# 软件系统开发流程

民航数据通信有限责任公司

杨敏行

yangmx@adcc.com.cn

2020-12

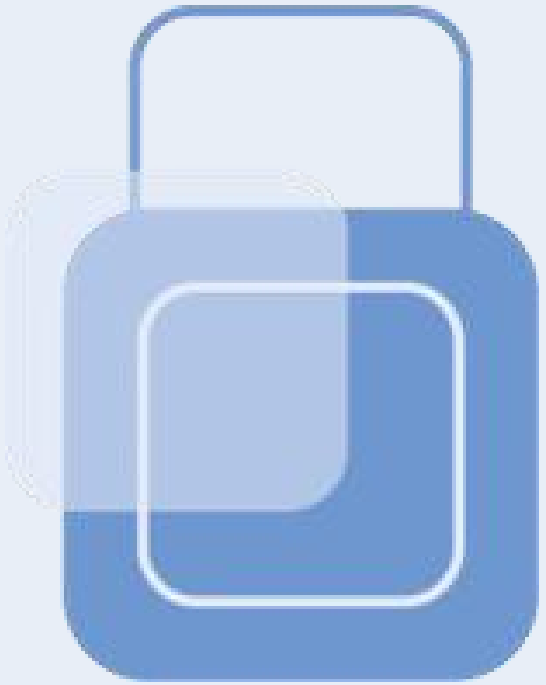
# 目录/CONTENTS



# 01

Part One

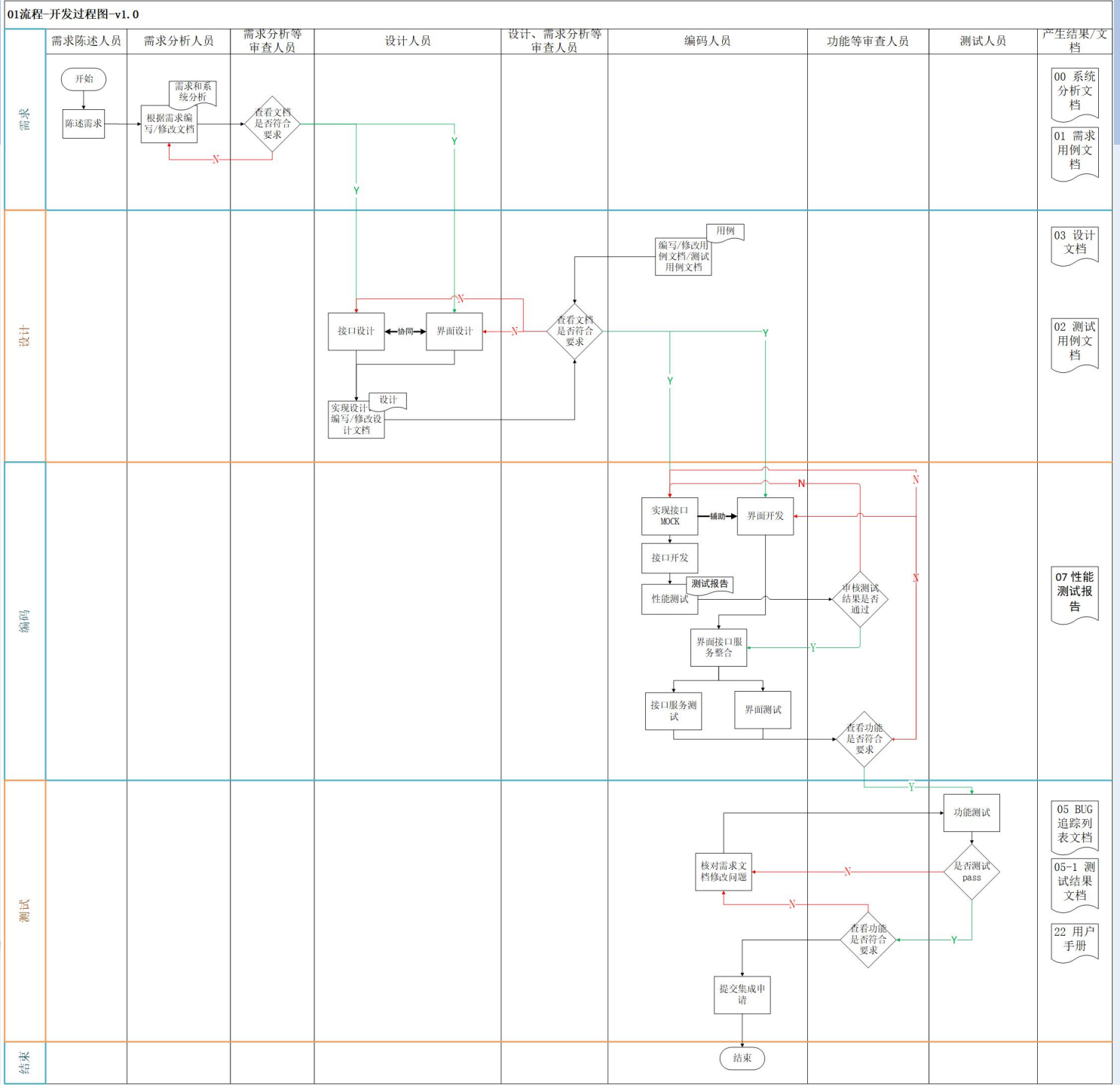
# 软件开发流程概述



# 软件开发流程概述

- 主要阶段：
  - 需求分析：初步了解用户的需要，形成需求用例文档。
  - 概要设计：基于需求用例，从用户场景、领域对象、进程交互、开发架构、部署结构5个方面4+1设计视图。
  - 编写测试用例：在详细设计之前就编写测试用例。
  - 详细设计：从接口、算法、前端页面、存储四个方面对功能进行详细设计。
  - 编码实现：IDEA开发代码，Git管理代码，Sonarqube保证代码质量。
  - 集成测试：按照产品周期提交功能到集成版本里，并按流程测试。
  - 部署和维护：按照流程对系统进行部署和维护。

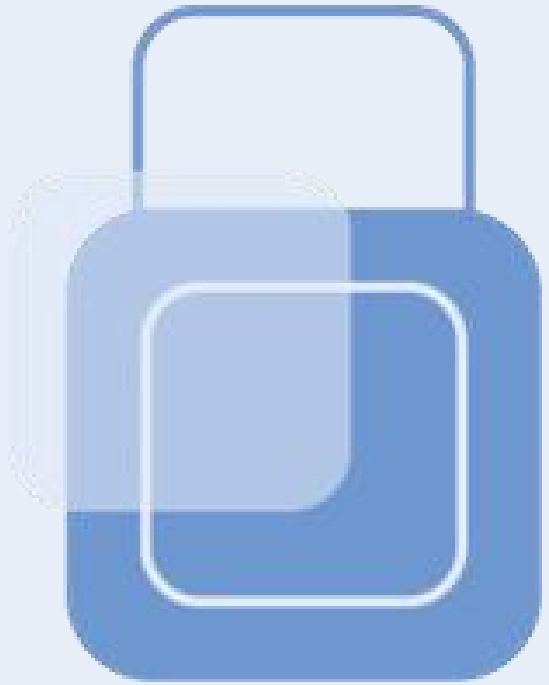
# 软件开发流程概述


01流程-开发过程图-v1.0
需求陈述人员
需求分析人员
需求分析等审查人员
设计人员
设计、需求分析等审查人员
编码人员
功能等审查人员
测试人员
产生结果/文档
需求
设计
编码
测试
结束
开始
陈述需求
需求和系统分析
根据需求编写/修改文档
查看文档是否符合要求
N
Y
00 系统分析文档
01 需求用例文档
用例
编写/修改用例文档/测试用例文档
接口设计
协同
界面设计
查看文档是否符合要求
实现设计编写/修改设计文档
设计
03 设计文档
02 测试用例文档
实现接口MOCK
辅助
界面开发
接口开发
性能测试
测试报告
审核测试结果是否通过
界面接口服务整合
接口服务测试
界面测试
查看功能是否符合要求
07 性能测试报告
功能测试
是否测试pass
核对需求文档修改问题
查看功能是否符合要求
提交集成申请
05 BUG追踪列表文档
05-1 测试结果文档
22 用户手册
结束

