step365思步网.com

会员自己的 精品 读物

记录我们的足迹　　分享成长的点滴

电子期刊

作者：思步网会员

主编：fishred

编委：step365he Scott
lily_014 坠入深海
cecilia steplv

校对：sungubbi steplv Scott

第六期

封面設計：steplv

发布：2008.05.18

著作权说明

由于编者水平所限，在文档中难免会出现错误，欢迎读者对本文档提出批评建议，意见请反馈至：service@step365.com ，谢谢！

阅读注意事项

在阅读本电子文档时，请留意以下字符（字母）所代表的含义。

本电子文档第一部分中：

Q（user name）：Q 代表问题 Question

A（user name）：A 代表解答 Answer

C（User name）：C 代码结论 Conclusion

括号中内容 user name 为思步网会员名

例如：**Q**（steplv）xxxxx，**A**（sungubbi）zzzzz，**C**（fishred） yyyyyy

表示：

提出问题者为会员 steplv，问题内容为：xxxxx

解答问题者为会员 sungubbi，解答内容为：zzzzz

做总结者为会员 fishred，总结内容为：yyyyy

依次类推。

思步在线交流

加入思不 IM 群，与众多朋友在线交流：

- MSN群：mgroup24287@hotmail.com
- QQ群：19894513（申请加入时，请注明职业）
- WebSite：www.step365.com

目　录

第一部分：思步简讯

2008 奥运年的五月一个令人喜、令人忧、令人愁的季节，“我为人人，人人为我”在这一季发挥得淋漓尽致。在这一特别的季节，我们喜的是：圣火的成功传递；我们忧的是：四川汶川地震还未解救出来的灾民；我们愁的是：灾后重建工作及对奥运会的影响。在思步群中，会员们时刻关注汶川灾情，并为灾区的人民贡献自己微薄的力量。思步论坛也开辟了两个火炬传递与救灾两个专题。

特别的季节，思步也在一天天的成长，一些激励政策和措施也相继出台，一些疑难也被一一解决，以下是我们近月所做的一些工作：

1. 思步会员积分策略新鲜出炉。让乐于分享资料者，获得更多的资料，实现公平、等值交换。
2. 实现论坛金币与个人空间金币的兑换， 满足会员个性化的要求，兑换比例为 1：1，思步网不收任何手续费！
3. 组建全员改进过程及 QA 学习组（PineTree Group），学习小组的目标是：让新手正确入门，少走弯路；让有经验者，经验更丰富，让经验丰富者，提炼思想；让有思想者，发挥影响力！打造一个致力于在国内推动全员改进过程的优秀团队。目前团队已经初俱规模，并将分享他们的第一次工作成果。
4. 完成全员改进过程培训调查问卷

[冰封伯爵　编辑提供]

第二部分：你问我答

1 全员改进案例

★ 如何做好 CMMI 过程改进工作？

Q（冰封伯爵）：

有以下疑惑需要大家帮忙解决：

1、如何有效的开展过程改进意见收集？

2、公司高层支持过程改进，但是中低管理层及基层员工不积极配合，碰到这种情况该如何解决或改善？

3、如何使改进的过程得到有效的推广实施，适合公司现状？

4、针对以上问题，需优化改进那些过程来缓解过程改进现状。

Q（coral）：

ＣＭＭＩ评估通过后，过程改进的速度就比较缓慢了.

公司９０％的项目都可以不按照制定的流程执行了．而且即使１０％的项目按照流程执行，也不是严格的执行的．ＱＡ上报的话结果高层说可以减低控制力度，但是该让他们减多少高层也不给明确解释，所以也导致了ＱＡ也关注力降低，其实是无法关注．长此以往组织级ＱＡ也无法采集一些数据进而分析.

目前觉得公司的ＥＰＧ处于混乱状态，根本就是挂名没有做什么事情，即使做也是1－2个月组内某个人做1件事.

该如何改变此现状呢？

A(lee_huo):

这个问题在国内是普遍存在的。

1.一定要引入一定的奖惩措施

过程改进工作是需要激情和投入的。无论是 EPG 组工作、评估项目组的工作，都会有额外的工作量产生。产生原因可能是对模型理解的不够，人员经验不足。在过程改进工作之初就要建立并得到高层同意的奖惩措施。奖的目的是鼓励、激励大家做事情。惩的目的是督促大家去做事情，按职责要求完成。有了奖惩措施方便 EPG Leader 执行工作。

2.高层经理一定要参加而不是参与

高层经理参加到过程改进工作中，其显示出的意义就不用说了。领导都以身作则了，我们还有什么可说的。高层经理切忌只高调的喊口号而无具体的行动，一定要找些切入点参加进来。例如参加 EPG 例会，解决会议中出现的问题；定期审批 EPG 报告，协调资源冲突等。让所有人感觉到领导在重视我们、重视过程改进工作。

3.EPG 成员的选择

作为 EPG 成员，首先要熟悉并达到熟知模型要求的内容，还要将模型中的实践融合到实际的过程体系文件中，还要为评估项目做指导等工作。所以对 EPG 成员的要求比较高，最好的工作经验在 3 年以上，时间太长容易变得顽固，时间太短有理解不上去。EPG 成员要包括技术能手、QA、测试、CM 等。最好不要把部门经理、项目经理等角色纳入到 EPG 中，因为他们在实际工作中很难把经理投入到过程改进工作上。搞不好这些人的任务要其他 EPG 成员协助完成。

4.评估项目的选择

如果公司为了减少评估的风险，可以使用无进度压力的项目；例如已经完成的项目。但这种项目对成员的锻炼价值不高。如果有进度要求的项目建议不要选择使用新技术、新人的，这样的项目风险比较大。项目经理的选择要选择喜欢接收新鲜事物的人，尤其是能接收过程改进、重视质量保证和测试的人。如果是评估 3 级建议选择 4 个评估项目，其中一个项目作为候选项目，以免在评估期间出现任何问题，而是评估结果收到影响。

5.咨询公司的选择

选择咨询公司要根据每个公司的特点和商业目标决定的。不过还是要 EPG 和 MSG 的领导清楚，如果公司的目的就是想尽快通过 CMMI3 评估，而并非是想通过改进工作机会提高员工的知识技能。在选择咨询公司上就有一定的说法了。如果是为了第一个目的建议选择小的咨询公司或主任评估师自己开办的评估公司。这样能起到加速作用。

A(冰封伯爵):

目前我们公司的 EPG 成员都是部门经理和项目经理担任,可能也验证你所说的,他们没精力投入 EPG 工作中来.

现在最需要解决的是,如何让公司更多的成员参与过程改进,现状况就是除了我们老总、软件部门经理和我外，其它人都默闻不问，不配合，公司领导组织大家开会鼓励大家参加过程改进并明确了一些奖惩措施，就差相关奖惩制度没有出来。但是没有什么效果呀，大家都装聋作哑，要么就是口头答应，接下来就是慢慢的拖和你磨时间；真的不知怎么办，怎么进一步改善公司现状，调动公司组织成员的积极性。

A(sungubbi):

1、如何有效的开展过程改进意见收集？

全面铺开的意见收集。可以获得底层员工的真实期望和需求，特别是他们实际在体系中的实施感受

对公司高层和部门的负责人进行意见的收集，结合公司的目标与他们的工作改进需要收集意见

2、公司高层支持过程改进，但是中低管理层及基层员工不积极配合，碰到这种情况该如何解决或改善？

责任落实到人，关于过程改进的任务和目标需由高层直接下达到相关的部门，部门产生这方面的需求就会找到 EPG，而尽量避免由 EPG 下达这些任务给相关部门。在部门中进行过程改进推进时，将部门的负责人、骨干作为切入点，取得他们的信任和支持，然后在底层员工中进一步推广。

3、如何使改进的过程得到有效的推广实施，适合公司现状？

同 LEE_HUO 意见，选择合适的项目或团队进行试点，再逐步推广。

4、针对以上问题，需优化改进那些过程来缓解过程改进现状。

需要优化的是工作的方式与方法

如果公司高层支持，并且也无法获取关键执行人的支持，实际上就很有可能不了了之了。

我始终认为过程改进的职责不应只落在 EPG，过程改进的职能应在各部门的职责中得到体现。每一个方面的管理工作总有一个归口部门，可以让归口部门的人参与 EPG 工作。

由于过程改进执行的周期较短，测量基础数据不够充分，起初很难定义控制目标或范围，比如进度偏差是控制在 5%还是 10%，此时可以结合业界的经验以及企业当前可接受的偏差来定义一个范围，例如 5%-15%，根据实际情况对于进度要求高的定小一些，进度要求不高的定大一些。随着基础数据地增加，就可以逐步缩小范围值，并且可以基于以往的项目类型做出不同的控制范围。

A(cecilia):

规范及流程的制定相对来讲更简单，因为咨询公司会给我们提供相应的模板，然后再由公司对其进行裁剪。不过要制定真正符合公司发展效益的质量管理体系，却不是那么简

单的，要经过 SEPG 大量的讨论，然后编制规范及流程。而且需要在项目中进行试运行，在试运行过程中又需要对这些规范及流程进行完善。

规范及流程制定好后，我觉得执行力度是一个很大的问题，执行力不强，那也只能是纸上谈兵，根本起不到作用。如果高层不支持的话，那在通过评估后，将又会回到以前未使用 CMMI 的情况。

A(iamredeye):

引用: 原帖由 coral 于 2008-4-17 15:13 发表

ＣＭＭＩ评估通过后，过程改进的速度就比较缓慢了.

公司９０％的项目都可以不按照制定的流程执行了．而且即使１０％的项目按照流程执行，也不是严格的执行的．ＱＡ上报的话结果高层说可以减低控制力度，但是该让他 ...

这正好是一个生产力 VS 生产关系的例子。这种情况说明生产关系相对生产力来说太超前了。流程改进不妨更缓慢和稳定一些。

A(w7a8):

引用: 原帖由 coral 于 2008-4-17 15:13 发表

ＣＭＭＩ评估通过后，过程改进的速度就比较缓慢了.

公司９０％的项目都可以不按照制定的流程执行了．而且即使１０％的项目按照流程执行，也不是严格的执行的．ＱＡ上报的话结果高层说可以减低控制力度，但是该让他 ...

每个项目都是唯一的，不可能所有的项目都按照一套流程执行！

要根据具体的项目进行具体的裁剪，不存在严格的执行。

一句话，适合项目的流程才是最好的流程，无谓坚持制定的流程是怎么样的！

A(丁字口):

首先确定过程改进的目标，然后逐层分解，最后将任务下达到具体责任人进行跟踪，定期反馈进度；

组织一定要有过程改进的决心，全员参与才能达到效果，否则只是两张皮；

要有一个强有力的 EPG，EPG LD 站在高处统览全局，确保目标达成，EPG 成员要有工程经验，做好基础工作，制定有效的规程，既符合模型又符合项目需求，对实际问题应有合适的解决方案，模板、指导书、检查单应有指导意义，模板最好样板化；QA 要多参与项目活动，多做引导；除对过程进行 QA 外，可适当组织有经验的技术人员做技术 QA；

把每一次的困难和困惑作为提升自己的机会，要坚持啊，各位老大。

在有过级的任务时目标明确，即使是形式化的东西也会做的有板有眼，一旦过级就会

皮软，这是通病；

首先要将过程稳定下来，做好执行与监督；

要有 MA，按规程做的与不按规程做的项目结果是不一样的，否则没有说服力。

C（lily_014）：

1、明确要改进的目标。

2、加强沟通，深入项目组，深入底层员工，全面展开建议收集，让企业全员参与其中。

3、高层的支持。对于中低层不配合情况，可适当引入奖励措施，责任落实到具体人。

4、合理选择 EPG 成员。

5、选择合适的项目试点，根据不同的项目情况作裁剪进行实施，通过实施并不断总结，并且收集数据做分析、度量来实行持续改进。

★ 大家的 WBS 都是如何写的？

Q（w7a8）：

分了几层，细到什么粒度，是按照两周原则呢还是有不同？里程碑如何设置？

A(lily_014):

通常情况下分三到四层 这样一个粒度来分

模块

子模块

功能点

模块

子模块

任务包

任务项

基本上是每个任务项（功能点）不超过两天（16 小时），不过这还是个设想，具体还没执行呢，这也是借鉴敏捷开发里面的。

我也有同样的问题，如果采用迭代的话，里程碑该怎么设呢？

A(iamredeye):

