

2018


-
2018 Android



2018-6-8



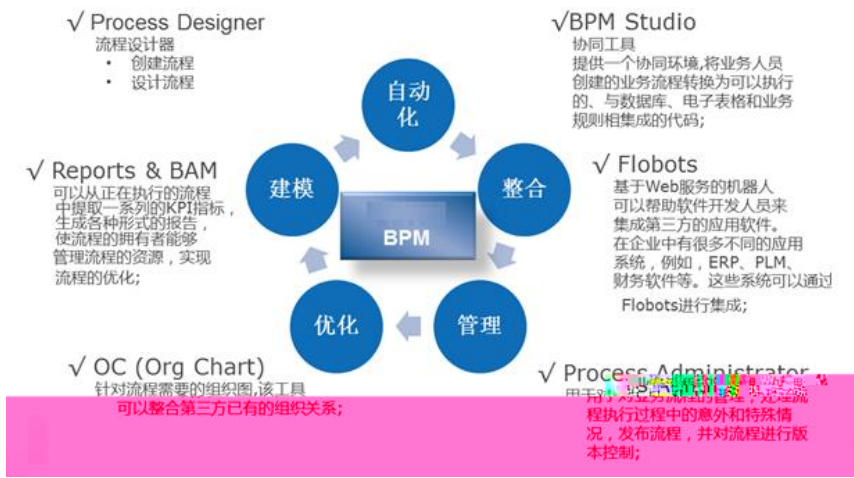
android