# 02

## 需求分析

编写需求用例文档



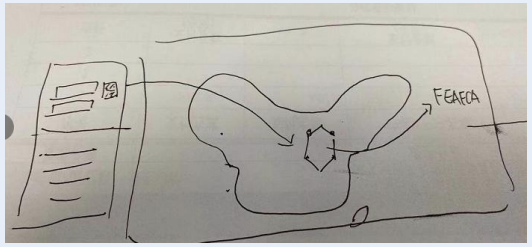
# 需求分析 – 需求定义

- 需求分析的主要目的是：明确用户对业务功能的需求。
  - 功能需求：定义开发人员必须实现的软件功能，使得用户能完成他们的任务，从而满足业务需求。例如某类业务数据的系统显示功能。
  - 非功能需求：定义除功能需求以外的特性。例如安全性、可靠性、扩展性等。
- 主要工具：Mindjet MindManager、 Microsoft Visio、 Microsoft Word
- 主要产出文档：需求用例文档

# 需求分析 – 主要步骤

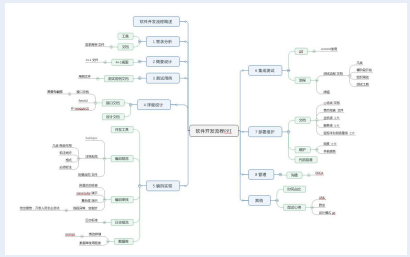
- 需求分析的主要步骤。
  - 问题识别：理解用户提出需求的原因和期望效果。
  - 分析与综合：对问题进行分析、然后在此基础上整合出解决方案。
  - 编写需求用例文档：对已经确认的需求进行文档化描述。
  - 需求分析与评审：对功能的正确性、完整性和清晰性，以及其他需求给予评审。

# 需求分析 – 主要步骤




◆ 理解需求

可以自定义一个区域，统计经过自定义区域的航班流量，解决统计航班流量时，只能用“固定空域”区域的问题。



◆ 分析问题：

FEA/FCA区域创建

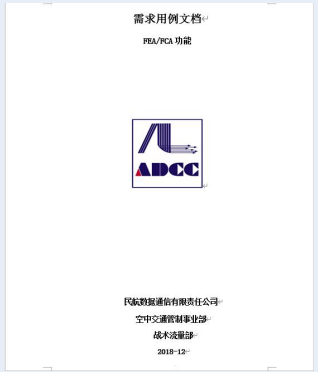
FEA/FCA区域修改

FEA/FCA区域删除

FEA/FCA区域地图显示

FEA/FCA区域合并

FEA/FCA区域在容流平衡里使用


需求用例文档

FEA/FCA 功能

ADCC

民航数据通信有限责任公司

空中交通管制事业部

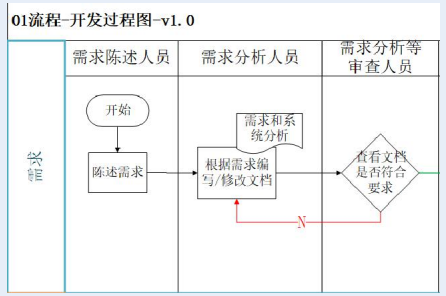
战术流量部

2018-12

◆ 文档描述：

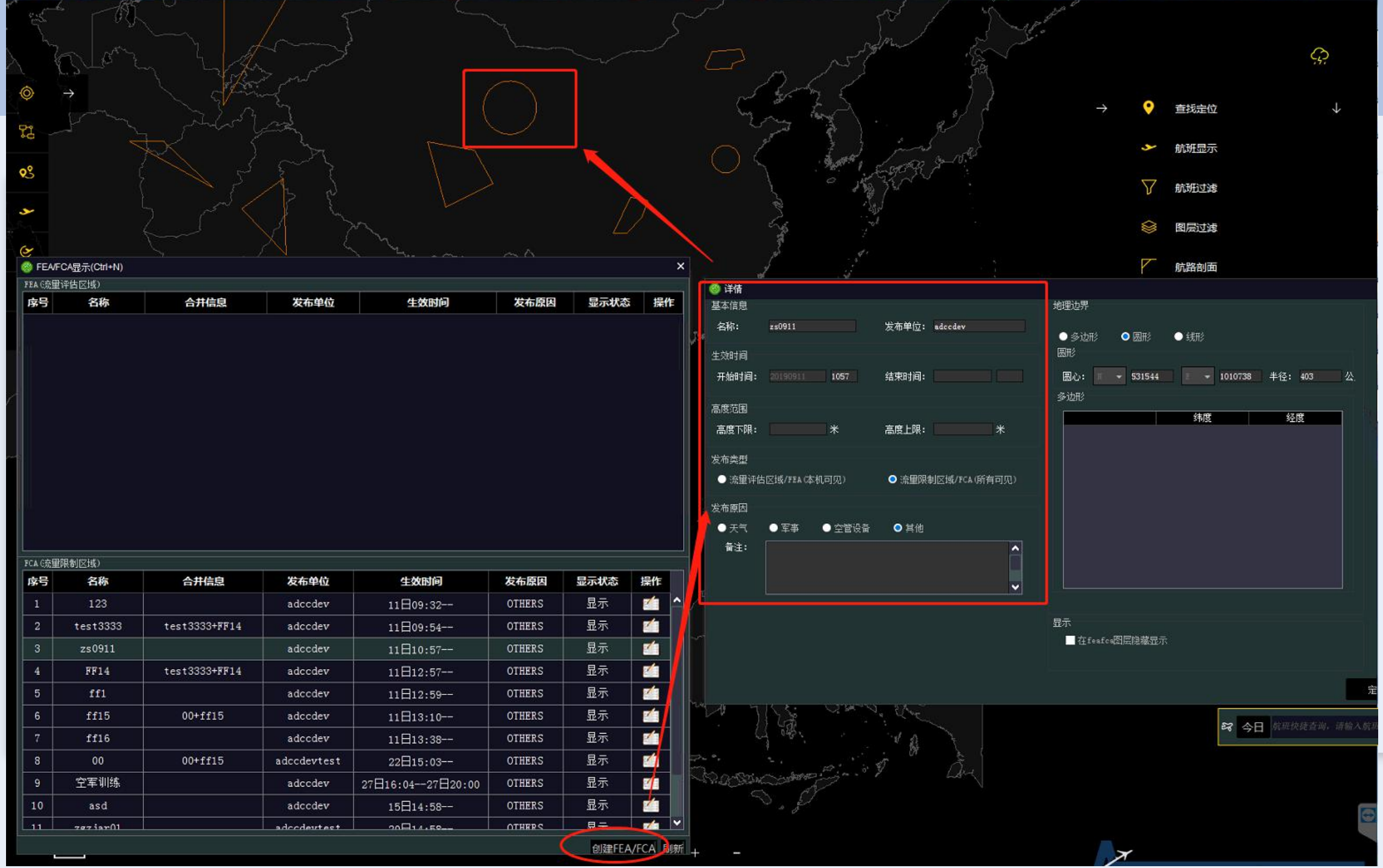
编写需求分析说明

编写需求用例文档




◆ 评审流程：

按照需求评审流程对需求用例文档进行评审

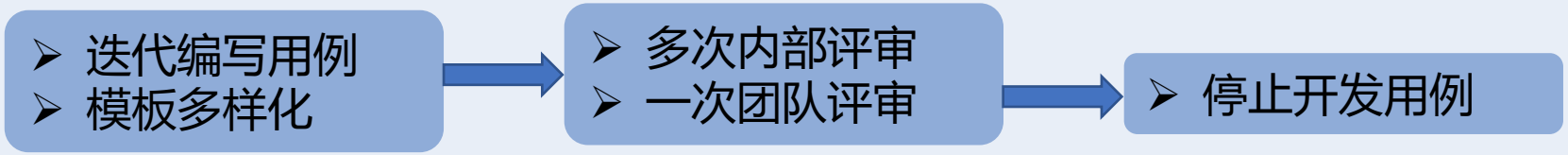

查找定位
航班显示
航班过滤
图层过滤
航路剖面
FEA/FCA显示(Ctrl+N)
FEA(流量评估区域)
序号 名称 合并信息 发布单位 生效时间 发布原因 显示状态 操作
FCA(流量限制区域)
序号 名称 合并信息 发布单位 生效时间 发布原因 显示状态 操作
1 123 adccdev 11日09:32-- OTHERS 显示
2 test3333 test3333+FF14 adccdev 11日09:54-- OTHERS 显示
3 zs0911 adccdev 11日10:57-- OTHERS 显示
4 FF14 test3333+FF14 adccdev 11日12:57-- OTHERS 显示
5 ff1 adccdev 11日12:59-- OTHERS 显示
6 ff15 00+ff15 adccdev 11日13:10-- OTHERS 显示
7 ff16 adccdev 11日13:38-- OTHERS 显示
8 00 00+ff15 adccdevtest 22日15:03-- OTHERS 显示
9 空军训练 adccdev 27日16:04--27日20:00 OTHERS 显示
10 asd adccdev 15日14:58-- OTHERS 显示
创建FEA/FCA
详情
基本信息
名称: zs0911
发布单位: adccdev
生效时间
开始时间: 20190911 1057
结束时间:
高度范围
高度下限: 米
高度上限: 米
发布类型
流量评估区域/FEA(本机可见)
流量限制区域/FCA(所有可见)
发布原因
天气 军事 空管设备 其他
备注:
地理边界
多边形 圆形 线形
圆形
圆心: 531544 1010738
半径: 403 公
多边形
纬度 经度
显示
在feafca图层隐藏显示
今日

