研发管理的那些事

许式伟

2014-3-27

研发管理

- 做对的事情：目标
- 加速产品演进的速率：产能
- 改善产品的品质：质量

最重要的事情 - 目标管理

- 走在正确的方向上：做对事情
  - 别折腾员工

做对事情

- 我们在做什么？我们公司的核心价值
- 哪些事情对提升核心价值非常重要？核心竞争力
- 我们是否把精力都放在了最重要的事情上了?

借力打力

- 时间成本才是最大的成本
- 轻公司运作理念
  - 非核心竞争力相关的事情，尽量外包
  - 尽可能服务外包（云服务），而不是技术外包（开源软件、外包团队）
    - 服务外包对结果负责
    - 技术外包通常伴随较大的隐性成本

开发效率 - 产能

- 架构的重要性
- 让任务并行
- 合适的工具

架构的重要性

- 决定一个模块能够走多远的，是架构
  - 一个大厦的地基，决定了大厦可以建多高
- 架构师在研发团队中至关重要
  - 导演

架构的关键点

- 模块的组合方式 - 需求如何被分解 - 正交分解
- 模块的接口 - 模块的外部依赖和使用方式
- 架构的话题可以很长，今天不展开讨论

让任务并行

- 多模块的并行
  - 及早稳定瓶颈模块（多个模块都依赖的业务模块）的接口
- 初期可mock瓶颈模块，使研发完全并行；
- 后期可更换为瓶颈模块的稳定版本（可选）
  - 后期维护一个mock模块的成本会越来越高

eg. WebService API重要性

- WS API — 瓶颈模块
  - 业务逻辑层
- 多终端支持
  - Web：PC/Mobile Web
  - 手持（手机/平板）： Android/iOS
  - PC：Mac/Windows/Linux
  - 其他：电视（机顶盒）/ 游戏机 / 车载终端 / etc

让任务并行

- 模块内的多人协作
  - 用 git 而不是 svn
    - 基于 branch 开发: feature/bugfix
    - pull request
    - git flow
- 优先考虑 github 服务，而不是自建 git 服务器
  - 你选择的绝非只是源代码托管，而是研发管理的外包

代码的质量保障

- 代码审查
- 单元测试
- 日构建
- http://xushiwei.com/software-architecture

代码审查

- 所有的代码都应该至少经过一次 Code Review
- 七牛建议的方式是：
  - 先做 Peer Review（同事间相互审查 / 结对审查）
  - 再做 Formal Review（正式审查）

代码审查

- 尽可能地减少一个pull request的规模
  - 当一个pr越大的时候，意味着它被merge的困难程度会急剧增加，review 压力会比较大
- 建议
  - 不要多个事情揉在一起做：一个 feature/bugfix 一个 pr，如果同时做多个事情那么就开多个branch
  - 事情分解着做：一个任务有可能可以分解为多个子feature独立上线，那么尽可能分解，分步上线

```
98  +/*
99  +       POST /insb
```

1

xushiwei added a note 10 days ago

应该同步修改 apidoc

Add a line note

```
100 +       Content-Type: application/bson
101 +
102 +       {Id: <Id>, doc: <Doc>}
103 +*/
104 +
105 +var ErrInvalidType = errors.New("invalid type")
106 +
107 +type insbArgs struct {
108 +       Id      string          `bson:"id"`
```

1

xushiwei added a note 10 days ago

冗余参数不可取。如果需要也应该在服务端获得这个Id。

Add a line note

```
109 +       //the Doc should containers '_id' field
110 +       Doc     interface{}     `bson:"doc"`
111 +}
112 +
```

单元测试

- 单元测试的重要性
  - 测试成本低（相比集成测试）
  - 发现问题的周期短（问题越晚发现修复成本越高）
  - 避免bug再现（系统refactor最容易引发bug，有充足的案例可以让你心里更有底气）

单元测试

- 为何开发人员抗拒单元测试?
  - 架构问题：环境复杂，模块测试成本高
    - 可测试性是验证架构合理性的最佳方法
      - 可测试性 = 低耦合 = 优秀的架构
  - 流程问题：做单元测试很麻烦
    - 好工具的作用：让单元测试变成效率工具，而不是负担

单元测试

defaultTransport #1182

Open [redacted] wants to merge 1 commit into [redacted] from [redacted]

Conversation 0 · Commits 1 · Files Changed 1

[redacted] commented 3 days ago

[redacted]

defaultTransport ✔ 476820a

Add more commits by pushing to the **feature/prefop_timeout** branch on [redacted].

✔ **All is well** — The Travis CI build passed · Details

This pull request can be automatically merged.
You can also merge branches on the command line.

Merge pull request

基础的研发流程

- 需求入库
- 优先级确定 & 研发计划
- 架构（模块设计）
- 开发
- 提交代码（自动触发 单元测试/代码风格检查/单元测试覆盖率检查）
- 代码审查（如果单元测试不通过直接打回）
    - 单元测试案例是否充分、合理
    - 业务逻辑是否考虑周全
    - 是否存在性能问题、设计缺陷等
    - 是否存在常规的坑、是否有代码风格问题
    - …
- 日构建 - 沙箱环境、集成测试
- 灰度发布
- 正式发布

相关的云服务

- github.com
    - 代码托管
    - 工作流
    - 缺陷管理
    - 代码审查
    - …
- travis-ci.com
    - 自动化测试
    - github 集成
- coveralls.io
    - 测试覆盖率
- drone.io
    - 自动化测试
    - 自动发布

总结：研发管理效率

- 目标管理：集中精力做正确的事情
- 借力打力：善用云服务
- 开发效率：模块并行研发、流程自动化
- 质量保障：白盒（CodeReview）黑盒（自动化测试）一起做

Q & A

@许式伟

@七牛云存储