以过程管理技术规避软件开发的管理风险

To Avoid Management Risk of Software Development via Process Management Technique

马慧　杨一平　张萍　(首都经济贸易大学信息学院　100026)

摘要：本文揭示了开发中常见的问题。同时描述了主要参考CMM过程关键域的具体内容，对于规避风险以及软件项目质量的提高均具有重要的指导意义。

关键词：管理风险　规避　CMM　过程关键域

1　了解软件开发的主要管理风险

我们必须承认：太多的软件项目开发失败了。以往虽然大多数项目也有计划，并具有项目里程碑，项目也提供了软件工程要求一系列文档(例如逻辑模型、物理模型等)，然而，由于项目缺乏有效管理与控制的方法，不少项目尽管花费了大量的人力、物力和财力，但开发到最后，却被描述成既超出预算，质量又很差。项目管理仍然很困难。基于IT行业的特殊性，我们应首先了解风险。孙子兵法："知己知彼，百战不殆"，对于软件项目管理者而言，这个"彼"指的就是"风险"。下面我们将具体描述项目开发常见的问题。

1.1　了解系统规划阶段常见的管理问题

(1) 规划缺乏总体性

(2) 不现实的计划

过紧的计划必然会降低工作的精确度与质量。开发队伍频繁地加人、换人，他们无暇顾及到设计的合理性。实现的功能范围缩小，项目仓促收尾，造成恶性循环，见图1所示。

(3) 缺乏管理上有效支持

1.2　了解软件配置管理薄弱的严重后果

(1) 软件产品缺乏可视性

(2) 缺乏完整性和一致性

(3) 造成软件产品跟踪与审核的困难

软件的跟踪与审核的基础是基于软件配置管理的基线库，而许多项目缺乏基线库的管理，对软件的跟踪与审核是非常不利的。

(4) 下一阶段计划缺乏依据

1.3　了解培训工作中常见的管理问题

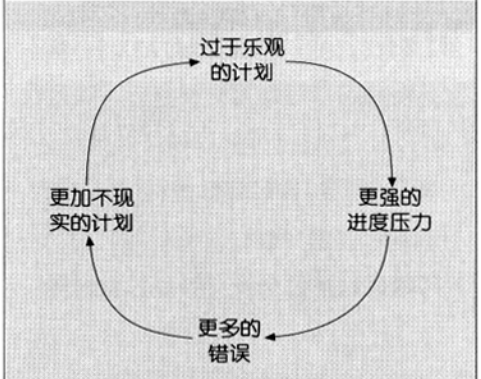

图1　过分乐观计划的不良后果

我们参加了各种项目管理，不时惊讶地发现，培训工作管理非常薄弱具有普遍性，培训的效果经常被不停的打折扣。具体表现形式如下：

(1) 认识上的错误

培训工作错误地被认为是仅针对程序使用的培训工作。

(2) 错误的选择了培训时间

错误的选择了培训时间，许多员工频繁的出差，日常工作较繁重。如果时间选择不好，培训的效果就要打折扣。

(3) 过窄或过宽的培训范围

培训范围选择的过窄或过大均会影响培训的效果。

1.4　了解软件转包的主要风险

软件转包对公司是有积极作用的，但绝不要因此而简单的认为：通过外包可以将原本复杂的开发工作变得及其简单。事实并非如此。当软件产品在一定范围内甚至在全球范围内转包时，开发方将面临着较大的风险，他们可能几乎失去了项目的必要可视性，可能造成有些专门的技术外

流，并且对进一步的开发失去了控制。具体描述如下：

(1) 可视性降低

软件转包后，最主要的风险之一是项目进度、质量、管理等情况可视性下降。

(2) 专业技术水平下降

软件外包后可能会造成公司开发能力的下降，同时，开发方不得不将一些内部技术公开于承包商，长此以往，开发方的技术水平会有所下降。

(3) 下一步开发失去控制

软件外包后，承包商可能故意编制一些秘密程序和技巧，其目的是限制其他人的修改。另一方面，开发人员很不情愿地去熟悉别人(承包商)编写的源代码。有时，在合同中还限制了开发方对外包代码进行修改的权力。

1.5 了解需求分析阶段常见的问题

捕捉真正的需求是困难的。需求分析中最具有挑战性的问题是如何获取真正需求见图2所示。对需求的误解将会直接影响着后期的开发工作，一旦需求分析中出现了漏洞或偏差，将会导致较大的风险。国外企业的业务较规范，需求很清楚，但国内企业或组织往往不能清楚地描述自己的需求，造成软件项目管理的难度加大，甚至项目失败。捕捉真正的需求困难的主要原因描述如下：

