一次有益的敏捷尝试

(One Beneficial Agile Trial)

版本 A0

“PRM 项目所反映的问题和现象是非常典型的。程序员高手和笃信编程技巧大于一切的观察家们会指着 PRM 案例说，这明显是开发人员的水平不够，页面处理太笨，数据库设计太次……要是我早就搞定了！可是，这果真是技术问题吗？”

—— 张恂

俊生：

你好！当我第一次在 DW 上读到这篇文章（《漫谈企业应用项目的软件开发过程：一个 PRM 系统实施的经验与教训》），就发现它是一篇非常难得的好文章——国内类似这样的软件工程案例分析太少了，很多人没有时间写或不愿与他人分享家中的宝贝，何况这篇文章还是专门针对 XP、RUP 实践的。不管结果如何，原文篇幅不长，却有很多值得我们借鉴、学习的地方。除了你总结的经验和教训之外，我还看出了一些另外的问题，有一些新的联想，于是写下来与你交流探讨。

印象最深刻的是原文中的 PRM 系统虽然准时交付使用，但是由于后来出现性能问题（数据库死锁、大数据量处理和显示慢等），过了 8 个月仍然没有通过客户的验收，不但有几十万尾款没有收到，而且还影响了其它项目的投标。我个人认为这个项目从商务角度看是比较失败的，至少曾一度让开发商和客户都陷于非常被动。

这些麻烦究竟是什么原因造成的，有没有可能事先避免呢？我想 PRM 项目出现问题可能有这几方面原因（在你概括的基础上）：

1）受 XP 影响对需求工程的重视不够

由于进度很紧（2 个月完成的进度计划可能一开始就定得不太合理），团队成员包括客户在内都把注意力放在了按时完成功能上，却忽视了关键的非功能需求（原文教训 2，“在需求获取阶段没有明确地了解系统的性能指标等非功能需求”），从而导致架构设计（对于多并发用户和大数据量的处理）存在缺陷，这可能是 PRM 项目所犯的根本性错误。正是由于需求不完备，没有及时发现遗漏了关键的、隐含的性能需求，才直接导致了项目后期的被动。

XP 用非正式的写在索引卡片上的用户故事来定义需求，像并发用户数、页面一次显示的数据量之类的约束往往很难通过这些简单的用户故事被发现，只有细致、认真、全面的需求分析才有可能捕获这些潜在的需求。不过，由于 XP 有现场客户，因而能在一定程度上减少遗漏需求的风险。

虽然 PRM 项目采用了用例来管理需求，在严格性上比 XP 前进了一步，但缺点在于只重视了实现功能需求，忽视了提取非功能需求（完善的需求工程非常强调捕获非功能需求），而且该项目的客户参与度不够，缺少正式、规范的需求

和架构评审过程，导致没有能及时发现遗漏的需求，给成功带来了很大的隐患。这一事例再次向我们强调了全面需求分析的重要性。

2）没有实施有效的风险管理，做到真正的迭代式开发，尽早排除架构（性能上）的风险

迭代不仅仅意味把整个系统开发任务逐一分解、按阶段分步骤来实现，如果迭代的含义仅仅停留在这个层面上，那么提出用迭代演进式过程取代瀑布型开发模型也就毫无意义，因为我们做任何工作本来就是要一周一周、一天一天、一块一块地来完成——不管瀑布型还是演进式，皆如此。

迭代真正的主要目的其实是为了通过加速客户反馈显著地消除开发风险，这就要求每次迭代结束必须有一个可运行、可演示的系统（这时的系统可能功能上还不完整，仅仅是一个骨架，但它总是系统开发中最难、最重要也就是风险最大的部分）。所以，真正的迭代式开发能够在项目早期就允许客户对可运行的系统进行验证，从而使项目的风险减到最小。开发工作也应该根据风险的大小来安排，通过迭代及时调整优先级，风险越大的任务越应该及早设计、实现、测试和反馈。

PRM 项目虽然模仿了 XP 迭代周期，甚至每天都开例会（这有点像 Scrum），保证了初始版本的准时交付（在保证 PRM 前期进度方面，迭代还是有功劳的），却仍然没有能够防止较大风险的发生（交付系统几个月后才逐渐暴露出性能和架构上的质量问题），可以说这并没有达到 XP 或 RUP 迭代开发的最终目的。在项目初期没有把合同中已经提到的数据迁移视为一个关键风险是一大失误。我想，如果严格按照 RUP 风险驱动的迭代演进式开发进行管理，在半年多的时间里应该还是有机会尽早发现这个问题的。

