尚未确定修复方案时，Bug处于该状态
* **验证中：** 对于测试场景比较复杂的Bug，需要进行较长时间的验证，比如性能类的问题，在完成验证前，可将Bug置于该状态

当然，流程是死的，良好、高效的团队协作最重要的还是团队中不同角色的互信互助。

不过我们理解正确流程中的不同环节和正确处理的原则，也有助于我们在出现问题或分歧时，减少互相争论、扯皮的内耗。

## Bug状态迁移

基于上述不同状态的分析，一个比较完整Bug状态迁移图如下，供参考。



如需以上完整大图，可回复 Bug状态 获取。

另回复大纲 可查看秋草测试技能全栈提升课详细目录。回复 进群可进入测试交流群和小伙伴们分享测试技术、交换资讯。