

一个构架重构项目的迭代实践

胡协刚

软件架构师 **UML/RUP**专家

szjinco@public.szptt.net.cn

中国软件架构师网

www.soft-arch.net www.chinaarchitect.net

软件项目的风险

项目管理应当使得项目：

- 推进有序，并保持稳定的节奏，让所有成员的工作步调一致；
- 最大限度地规避风险，让风险提前暴露，以提供解决风险的足够时间和机会；
- 减少相互依赖，尽量提早进度，同时提高资源的利用效率。

迭代开发模式

- 软件的不确定和高风险等特性，使得传统的瀑布式开发力不从心
- 迭代有助于尽快发现和解决风险
- 迭代有助于控制项目的节奏，加快反馈，增强项目的控制力度，实现过程的有序化
- 迭代符合人们对事物的认识逐步加深，解决问题的能力随经验逐步提高，人类最根本的技能就是实践、总结和学习的能力等客观事实

迭代计划

- 确定本次迭代的目标，这包括：技术与开发、方法与过程、团队与管理三个方面
- 定义迭代产出的工件，例如文档、模型、代码等
- 细化迭代的任务

迭代总结

- 回顾上次迭代目标的达成情况
- 回顾迭代进度执行结果
- 统计迭代工件产出量，例如文档页数、代码行等
- 简述迭代集成测试的结果
- 总结经验与教训，例如团队建设、内外沟通、项目开发过程、需求、设计等各方面的成功经验和失败教训
- 提出改进建议

迭代实例——重构背景

- 某个企业自身开发的核心业务支撑系统，已经运作并不断演进了一年半，期间实施了众多的需求变更，虽然通过项目组的努力一直还能跟上业务部门迅速增长的需求变化，但在初始构架缺乏设计、实施进度压力过大、开发控制力度不够等因素的积累效应下，系统开始显现出硬化Rigidity和脆弱Fragility的不良症状

迭代实例——重构项目

- 为了确保适应业务部门未来迅速增长的需求变更，消除系统硬化和脆弱的不良症状，项目组决定实施一次重大的构架重构，力求根本扭转目前的不利局面
- 重构的目标初步定为：对软件构架做较为系统的重构，解决重大的不合理问题，并建立一个可持续演进的框架性基础；另外引入单元测试、初步改进代码质量（例如消除立即数、硬编码等）

迭代实例——重构挑战

- 重构面临的主要困难和风险：工期压力巨大，限制为一个半月；系统规模较大，整个有效代码行数超过十多万行；项目组对构架级的重构没有经验；重构如果失败对今后的工作负面影响很大
- 通过深入考察原有系统，暴露了更多问题：系统结构不够清晰，依赖关系复杂，绑死Immobility的症状明显；系统的接口设计存在重大缺陷，职责的分配粒度严重失衡

迭代实例——重构思考

- 如何解决这些困难
- 如何规避和降低这些风险

迭代实例——明确目标

- 制订可行的目标是决定项目成败最关键的第一步:

短暂的工期和较大的代码规模决定了重构的范围必须有限，不可能包含对代码级的重大变更等高强度工作；

重构的真实意图在于建立一个可持续演进的框架性基础，以支持业务的发展和持续并可控地不断小步改进

迭代实例——项目目标

- 项目的开发目标定为：

将系统的主要层次结构梳理清楚，消除跨层的循环依赖；

对业务逻辑层重新划分子系统架构，并设计出更为合理的接口；

以最小的代价将老的代码调整到新的子系统框架内；

新的子系统划分将成为今后演进的基
础，将以子系统为单位来实施持续改进

迭代实例——迭代过程

- 为了最大限度地规避风险，项目组决定实施四次迭代：

第一次，定义合理的系统层次结构，交付一个符合新层次架构的可执行系统版本；

第二次，重新划分业务逻辑层子系统，完成新的接口规格定义，交付一个满足新子系统接口的由桩替代组成的部分可执行系统版本；

第三次，使用老的代码来按照新接口规格实现主要关键子系统，得到一个混合桩的可执行系统版本；

第四次，完成所有业务子系统的老代码调整实现，最终实现项目的开发目标；如果可能还直接改进若干核心子系统的内部实现

迭代实例——规避风险

- 项目迭代计划的出发点:

尽量将风险的暴露时机提前，从而为项目留出足够的补救时间；

为项目的进展提供最为客观和可视的评估途径，每次迭代都有可执行的交付以便进行系统级测试，让项目组不会为表象所蒙蔽；

阶段交付的另一好处是在后续迭代失败时提供标准降低但却可接受的替代成果

迭代实例——事后教训

- 项目迭代计划仍然存在一些不足：

实践结果是第一次迭代非常顺利地完成了，但在第二次时遇到巨大困难，提前爆发风险的意图倒是实现了部分；但未能让项目组主要成员提早认识到项目的艰巨性，在设计细节上浪费了宝贵时间，而没有尽早完成需求（本项目的完整需求就是老的Webservice接口）到设计的对应，提供给开发人员一个有质量保证的设计成果；

在第二次迭代交付一个由桩组成的部分可执行系统版本的目标，最后证明是不可实现的，最后及时调整目标为在单元测试层面做到可以验证；

迭代实例——事后教训

第三次迭代即使按计划完成，它仍然得不到一个可以正式发布的版本（相当功能没有真正实现），做不到真正的阶段交付，其后调整计划为：在Webservice层直接调用老代码，以交付一个可以真正发布的版本，然后逐步以新接口实现来替换直接调用，从而一直维持一个可交付版本；

值得注意的是，这里同时增加了在Webservice层加强单元测试的决定，它保证了在以新子系统接口实现完全替换直接调用之前，系统的质量是可以控制的（在第四次迭代可能取消的情况下，也能达到建立一个可持续演进的框架性基础的目标），另外也可以用来进一步暴露老代码隐藏下来的Bug（毕竟系统级测试覆盖率有限）

迭代实例——经验总结

迭代开发的方式最大限度地减轻了项目的自身风险，为项目管理者提供最迅速和最可靠的反馈，从而为项目组提供了补救和改进的众多机会，大大增加了成功的概率；

项目组面对的众多不确定性、风险，通过迭代计划的及时调整，最终都能加以消解，虽然不一定能得到最好的结果，但确实避免了最终失败的最坏结局；

建立一个反应迅速、适应力强的团队，成为本次项目的重大成果。

Q&A

谢谢！