原文提到“该项目在系统维护期间，对代码进行走查，修改了很多对于性能有影响的语句”，需要提醒的是，这种方式可以消除局部的缺陷，但却很难发现全局性的架构问题。对于软件架构，“头痛医头，脚痛医脚”的做法往往是行不通的。曾经有朋友告诉我这样一个真实的故事。在进度高压下有一位架构师成功开发了 1 个 J2EE 应用系统，正当他为此兴高采烈时，客户却告诉他原来的性能要求需要修改——平均响应时间应该从 15ms 减少为 5ms。这时他才发现原来自己辛辛苦苦花了 3 个月时间设计出来的架构对于这样一个性能指标的调整完全不能适用，需要推倒从来，此时他的心情是何等沮丧可想而知。PRM 项目的性能出现问题是在系统交付几个月之后，如果事先没有经过充分的架构设计和评审，谁知道以后还会发生什么情况，谁又能保证今后不会出现类似的或新的问题？

3）合同缺陷

作为使用者代表的客户方 IT 负责人、负责销售和签订合同的开发商客户经理以及开发商内部负责系统实现的项目经理/架构师这三方之间能否对合同的范围和技术可行性进行充分的沟通，达成清醒的共识，对项目的最终成败起到至关重要的影响。

从原文分析看，与客户签订的这份合同明显存在着缺陷。它对需求的理解是片面的（“主要描述的是系统功能性的需求，而没有非功能性需求的规定”），对进度计划也过于乐观。事后看，即使用去了 10 个月，开发商也付出了不懈的努力，客户仍不愿意验收和支付尾款，显然项目的实际进度已经远远超出了当

初的估计。既然如此，何必当初？一开始 PRM 团队其实并没有必要心急火燎地赶在 2 个月内交付系统，如果事先能多做些客户的工作（早期客户一般都对开发商充满着期盼和信任），把合同进度和条款定得宽松一些，把数据迁移和系统验收/β测试以及适当的返工时间和必要的补偿措施也考虑在内，给双方留有一定的余地，把基础打好、做得更扎实一些，结果就可能要乐观得多。

对此我们可能有些一厢情愿。为了应对激烈竞争，赢得商机，开发商往往对项目的前景作出过于乐观的估计，而客户也往往对软件开发的复杂性估计不足，甚至凭借买方市场优势对开发商要求过于苛刻，这些因素迫使开发商哪怕不行也要缩短工期、压低报价硬着头皮上，结果必然是两败俱伤。然而，这不意味着在此种情况下规范的软件工程就无能为力了，没有必要做出正确的项目估算和计划。如果开发商有了充足的理由和数据支持，至少可以据理力争，在知情的情况下有意而为之、防范于未然比因超时超量的意外而惊慌失措、亡羊补牢要好得多。

4）对客户的启发、引导、反馈和协作不够

原文提到，“在开发过程中，客户经理由于业务拓展的原因，并没有在项目上分配多少时间进行审查；而客户在交付前也没有花费很多的时间研究系统，也没有提交很多的反馈报告”（原文教训 5，我把它更多地视为一种项目管理制度的缺憾）。客户以及作为“现场客户”替代者的开发商客户经理没有依据迭代计划及时提供反馈，这也是导致 PRM 项目失败的重要原因。

现阶段，国内项目完全实施现场客户的难度较大，因而我们不能指望通过现场客户来提高项目的成功率，而实际上很多成功的项目也没有或者没有必要做到现场客户。其实 XP 的反馈环不一定要全部通过现场客户来实现，它只是达到明确需求、强化反馈的一种形式，在不同条件下可以有不同的变通处理方式。对于 PRM 项目而言，我以为在迭代周期中引入由客户参加的正式架构评审，使关键问题能够被尽早发现，可能比现场客户起到的作用更大。

人们爱说“客户是上帝”，从事软件工程千万不能抱着这样的观念（其实是一种欺骗）——因为上帝是无所不能、无所不知的，而客户绝不是万能的，他/她们经常会犯错误，很多内幕细节客户不了解、不知情，也不可能比你更懂软件，所以我们不能等着客户来主动告诉我们应该如何去做，抑或对客户百依百顺，客户意见照单全收，结果只能是自吞苦果。软件工程更应该提倡“客户是朋友，是合作伙伴”的正确观念。只有通过规范的软件工程过程和正确的软件项目管理措施，在客户和开发商之间建立起良好的沟通桥梁和协作机制，双方将心比心，互相理解、引导和启发，才能增加共赢取胜的机会。

我还想起了 D. Dikel 在《软件架构：组织原则与模式》（拙译，机械工业出版社，2002 年 8 月第 1 版）中总结的架构组织反模式。不幸的是，PRM 项目出现的情况好象正好符合“电话铃未响”和“遗漏的部分”两个反模式——由于“电话铃一直未响”，在开发过程中客户沟通不畅（很可能是因为客户放心地认为开发商的客户经理非常了解业务，所以不愿亲自操心），项目组也就没有预见到在数据迁移后的大用户量情况下可能出现的问题并提前采取措施；项目组知道了明确的用户需求（合同规定的内容），却遗漏了为了向客户提供有价值的东西（合同之外隐蔽的需求）所应该做的事情（调整架构，数据库优化等等）；系统虽然及时交付了，却由于缺少一些客户必需的关键功能特性（保证大数据量处理和显示的良好性能），无法被客户所接受，结果用户得到的是不完整的解