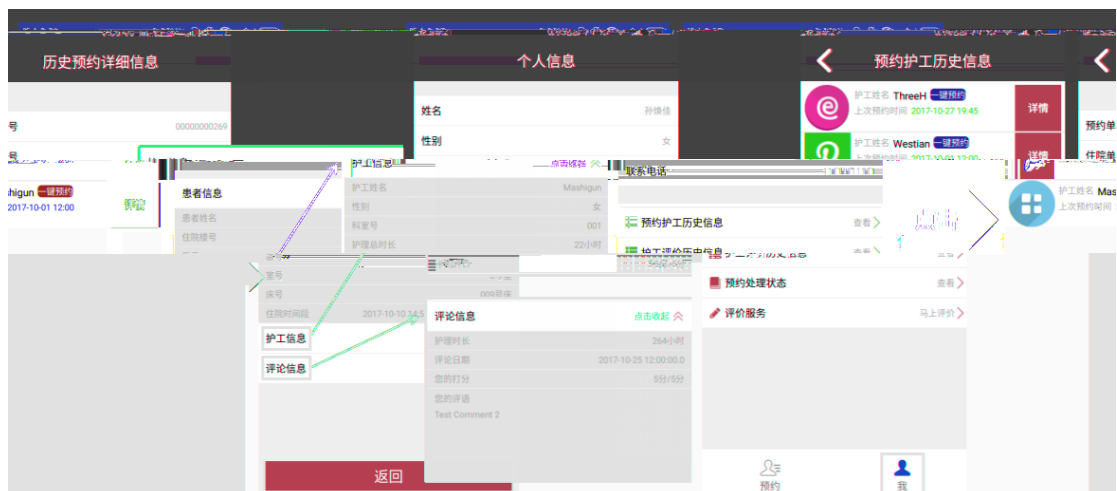
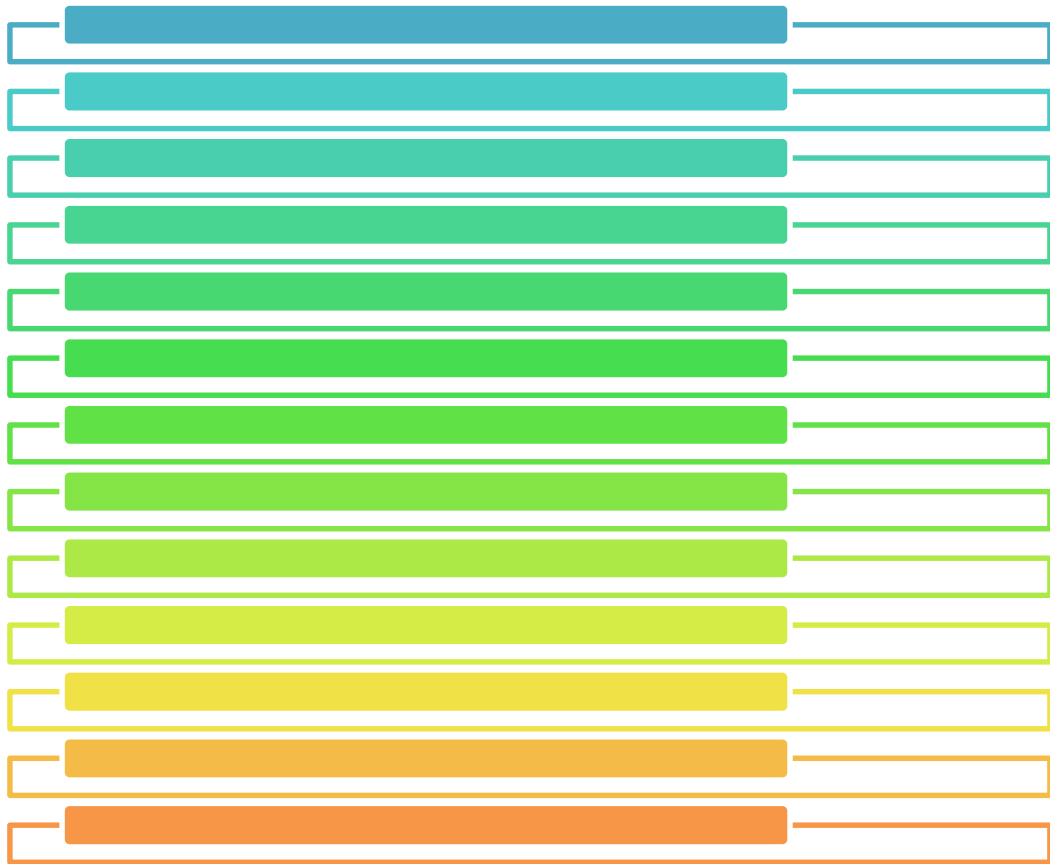