# 需求分析-用例文档

- 目的：
  - 项目经理根据用例进行估计和发布进度；
  - 数据及业务规则制定人员可以把自己的需求和所需用例联系起来；
  - 用户界面设计人员可以进行UI设计，并将其与相关用例联系起来；
  - 测试人员可以根据用例中描述的成功和失败情况构建测试场景（测试用例）；
- 参与人员：
  - 要求具有不同观点和专业知识的人共同参与，以维护多方利益。一般包括项目经理、开发人员、测试人员。

# 需求分析-用例文档

◆ 编写过程




◆ 注意事项

- 用户界面信息不要太多，鼠标、键盘内容不应出现在用例中
- 低层次用例不要太多，无法展示其最终用户提供什么功能
- 不要太冗长，在3-9步
- 句子片段，主、谓、宾尽量完整，不要有歧义
- 删除不会为系统添加任何价值，或者已不在现有用例清单中的那些用例

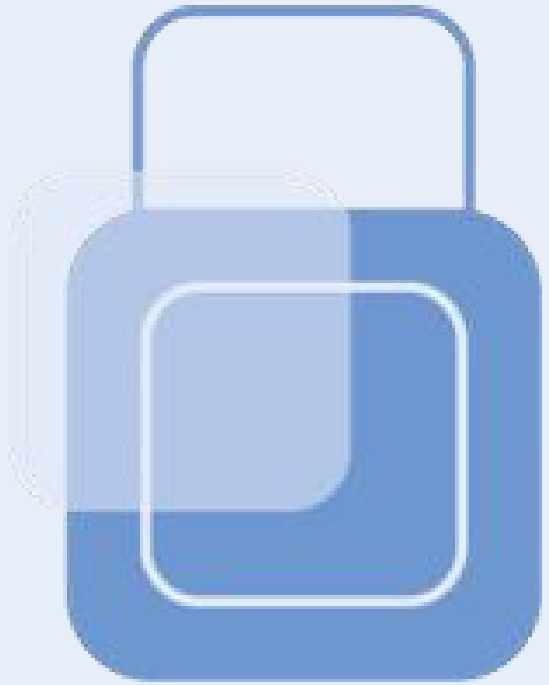
# “4+1”视图模型 – 场景视图

- 文档演示：FEAFCA功能需求用例

# 03

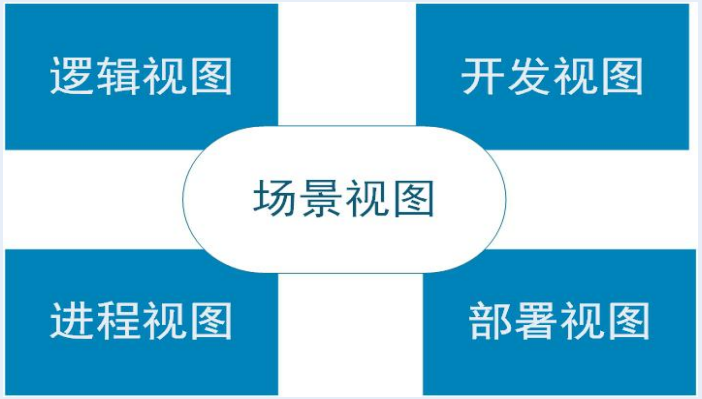
## 概要设计

“4+1”视图模型



# 概要设计

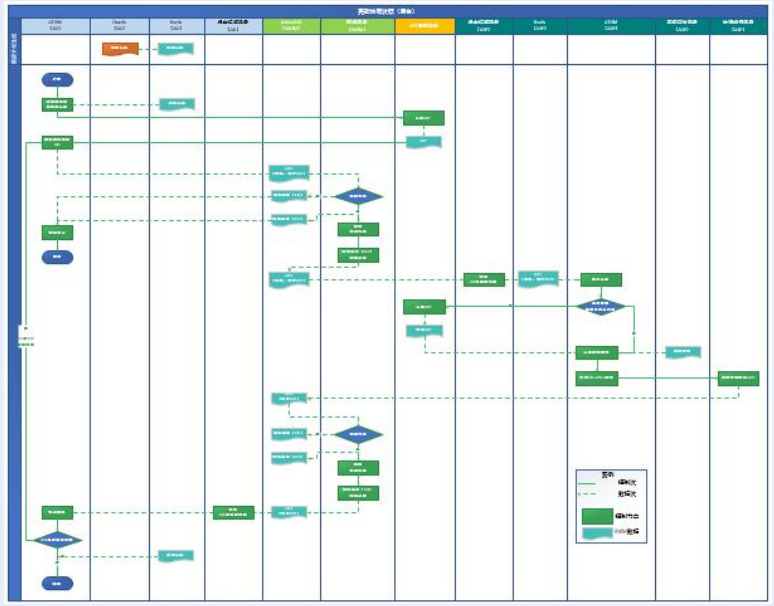
- 概要设计的核心：基于需求用例文档，完成4+1视图
  - 4+1视图模型从5个不同的视角来描述软件体系结构，包括逻辑视图、进程视图、开发视图、部署视图和场景视图。如下图所示：