粒度到 manageable & measurable。具体时间和项目大小特性都有关。

里程碑就是你特别想去 measure 的点，对迭代和瀑布来说有什么本质上的不同呢

A(w7a8):

每个任务项，即最小工作包，有个 2 周原则，即 80 小时原则！

对里程碑而言，无需区分用的是什么 life cycle！

同意楼上的，粒度应该视项目具体情况而定，PM 觉得能管理能度量就可以了！

文无定法，符合项目实际（考虑各种约束）的才是最好的！

A(lily_014):

前面我说的那个是实际的情况，之前楼主不是问各个公司是怎么做的？

任务的细分程度当然是视情况而定，前面也说了是借鉴敏捷里面的，因为要做到可视度，而且如果要引入每日构建的话，通常是开发人员拿到任务后就不用再去细分了，直接可以 coding。

纠正一下 每个任务项，即最小工作包，有个 2 周原则，即 80 小时原则！

我理解的是，工作包里面含每个任务项，工作包 2 周原则，而任务项是 2 天。当然这是理想的状况。如果有误解欢迎大家纠正。

A(iamredeye):

你说的任务项大概是指 schedule activity。还是要看项目特性，比如规模，对大的项目 2 天显然太细，也做不到。计划也是有成本的。对需要特别 monitor 的成员，2 天却是一个好办法。只有一个原则，就是粒度让 pm 感到安全。

另外 WBS 是 project scope，千万不要又变成了 product 功能分解

A(lily_014):

对，很多时间领导会质疑开发人员的工作量

A(xixiaojing666):

我也很想听听迭代时里程碑的设置？

我们不管多大的项目，最后任务的粒度也会在 2-3 天，或更小，一般在项目开始的时候粒度会大点，当进入该阶段，进行制定阶段详细计划的时候，粒度就缩小了。

感觉在项目执行的时候，制定计划对项目经理来说很费劲，而且很多计划到最后都不能执行，也就是说计划变化很快！

A(冰封伯爵):

做 WBS 的分解前提条件是已选择了相应的软件生命周期模型，根据相应的生命周期模型做初步分解，然后在日常工作中逐步细化，没有不变的需求和计划，时时在变，如果跟不上更新的脚步，计划将无任何实施意思。

A(iamredeye):

引用: 原帖由 xixiaojing666 于 2008-5-7 16:45 发表

我也很想听听迭代时里程碑的设置?

我们不管多大的项目，最后任务的粒度也会在 2-3 天，或更小，一般在项目开始的时候粒度会大点，当进入该阶段，进行制定阶段详细计划的时候，粒度就缩小了。

感觉在项目执行的时 ...

项目有大有小。你们不管多大的项目，最后还是 2－3 天或更小－－这是因为你们不管多大的项目还是很小。

另外你们的项目计划执行成问题也反应了这个计划有问题。可能的原因是：

－粒度太小

－这个计划只是 PM 一个人拍脑子定的吗？如果是，那就可以理解了。pragmatic？

－没有 change mgmt

－没有 risk mgmt

－maturity…

A(Scott):

还是要看项目特性，比如规模，对大的项目 2 天显然太细，也做不到。--同意。

WBS 的制定方式最好考虑交付件，按照交付件来制定会比较合理，也比较好度量。比如有需求说明书，那就制定

xx 功能

需求说明书编制

评审

基线

里程碑

如果是迭代开发，可能就要以迭代为里程碑。

比如

xx 需求编制

yy 需求编制

评审

迭代里程碑

粒度要看从哪一层算起，如果是项目经理一层，建议最多 3 层，一层项目（包含公共任务，比如培训、会议等），二层迭代，三层活动。

C（lily_014）：

1、根据项目的特征、规模、生命周期的不同来制定；

2、不要把 WBS 变成功能的分解了；

3、粒度易于管理和测量即可。

2 全员改进过程

★ 一起聊聊，如何量化评价项目执行情况？

Q（sungubbi）：

如何量化评价项目的执行情况呢？

我先扔几个常见的，大家一起补充啊。

1）用符合项的百分比：如果有 100 个检查项，其中 20 个不符合，而符合率为 80%，越高代表执行情况越好；

2）用 SCAMPI 方法：对检查的内容分别评价 S、W 等，最后得到整体的 FI\LI\NI\PI

3）分值法：对每一个检查项评分（如 100 分），得到平均分，平均分越高代表执行情况越好。

A（丁字口）：

如果组织级过程有效性比较高，可用第一种方法，应细化到过程单元。

A（iamredeye）：

是不是 80%符合流程的项目就比那些 60%的成功呢？量化本身也没错，但是用来评价项目的执行情况我感觉有些困难。这样的话又变成了用度量数据来考核项目，尤其是 pm。那么，比如，也许会导致 pm 做项目计划的时候会放过多预算或 buffer。总之他心里真实的 plan 和给你看的 plan 会不一致。这不是度量的初衷。

Q（sungubbi）：

决定一个项目的成功，有很多决定性的因素，关键在于你对这个项目的要求是什么，也许产品的质量很差，但因为抢占了市场，组织认为这个项目是成功的。

决定一个产品的质量，我们通常所讲的是质量三角架：人、技术与过程。如果单单是从过程的角度，当然不能片面地说过程执行得好，质量就一定是好的。

量化的结果在目前的企业文化底下，的确容易被用于考核，但是不是因为这样就不需要量化了呢？

就好像做项目的估算和预算，是不是因为项目经理真实的想法与拿出来的不一致，就不需要做估算和预算吗？

iamredeye ，其实关键不是在于要不要测量，而是在于怎么利用测量的结果，这个想法跟你没分歧吧。

A（iamredeye）：

是的。我是觉得这个百分数出来之后要发布给所有的 PM 可能会有些误导。可能 QA 或 SEPG 自己参考这个数字来做进一步流程改进比较合适。

A（scott）：

"如何量化评价项目执行情况"

who 来量化评价：QA

量化评价 what：进度、缺陷、工作量、规模。。。

why 量化评价：向高层经理提供项目可视性；给项目经理提供建议；

when 量化评价：各个阶段结束的时候

how 量化评价：

1、量化数据必须准确且可重复，即数据来自所有项目成员，并且重复统计会得到相同结果。

2、量化数据不能用于绩效考评。

3、量化数据对其他项目有横向可比性，可供组织级参考和使用。

4、量化数据至少要区分为业务数据和过程数据。

5、量化数据只是实际情况的反应。

6、量化数据应该不局限于本阶段，应该从项目的整理评价。

接下来使用什么方式或采用什么手段，我觉得都不重要了。

Q（sungubbi）：

SCOTT 把如何量化项目进行了更大范围的扩展,不过我的原意确实如 *xueer0124 所说*是指 QA 对过程执行情况的评价。

关于如何量化评价过程执行情况，请大家继续补充。

xueer0124 问的问题，也是我计划抛出第二个问题：如何使用测量的结果。欢迎大家讨论

A（sungubbi）：

接着如何量化评价项目的过程执行情况的话题吧。

我罗列了几人我曾经见到过的评价方法，不过我发现我遗漏了一些东西，虽然那些指标能够大致反应过程执行的情况，但是在进行过程检查时，我们还产生了其它一些数据，例如发现的问题、问题的归类、关闭的情况、问题分布等等，这些数据也能很好地反应执行的问题、定位改进点

A（iamredeye）：

问个问题：

－如果收集到的数据不能准确反映实际问题所在呢？（上帝心里一定知道这个答案。）那按照这些数据定位出的改进点和上帝心里那个答案会不会有比较大的偏差呢呢？会不会有负面效果？

A（sungubbi）：

为什么不能准确反应问题呢？

A（iamredeye）：

我就反过来说吧。如果数据能完全说明问题，那么极端一点你开始说的第一项－100％的符合流程要求是不是产品质量就有完全的保证了？答案当然是 no。那么到底问题在哪儿呢？

设计 metrics－》收集数据，这两步每一步都存在偏差的，尤其是第一步

A（sungubbi）：

约定一个前题，好像之前我们也有共识：数据不能完全说明问题，但不能因为不能说明问题就不进行收集。

对啊，为什么存在偏差？metrics 还是人？还是收集方法？

评价标准也需要不断地完善和精确。

A（iamredeye）：

“数据不能完全说明问题，但不能因为不能说明问题就不进行收集”

－ 这个没问题

“这些数据也能很好地反应执行的问题、定位改进点”

－这个我有不同看法。1。数据会不准确，甚至有时非常不准确；2。即使 100％准确，也不能全部反应客观现实，毕竟定义的 metrics 是有限的，而且很多东西是没办法完全用数据说话的，比如人的 ownership。这是很多人置疑 process 的地方。process 只是一个 support 的地位，收集到的数据可以作为流程改进的参考，但不是全部改进点都可以从这

里找到，甚至最重要的在这里找不到。

举个极端的例子，我不知道有多少人碰到过这种情况：testing engineer 也按流程走了，但是不认真，或没那么认真，这时候很多 bug 就溜走了。这是流程的问题吗，能完全反应在数据中吗？很难

Q（sungubbi）：

首先：质量三角架：人、过程、技术。如果割裂这三者的关系，而仅凭其中一个方面对产品质量进行评价、判定，都是不客观的。如果 EPG 在收集改进的参考信息时，只是参考了 QA 的数据，呵，我不晓得高层对那些改进点有多少信任，EPG 本身的工作也是失职的。如果指望着过程执行情况 OK，产品质量就没问题了，那我看目前 QA 的地位就不会这么难堪了。

其次：我可能没有表述清楚我的讨论前提：是对过程执行情况的评价，而不是对产品质量、组织性能的评价。准确地来说是 QA、QAG 对过程执行情况的评价。不可否认，QA 需要关注人、技术对组织性能、产品质量的影响，不过我认为这不是 QA 的重点，如果人的能力和技术的评价也由 QA 来完成，我看那些人事部门、资源经理、技术控制部门也该下岗了。

最后：过程执行的数据不能很好地反应问题，为什么不能反应？这本身也是一个改进点，组织需要进一步改进他的评价体系。

A（iamredeye）：

我在说 process，没有说 QA； QA 关心什么我并不关心

process improvement 目标如果不是为了提高 quality 或 productivity，那又是什么呢？提高 process 自己？这个。。。

“过程执行的数据不能很好地反应问题，为什么不能反映？这本身也是一个改进点，组织需要进一步改进他的评价体系。”

－这个没问题。不过这个不是 process improvement 的主要目标－－quality 和 productivity

Q（sungubbi）：

得保证评价体系的完善，才能更好地为提高 quality 和 productivity 提供信息，包括优化评价体系本身

A（iamredeye）：

理论上，这个完全没有问题。

但这个所谓的评价体系什么时候才能完善呢？never！it's changing all the time, in order to serve or match the varing quality & productivity goals.

what if there're conflicts between the so called evalution systems and the project success? what's the focus? if you made the wrong decision, you got complaints and challenges from project teams.

again, of course there's no doubt what you said is true. however in reality we gotta have the focus. The evalution system would never be the focus. At least the project teams think so. They're in the spotlight, not the process guys.

I typed a lot. Maybe I should give up arguing... It's not wise to argue here. I don't think it's clear enough for most people.

