|  |
| --- |
|  |
| 开发过程 |
|  |
|  |
| ©MetaShare Inc. |
| **4/4/2018** |

|  |
| --- |
| 本文eService项目的开发过程 |

目录

[1. 引言 1](#_Toc511203094)

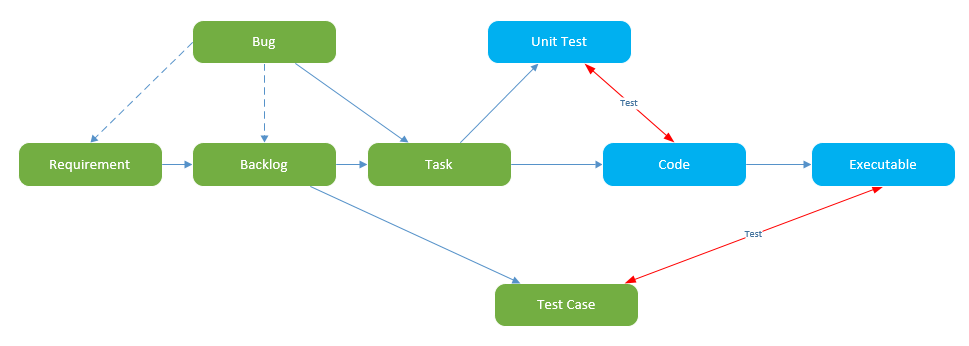
[2. 开发过程 1](#_Toc511203095)

[3. Trac Guideline 2](#_Toc511203096)

# 引言

Eservice项目开发过程进行了调整，Adam结合Trac对开发过程进行了汇总整理，后面我们在开发过程中不断反馈和完善。

# 开发过程



* Requirement (aka, User Story or Use Case), 是用户提出的需求
* Backlog，经过对需求的分析，通过设计，形成的一系列工作目标，每个工作目标应该只局限于一个系统或子系统或组件或模块的系统边界内。如有必要，可以根据实际需要形成层次结构。一个需求所产生的backlog的集合逻辑上与需求是等价的，并可以相互验证。
* Test Case. 根据Backlog中的目标定义，形成可操作的目标检测流程。通过Test Case的设计，可以对应设计数据结构、交互界面。
* Task。通过在Backlog基础上的在系统边界内的进一步分析和设计，可以形成程序员可执行的任务，任务的粒度应控制在单个子系统环境可测试的规模，测试要求能够自动化运行。
* Code。代码包括程序代码、数据库脚本及实际。
* Executable。最后形成的可执行程序。可执行程序必须要保证通过Test Case定义的测试。
* Bug。这里指的是在系统交付运行后，用户反馈的系统实现缺陷。它可以是分析、设计及编码中的缺陷导致的，在修补此缺陷时，同步修补更新相关文档。

## 过程

1. 获得需求。
2. 分析需求，识别Use Case.
3. 分析Use Case，识别实体、工作流及状态。这个时候可以用决策树来辅助分析，形成判定表。
4. 细化Use Case，或者是识别Use Story（或者叫Use Case Scenario）。
5. 设计Test Case，这时候可以用跟踪表来保证测试的覆盖率。Test Case并不需要与User Story一一对应，只要保证覆盖率就可以了，这点需要自己去体会和学习。Test Case可以用来实现系统测试，以实现测试自动化。根据Backlog中的目标定义，形成可操作的目标检测流程。通过Test Case的设计，可以对应设计数据结构、交互界面。
6. 设计实现任务
7. 设计实现方案，同步设计单元测试。注意单元测试是组件级的，不是系统级的。这里会产生复用，效率会更高。
8. 代码实现。
9. 倒序测试。即先通过单元测试，再进行系统测试，最后进行用户测试。
10. 回归测试。保证其他已有的逻辑不会因为新的实现而被破坏。即原有测试应该正常通过。

# Trac Guideline

1. 对文档本身的修改和完善使用修改功能，Trac本身具有版本管理功能，比较容易进行对比
2. 关于文档内容的交流使用回复功能。它具有线程的连续特点，比较容易理解上下文。
3. 在分析过程产生的非代码文档，如流程图、交互图、类图、状态图、数据结构定义、测试数据定义等，需要做为附件上传到相关的Trac 记录中
4. 谁最后一个改动Ticket，把Owner改成自己，这样便于相互沟通。
5. 所有的Ticket的Version必须正确填写，以便于管理和查询

# 对开发过程的思考

## 存在的问题

1. 没有单元测试；
2. 没有集成测试，最好能像eService的测试架构那样，实现集成测试，减少错误率；
3. 没有专门的测试人员，我们需要培养一个能够非常了解整个Sanjel业务的测试员，不仅仅是测试目前开发的一小部分功能，更多的是测试对其他子系统的影响和依赖；
4. 对需求的上下文理解不够透彻，一般只清楚目前开发的一小块业务，而这块业务对其他系统的影响不清楚；
5. 整个团队对架构理解不够深，应加强这方面的培训工作；
6. 涉及到重构类的工作，因为涉及面很广，eService这么庞大的系统群，要考虑各个子系统之间的依赖和影响，我们团队在重构过程中对整个依赖关系理解不够；
7. 没有一个比较全的需求文档，不连贯，不易跟踪和更新，导致在修改之前功能时对上下文理解缺失；

## 理想的开发过程

1. 需求获取：根据客户提出的需求，整理需求文档（后面应该是在一个大的需求文档上不断更新，有助于了解上下文），在此阶段BA要与客户充分的交流，遵循5W2H原则完全了解每个需求的上下文，以及涉及到的其他各个子系统的功能到底有哪些？主要包括：
   1. 需求描述
   2. 应用场景描述（主要分为几类场景）
   3. 涉及到的子系统，相关的功能点描述，尤其是服务调用
   4. 如果业务流程复杂需要画出业务流程图，完成初始的Domain Model
2. 识别UseCase：
   1. BA根据业务流程识别出本次开发所涉及的UseCase
   2. 在分析UseCase的过程中识别出更多的业务实体，不断完善Domain Model
   3. 给客户和开发团队一起讲解整个业务流程和Domain Model
   4. 修正错误记录问题
   5. 如有必要返回a继续
   6. 整理UseCase并对每个UseCase进行细化
   7. Tester测试员根据UseCase写测试用例；
3. 录入MDD：
   1. BA将Domain Model录入到MDD平台，作为该系统的CIM模型；
   2. Developer检查已录入的CIM模型，并更新PIM和PSM；
   3. 生成Code，交给BA使用并测试（理想的情况是生成系统包括单元测试和集成测试）
   4. 生成系统由BA协助客户去使用，并反馈问题，返回a继续迭代
4. 二次开发：基于生成的系统按照UseCase进行开发
   1. Developer根据TestCase进行开发
   2. 开发完一个功能交由测试人员去测试
   3. 测试人员进行集成测试
   4. 系统测试
   5. 部署测试版