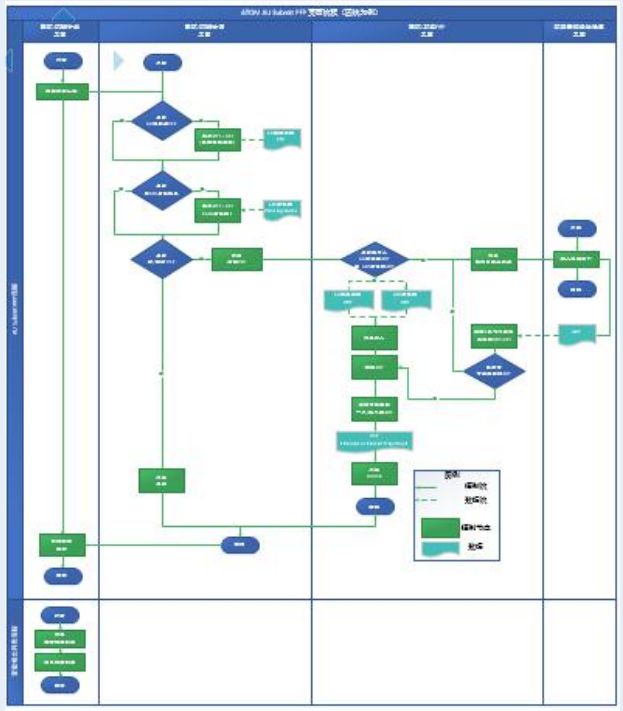



- 主要工具：Enterprise Architect 、 Microsoft Visio
- 主要产出文档：4+1视图

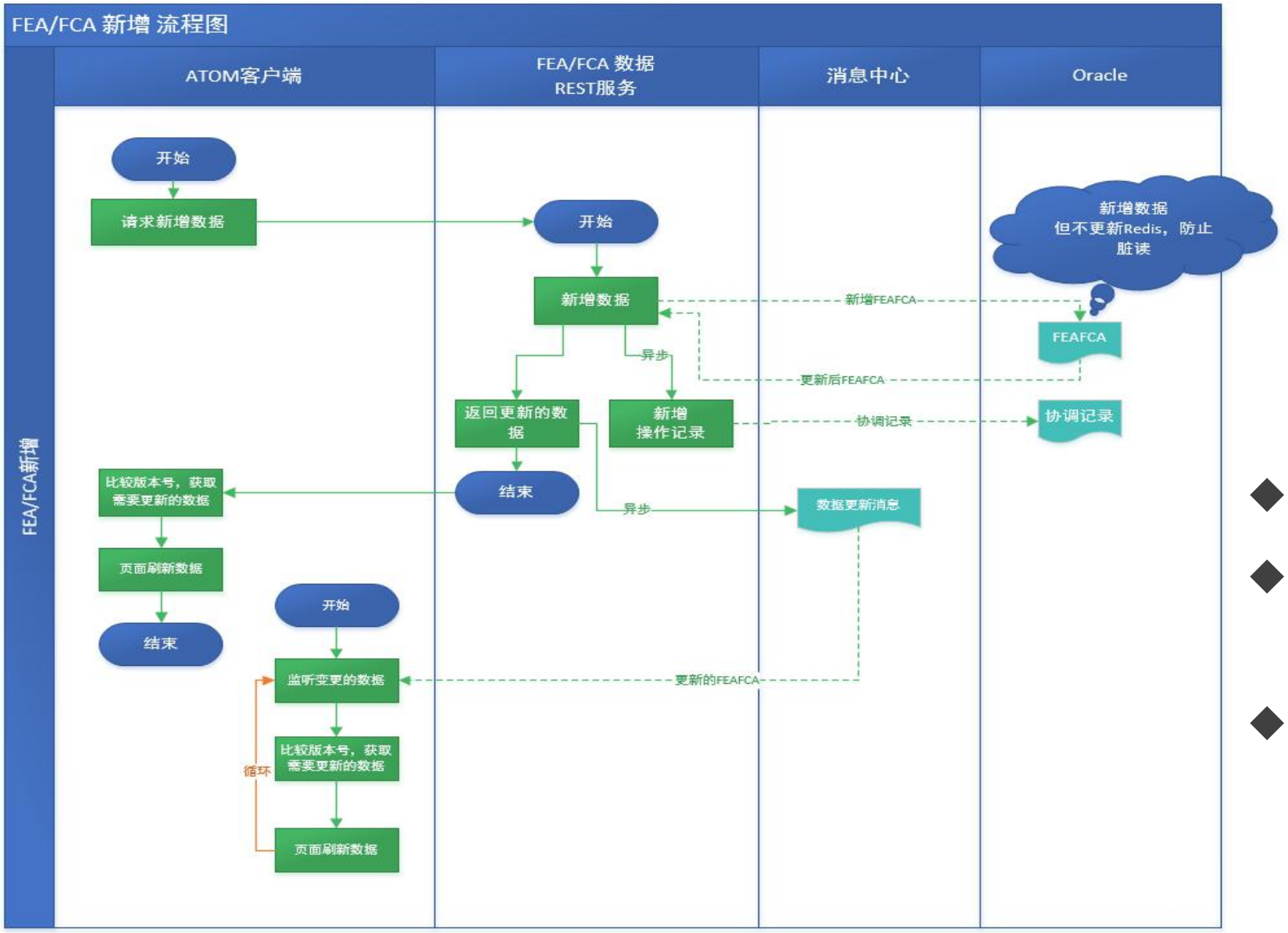
# “4+1”视图模型 – 场景视图

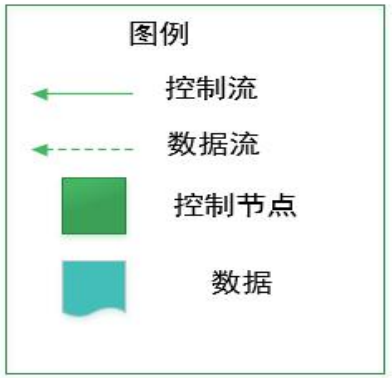
- 场景视图模型是重要系统活动的抽象，它使四个视图有机联系起来，是系统架构中最重要的需求抽象。在开发体系结构时，场景视图体现了系统构件以及构件之间的作用关系。它主要描述了现实中的一个系统运用场景的过程，把其中涉及到的对象、服务和操作都展示出来。





# “4+1”视图模型 – 场景视图




- 每个泳道是一个进程。
- 进程可以是客户端、应用服务、数据库等。
- 通过各种联系展示进程间和进程内部的关系。

# “4+1”视图模型 – 场景视图

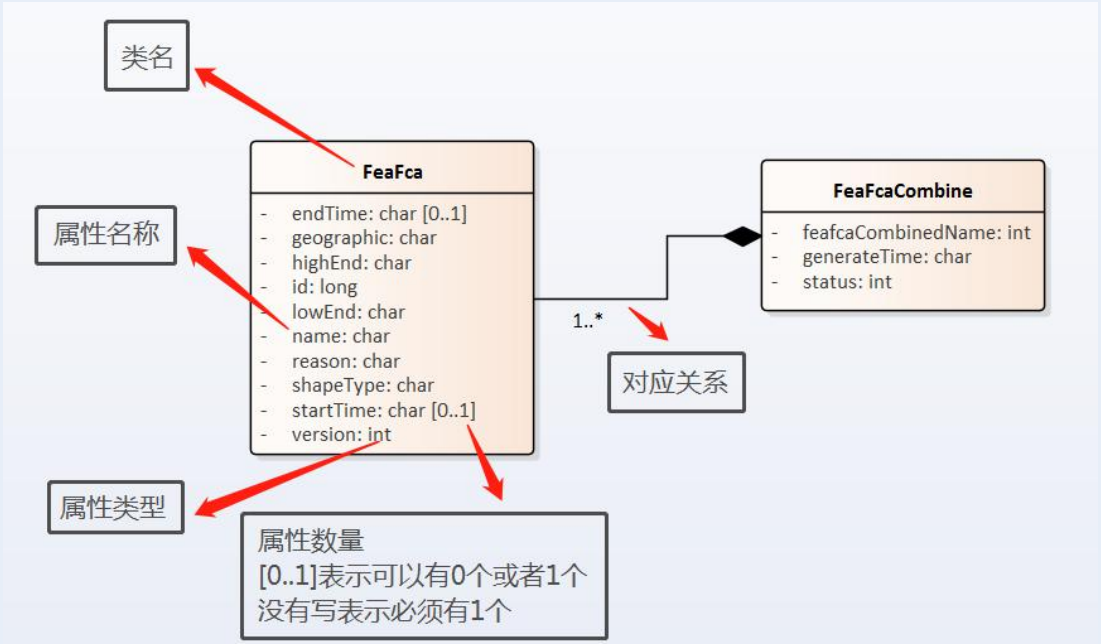
- 文档演示：FEAFCA场景图

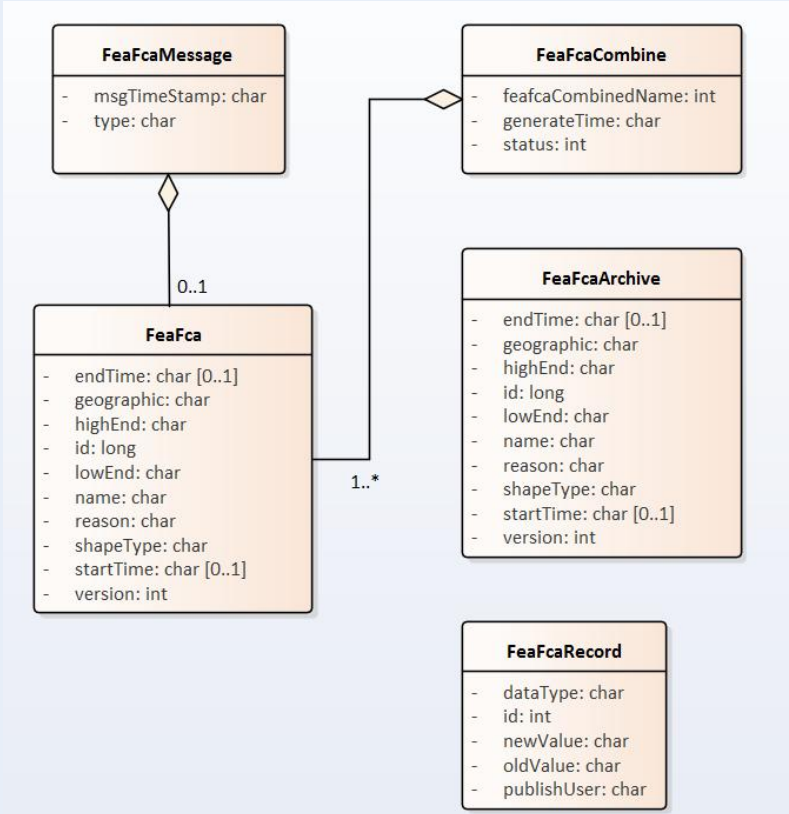
# “4+1”视图模型 – 逻辑视图

- 逻辑视图以UML类图的形式进行领域对象描述，类的设计遵循抽象、封装和继承的原则。
  - Union 统一：表示是一种通用的标准，称为软件工业界的一种标准。UML表述的内容能被各类人员所理解，包括客户、领域专家、分析师、设计师、程序员、测试工程师及培训人员等。
  - Model建模：建立软件系统的模型。
  - Language 语言：表明它是一套按照特定规则和模式组成的符号系统，它用半形式化方法定义，即用图形符号、自然语言和形式语言相结合的方法来描述定义的。