Q（sungubbi）：

OK，我明白你的意思。就像我们在两个层面一直讨论这个问题，而没有聚焦一样，正确有效的信息需要从各种途径的评价中发现共同关注点，而不能单方面片面的进行。

A（iamredeye）：

我在这个论坛里一直在发表谬论，是因为我对当前很多公司的做法不是很认同，我觉得过于教条了。可能中国人比较容易走极端吧，最早毫无 discipline 的 cowboy coding 到今天严格执行从外面传进来很多思想和方法论包括度量。这些东西本身都是很好的，但我觉得它们只是用来帮助管理的，怎么执行是个很大的问题。还是学好了理论忘掉再实践比较好。我 argue 的是这个针对流程的量化的评价体系，目前来说 ROI 是不划算的，甚至是负的 CMM5 一定比 CMM3 对公司好吗？。。。中国这些 CMM5 的公司能成为世界级的百年老店吗。我就不说了

A（iamredeye）：

中国的软件不行，当然各个层面都有问题（比如 maturity），但最根本的是管理。当然其它行业也一样。

公务员的管理水平也是令人赞叹。。。比如，那个 12 万申报的流程，“嘭”就蹦出来了，无语。有一个 user 对它表示赞同吗？

Q（sungubbi）：

我倒觉得不是谬论，其实蛮提醒我的，这些“谬论”正是源源不断地从项目组、高层、其它部门抛来的疑问。很多时候我们正努力实践这些“谬论”，是因为我们已经忘了为什么要这

么做，只是因为要做所以做。但是，我们讨论了这么久的结论是暂时不适合流程量化，那到底什么时候更适合呢？又该怎么做起？不知道 iamredeye 有啥好办法？

A（iamredeye）：

我并没有说完全不要收集数据，不要量化；我只是觉得不要走极端。比如收集计划和实际的偏差，defect analysis，这些都是很有意义的东西。

但比如你开始提的：

1）用符合项的百分比：如果有 100 个检查项，其中 20 个不符合，而符合率为 80%，越高代表执行情况越好

2）用 SCAMPI 方法：对检查的内容分别评价 S、W 等，最后得到整体的 FI\LI\NI\PI

3）分值法：对每一个检查项评分（如 100 分），得到平均分，平均分越高代表执行情况越好。

我觉得这些就是价值不大的投入。甚至很可能误导。

另外很多值得改进的地方不是靠数字推导出来的，走过去和 engineer 聊两句就说不定感觉到哪里出问题了。我们周围的环境真到了要靠数字微调的阶段了吗？听听 prject team 真实的声音也许更好。

当然也许我也走到了另一个误区。或者你能介绍一下上面这几种方法实施后给项目带来的价值以及大家的反馈。：）

C（fishred）：

1. **前提是项目过程执行情况；**
2. **量化的数据如果能能实际反映项目执行情况，在量化的过程中，可以为问题类型、严重程度等设置权值来进行计算；**
3. **另外对于过程改进的来源不仅仅来于量化的数据，和项目组成员的沟通也是过程改进的来源；**
4. **我们的过程改进是以提高产品质量、生产率为最终目标的；**

★ 怎么着手质量管理？

Q（谁的眼泪在飞）：

我之前学开发，而进公司后是“项目专员”职位，后期还要做质量管理这块，可怎么着手呢？请各位前辈指点一二，小女将感激不尽。

A（flyee_cn）：

一步到位还是循序渐进？

A（stop_start）：

不知道现在你们公司的情况是怎样，规范程度如何，但从你目前的情况来看，我认为可以先从以下三个方面入手：

1、如果你对质量管理一点都不了解，那应该先要去了解一下什么是质量管理，一些基础的理论知识还是有必要了解的。建议可以从 ISO 质量管理体系入手，这是比较简单、基础的体系，入门比较快，如果精力允许的话，可以再结合着了解一下 CMMI 的体系要求，因为你有开发的一些实践基础，了解这个体系应该也不会太吃力；

2、上面说过不知道现在你们公司的情况是怎样，但我想一个公司再怎么样应该都会形成一些特有的规范，你在学习理论的同时，可以结合这些已形成的公司规范进行比照，这不仅可以加快你学习的进度，而且也会加深你对公司管理的了解；

3、对于初学者、入门者来说沟通是很重要哦。所以你应该多向你们公司做这块的前辈学习，有什么困惑可以向他们提出，另外一个沟通渠道就是 step365，这里面聚集了很多有质量管理经验的同仁，大家会热心地为你解答的。如果你遇到什么具体的问题时，可以上 step365 中寻找是否以前有人问过类似的问题，如果找不到想要的答案，那你也可以提出来，让大家一起帮你解决哦。

另外我始终是认为质量管理是个循序渐进的过程，不论是在学习还是推行还是实施方面，楼上的，有什么一步到位的方法可以推荐的吗？

Q（谁的眼泪在飞）：

谢谢各位前辈的指导，其实我公司目前几乎没什么规范可言，之前也没人在做这块，所以我才感觉比较迷茫，不知如何下手，我经理叫我尽快出一套质量管理可行方案，不知可有借鉴之方案

一步到位的方法对我这个初学者来说难度太高了，而且公司还没有一定的规范，所以只能是循序渐进的方式了

A（sungubbi）：

建议使用 PDCA 或 IDEAL 方法，先找出公司最关注的地方、最需要改进的地方，然后着手改进。

技能方面，建议逐步补充对软件工程、项目管理以及方法论的学习。

组织层面，根据对企业的调查结果和改进方向，可先设置改进小组来推进改进工作，记得要有高层的参与哦。

A（chyfx）：

提高质量最有效的方法就是 同行评审和测试。

你可以先从这两方面着手，同行评审主抓 需求评审、设计评审、代码评审，测试主抓单元测试，只要这两块抓好了，保证质量会有很大提高。

A（sungubbi）：

呵，我不太赞成。同行评审和测试做好，质量就提高了吗？

如果是这样，就不会有质量免费、零缺陷，不会有“质量产生于每个人之手”之说。

我记得在另一个帖子中曾经讨论过，发现越多 BUG 的那个产品其质量一定是次于发现少 BUG 的那个产品。所以我们的目标是在每一个环节尽可能地不要出错，而不仅仅是在测试和评审环节发现错误。所以如果问：你的企业进行了哪些质量管理活动？而回答是测试、评审之类的，十有八九 TA 是搞技术出身的。

什么才是好的质量？质量是满足客户需要的程度，用户的满足感越高，质量越好。产品的实物质量、服务、培训、故障解决、进度、价格等都决定了用户的满足的程度，质量管理工作应围绕它们展开。

A（chyfx）：

哈哈，同行评审、测试当然不是保证质量的全部因素，但确是最容易见效果的，很容易就做到的。

如果让我说如何快速的提高产品的质量，我倾向于评审和测试哈。

A（chyfx）：

我对质量管理进行了一下总结，请到我的空间参考“全面质量管理”一文

http://www.step365.com/index.php ... iewspace-itemid-206

A（sungubbi）：

关于PDCA方法详见：http://www.step365.com/bbs/thread-627-1-1.html

关于IDEAL方法详见：http://www.step365.com/bbs/thread-626-1-1.html

chfyx的总结我看啦，总结得不错，把常见的质量控制的实践进行了概括。但我觉得对于全面质量管理而言是不是应该把范围更扩大一些呢？毕竟质量管理不是强调检查、验证，而更多的强调持续改进、全员参与、预防为主。

换而言之，如果编码一直BUG不断，测试人员找出了很多虫子，那是不是质量就提高了？问题的根源在哪里？也许进一步的分析，我们会发现那个需求根本就做偏了，不论是评审组还是需求分析师都没有正确理解用户的需求、没有充分地了解客户的本意，那是不是应

该要通过更有效地方法来提高用户需求调查的质量呢？这不是评审就可以提高需求调查的质量。也许分析的结果即不是需求也不是设计也不是编码的问题，而是在进行集成的时候代码并没有被完整包含，那么，是通过测试来发现这个BUG的方法更好呢，还是改进集成的工作方法更有效？

质量是每个人的责任，而不是依靠检查和纠错。评审、测试可以发现并排查问题，但不能依赖他们。

A（sungubbi）：

这张图，昨天我放到了群里，本来只是博众一笑。但是今天在讨论这个话题的时候，我又想起这张图，并且想起偶家的一件事。有一阵子，偶家的卫生间地板老是湿湿的（发现问题），然后偶去拖一下地板（纠错），过一会儿老爸去卫生间又看到地板湿的，又拖干，如此几天，每次湿了拖、拖了又湿，并且湿得越来越利害了。后来当家的男人出手，发现是止泄带坏了，重新用上新的，并且还重新对其它可能漏水的地方均检查了一遍，该补胶的补、该换的换，卫生间再也没有无缘无故湿了。

chfyx 说的没错，在有些企业测试、评审、验证是提高产品质量的最快速的方法，但一定不是最有效的方法。

如果我到一个新公司去，这个公司一穷二白，啥也没有，我可能会先全面了解一下企业的现状、哪些是最影响目前的产品质量、工作效率的环节、哪些又是最容易出效果的环节，从中选择一两项进行试点改进，工作方式可以参考 IDEAL 方法。过去的经历上我都是选择项目管理、需求做为入手点。通过项目管理，可以先掌握和监控项目进度，通过需求从项目的最源头开始改进。

A（chyfx）：

其实，我说“评审和测试是提高质量的见效最快的方法”主要是根据我目前企业情况得出的，而且我也问过其他的同行，都反映这两块较差。当然每个企业不同，可能并不是这样子。

我曾对我公司的产品开发类项目系统测试阶段时间占项目总时间的比例进行过统计，一共 6 个样本。

最大值 75%；

最小值 42%；

均值 57%

主要是解决 BUG 花费了大量的时间，分析得出最主要的原因是代码质量较差，编码阶段时的代码评审、单元测试基本没做，导致系统测试暴露很多问题，修改 BUG 难度加大，致使项目延期，客户投诉的问题不断发生。

A（stop_start）：

我认为代码评审与单元测试是一种事后补救措施，是捉臭虫的过程，当然它们会减少错误，但并不能真正解决代码质量较差这个问题。

所以我认为可以把这些问题代码拿出来，看看产生它们的真正原因是什么（当然这需要项目组相关人员的支持），比如只是新人经验不足引入的简单书写错误；还是因为编程习惯不好，编制了复杂的语句，以至引入错误；或者是详细设计时未对一些必要的内容进行说明造成误解而引入了编码错误等等。然后可以把这些原因进行归类，找出对应的解决对策，比如对新人容易写错的地方进行总结和培训，必要的时候加入到 MK 的 checklist 中；再比如对于不好的编程习惯是否需要在代码编程规范中进行规范，并对员工进行培训同时还可以制定一些自动检查工具进行检查，又如针对详细设计的某些细节描述不足是否可以在详细设计模板中固化这些说明等等。

总结以上所述，我很认可评审与测试是一项很重要的保证和提高质量的方式之一，但并不是一定的哈，不同的企业改进的需求是不同的，方式与着手点也是多种的，关键是要去发现真正的改进需求是什么，然后再对症下药。个人的一些拙见……

C（fishred）：

1. **首先先去了解下质量管理的理论知识，可以熟悉下 ISO9001 和 CMMI 体系；**
2. **熟悉公司现有的过程；**
3. **对于初学者，与公司中的前辈多请教；**

4. **多跟踪项目，会发现目前存在那些比较紧急的问题，然后着手解决这些问题；**
5. **评审和测试是质量保证的一种方式，但并不是做好这两个方面就可以保证质量的，关键要了解项目目前的改进需求是什么，然后围绕这个改进需求细化任务；**

3 全员改进工程方法

★ 关于基线变更后，上下游工程文档的更新方式，头疼哦！

Q（fishred）：

基线建立后，遇到需求变更，对应的产需、概设、数据库设计等文档不知道如何更新，如果更新原文档的话，工作量可是巨大无比；大家讨论下，如何操作会比较好；或者大家都是如何操作的，分享下经验也好；

A(tyrones_nc)：

文档一致性的问题在任何实施 CMMI 的企业中都会遇到，这是一个一直没有很好解决的问题。

我们曾经想过彻底的解决办法，就是模型驱动开发，让代码、模型和文档三者保持一致，只要更新其中的一个就能保持三者同步。我们是嵌入式系统开发，使用 C++语言，所以当时选用的 Telelogic 公司的 Rhapsody。这个工具很强大，基本上能达到目的。但是同样出现了两个很大的问题：一是以前是写 c++代码，但是如果采用模型驱动开发，相当于要采用 UML 进行设计，可公司 C++的专家要远多于 UML 的专家；二是工具生成代码时，会自动生成一堆无用代码，而嵌入式系统对缓存和内存的要求是很高的。再后来，我就离职了。。。听说后来这件事也没有解决，EPG 转而研究敏捷方法，希望能解决这个问题。

就我个人而言，通过引导项目进行最佳工程方法的推广中，我只能提供一些体会和建议。

1、需求文档：无论采用 UseCase、IPO、Feature or User Story，需求文档是必须的；如果开发或维护过程中涉及需求文档的改动（注意这里不是需求变更，包括需求增加、删除和修改，所以情况较少），建议（方案 A，下同）每月或在阶段点或每个版本统一修改一次；对于需求变更，必须同步修改需求文档（方案 B，下同），否则测试人员无法更新测试用例，甚至于无法有效测试。

2、设计文档：概要设计文档一般描述架构，除非需求变更或重构，这篇文档很少改动，可采用方案 A；详细设计文档建议多画图，少写文字，好好学习学习 UML，如果采用伪码

描述也要仅描述算法，另外，逐渐考虑使用敏捷方法，编码即设计，把设计思路写到代码的注释中，然后通过 xdoclet 等工具生成 API 文件，作为详细设计文档的替代，采用方案 A 定期发布；