(1) 用户不能准确地表达需求

(2) 用户参与不够深入

(3) 研究导向造成的目标偏离

一些开发团体尤其喜欢甚至着迷于新的技术，他们渴望尝试新的开发语言，新的环境或是建立新的算法。使用新的算法虽然有利于研制人员发表文章或通过技术鉴定，但是，容易引导开发人员将太多的时间和精力陷入一些难度很大、风险较大但并不是主要的需求模块，从而偏离了需求分析的主要目标。

(4) 开发人员缺乏交流与说服能力

1.6 了解系统设计常见的问题

(1) 没有充分利用系统分析的阶段结果

系统设计没有以系统分析为基础，设计师只是凭借经验和自己的理解设计系统。

(2) 设计难度不适度

设计过于简单，甚至无法确定主要事件和相互关系；或是过于复杂，导致不必要的投入。

(3) 混乱的设计易造成水波效应

系统内部模块之间的耦合度太大，由于设计的混乱，在修改时易于产生水波效应，即一个模块的修改而导致缺陷的隐含、放大以及一连串的新错误的出现，见图3所示。

(4) 对开发工具的过高估计

过高的估计开发工具带来的效益，从而不切实际地估计了设计进度。

(5) 忽视总体效果

设计时只关心个别用户的需求，而忽视了总体设计。

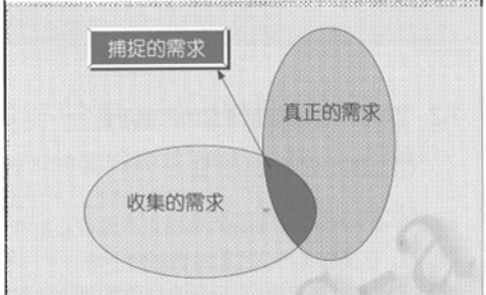

图2 捕捉真正的需求

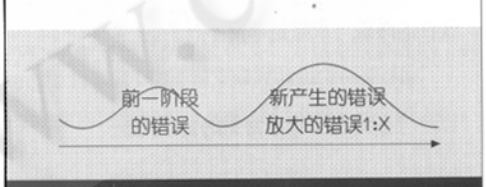

图3 错误放大易产生水波效应

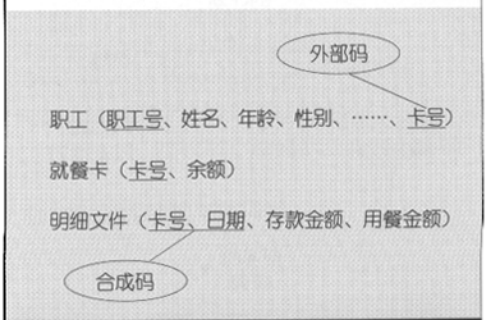

图4 外部码的设计

(6) 综合查询难于实现

数据库设计的不合理。在数据库规范化、分解的过程中，漏掉了必要的合成码和外部码。这样会造成综合查询实现的困难，例图4中的职工文件中的卡号就是基于综合查询而设计的。

1.7 了解系统实施阶段常见的问题

相对于系统分析、系统设计阶段而言，系统实施阶段的工作量较大，投入的人力、物力多，出现不可预测的错误会较多。具体描述如下：

(1) 人员安排不妥

为了测试相对客观，测试工作人员和程序编写人员不应是同一些人员，即测试工作应尽量不安排那些编写程序的人员。

(2) 人员分工不明确

没有建立较严格的管理制度，人员分工不明确，从而造成有些工作遗漏，而另一些工作重复开发。

(3) 测试用例不全面

测试用例不全面，用例仅包括合理的数据，而没有包括无效和不合理的数据。测试的用例缺乏全面性。

(4) 鲁莽的编码

开发人员为了加班加点追赶进度，甚至在比较恶劣的工作环境下工作。编码质量差，重用性差，文档不齐，说明混乱，鲁莽的编码均是遗成以后问题的隐患。

(5) 缺乏复审

软件是个逻辑产品，产品中隐含大量的错误。软件复审是软件工程过程的过滤器。没有复审的软件不仅包含大量的错误，而且一个错误还会连带若干个错误造成恶性循环。

(6) 对并行转换时间错误的理解

有的单位把手工和计算机的双工方式的时期简单的理解为对软件的可靠性、准确性的测试。因此，把双工并行方式的开始时间定位在系统开展软件使用以前，例如，假设单位决定今年6月份开始使用新的系统，而将3，4，5月的数据输入作为验证系统正确的数据，上述做法是不对的，正确的做法应该是用系统开展软件使用以后