# “4+1”视图模型 – 逻辑视图

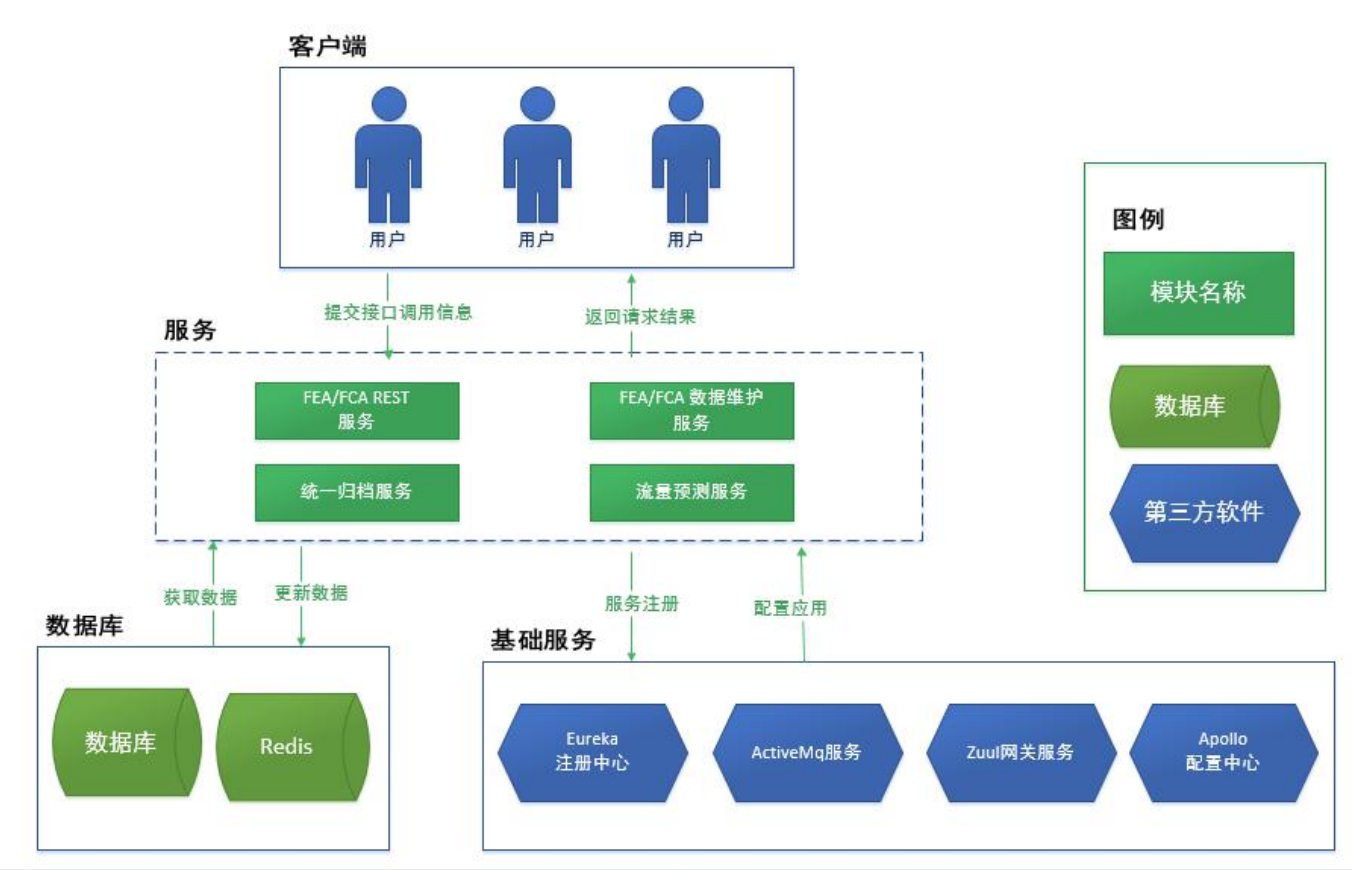
- 类图以反映类的结构(属性、操作)以及类之间的关系为主要目的，描述了软件系统的结构，是一种静态建模方法
- 类图中的“类”与面向对象语言中的“类”的概念是对应的，是对现实世界中的事物的抽象