3、数据库设计文档：这个我接触较少，但我觉得应该可以参见需求文档。

采用方案 A 时，QA 要定期审核，监督项目组完成；采用方案 B 时，公司最好有一个非常好的变更管理工具（可以访问配置管理工具），如果没有这样的工具，跟踪起来工作量很大，这时建议分开管理，一是变更管理工具仅记录 CCB 变更决策过程和结论，二是项目组维护的问题列表中对需要变更的文档和内容以任务形式进行跟踪，防止遗漏。

总之，我是认同部分文档同步修改的，因为在考虑同步修改内容时，也会衡量修改的工作量以及影响程度，否则很小的问题也会当成一个重要任务来跟踪解决。

A(Chfyx)：

建立需求跟踪矩阵 来跟踪需求的变更对相关文档的影响。

当需求变更时，就要分析需求变更的影响，当然其中包含了修改设计、代码的工作量。

需求的变更肯定是要带来工作量的增加的，如果不更新相关的文档，而只是修改代码，那就会产生需求、设计、编码文档的不一致问题。对于后期的交付、维护是不利的。

仔细想来，对于文档的修改，确实需要很大的工作量吗？不一定，很多时候是开发人员不愿意改这些文档，觉得麻烦，而更愿意直接编码。其实真正改起来，也并不复杂，反而也要求了开发人员，根据需求，先设计再编码的习惯。

Q(Fishred)：

目前我们的情况是需求文档、数据库设计文档更新起来，工作量都不是很大的；主要是设计文档，我们没有详细设计，目前的设计文档把概设、详设整合在一起的，开发人员看到设计文档就能进行编码的；所以需求变更时，维护设计文档的工作量有时是很庞大的；有人建议我们做设计文档分册，这样维护的工作量会相应地减少，但是针对我们公司这种情况，还是解决不了根本问题；因为一旦业务流程变更了，会涉及到多个设计文档分册的修改；现在真的不知道该怎么办了？即使采取定期维护设计文档的方法，巨额的工作量还是存在的，实施起来比较有难度，即使强制项目组维护，维护的质量就可想而知了。

C（深海）：

基线建立后，在遇到需求变更时，如何保持上下游工程文档的一致性，这是很多实施 CMMI 的企业中遇到的问题，修改需求、设计、数据库设计等文档会带来很大的工作量，项目组往往因为巨额工作量而不愿意维护或者维护质量低下。

针对这类问题的建议方案有：在开发过程中建立需求跟踪矩阵，跟踪需求变更对相关文档的影响，当需求变更时候，分析需求变更的影响，要求开发人员进行修改。

- 针对需求和数据库设计文档：
  - 如果仅是文档本身的改动而非由需求变更引起的改动，则可以按阶段或版本统一修改，由 QA 定期审核监督项目组完成；
  - 如果是需求变更，必须同步修改涉及到的相应文档（此时最好有强大的变更管理工具支持，如果没有必须做好相应的跟踪工作，防止遗漏）；
- 针对设计文档：
  - 概要设计文档：一般较少改动，可以按阶段或版本统一修改，由 QA 定期审核监督项目组完成；
  - 详细设计文档：建议多画图，少写文字，逐渐使用敏捷方法，把设计思路写到代码的注释中，然后通过 xdoclet 等工具生成 API 文件，作为详细设计文档的替代，按阶段或版本定期发布；

文档一致性是一个一直没有很好解决的问题，各企业可以在实践过程中总结适合本公司的方法（如果有条件的公司可以采用模型驱动开发这一彻底的方法），来达到既修改了需求变更涉及的所有文档，又不会带来巨额工作量的目的。

4 QA/EPG 专区

★ QA 和 SCM 可以是同一个人担任吗？

Q（歌唱吧）：

我刚来我们公司时是 QA 的角色，但是当时由于没有项目可跟踪，所以就兼任了配置管理员的角色，而目前新的项目启动了，领导要求我担任这两个角色，我一直很困惑，因为 SCM 的工作是由 QA 来检查的，基于这一点考虑 QA 和 SCM 应该是不同的人来担任的；但是公司为了节省成本吧，就不想再招人了，请问如果 QA 和 SCM 是一个人的话，那么这样会对项目带来什么风险吗？而我又该如何做？

A（steplv）：

看到这个主题，我先扯一些题外话，在我看来：

首先，应该看这个人本身心理特点与自身特点说起。

一般意义来说，这两个角色，都有一个共性的要求，这就是：做事踏实、仔细，富有耐心！

其次，就是专业素养。尤其是 CM，思维里面一定要整个库的架构在脑海里，哪些资源该放在什么地方，由什么角色来承担；何时产生基线；分支与版本的控制；分支与合并时的冲突等。这些都应该是重点，当然，更重要还有赋权！

说到 QA，不要一提起 QA 就想到检查，我一向很忌讳说这两个字——“检查”，不要把检查经常挂在嘴上，要学会引导。

好了，我们回到正题：

1、SQA 与 SCM 是同一个人的话，对于项目有什么风险？

首先，考虑同一个人承担这两个角色，能否胜任？如果不能胜任（指技能），一来直接地影响项目的支持工作，会间接地影响项目的工作环境（软件）与进度，二来会间接打击你的积极性，加大工作压力，造成工作中的逃避心理；如果能胜任（指技能），能否在正常的工作时间内完成相应的任务？因为工作量加大的原因，是否会有经常性的加班情况出现？久而久之，有可能不堪重负，压力过大，将会降低工作效率，并引起精神的健康问题。

当然了，SCM 日常的工作量可能相对来说少一些，SQA 的工作量，也有可能会少一些，一般情况下（有特殊情况存在）：在项目启动时，SCM 忙，SQA 相对闲一些（如果公司体系规范都比较规范，不再需要单独为项目做培训的话）；在项目进行时，SCM 忙，SQA 忙；在项目结尾时，SCM 忙，SQA 忙。要衡量好。

2、如果这两个角色由同一个承担，你该如何做？

既然公司有这个决定，说明你是有这个能力来胜任这两个角色的。至于这两个角色之间的权衡，我个人认为，在不同的阶段应该有所侧重，也不要顾此失彼，重要的一点：就是在按原则办事的基础之上，适当灵活一点，别太死板。

监督别人，往往比较容易，但监督自己，往往比较难，因为每个人对自己都有“宽容”这个倾向。如果你能在规范体系下执行 CM 与 QA 活动，只要找到平衡点，即可！

至于这个平衡点是什么，我觉得有如下一些情况：

A、这两个角色，对于你们目前的项目来说，孰轻孰重？

（如：既要控制好版本，又要保证自主知识产权的产品的源代码的安全性，这些重要吗？想监控项目，并不完全是 QA 的事，PM 的责任不能推托）

B、QA 的活动真的很需要吗？对于你们项目？

（如：项目中存在 QA 与不存在 QA 几乎没多大差别，那么，QA 的角色就要相对轻一些）

另：CM，我仅仅是站在项目级的角度来说，如果你还是组织级的 CM，那情况会更复杂一些。

A（歌唱吧）：

非常感谢 steplv，你分析的很全面很详细！就象你在第一条中分析的风险，我也正是基于这方面的考虑所以想推掉 CM 的工作，而只想做 QA 的相关工作。现在看来我就可以从规范的角度及风险的方面来说服我们领导：两个角色最好是不同的人来担任！你还有什么好的建议吗？再次谢谢。

A（steplv）：

我第一次的回复可能有些极端的倾向，其实，我觉得不妨一试。

凡事，只有经历过的人，才知道是对是错，别人的想法与意见，仅仅是一种决策的参考而已。

如果是单个项目的项目级的 CM，其实日常的项目进行过程中也并不是天天都忙的一塌糊涂；QA 也是，如果是单个项目的 QA，也不是天天忙。只要将两者的时间协调好、分配好，我觉得还是可以非常出色地完成这份工作的。

1、只要公司在项目研发管理规范方面做的不错，项目级的 CM 其实也很容易做。不用给项目组成员过多的培训，大家都能很规范地 check in 或 check out，做好版本控制就不错了。诸如：CI、Branch、Merge、Label 等等，熟悉配置管理的人，都将做的很好。

2、只要公司在项目研发管理规范方面做的不错，项目级的 QA 更容易做。过程规范执行的相对简单一些，在什么 Process 该出什么 Product，大家心里都有底的。你也不用天天叫嚷着要 Check……Check……，学会引导！人们往往都有一个习惯：在某件事上，当自己忙的不可开交时，外部要求的一些额外需求，往往会产生抵触情绪。所以，切忌不要说“你该给我交付什么成果了……”等等之类的，做好服务工作为优先！

3、CM 和 QA 都是服务与支持类的工作，注意定位。

以上，供参考！。

A（monica）：

呵呵，俺的亲身经历，SCM+SQA，如果你作为一个初级的 QA，一开始并没有什么实际的经验，所作的只是 follow 和 check，其实工作量不大；配置管理的工作可以作为你切入项目的点，能够更加方便的让你了解到项目的实际情况，方便你开展一些 QA 的工作。当然，如果你是已经有很多项目经验的话，这个经验可能不适合。

C（sungubbi）：

如果能够合理地分配其工作，QA 与 SCM 可以由同一人担任。但建议不要在一个项目内

同时兼任 QA 与 SCM，可以在一个项目任 QA 而在另一个项目任 SCM。

★ CMMI 在做项目和做产品之间的实施相同点与不同点

自打跟 CMMI 有了接触后，所在的公司一直是以实施项目为主的，还没有形成产品生产线。今天，忽然有个朋友问起对于产品如何实施 CMMI 或 CMMI 如何提高产品生产的质量和保证进度，一时间没有好的建议。同时，也引出上述问题，希望增长下见识。

这个朋友面临的情况是这样的：

他们主要是做产品的，目前产品在各个省都有使用，各省根据本地业务需要也提出需求，这些需求需要开发部门修改产品去实现，几个省每个月的需求有一百项之多，由于不同的需求要求的开发进度不一样，而且有的不同省份的需求有相互关联性，造成开发过程，测试过程和发布过程会经常出现很多问题，最终经常导致实现需求的进度滞后

Q（tyrone）：

如果这两者有不同的话，SEI 就会去界定两种不同的框架去处理这两个议题，可是，事实并非如此。所以你可以安心地，学一套 CMMI 的 Best Practices，不论是项目或是产品开发都可以适用。所以，个人认为，不论是项目或是做产品，其实施没有什么不同，重点是，你得了解你的客户与产品。

当然，如同你所提出的一样：由于不同的需求要求的开发进度不一样，而且有的不同省份的需求有相互关联性，造成开发过程，测试过程和发布过程会经常出现很多问题。好像真的项目与产品的实施有很大的差异！

不同的客户，在同一个产品基础上，有不同的需求，但是当某一个需求被改动时，另外一个客户的需求可能也会受到影响，这个时候，最好将每个客户视同一个项目，因为每一个客户要的产品都不太相同。但是这并不意味着，所有的客户的产品都得由零开始，反而是要使用衍生产品或是从配置管理去思考管理的方法。

既然是一个以同一个核心产品(core product)针对不同的客户需求做改动的，那么，可以就不同的客户需求，去实现差异的部分，这个就有点像是针对特定客户的需要去做需求变更(change requests)或是产品提升(enhancement)，你可以某个产品架构为基底，不同客户的需求另外实现，针对不同客户的需求各有一套配置管理的机制。当要针对某一个客户发行的时候，就是依照所有适当版本的构件，去 build 一个产品以交给客户。我们举例如下：

某产品是由 A(v1),B(v1),C(v1),D(v1),E(v1),F(v1),G(v1)等六个构件所组成，这是一个核

心产品。

针对甲客户的需要，把 B(V1)修改为 B(V1.1)，F(V1)要改为 F(V1.1)

对于乙客户的需要，把 B(V1)修改为 B(V1.1)，F(V1.1)改为 F(V1.2)、G(V1)改为 G(V1.1)交付。

因此交付给每个客户的产品配置为：

甲客户：A(v1),B(v1.1),C(v1),D(v1),E(v1),F(v1.1),G(v1)

乙客户：A(v1),B(v1.1),C(v1),D(v1),E(v1),F(v1.2),G(v1.1)

对于两个客户的配置管理系统中，可以只管到：

甲客户：B(v1.1),F(v1.1)

乙客户：B(v1.1),F(v1.2),G(v1.1)

当然每一个配置库当中还要包括一份 VDD(版本说明文件)，以描述各客户产品的配置为何。

另外，针对不同的客户，可以排定不同的发行时间，千万不要想搭便车，“愈急愈慢”是金科玉律，宁可为不同的客户律定不同的交付期程，以稳健的步伐满足每一个客户的需求方是上策。