即用6，7，8月的数据进行验证。

(7) 不恰当的纠正错误的方式

在测试出错误后，没有经过纪录、上报，统一安排，而是匆忙的进行编码。易造成是水波效应。

(8) 缺乏安全管理的系统转化

系统转化是存在风险的，如果转化工作的随意性较大、缺乏安全管理，势必会造成严重后果，其中包括：原系统数据的丢失与混乱、原系统程序的丢失与混乱、当前阶段原始文档的丢失与混乱、新系统数据的丢失与混乱等以至影响正常工作的严重后果。见表1所示。

1.8 了解维护阶段中常见的问题

对"老"的程序维护是困难的。软件维护是件不吸引人的工作，它的工作难以表现出成果，而且工作量大、困难大。造成维护工作困难的原因是由于系统分析和系统开发过程的缺陷以及软件本身的性质造成的，具体描述如下：

(1) 阅读程序的难度

阅读别人的程序是困难的，一般人都有这样的体会，与其修改别人的程序还不如自己重新编新的程序。如果还缺乏程序的说明文档（数据字典），那将造成维护人员不知所措的局面。

(2) 文档不规范、信息不一致

文档不一致、不规范是造成维护工作困难的又一个原因，这种不一致包括文件和程序的不一致以及文档之间的不一致等情况。

(3) 修改程序产生副作用

修改容易造成一系列新的错误即水波效应，有时，对一个错误做一个简单的修改，都有可能导致灾难性的后果，而产生这种副作用的机会很多，其中包括修改程序代码的副作用、修改数据的副作用、修改文档的副作用等等。

2 借鉴CMM过程管理规避风险

软件过程改进的理论以及软件项目开发中遇到的严重的、长期的、顽固的压力、危机与风险，告诉人们一个朴素而深刻的道理：软件开发中出现的一系列成本、进度、质量等问题，仅仅依靠技术是不可能提供较完整的解决规避风险方案的，项目开发中暴露出来的深层次的问题与长期纠缠的风险引发我们进行深入的思考，即应借鉴国外软件过程管理CMM的思想，指导国内软件工程开发的实践。其中，CMM的公共特性主要揭示了有关职责和目的，关键实践揭示了基础设施和活动等内容的过程管理要求。

表1

信息缺乏安全管理					人员、进度、成本缺乏控制
原系统的信息被破坏		新系统的信息被破坏			分工、计划、 资金的管理混乱人员不到位、 资金缺乏
数据丢失	数据混乱	数据丢失	数据混乱	日常的工作文件丢失	

表2 捕捉真正的需求

关键过程域 / 开发阶段	主要参考的关键过程域	一般参考的关键过程域
项目准备与计划	软件项目计划 软件配置管理 软件转包合同管理 培训程序	软件产品工程 缺陷预防
需求分析	需求管理	组织过程焦点 组织过程焦点
系统设计	软件项目跟综和监督	同级评审
系统实施	软件质量保证	组间协调
系统的评价与维护	软件配置管理、基线管理	过程改革管理 技术改革管理

在CMM中有18个关键过程域，分布于2~5级中，他们在CMM的实践中均起到了重要作用，但每一个KPA的侧重面是不同的。在软件工程中的每一个阶段的工作特点不同，容易出现的管理漏洞不同，应重点加强的管理内容不同，因此，针对软件工程的阶段管理风险，

我们建立与关键过程域的联系，在框架中描述了在软件开发中每一个阶段应重点借鉴CMM的重点过程关键域的不同，见表2所示。

需要说明的是：CMM的每个关键过程域本不是面对某一个特定的过程，它们对整个开发过程起作用，上述表格并不妨碍关键过程域对全过程的指导作用。但毕竟在开发过程的不同阶段，技术要求不同，容易出现的管理问题也是不相同，关键过程域的应用也应有所侧重。我们总结了长期开发软件项目的实践，尽量考虑以简单并便于操作和实施的方式，描述了上述表格框架。鉴于CMM细节的具体内容较多，在这里不再赘述。

根据上述框架，开发方可以根据具体情况拟定自己的过程管理规划，以往的事实已经证明将来也将进一步证明CMM的过程管理技术对于规避风险、化解管理问题具有重要的作用。

参考文献

1 杨一平，"现代软件工程与CMM技术的融合"，人民邮电出版社，2002，12。