决方案，它带来的负面影响足以超过任何已完成的新功能的价值（不满意的客户不愿继续支付）。回想起来，这些遗漏的细节却似乎又都是显而意见的。

如何避免重蹈覆辙？书中建议，需求调研的“覆盖面很重要”，“规范的东西可以确保没有东西被遗漏”，采取措施让客户参与协作，了解你的受益人、架构评审和客户互动等模式也有助于预防……

5）开发过程不尽完善

PRM 项目灵活采用了许多敏捷做法：“每日晨会、交叉审核、持续集成、测试先行、小步发布”，因而在项目进度的控制上的确取得了局部成功——项目前期进度得到了保证，按时交付了系统。

然而，质量、内容和速度永远是个矛盾。由于风险管理没有到位，PRM 项目看似满足了合同的进度要求，却在项目后期遇到麻烦，客户对系统的质量并不满意，项目结案的总体进度实质上是远远拖后了。我们在实施软件过程改进的过程中，往往容易机械地理解一些软件过程的规范和教科书，单纯地学习某些具体、个别的做法，却忘却了软件工程的基本原则，没有从根源上明白 XP、RUP 为什么要这样做。

原文提到，“主要原因在于项目开发商之前没有大规模业务系统开发的经验，对于非功能性需求没有足够的重视；同时，在测试阶段，也没有对于系统负载和性能做过测试。”（PRM 项目做到了单元测试、集成测试，但还缺少系统验收测试，至少在合同中对此没有明确规定）。我觉得可能不能简单地把问题归咎于开发经验不足，与其说开发团队没有大规模业务系统开发经验（事实说明大家的技术技能还是很不错的，出现的一些问题后来也逐步得到了纠正），还不如说组织的开发过程不尽完善，缺乏对实施规范软件工程（需求、架构、评审）必要性和重要性的强烈意识，被 XP 的个别做法迷惑了？我猜想完善过程和制度很有可能可以避免类似情况的发生。如果通过迭代交付，使系统尽早上线进行数据迁移方面的试验（既然合同上已经明确提到了这个需求），并通过严格的需求和架构设计评审，对付页面显示和并发用户访问的性能问题就将更加从容。

6）OOAD、设计模式与 DB 的关系（对教训 3、4 的补充）

软件 OOAD 与数据库设计、优化并行不悖，两种技术解决的是不同领域的问题，在项目管理中应该并重。无论软件开发采用的方法是传统结构化还是 OOAD，数据库和 DBA 的效能对于企业应用系统来说始终都是关键因素，对系统性能和质量有着重要的影响。有些人认为 RUP、XP 对数据库介绍的篇幅不多，好像对于软件开发方法论数据库技术就不重要了，这完全是一种误解。另外值得指出的是，面向对象设计同样可以产出高性能的软件，两者并不矛盾。不要被设计模式所迷惑，应该避免在错误的场合运用模式。

7）CASE 工具的局限性

PRM 项目采用了不少先进的 CASE 工具（例如 Rational ClearCase、ClearQuest、Robot）来进行代码、缺陷和测试等管理，应用水平在国内是属于领先的。自动化工具对于提高效率、保证进度必不可少，但工具的作用也是有限的，工具是为人服务的，它们不能替代正确完善的软件项目管理和开发方法论。PRM 案例恰好验证了即使再好的工具（“银弹”）也无法弥补、挽救在做事方式、方法上存在的缺陷（归根结蒂还是人的因素）。

总而言之，PRM 项目给我们的教训是很深刻的。我想它遇到麻烦的主要原因在于前期需求分析、架构设计不足，加上系统测试存在漏洞，迭代式开发也没有起到应有的客户反馈效果，结果造成虽然 PRM 系统被及时交付使用，但是一些关键需求被进度顺利完成的假象掩盖了，导致项目后期才暴露出缺陷，给开发商造成了很大被动。这正是我们通常不愿看到的最坏结果（可谓经典案例）。

以上事例反映了 XP 应用存在的一些典型陷阱。不能因为 XP 强调敏捷、极限就省略软件过程的一些关键环节和必要步骤，这方面 RUP 能给我们较大的帮助。我们不能只学习 XP、RUP、CMM 的表面知识，更应该注重掌握软件工程的基本原则和软件过程的核心设计思想。该案例还提醒我们不能把项目的成功完全寄希望于事后重构（除非你“艺高人胆大”），针对架构打补丁实在是很不划算、很不聪明的做法，会让组织和个人冒很大的风险（一旦架构瓦解）。宁可与客户一道多花些精力做好前期工作，尤其要掌握全面需求分析、风险分析和控制的方法，做到善于捕捉潜在的需求和问题。

IT 之源　张恂
zhangxun2001@hotmail.com
2003 年 7 月 29 日