SQLite

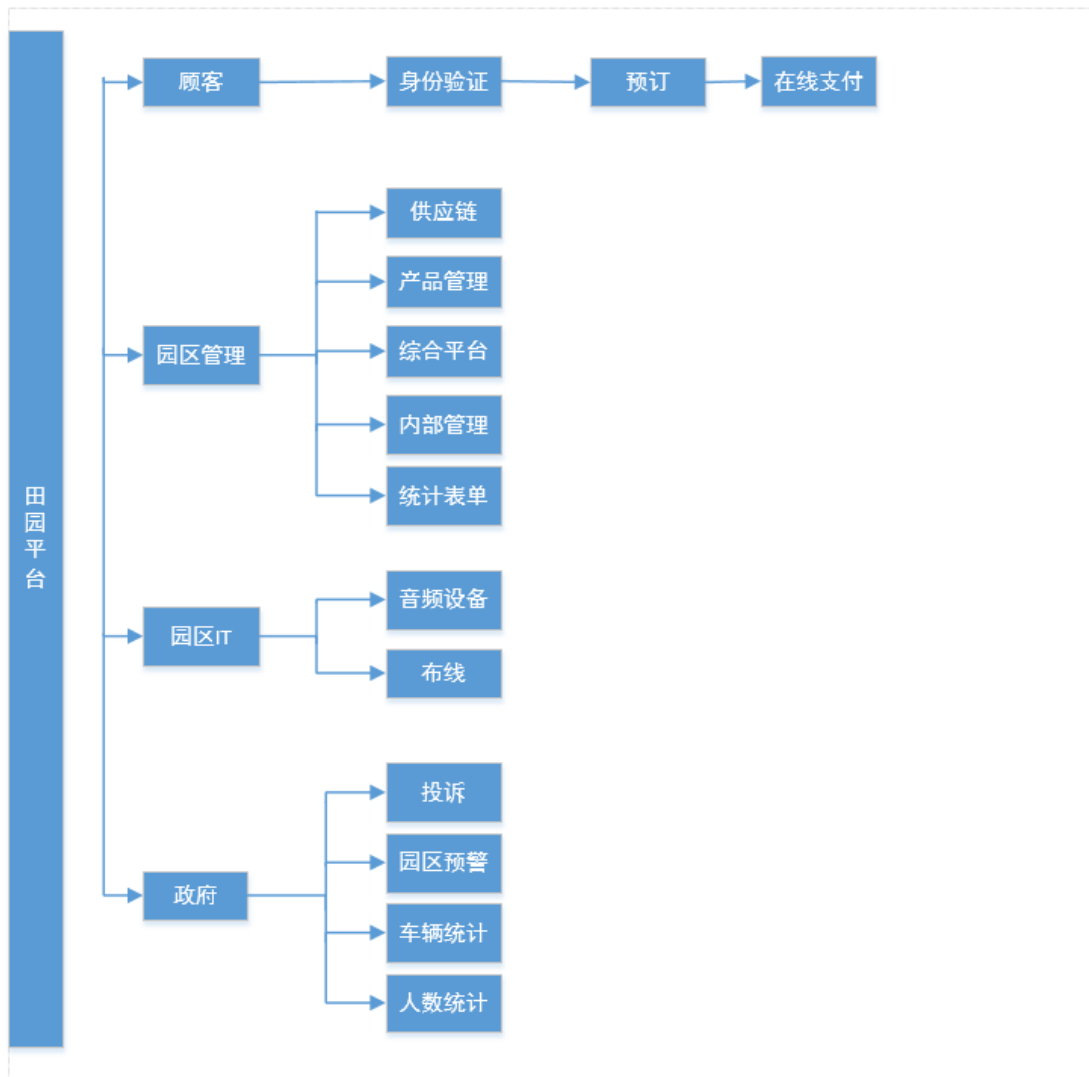
SQLite

| | |
|--|--|
| | |
| | |









车流 通过公众号，实时查看园区停车场车位状况。

进出 通报园区开放时间以及关闭时间。

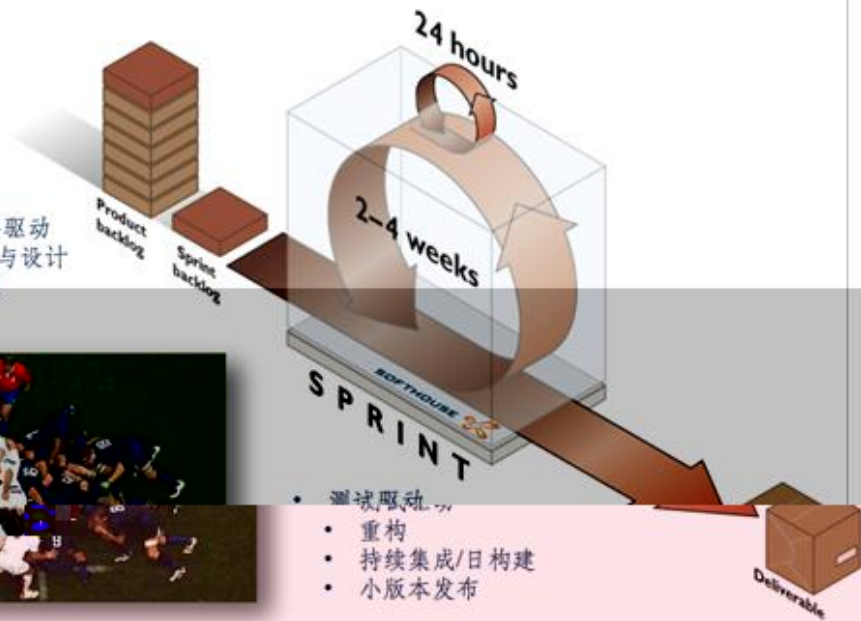