5 Test（测试人员）专区

★ 测试用例的编制时间

Q（cecilla）：

我想请大家结合自己的经验说说测试用例编制需注意的一些事项：比如测试用例的编制时间应该是什么时候？是在需求规格说明书完成之后？还是在技术说明书完成之后？还是分阶段编制？

依据需求规格说明书编制功能测试用例，这样的话是不是意味着需求规格说明书需要写的非常详细，将所有的功能及操作都描述清楚？。

A（steplv）：

按照我自己的理解，测试应该分几个阶段：

1、UT-单元测试（静态测试）

2、ITA-集成测试 A（软件各功能模板之间的集成测试）

3、ITB-集成测试 B（软件、硬件、网络设备等之间的集成测试）

4、ST-系统测试

5、用户验收测试

此外，还有 Alpha、Beta 测试

如果按照上面的四种归类的，不同的阶段，针对的用例不同，按照 V 型模型的话：

1、UT 对应编码阶段

2、ITA 对应详细设计阶段

3、ITB 对应概要设计阶段

4、ST 和用户验收测试对应需求阶段

所以，每种类型对应不同的测试用例。

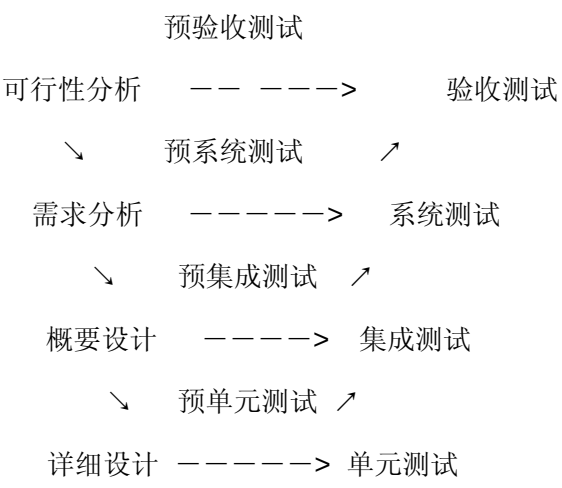
A（lily_014）：

针对你的问题说一下我的看法。首先，测试用例什么时间写，根据你开发选择的生命周期来定，是按不同的模型分不同的阶段，但一般情况下是在需求出来后（并不是说等到需求完全确定下来，至少是在第一次评审后）这时要开始入手编写测试用例，没有不变的需求，因此测试用例也是随着需求的变化不断地去修改和完善。

至于你问的是不是需求要写得非常细，当然，如果需求写得很好很完善的话，那么后期测试用例也无需花大量的工作量在更改测试用例上面。

A（cecilla）：

最典型的 V 模型版本：



↘ ↗

编码

单元测试所检测代码的开发是否符合详细设计的要求。集成测试所检测此前测试过的各组成部分是否能完好地结合到一起。系统测试所检测已集成在一起的产品是否符合系统规格说明书的要求。而验收测试则检测产品是否符合最终用户的需求。

A（step365he）：

1. 单元测试的测试代码一般可以先于产品代码，也就是常说的测试驱动开发。这部分的测试用例（测试代码）由程序员自己完成（不建议其它程序员负责）。

2. 集成测试是测试模块之间的接口，以及模块集成过程中各个模块的运作是否符合设计要求；这里的模块以及模块集成后一般是构成了个子系统（如果产品规模不大）。这部分测试可以由测试人员和程序员一起完成。

3.系统测试是测试人员根据需求说明书，对产品进行测试。所涉及的不仅是产品本身，而且有用户说明书等。

4.如果需求规格说明书需要写的非常详细，将所有的功能及操作都描述清楚，那就不是需求规格说明书了。每步操作（正确流程，旁支流程等，操作步骤，期望结果等）描述是测试用例的工作。

5.我觉得快速划分测试用例的优先级，以及设计合理可行（成本低）的测试用例是两个关键。

C（step365he）：

测试用例可以分为基本事件、备选事件和异常事件三类。

设计基本事件的用例，应该参照需求以及设计规格说明书。基本事件一般使用路径方法设计。如果模块多，模块之间关联多，会使得路径数量迅速增加，导致测试不可行。这时我们按照功能设计测试用例。基本事件的测试用例应包含所有需要实现的需求功能，覆盖率达 100%。

设计备选事件和异常事件的用例，则要复杂和困难得多。我们测试目的是要求验证全部非基本事件，并同时尽量发现其中的软件缺陷。采用的基本方法有等价类划分法、边界

值分析法、错误推测法、因果图法、逻辑覆盖法等设计测试用例。

在上述基础上，掌握良好的划分测试用例优先级也是很重要的技能。否则当产品有新版本发布或者 bug 修复后，需要在限定的时间内快速测试以确保系统基本功能正常，测试人员却没有时间进行有效快速测试的情况。

6 CM（配置人员）专区

★ 关于不同配置项的管理方式的讨论

Q（Xueer0124）：

一直以来，几个操作层面的问题，都非常的困扰我，希望能和大家讨论一下：

1、代码，作为配置项，哪些管理是必须的？如何通过工具和日常的开发工作结合起来？代码类配置项的配置审核工作如何进行更有效？

2、文档类的配置项，比如需求文档，如何管理会更好？

3、配置管理环境上，有些公司会建“开发库”“受控库”“产品库”等从物理上隔离不同的库，管理过程中，需要蚂蚁搬家一样将配置项搬来搬去，但在使用这些配置项时，在“受控库”中保存的宝贝却不能得到很好的使用。尤其对于迭代开发的项目。如果一个文件作为一个配置项，那么即使已经进入基线，其下一迭代的相应信息还会补充到这个文档中；如果将这个文件的某些部分作为一个配置项，操作起来，比较难以控制。一直以来的理解，配置项的版本可以通过工具自动保留，单个配置项的重要版本（或基线）可以通过标签来实现。那么，为什么大家还会那样规划呢，是我理解错了吗？。

A（amandating）：

对于蚂蚁搬家的问题，我个人认为这也是为了管理方便，不至于需要某份受控文档时，还需要在海量文档中寻找。同时方便了后续需要使用这个工作产品的人员在统一的基础之上进行开发，不至于造成因为基线的问题而影响开发。

A（xixiaojing666）：

to xueer0124

不是特别理解你的有些问题，我谈谈我们的配置项是怎么管理的，希望对你能有帮助。我们设置了开发库、受控库、产品库，使用的配置管理工具是 CVS。

开发库专供开发人员使用，其中的信息可以做频繁的修改，对其控制相当宽松； 由配置管理员定义最上层目录，下层目录可由开发人员自行控制。

受控库里面的工作产品一般是经过评审的管理文件（一般是项目计划、里程碑报告、度量数据等），或是进入基线的工作产品（需求文档、设计文档等）。由于软件配置管理控制和管理的最重要的部分几乎都在受控库中，所以对受控库中的相关配置项进行修改时，需要履行一定的变更程序（一般指软件工程的工作产品，需求文档、设计文档等）。

产品库就是项目结项之后，形成的最终库。

我们开发库和受控库使用了一个库，通过在开发库上打标签的方法建立了受控库，我觉得开发库和受控库是否采用一个物理库差别不大，各有各的好处。就看哪个更适合自己了。如果不想搬来搬去，就用一个库上打标签的方法。

产品库是使用的单独的一个物理库。

我们很少有迭代的模式，我觉得对于迭代开发的配置项比如 SRS，应该是这样控制的，刚开始作为 V1.0 建立了一个基线版本，当下一迭代的相应信息补充上去的时候，就作为 V1.1 再建立一个基线版本，然后在变更履历上写明版本之间的差别就可以了。

如何更好的控制基线后的配置项，请参考：
http://www.step365.com/bbs/thread-618-1-1.html

Q（xueer0124）：

多谢各位，对于蚂蚁搬家的困惑，可能是我有点钻牛角尖了。使用时，充分分析优劣，并根据对组织的情况确定好了。

对于文档的管理，我想可能也是比较容易的。主要是一个项目的文档类配置项，还是数得过来的，如果本身就只管理有限的几个配置项，那么怎么管，都不至于乱。重点是要控制好基线版本。

我最头疼的是代码类配置项：代码之间，除了用配置管理工具来进行版本的管理，通过标签（或物理上建立）控制测试版本、发布版本等重点版本以外。还有哪些控制呢？（好象和 xixiaojing666 的问题一样哦)

A（xixiaojing666）：

大家还有什么其它的方法？关注中！不过我觉得，不管用什么方法，只要能管理起来

就好了。

A（cecilia）：

谈谈我的一些看法：

我们现在主要使用 VSS 来管理配置项，主要分为工作库、基线库和受控库。工作库主要是项目组成员对自己工作成果的一个保护库，可以对其进行 checkin 和 checkout，进行文件版本的控制，文件也不会丢失，可以对历史版本进行追溯，配置管理员会给相关人员赋予不同的权限操作，在工作库的开发类还建立了一个公共区，开发人员可以将需共享的代码放到该区内，实现代码的共享及检测。

基线库又详细分了需求基线、设计基线、编码基线、测试基线等。将通过评审的工作产品纳入相应的基线，以后的每一次更改都将进行评审，并将更改的内容（与上一版本比较）记录到文档的首页，实现版本的有效控制，不致混乱。

受控库主要是和工作库对应的，工作库里的文件项目组成员可以修改自己的文件，保护工作成果，对于受控库的文件（经过评审）可以查看，但是只有配置管理员有权限更改，其他人只有查阅的权限。通过项目变更申请流程实现文件的变更，并且是可控的。

配置管理还有待研究，毕竟自己没做配置管理员，只是针对公司目前配置管理及 CMMI 相关资料发表一些看法，期待其他人的精彩回答。

A（fishred）：

1、代码，作为配置项，哪些管理是必须的？——测试通过后的版本控制是必须的。

如何通过工具和日常开发工作结合起来？——主要通过打标签来实现。

代码类配置项的配置审核工作如何进行更有效？——写脚本来实现审计。

2、文档类的配置项，比如需求文档，如何管理会更好？

——走变更流程，一般对于基线变更会有所控制。

3、配置管理环境上，有些公司会建“开发库”“受控库”“产品库"。

——我们受控库是用来放基线的，产品库是用来放上线的代码，工程文档放受控库中。

Q（xixiaojing666）：

to fishred：代码配置项的配置审核你们一般都审核哪些内容？能举个例子详细说说你们怎么用脚本来实现审计吗？

A（fishred）：

我们是 CM 组提出需求，由 PM 安排开发人员写脚本，审计内容大致是：标签是否与代码版本一致、版本是否为该次上线的版本、版本是否有冲突等。

Q（lily_014）：

看了大家的激烈探讨，我觉得 xixiaojing666 和 cecilia 他们俩的配置管理做得不错了。也了解到对于一些共用的产品的处理方式。

不过我有两个疑问：

1、通常开发人员任务分配可能是交叉式的，那么与他有关的每个模块的核心代码都能看到吗？

2、如果我共用的东西也是我这个企业核心的东西，请问大家是怎么来控制的呢？

A（xixiaojing666）：

to lily_014

在一个项目组内，我们公司对代码的管理权限不是很严格，理论上如果两个人的任务是交叉的，那么我们的做法就是两个人可以读并取对方的代码。但是没有修改对方的代码的权限。（但是实际操作 CM 比较麻烦）目前我们的实际做法是，一个项目内的所有人员都有读写其他人代码的权限。但是非一个项目的人员，就没有任何权限了，如果需要查看，得经项目经理的批准，如果看跨部门的代码，就需要两个部门经理的批准。

如果我需要的是企业的核心代码，那么一般是需要两个公司的领导的批准，然后由 CM 开放这部分代码给相关人员。

解释一下我们公司目前的组织结构是，集团公司-各个子公司-部门-项目组。

A（xixiaojing666）：

to fishred

看来你们配置审核做的还是不错的，能使用脚本进行并且开发人员还配合开发脚本，我们都是手工的了。

A（xueer0124）：

我也总结一下我们当前的做法以及目前存在的问题：

物理上，也曾经分了开发库和基线库。其实和大家的管理方式是一样的。只是，对于基线的认识和使用上，感觉没有达到预想的效果。大家很少查阅基线库里的文件，文档还是喜欢用 mail 来传递。主要原因是，因为项目周期短，而且对于工期的要求严格，所以如