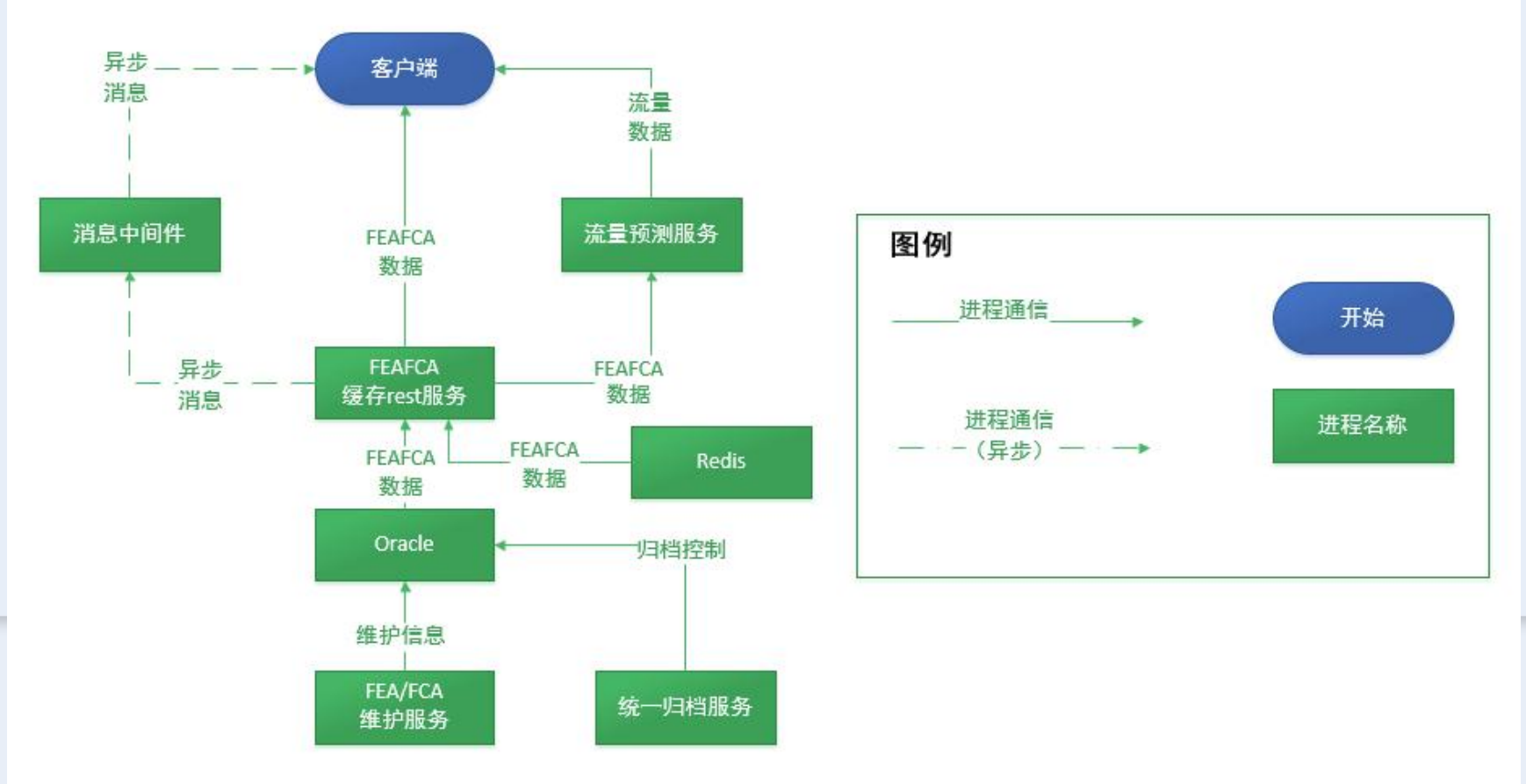
# “4+1”视图模型 – 开发视图

- 开发视图描述软件在其开发环境中的静态组织。采用哪些现成框架、哪些第三方SDK、哪些中间件平台，都应该考虑是否由软件架构的开发视图确定下来。



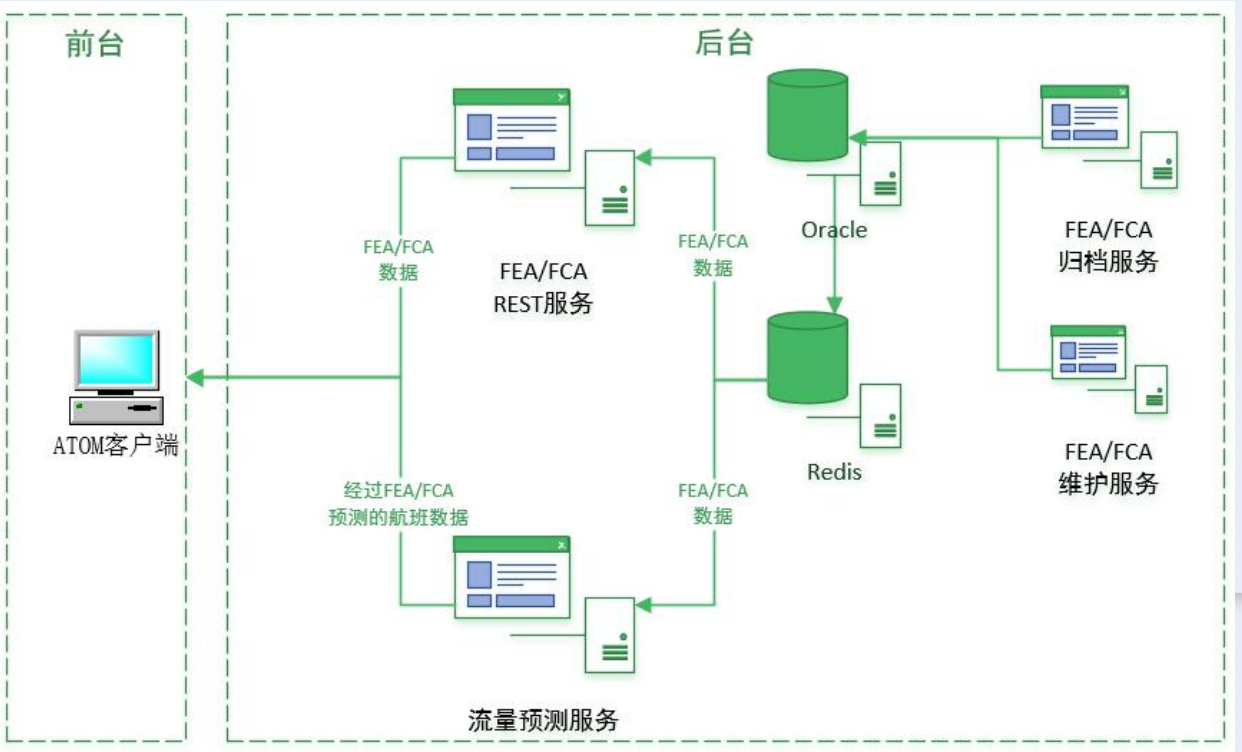
# “4+1”视图模型 – 进程视图

- 进程视图关注进程、线程、对象等运行时概念，以及相关的并发、同步、通信等问题。进程视图可以描述成多层抽象，每个级别分别关注不同的方面。在最高层抽象中，进程结构可以看作是构成一个执行单元的一组任务。



# “4+1”视图模型 – 部署视图

- 部署视图描述软件到硬件的映射，主要反映在分布式方面。它通常要考虑到系统性能、规模、可靠性等。解决系统拓扑结构、系统安装、通讯等问题。当软件运行于不同的节点上时，各视图中的构件都直接或间接地对应于系统的不同节点上。因此，从软件到节点的映射要有较高的灵活性，当环境改变时，对系统其他视图的影响最小。



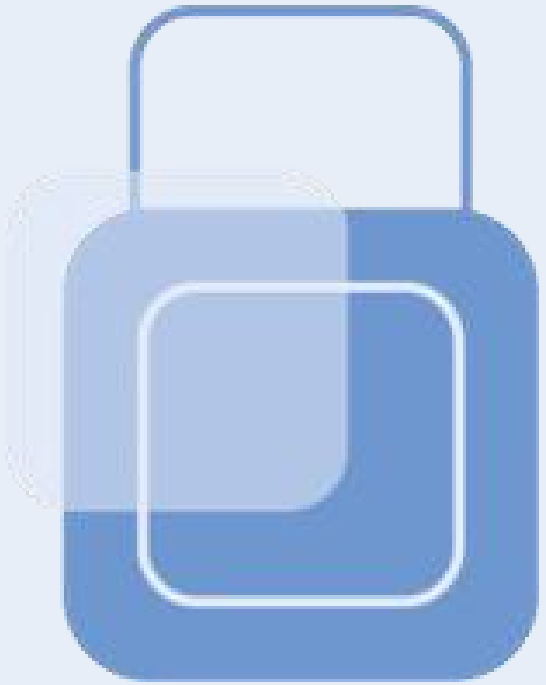
# “4+1”视图模型 – 4+1视图规范

- 文档演示：4+1视图规范

# 04

# 编写测试用例

测试用例



# 编写测试用例

- 在详细设计前编写测试用例的好处是：
  - 尽早发现设计问题：在软件研发的整个过程中，需求分析、设计、编码、测试、发布维护中，都有可能引入软件缺陷，尽早测试并发现软件缺陷修正时所投入的人力物力越少。

- 介入阶段：概要设计阶段，之后会贯穿于整个项目开发流程
- 主要工具：Microsoft Word
- 主要产出文档：测试用例文档

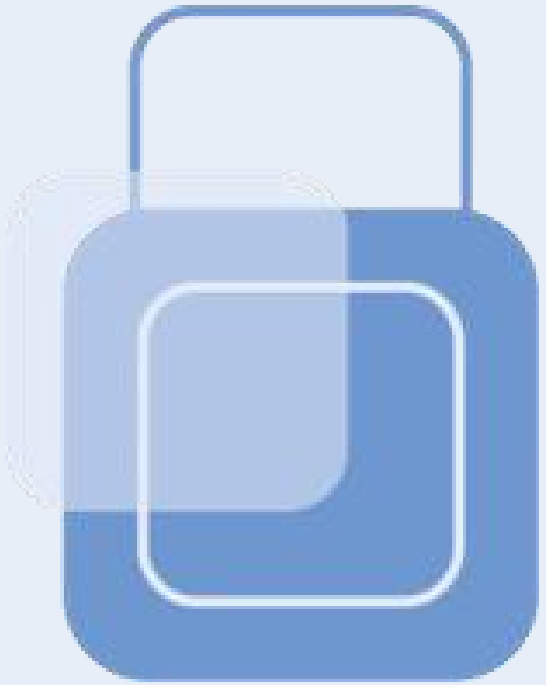
# 编写测试用例

- 文档演示：测试用例文档

# 05

## 详细设计

RESTful标准、接口文档、设计文档



# 详细设计

- 设计细化，基于需求用例、4+1视图、测试用例，对功能进行详细设计。
  - 接口设计：确定后台服务提供哪些接口、接口调用方式、接口参数格式、接口返回结果格式。
  - 算法设计：确定后台核心算法计算思路和计算流程
  - 页面设计：确定前台操作流程、页面流转关系、窗口页面布局需求、原型图
  - 数据库设计：确定关系型数据库物理表结构、非关系型数据库对象结构

- 主要工具：Microsoft Word
- 主要产出：接口定义（文档、或JavaDoc、或WebUI），详细设计文档

# 详细设计 – 设计标准

- 接口设计遵循RESTful风格规范

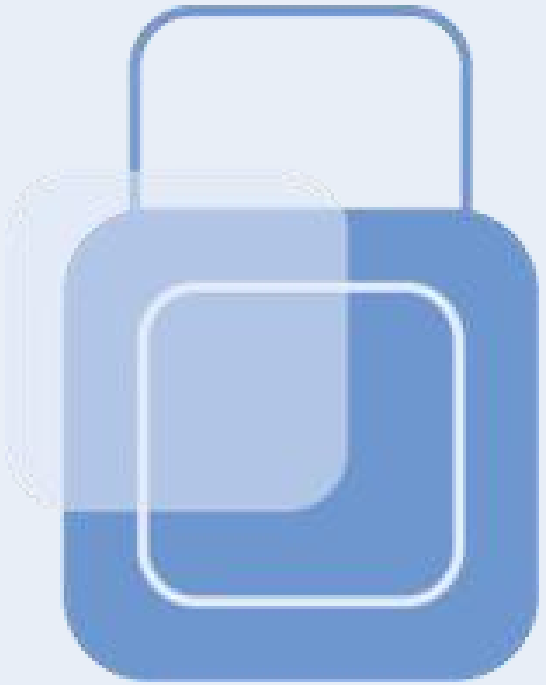
- 文档演示：RESTful接口标准

# 详细设计 – 设计标准

- 文档演示：设计文档

# 06

## 编码实现



# 编码实现

- 使用IDE工具实现设计逻辑
  - 遵循Java相关开发规范，正确地根据设计模型进行程序设计
  - 使用Git提交代码
  - 使用Sonarqube对代码质量进行评估
  - 使用Jmeter对接口进行压力测试