人流 监控园区总流量，在园人数、离园人数，等入人数。

园区天气

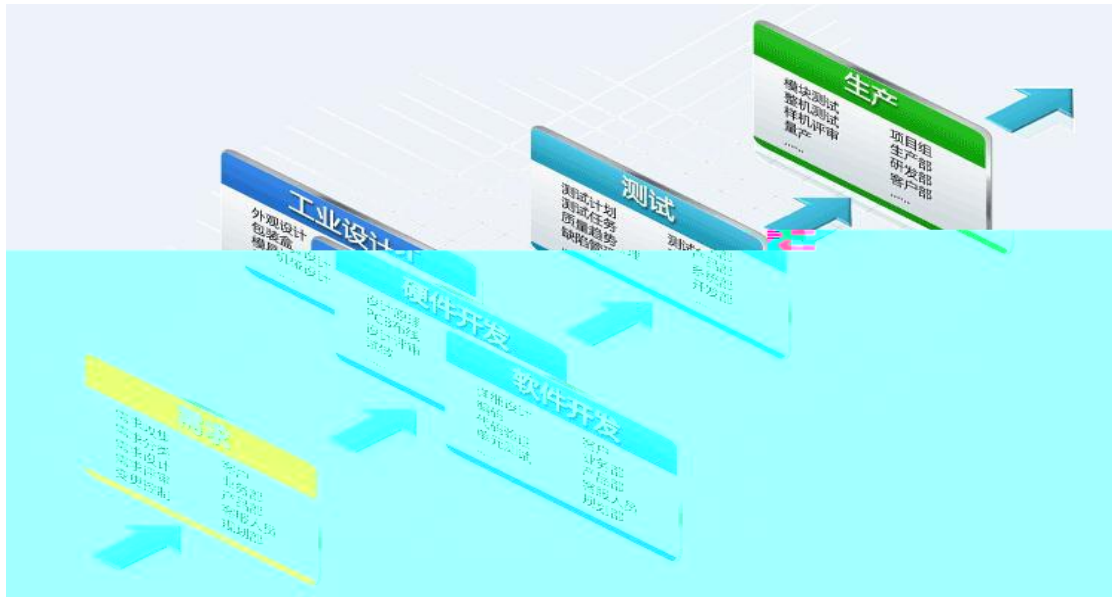


Overview

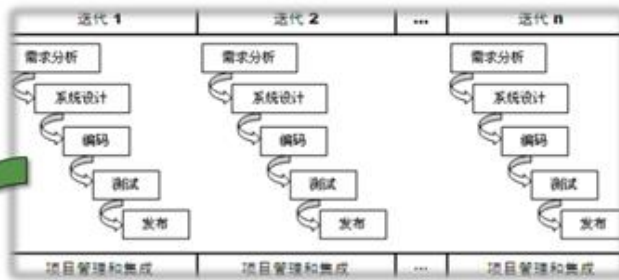
- 迭代开发
- 以架构为中心
- 用例/用户故事驱动
- 面向对象分析与设计
- 全程UML建模