果有变更，一般情况下，变更是会被审批通过的，为了节约变更流程执行的这段时间，往往是流程一边走，工作一边做，后续工作基于最新版本，而不是基线版本；如果没有变更，那基线版本就是最新版本。真正的基线最多只能供项目组、基线库只供 QA 分析问题时使用。

对于代码：在开发库进行开发。开发过程中，仅使用配置管理工具进行版本控制。在提交测试时，使用标签来作版本管理。测试人员通过标签获取版本进行测试。开发人员继续在开发库 fix bug、以及新的编码工作，直到下次提交测试。测试通过时，做发布基线（也是在开发库通过标签完成）。存在的问题是，对于代码的管理，只有到基线，配置人员才会进行审核，而且审核往往也只是浮于表面，顶多查一下标签是否正确、版本是否通过测试等等。项目组认为：代码之间的冲突问题一般是通过编译识别的，daily build 的工作，当前公司是作为开发工作的一部分，而不是配置管理工作。因此，配置审核对于代码的审核做不深，无法真正体现配置审核的重要性。请大家帮忙分析一下，看看我们存在的问题如何解决。

A（xueer0124）：

to lily_014

对于 lily 的两个疑问，感觉更多的是与组织结构相关的，高层对于相关资源的共享层次的一个策略问题。如何控制权限，应该由对于这些问题能够作决定的人员判断、规划。同时，如果操作上与管理策略上有冲突，那就权衡两者的利弊，再决定如何控制。

C（cecilia）：

从大家的讨论中可得出，大部分公司的配置管理基本上都是从物理上来区分，对于代码只是使用标签的方式来加以区分，如果标签中的描述不够详细，后期管理也不大方便。像 fishred 她们在这一块就挺不错的，PM 会指派开发人员编写脚本来控制代码的管理。配置这一块做的好的话，对于公司来讲是一种财富，将不会随着人员的流失而有所变动，后续招聘进来的人员也能有所参考，依据。

综上所述，配置项（Configuration Item，CI）主要有两大类：① 属于产品组成部分的工作成果，例如需求文档、设计文档、源代码、测试用例等；② 项目管理和机构支撑过程域产生的文档，如项目计划、里程碑报告、风险跟踪表、评审报告等。

由 CM 在配置管理工具上为项目创建配置库（工作库、基线库和受控库），并给每个项

目成员分配权限，各项目成员根据自己的权限操作配置库。公司应制定相应的配置管理规范，对于草稿、正式发布和正在修改的配置项的各类标识应详细定义，每个库的工作成果所对应的使用人员及权限也需规范，这样 CM 才能更好地管理项目各阶段的工作成果及管理类的文件。同时应有配置项版本控制流程，每一次版本的变更都应有修改履历，都能进行历史追溯，实现版本的有效控制。建议对全员进行配置管理培训，要求所有人员对其工作成果进行配置管理，并严格执行配置项变更控制。

第三部分：原创推荐

1 生如夏花－EPG 和 QA 人员的价值定位思考

作者：何丹博士

QAI China 高级咨询顾问

美国SEI 认证的CMMI 高成熟度主任评估师

(SCAMPI High Maturity Lead Appraiser for CMMI)

电子邮件：danh@qaichina.com

当我死时，您的洪名，如最后一瓣花，自我的唇上飘落。（那名号一定是阿弥陀佛，因为我早已在十方三世一切诸佛菩萨面前发过大愿，用永不退转的菩提心去利益一切众生）。

一、引子

清晨，在深圳弘法寺旁放生完毕，漫步在寺外幽境的山谷，周围美丽的山花似乎也在用清芬为那些回归自由的生命欢欣着、祝福着，心情也随着清晨的阳光如花般绽放着.....

“当你生命花开的时候，会像艺术家一样吸引众生。花开是生命力展现的最高境界，也是生命力最旺盛的时候。看到花开，就要自省我们的生命力有没有开花?” 这是台湾一位高僧的开示。一朵美丽的花朵不会说：“我正在帮忙、给予和服务？”而它却是是这么做的，正因为无心去做，因此能够覆盖大地。那么当生命花开，用倾情绽放的智慧，去为他人服务，那生命将何等地辽阔……

很多企业的EPG（Engineering Process Group，即过程改进团队）成员和QA（质量保证团队）成员经常问我：“我们工作很努力，却经常得不到大家的认可，我们的价值和职业

发展方向在哪里？CMMI 评估通过后是否就意味着我们要失业？公司是否会卸磨杀驴？”我曾尝试着给出一些答案，但一直感觉不够满意，凝望着身旁静静开放的山花，似乎如老僧顿悟般找到一个比较圆满的答案……

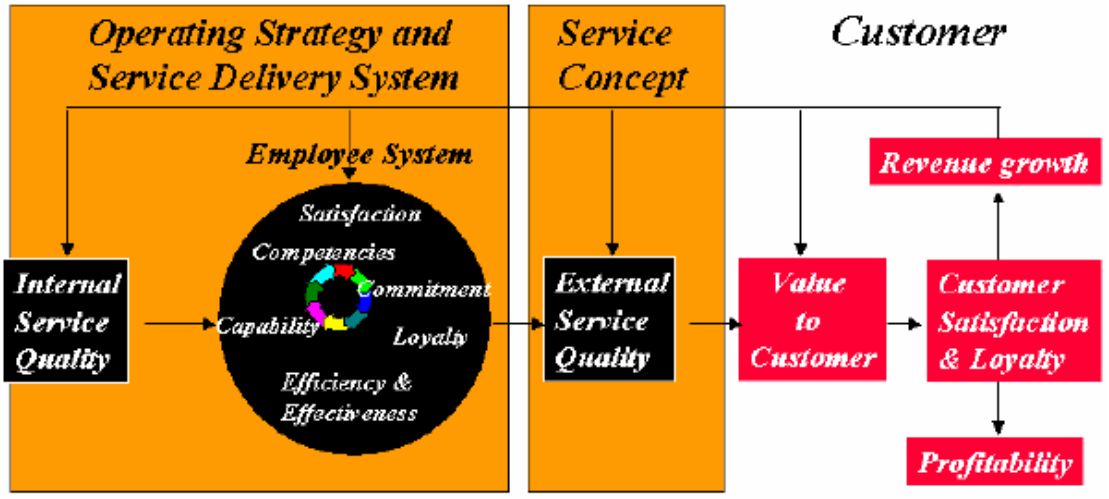
二、EPG 的定位和发展-从流程管理到服务价值

2.1.关注价值–提供高价值的服务

大部分公司的EPG 团队的工作主要内容是建立和管理工作流程体系：基于CMMI（或ITIL、TL9000……）等一些质量模型来识别公司的弱点，并建立和改进相应的流程体系、知识库等。通常在一定时期内会给公司带来一些比较大的价值，而改进工作进展到一定程度，就会出现一些饱和现象，很难再找到新的改进突破口，因此得不到一些人的认可。为此EPG团队需要重塑自己，实现以下几方面的转变：

2.1.1 从管理到服务的转变

美国著名的管理大师Tom Peters 在其作品中屡次提到[i]：我们大部分人都在服务业工作，有96%的人经常在各种服务业中穿梭往来。其中中79%从事正式的服务业（交通、娱乐、零售等），而从事所谓制造业的其他19%的人中，有九成做的是服务工作（设计、财务、营销、采购等）。他更是一针见血地指出：未来的组织里的各个部门和团队都应该按照专业服务型公司运作，不再需要高高在上的管理部门和团队。不带来价值的团队和部门必然会走向消亡，只有不断地给客户、给组织创造价值的团队和部门，才能够持续存在。服务就是去协助构建公司整体的价值系统，就是实现和价值链的零距离。只有和价值链零距离才能生存。Heskett, Sasser, and Schlesinger 等国外学者更是明确设计出公司的服务利润链（Service Profit Chain）： 一个公司如果期望不断扩大利润和产值，就需要不断提高客户的满意度和忠诚度，而这就要求我们关注客户价值取向的研究，并通过持续提高服务的水平和增加服务价值不断给提供客户增值的服务，为此公司内部只应该存在高价值服务的内部部门和团队，给内部客户和员工提供高质量的增加值服务。



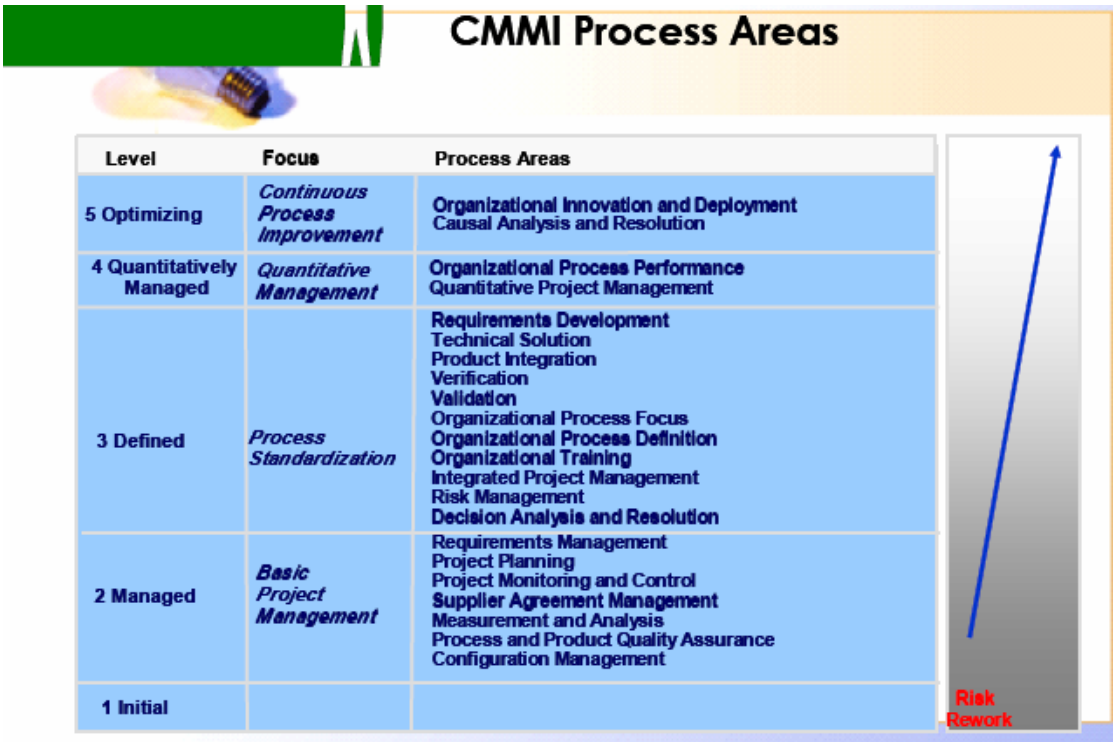
曾经走过的一家企业，其市场部的辞职率居高不下，究其原因，辞职的员工抱怨说：我面对客户的时候是孙子，回来面对我们内部各个部门的时候是龟孙子，因为要努力求着各个部门去合作，去给客户提供服务。显然，在这样的运作模式下服务价值链已经断裂。

那么，作为EPG 团队，应该作为服务文化的楷模，从意识和行动上谦逊地给他人提供高价值服务。扪心自问： 我们定的模板和流程是给公司给客户带来价值的吗？是否仅仅是满足某种质量模型或标准的要求。很多企业的人都问我：这样定义的模板、表格、检查单、过程资产库是否满足CMMI 模型的要求，但极少有人问我，这样做是否更能给客户、给公司、给员工带来最大的价值。生命太短，如白驹过隙，无法容忍自己的庸庸碌碌，没有价值的服务是否不要拿出手。只要对他人有价值的服务，就应该以喜悦的心情全力以赴。并与他人

分享服务带来的欢欣与喜悦。生命的意义也许就在于让自己成为燃烧的薪，让薪尽火传。（指穷于为薪，火传也，不知其尽也。-庄子.养生主)

2.1.2 寻求超越-从运作卓越(OperationalExellence)到业务卓越(BusinessExcellence)的转变

CMMI 是经过美国SEI 收集和整理的业界优秀公司的过程改进最佳实践的大全（共400多条最佳实践，分类成22 个过程域，参见下图），借鉴别人的经验，可以大幅度缩短我们走向成熟的时间，因为别人吃一堑，我们长一智是经济的学习方式。但学习和效法后必须超越，否则是只是东施效颦，只有其形，而没有其神。