- 主要工具：IDEA Intellij、GitLab、Sonarqube、Jmeter
- 主要产出文档：项目源代码、压力测试报告、代码质量评估结果、接口压力测试结果

# 编码实现 – 开发标准

- 文档演示：JAVA开发规范文档 V2.0

# 编码实现 – 代码展示

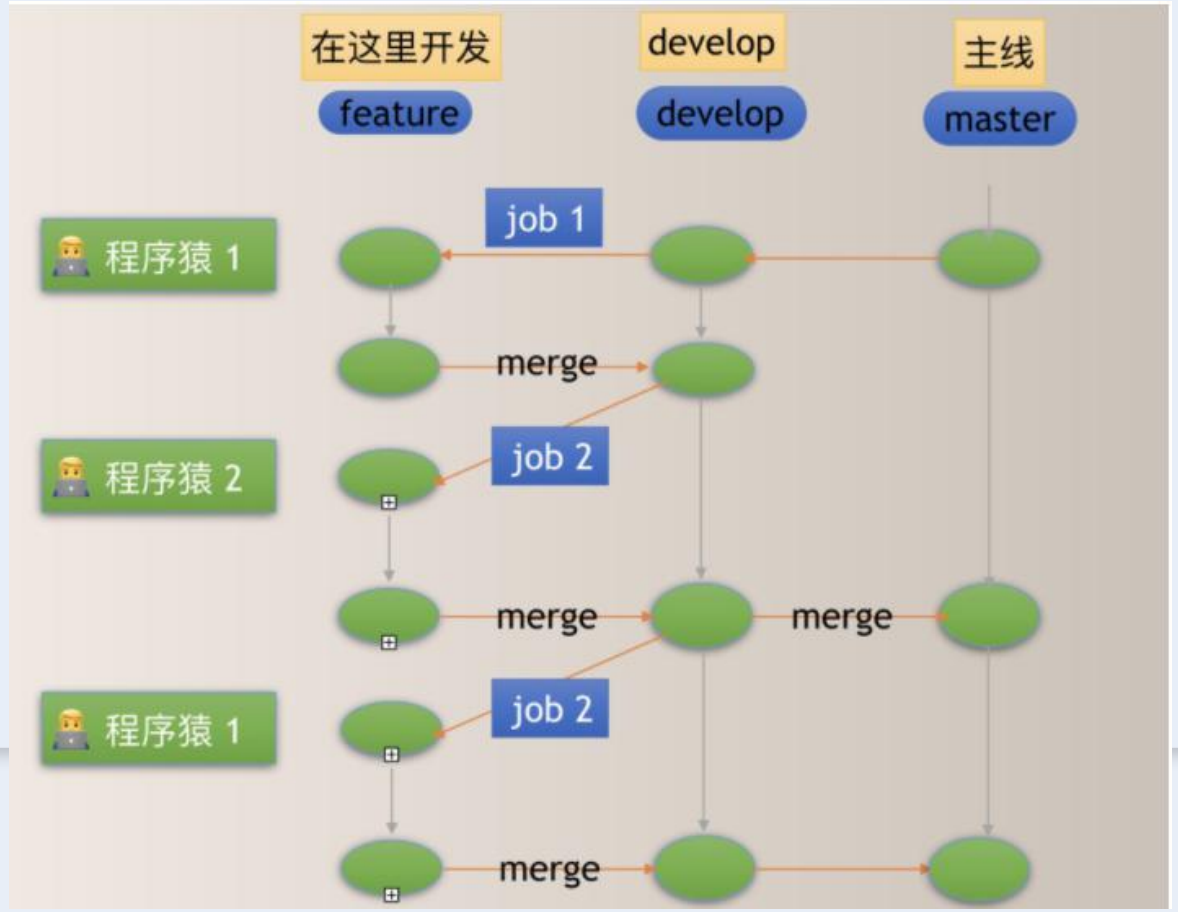
- 文档演示：项目结构和代码（Maven）

# 编码实现 – 工程类代码编码心得

- 随手写注释，不要最后再补注释。少比没有强。
- 先定义好代码Interface类，再写实现。
- 抽离公共逻辑或者方法，形成抽象类或者工具类。
- 时刻牢记NPE（空指针），CCE（类型转换错误）
- 适当使用设计模式，不要滥用。过度滥用会导致后续代码维护困难。
- 底层代码、被依赖地项目尽量不用注解，否则依赖结构不容易控制。
- 算法类逻辑，要用日志输出计算时间。