- 测试驱动
- 重构
- 持续集成/日构建
- 小版本发布



迭代开发

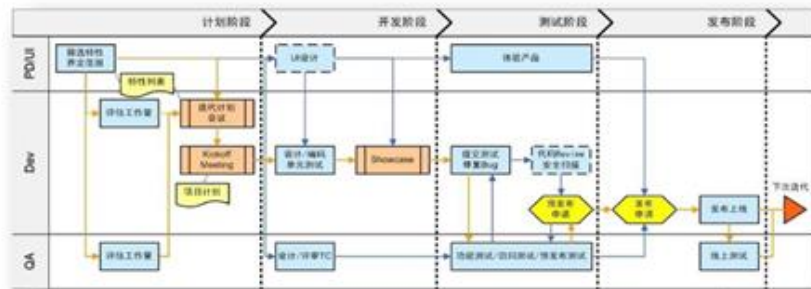


迭代式开发通过反复逼近、迭代增益的方法，在学习和探索的过程中逐步接近更为贴近真实目标的软件实现，开发效果和效率俱佳。

每一次迭代都是一个完整的开发过程——从需求到设计，从编码到测试，善始善终。

每次迭代完成时应该有完整的可交付的成果（软件成品），这是迭代开发和瀑布模型的阶段化开发的重大区别。

在实际开发中，我们通常将一次迭代称之为Sprint（冲刺），一般将1-2周作为一个冲刺周期。



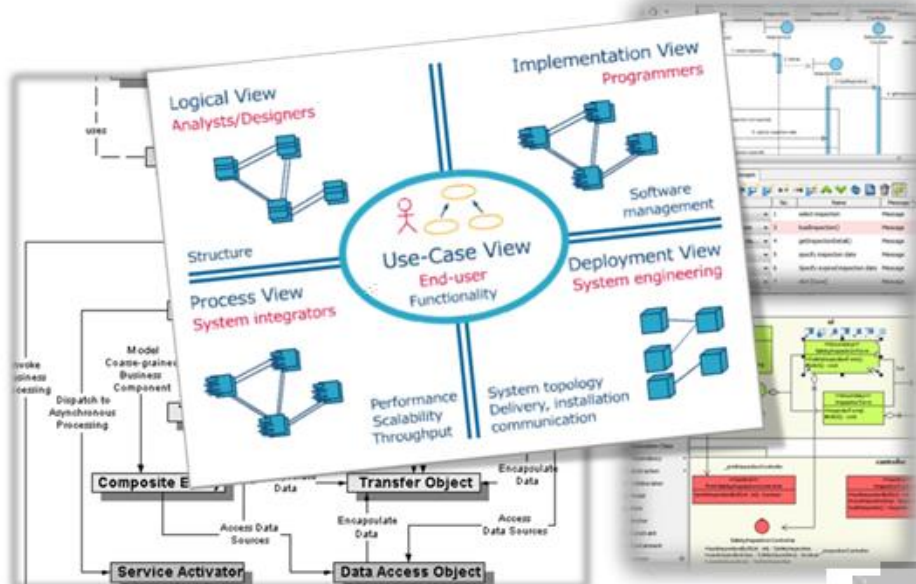
用例驱动



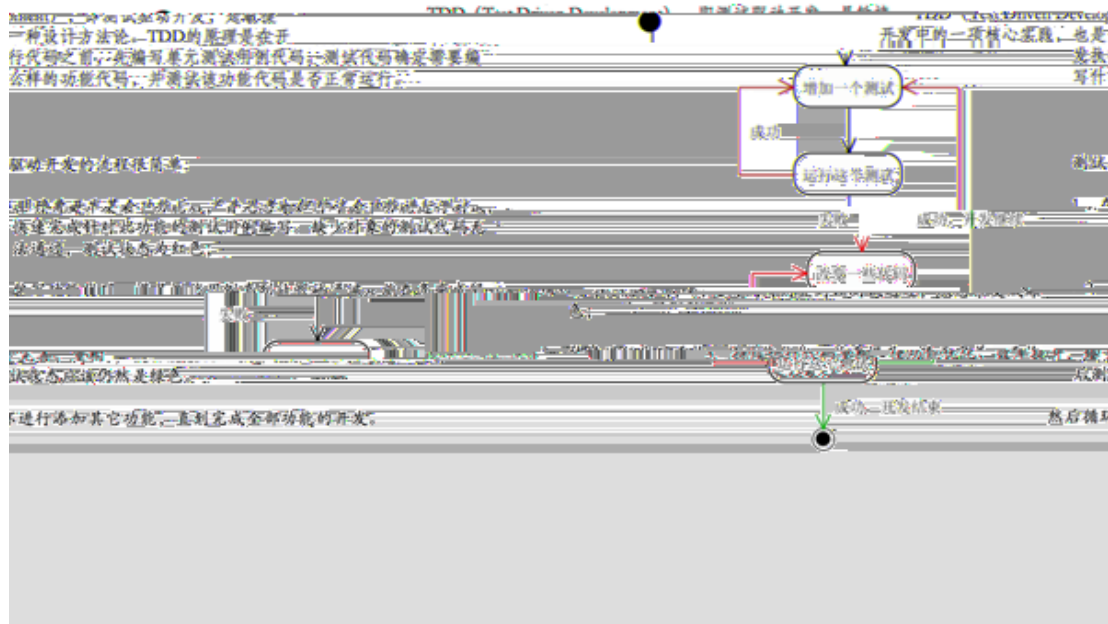
需求分析是从使用者的角度而不是开发者的角度描述用户对开发产品的需求。根据用户对产品功能的期望,对采集到的原始需求进行分析、整理、辨别和归纳,“去伪存真,查漏补缺”,提取出产品外部功能以及系统和外部环境关系的描述,最终形成系统、明确的软件需求。