CMMI 模型更多关注在运作卓越(Operational Excellence)，也就是说更多地在强调开发体系的提升，即开发效率和生产率提升。但这样一定会使业务卓越吗？会使公司腾飞吗？中国是世界上的生产基地，很多工厂都把生产率等指标最为最重要的改进目标，却有很多产品积压和滞销。一个企业要优秀，仅仅靠CMMI等关注运作卓越的模型是不够的，也许需要寻求公司内部和业界的最佳的经验和教训，从更系统的角度去构建自己的业务卓越模型(Business Excellence Model)。QAI 公司的咨询顾问走过那么多优秀的企业，搜集了很多优秀的实践，然后整理成下图所示的一个业务卓越模型架构： 一个走向伟大的企业如同一个即将发射的火箭，包括几个组成部分：

(1) Leadership (领导团队和领导力)

(2) Strategy Planning and Execution (战略计划和执行)

(3) People Development (人力资源开发)

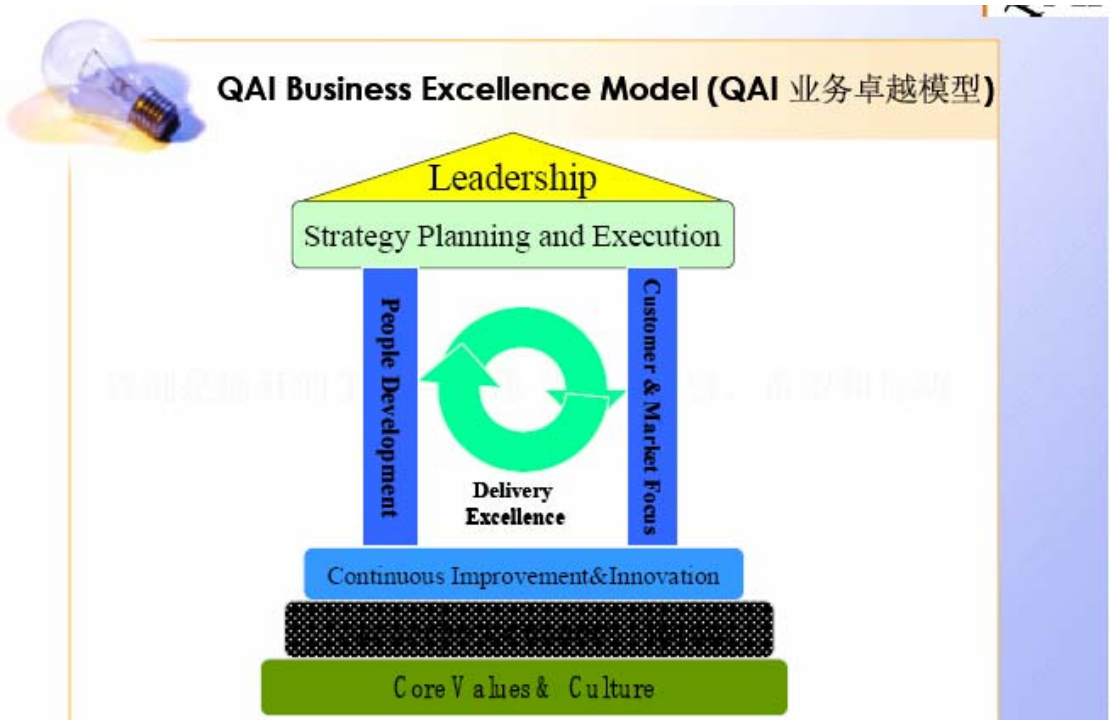
(4) Customer and Market Focus (客户导向和市场导向)

(5) Delivery Excellence (卓越的开发、实施和维护)

(6) Continuous Improvement and Innovation (持续改进和创新)

(7) Continuous Learning and Sharing (持续学习和分享)

(8) Core Value and Culture (核心价值观和文化建设)



EPG 团队可以根据各自公司的特点去构建自己的业务卓越模型（BusinessExcellence Model），然后针对各个分类来寻找公司内部和外部的最佳实践（即最好的经验教训），并迅速在公司内分享。这样，EPG 工作内容就已经大大拓展，从仅仅关注开发和维护等流程体系的改进和创新扩展到关注公司业务卓越体系的改进和创新。因此很多优秀的企业把 EPG 的使命重新定位，甚至更名成BEG（Business Excellence Group，即业务卓越团队）。

对于一个公司，业务卓越模型中最重要的两点也许应该是Leadership和Strategy：

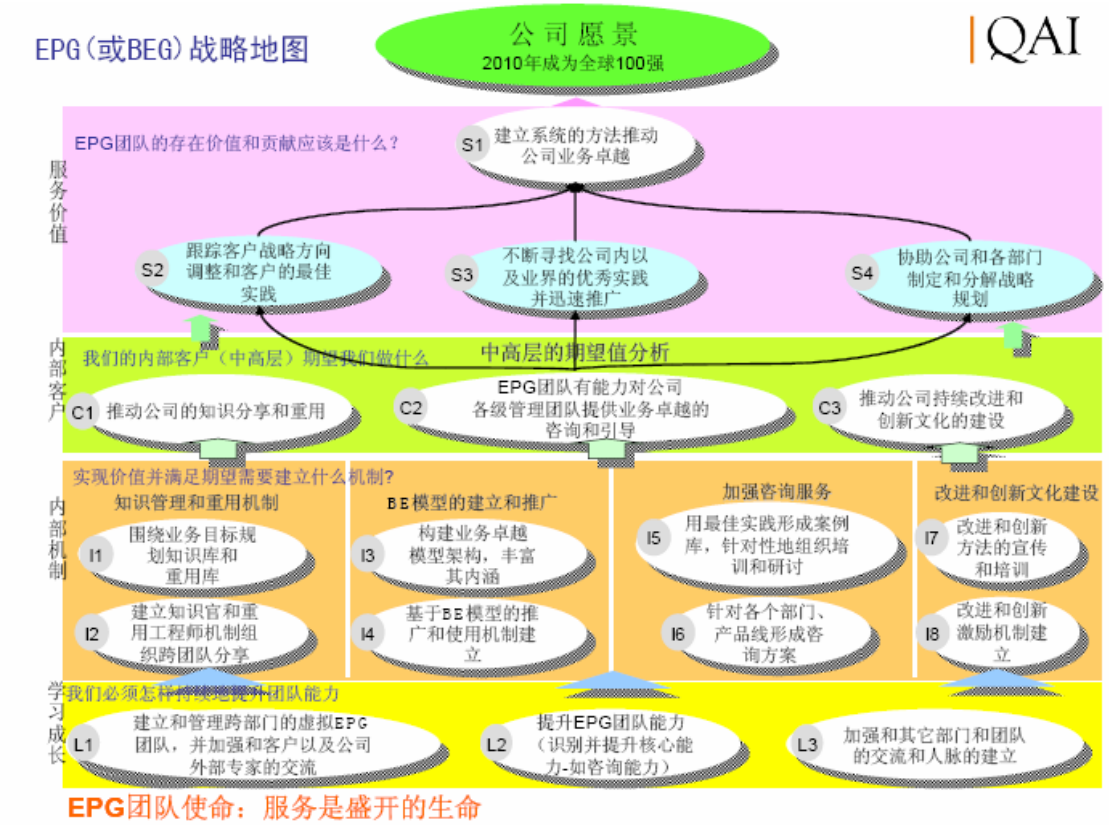
(1) Leadership – 各级领导团队的打造

高建华先生在《笑着离开惠普》[ii] 中提到：美国德州仪器公司1998 年曾经委托麦肯锡顾问公司做过一个“Top Talents（顶级人才）”的研究。在这项研究中，德州仪器公司发现，这些顶级人才几乎一致地认为他们工作中最重要的一个因素是“老板的质量”。相对于较好的公司薪资福利、工作环境和条件以及企业策略而言，“老板的质量”是最不可能被这些顶级人才放弃的。盖洛普（Gallup）调查公司也曾在20 世纪90 年代作过一个超过一万名经理人员的调查。调查发现：“拥有优秀的第一线经理”的企业，在企业评比的四项主要指标中脱颖而出。这四个项目包括了：生产力、利润率、留住人才以及客户满意度。我走过这么多企业，有相同的感受，一个优秀的领导团队能够把整个公司治理得生机勃勃、欣欣向荣。但在有些公司内部，各个领导的观点、价值观和领导方式都完全不同，各有优点和不足，EPG 如果能够找出公司内部和外部的领导团队的最佳实践，和各级领导团队取得共识后，尽快分享这

些好的理念和做法，可能会很快让公司形成有效的合力，形成激光。

(2) Strategy Planning and Execution (战略计划和执行)

公司和部门战略规划是航海的罗盘，特别是在这个市场和客户瞬息万变的时代，更需要及时调整规划。而很多企业中各个部门的规划方向完全不一致，甚至有些重要部门没有规划。如果EPG（或BEG）在这方面寻找最佳的实践，形成样例库，并迅速推广，对公司的腾飞会起到巨大的作用。接下来，EPG（或BEG）可以根据公司的业务愿景和规划去建立团队的使命和战略规划，尽自己的力量服务于公司走向辉煌。下面是一个EPG 的战略地图样例。战略地图其实就是战略因果关系图，因为我们无法制造一个只有果没有因的系统[iii]，所以需要清晰识别出最重要的结果和导致这些结果最重要的环节。



(1)EPG走向成功的重要结果分析-即战略地图中前面两层的分析：

第一层：服务价值– 服务于公司的愿景，通过图中所示的三方面更系统地协助公司走向成功，协助公司一起走向成功应该是EPG 的使命。

第二层：内部客户– 针对中高层的期望值分析，服务于各级主管走向成功。

(2) EPG走向成功的重要因素分析-即战略地图中后面两层的分析：

第三层：内部机制– 为实现上述结果需要建立的内部机制。

第四层：学习成长– 为了使得内部机制发挥价值，需要调整EPG 架构以及加强EPG 的核心能力，使得EPG 团队组成以及核心能力使其能更好地扮演BEG（业务卓越团队）的角色。然后EPG 可以根据战略地图中分析出的因果环节，根据优先级排序后，形成详细的行动计划和方案。（注：针对有些重要结果是无法采取任何措施的，而是针对相对应的因采取措施）

2.1.3 扬长避短-选择企业和各部门的强项进行进一步改进和创新

EPG 团队协助公司改进和创新的过程中，更多地是采用木桶原理，寻找公司和各个团队的短板进行改进和创新。但也许公司达到一定成熟度，应该把关注点放到识别强项，而不仅仅是弱项，然后去进一步改进和创新强项，加强优势，也许扬长避短是走向卓越最短的距离。正如咨询界泰斗彼得.布洛克所说的： “识别优势，而不是劣势；选择希望，而不是失望”[iv] 。EPG 在组织改进和创新过程中是把重点放在失去的东西上还是放在现有的东西上？如果只关注缺陷和弱点，纵然将一生的精力都用于改善缺陷而取得成果，所有的努力只能产生很少的结果。关注缺点或需要的结果，只能是以牺牲客户或员工为代价。我们的目的是为了使大家都想起曾经失去的东西还是为了使大家思考什么是未来可能的东西？如同一个汽车司机，应该把80%的精力关注前方，20%的精力关注后视镜中的出现的问题。

2.2.关注人和文化–改进和创新文化的建设

EPG 仅仅关注在流程、模型的改进和创新是远远不够的。譬如我们能够从日历和天气预报中知道春天来了，却没有的心灵和慧眼去体会“苔痕上皆绿，草色入帘青”，那么日子或许是满的，生命却是空的。没有灵魂的流程和模型是不完整的，是没有生命力的。像无源之水，弄得再漂亮，也不过像个游泳池。无本之木是长不高大得，弄得再好看，也不过像个大盆景；EPG 缺少了对文化和理念的传递就失去了持续走向优秀和伟大的灵魂。因此EPG 需要更多地关注一下人文、理念的传递。

2.2.1 从流程和质量的医生到价值链的医生，再到改进和创新文化的牧师

海尔的CEO 张瑞敏先生指出：“质量管理不是一种权能管理方法，而是价值观。”[v] 因此当海尔还在很起步阶段，发现76 台有些许质量问题的冰箱，就要始作俑者当着全场员工的面把它们砸掉。我们看到砸掉的是冰箱，而张瑞敏却说：“我们把76 台冰箱砸掉，是要改变的理念，而不是冰箱。起到了震撼的效果。”他还提出：“改革开放这些年也是观念的转变。一件事情，随着一个观念的改变，这件事情的本身含义就改变了。这件事情本身并没有改变，但对自身的认识改变了。”“ 流程再造就是再造人。”其实所有的质量和流程管理都不应该是管理工具，而应该是价值观和理念传承的载体，任何优秀的公司都是有理念的公司，这样才