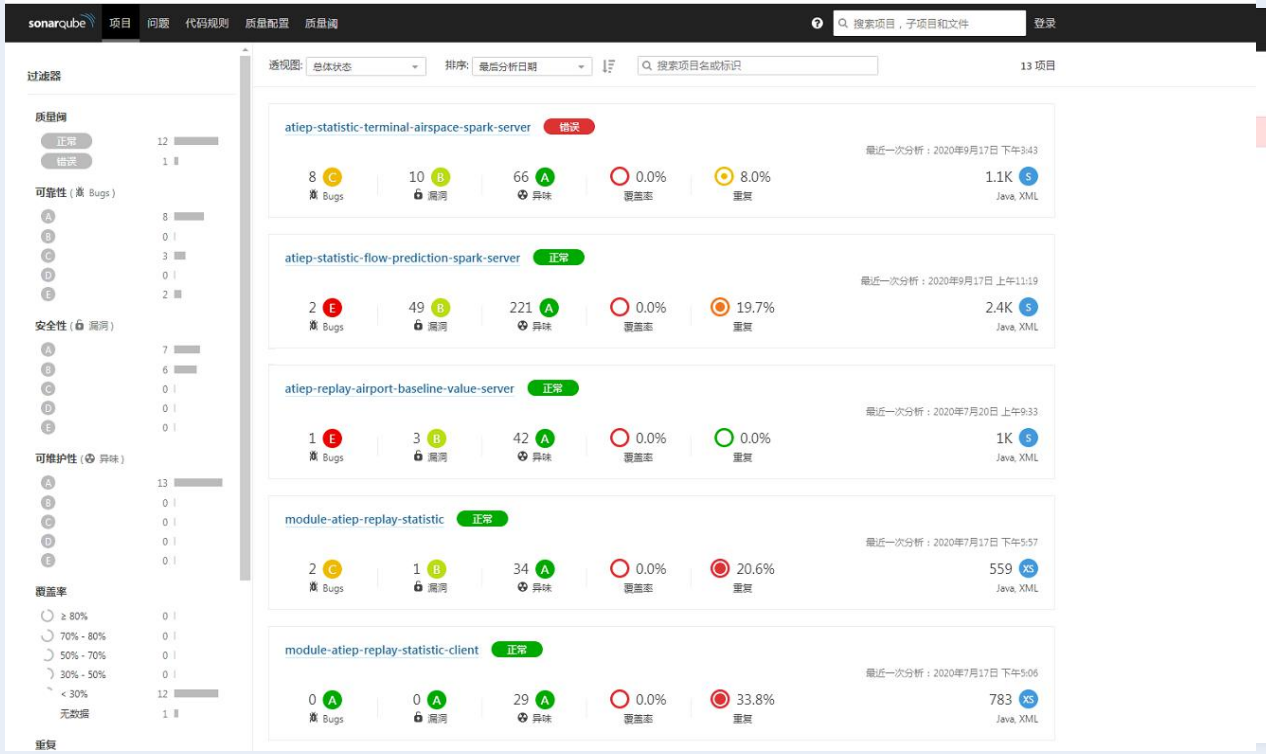
# 编码实现 – 提交代码Git

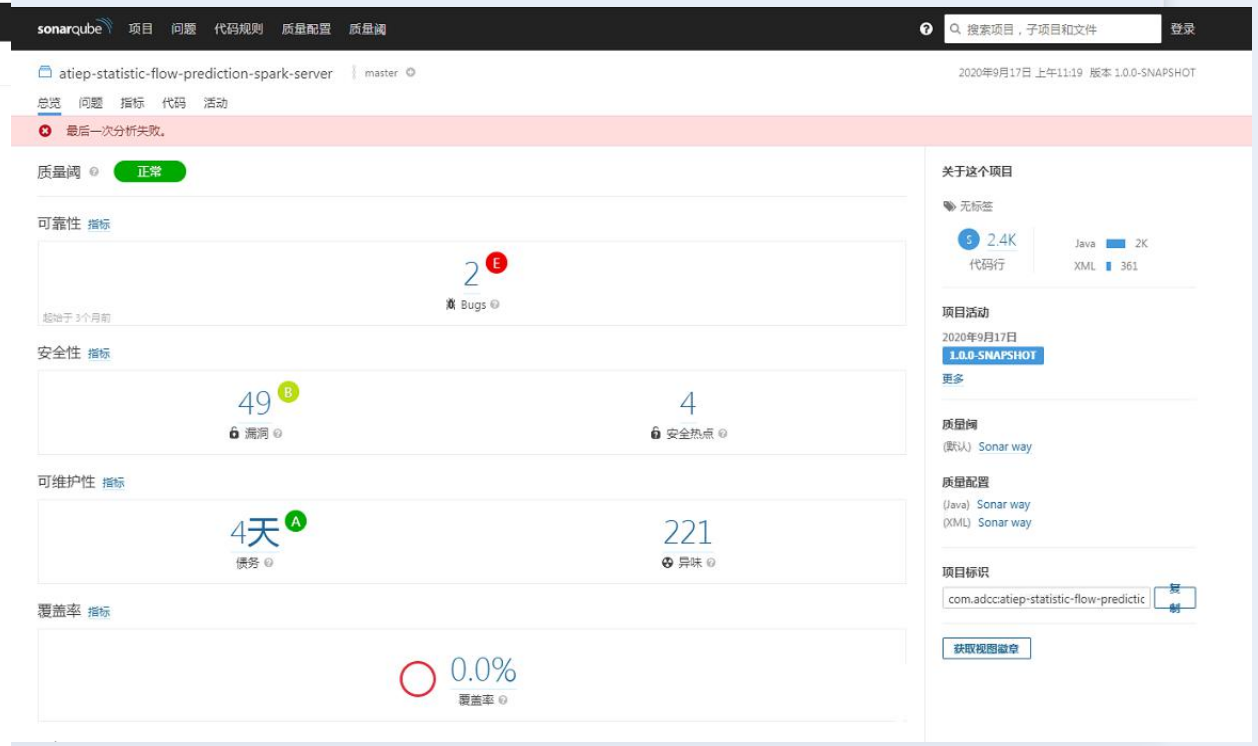
- 在使用Git的过程中如果没有清晰流程和规划，否则,每个人都提交一堆杂乱无章的commit,项目很快就会变得难以协调和维护。



# 编码实现 – 代码质量

- 使用Sonarqube平台对代码质量进行自动化检查
  - 发现非标准代码：基于代码标准，识别不符合规范写法。
  - 发现潜在BUG：通过PMD，CheckStyle等工具检查代码中潜在BUG。



# 编码实现 – 压力测试

- 代码实现完毕后，开发人员（非测试人员）需要对服务接口进行压力测试，通过压力测试，侦测设计缺陷和潜在代码缺陷。

测试用例一

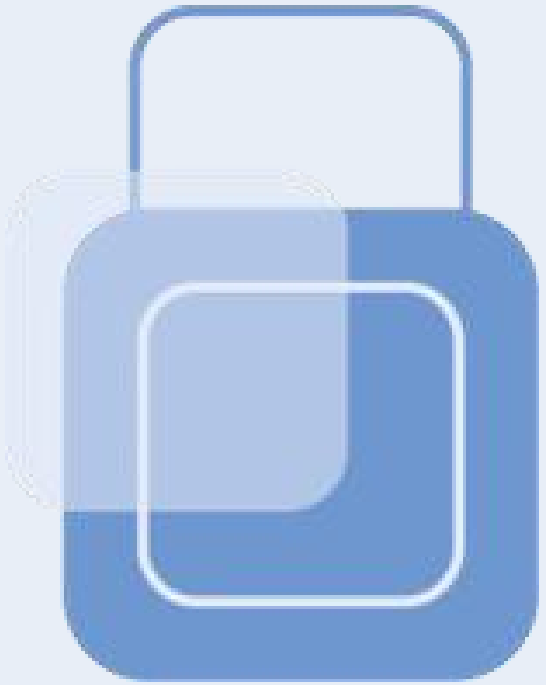
| 接口 | http://192.168.243.197:28080/flow-prediction-history-server/flow/national/monitor?startDate=20200227&endDate=20200227 |
|---|---|
| 参数列表 | ["DEPARTURE","ARRIVAL","EXIT_BORDER","ENTRY_BORDER","OVF"] |
| 返回结果 | {"generateTime":"20200227134419","status":200,"code":"20200227133900381","nationalMonitorStatistics":[{"statisticDate":"20200226","statisticType":"NATIONAL","scheduleNum":19571,"executeNum":4360,"cancelNum":0},{"statisticDate":"20200227","statisticType":"NATIONAL","scheduleNum":20870,"executeNum":2379,"cancelNum":0}]} |

测试结果

| 线程数 | 单线程执行次数 | 测试样本数 | 响应耗时(平均值(ms)) | 响应耗时(中位数(ms)) | 90% 的样本响应耗时(ms) | 99%的样本响应耗时(ms) | 耗时最大值(ms) | 耗时最小值(ms) | 样本异常率 | 吞吐量 |
|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 10 | 100 | 4 | 4 | 5 | 14 | 15 | 3 | 0.00% | 106.0/sec |
| 10 | 10 | 100 | 3 | 3 | 6 | 15 | 18 | 2 | 0.00% | 106.0/sec |
| 10 | 20 | 200 | 3 | 3 | 4 | 7 | 12 | 2 | 0.00% | 203.7/sec |
| 10 | 20 | 200 | 3 | 4 | 5 | 7 | 12 | 3 | 0.00% | 204.9/sec |
| 20 | 10 | 200 | 10 | 4 | 14 | 106 | 195 | 2 | 0.00% | 200.9/sec |
| 200 | 10 | 2000 | 51 | 39 | 119 | 159 | 169 | 2 | 0.00% | 1218.0/sec |

# 07

# 集成测试



# 集成和测试

- 按照流程对功能进行集成和测试
  - 按照周期发布计划，编写发布计划、升级概述。
  - 基于测试用例，测试功能是否正常。
  - 在发布前开发人员对即将上线地功能签字确认。

- 主要工具：Microsoft Word、开发管理工具（禅道）
- 主要产出：版本发布计划、系统升级概述、测试报告、发布计划签名表

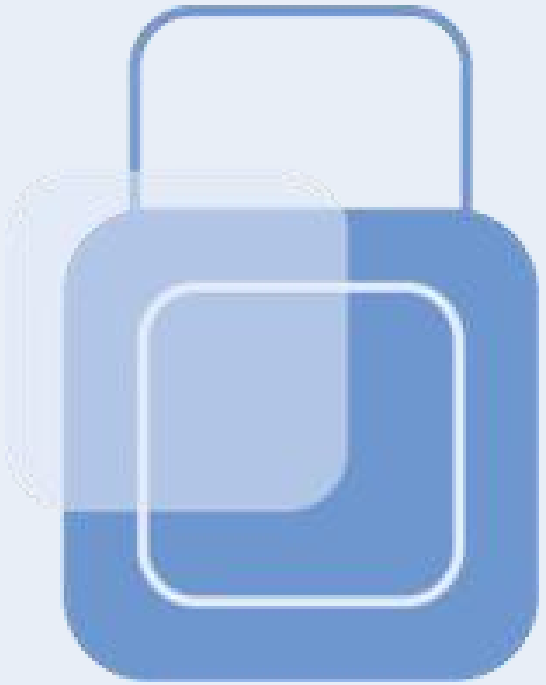
# 编码实现 – 代码展示

- 文档演示：集成测试流程
- 文档演示：版本发布计划
- 文档演示：系统升级概述
- 文档演示：测试报告
- 文档演示：签名计划表

# 08

## 部署和维护

系统部署列表、系统部署更新文档、系统维护文档



# 部署和维护

- 1. 文档演示：部署流程
- 2. 文档演示：部署文档
- 3. 文档演示：更新记录一览表
- 4. 文档演示：事后检查分析报告
- 5. 文档演示：服务表
- 6. 文档演示：数据库表
- 7. 文档演示：主机表
- 8. 文档演示：系统拓扑图
- 9. 文档演示：升级经验教训总结
- 10. 文档演示：故障处理手册
- 11. 文档演示：用户使用手册

# 谢谢！

提问时间