本系统的需求分析主要采用用例建模 (Use-Case Modeling)

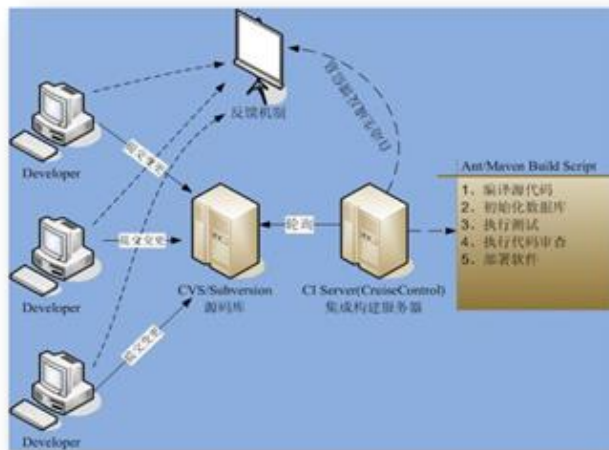
以架构为中心, OOAD与UML建模



测试驱动与重构



持续集成/日构建、小版本发布



1. 开发者每次将代码提交到SVN之前, 必须进行本地构建, 确保本地任何变更不会破坏集成构建。
2. 开发者每天进行多次提交, 小步前进会大大减少服务器构建失败的概率, 并且使得修复失败构建的时间大大缩短。
3. 自动化集成构建每天在一台独立的计算机上进行多次, 每次构建都必须 100% 通过测试, 保证软件现有的功能不被破坏并且没有引入新的缺陷。
4. 生成构建结果, 并以邮件、网页报表等各种可能的通信手段通知项目团队成员, 用以开展下一步的工作, 譬如修复、QA测试等。
5. 生成可以进行功能测试的产品 (如 WAR、插件、可执行程序等)。
6. 修复失败的构建是优先级最高的事情。

持续集成是敏捷开发不可或缺的重要环节, 没有持续集成, 整个开发就算不上敏捷开发, 而且持续集成也有利于整个项目团队找到开发的节奏感。

每次集成都通过自动化的构建 (包括测试) 来验证, 从而尽快地检测出集成错误, 尽早地解决风险。

在实际开发中, 我们一般进行每日构建。每个开发者每天将自己的开发变化提交至源代码库, 构建系统将这些变化提交源代码库, 进行自动化构建和单元测试。

如果一切顺利, 团队一天的开发工作才算是成功的。每日一次的构建能保证第二天的开发集成顺利进行。

为了达到高度迭代的目的, 将在一个或几个迭代周期结束后就发布一个预览版本。通过频繁地发布小版本 (Small Release), 开发人员可以更快地得到更多的反馈。

这些反馈可以帮助开发人员更及时全面地掌握客户的真实需求, 做出更为准确的计划。

会议与报告制度

- 站立式日例会
- 迭代计划会议
- 迭代评审会议
- 迭代总结会议



- 日报
- 周报
- 迭代总结报告

考核标准

| 等级 | 得分 | 说明 |
|-----|--------|--------------------------|
| 优秀 | 90~100 | 表现出色，创造性地、超出预期地完成实训质量目标。 |
| 良好 | 76~89 | 表现令人满意，达到期望质量目标。 |
| 合格 | 60~75 | 表现尚可，基本完成实训目标，但还有待改进和完善。 |
| 不合格 | <60 | 没有达到实训目标。 |