能赋予贫乏的流程和模型丰盈有感的生命。因此EPG 应该把更多的关注点从流程的改进和创新转移到改进和创新理念的传递。不仅仅是流程和质量的医生、而且是价值链的医生，进而成为持续改进和创新的文化牧师。积极主动去搜集企业内部和外部的一些能触发人思考的案例，因为理念和文化需要故事才容易传承。

2.2.2 把改进和创新植入到公司的每一块肌体

只有形成草根文化，文化才真正的生根发芽。改善和创新文化的形成也需要渗透到每个部门和每个员工，需要全员的参与。那么EPG 就需要和高层考虑设计在企业和各部门内部建立一个支持持续改进、创新的文化和架构。而真正有多少企业、多少部门、多少个人把持续改进和创新变成自己的日常工作呢？管理学开山鼻祖德鲁克明确提出[vi]：传统组织的设计上就是为了维持原状。因此，所有现存的组织，不论是企业、大学、医院或教会，在接受变革和具备变革能力方面都力不从心，而且变革的阻力也很大。忽视变革，假装明天会和昨天一样，是没有用的。特别是以前曾经享受过成功的公司最容易采取保守的态度。以为明天就像昨天一样的错觉。唯一可以成功的策略是，尝试着去塑造未来。有生命力的肌体一定是在新陈代谢，肌体的80%相对稳定，20%不断推陈出新。那么生命力旺盛的公司中也应该是80％精力投入相对稳定的业务、管理、技术等，20％的精力不断改进和创新技术、管理、业务等。有专注才有成功，EPG应该是公司20%的一部分，集中引领主要的改进和创新。同时建立分布式系统，让每个部门和个人都承担其一些改进和创新的任务。这样一个集中式-分布式相结合的改进和创新肌体才能在公司生根发芽。

在通用电气（GE），专职的黑带大师（MBB）在扮演EPG 角色，而每个经理和员工都要成为黑带(BB)或绿带(GB)，兼职参与公司的改进和创新。同样，在新加坡航空公司[vii]，专职的创新部扮演着EPG 的角色，主要关注重大的创新和改进，研究未来3－5 年需要的创新。但同时也鼓励员工参与创新，所有员工都有机会被选中参加一项“未来工程”，把不同部门的人员集中到一起，来进行自由讨论创意，发表意见。

另外EPG 也要和高层共同考虑建立相应的报酬、认可和奖励激励来鼓励改进和创新，真正把持续改进和创新注入公司的每一块肌体。

2.3.关注自我的重塑-建立开放的学习型团队

进入很多寺院，都能看到千手观音菩萨，为了帮助更多的众生，需要那么多手，她的每只手掌里都有一只眼睛。手代表着行动，而眼睛代表着智慧。如果没有智慧，我们的行动就有可能会给他人带来痛苦。因此我们需要让自己智慧充盈，才能去给很多人带来价值和幸福。

2.3.1 感觉全开–开放地学习

EPG 和QA 成员不管以前有多少辉煌的历史，都要谦逊的不断重塑自己，让知识更新成为每个人的首要工作，因为我们都是快速贬值的财产。这个时代人们最需要的能力不是具体某个能力，而是持续学习的能力。如果要学会这个能力，最好在公司内部或外部寻找一位真正的良师益友，也许你受益的不是他具体传授的知识，而是感染了他那份对改进和创新热爱的激情[viii]。

有了这份激情，再去开放地向客户、向同事和公司外部的同仁学习和交流，不要仅仅“摸着石头过河”，那或许要付出巨大的代价。同时养成读书的习惯，阅读经典，常常让我们有灵魂觉醒的惊喜，也许能让身心更加澄澈，从而更有灵感和创意去主持公司的改进和创新。

2.3.2 全然地去服务–天使能够飞翔，是因为他把自己看得很轻

艺术是“我”的缺席[ix]，任何真正有价值的艺术品都是生命的，任何为了彰显“自我”的艺术家都无法做到真正的优秀。琴艺很好的小提琴家如果很在乎自己的名声，他就是志不在小提琴，而是觉得“我”比音乐重要。罗丹在雕刻的时候忘我了，也忘掉了身边等候他的朋友，因此能够创造出伟大的《思想者》”。

米开朗基罗在雕刻大卫的时候一定是忘我的，才能雕刻出如此有生命张力的作品。“采菊东篱下，悠然见南山，山气日夕佳，飞鸟相与还” 这样美丽的诗词也根本没有“我”的存在。乔丹打球的时候一定也忘我了，因此打出了很多叹为观止的球。您在开发最优秀产品的时候也一定是忘我的……

正如西方的一句谚语：天使能够飞翔，是因为把自己看得很轻。那么能否以忘我的心态，全然地去服务，把公司的改进和创新打造成艺术品，而不是千篇一律的模型和商品。

三、QA 的角色转换-从过程监督员到项目咨询师

一些公司设定了QA 的角色，让其负责流程的监督和推行，而且经常引用以下猴子吃香蕉的故事来解释流程的执行手段和方法。

5只猴子和橡胶的实验设计：

笼子里面有5 个猴子。笼头上挂了一串香蕉，实验人员装了自动装置，若是侦测到有猴子要去拿香蕉，马上就会有水喷向笼子，这5 只猴子马上会被淋湿。

实验开始：

1. 每只猴子去拿香蕉都把大家淋湿了，因此猴子达成共识，不去拿香蕉。

2. 后来试验人员把其中一只猴子换掉，换一只新猴子（A 猴子），A 猴子想去拿香蕉，结果被痛打一顿，只好作罢；

3. 后来试验人员又换了一只猴子（B 猴子），B 猴子想拿香蕉时，又被大家痛打一顿，打得最卖力的是A 猴子；

4. 后来把所有的猴子全部换掉，也没有猴子去拿香蕉，至于为什么不能拿，大家不知道，只知道会被痛打一顿。

这样的流程推行方式用于猴子是否是最佳模式仍然有待商讨。让QA 是扮演那个检测装置，在以人为核心高科技企业采用类似的推行方式是否合理？是否会大大扼杀创造力？而创造力恰恰是目前这个时代最不可或缺的无价之宝。

3.1 内因和外因-有效的过程体系执行需要内在的承诺

有些公司在流程体系推行中，公司高层规定QA 一旦发现一些流程没有被完全遵守就要对相关责任人（如项目经理）进行罚款。所以当QA 人员出现在项目组时，项目经理胆战心惊地询问QA 自己又有哪些表格没有填写、哪些模版没有使用……然后赶紧补上，这样的白色恐怖的流程监督方式对公司真的有价值吗？对员工的成长有价值吗？会给客户带来价值吗？

咨询大师彼得.布洛克在其成名作《完美咨询》[4]中阐述到：影响流程的执行力因素包括内因和外因，内因的提升需要激发内在的理解程度和目标愿景，而外因的建立则是采用监督机制。但如果没有有效的沟通和培训，员工不是很理解流程，那么员工会以为你要求人们去做的事情与他们想做的事情之间没有任何联系，那么人们全力以赴的可能性就微乎其微。你可以命令人们做事，并且通常他们会遵从至少在你监督的期间他们会如此，但如果你希望他们全身心地投入其中，内在承诺是很不可少的。否则，可能过一段时间，监控力度减弱，那么流程体系就会被束之高阁，不管以前花费了多大的代价来定义。只有告诉大家流程背后鲜活的思想，流程才能在使用者面前变得生机勃勃，才会更灵活地去裁剪和调整流程以更好地适应项目组和自己的工作，为客户和公司带来价值，才能激励大家带着谦虚的心去学习和执行业界一些优秀的实践，带着青出于蓝的勇气和热忱去寻求进一步超越和创新。一个伟大的企业的核心竞争力一定是开发出优秀的人才和文化，而优秀的流程和产品则是副产品。“就像抚养孩子，你把正确的价值观铭刻在他们的心上，等他们长大后，就会做出自己的决定。有时候，他们也会让你失望，有时候，他们也会犯错，但如果他们已经接受了正确的价值观，他们就有一条得以回归的准则。”[x]

3.2QA 的 4 个角色-从监督到咨询

在一些国际上著名的公司，给QA 人员定义了以下4 个角色：

(1)Teacher

老师

QA 人员由于工作之便，可以同时介入很多项目的工作，因此可以看到很多项目组的优势以及不足，让自己像海绵一样迅速地吸收、反思和总结，并迅速地传递给其它相似的项目组，作为一个知识大使去服务，一定会得到大家的工作认可。“师不必贤于弟子”，QA 人员并不一定要比项目经理强，但由于专注于知识收集和分享服务，因此很容易获得认可。

(2)Doctor

医生

如同中西医的结合，总结和整理以前公司各个项目生命周期中常出现的问题，形成咨询方案(FMEA)和咨询工具包。以预防为主，在一个新项目启动时，针对项目特点识别可能出现的问题，形成项目咨询计划，在项目中各个重要任务启动前，提供一些有价值的咨询引导服务。

并针对项目组的数据和已经出现的一些问题，协助或引导项目组进行根源分析，必要时邀请其它一些优秀的项目经理和QA 同事一起参与会诊，发现病灶，然后建议项目组采取一些有效的措施。

(3)Lawyer

律师

任何一个优秀的律师都不是仅仅会背诵法律条文，一定会旁征博引，去让别人心悦诚服地接受。特别在流程推行的初期，如果能告诉项目组流程背后的思想和故事，才能点亮流程的生命，让流程执行者知其然、知其所以然，从而有能力根据项目特点量体裁衣。

(4)Police

警察

人要高标准地完成一项工作可能不难，但真正的卓越，需要一辈子孜孜以求才能拥有。人是有惰性的，因此还是需要一些必要的监督和提醒机制来辅助我们不退步。在流程的执行中，QA 人员仍然需要扮演警察的角色，去发现一些问题，特别严重的问题，要上报给高层经理。

从上述4 个角色可以看出，前3 个角色其实都是在做项目的咨询和引导工作，只有最后一个角色关注在监督实施。因此QA 人员75%的情况下应该是一个咨询师，设身处地站在

项目的角度，用心去提供高价值服务，协助项目走向成功。正如咨询大师彼得.布洛克所提倡的，最好的咨询活动就是一种爱的行为[4]：真诚地帮助他人……运用我们所知道的、所感觉到的或者曾经承受过的来减轻他人的负担。用心去服务，用热情去点燃别人对持续改进和持续优秀的激情，如邱吉尔所说的：

在你激动的鼓舞别人之前，你自己先要充满激情

在你感动别人流泪之前，你自己先要流泪

要说服别人，你自己先要相信。

要别人去微笑服务之前，你自己必须献身服务

四、总结-服务是盛开的生命

总之，EPG 需要不断重塑自己，努力提高自己的能力和视野，关注于公司价值体系，去引导和规划改进和创新。并作为改进和创新文化的牧师，去传递理念。理念如阳光，似乎什么直接价值也没有，但却能给花开花谢、万物的滋长带来一些微妙的不同。QA 也应该从项目组流程的监督员更多地变成咨询顾问，用自己整理的咨询包和样例库去给项目组传递知识、经验、分享希望，提供高价值的咨询服务。其实不仅仅是EPG 和QA，也许我们每个人都是带着服务的使命来到这个世界的，正如高僧一行禅师xi所说的那样：世界是一个有机的整体，我们的每一瞥、每一个微笑、每一句话、写的每一个字，都能到达远近他方，服务并影响着其它的生命。

泰戈尔在诗中写到：“生如夏花之绚烂，死如秋叶之静美。”

纪伯伦说：“灵魂像一朵千瓣的莲花，自在开放着。”

亚里士多德说：“生命的意义在于尽情绽放”

佛教《华严经》的华严世界更是花的世界，因为只要心灵变成了一朵洁白的千瓣莲花，我们用来看东西的眼睛变成了花，用来听东西的耳朵变成了花，用来讲话的嘴唇变成了花，用来服务的手也变成了花……

生如夏花， 把生命无限拓展— 把生命的美发挥得淋漓尽致，充分地绽放，永远带着谦虚的灵魂和服务的心，去改进、去创新、去学习、去分享，让每一刻当下和未来的生命相视而笑，莫逆于心。

近六年的咨询生涯中，走过那么多企业，真的想说声谢谢，谢谢我的每一个客户。是你们让我深刻地了解到服务是盛开的生命- 用开放的智慧和心灵去服务他人。是你们教会了我

把生命的知识和实践合二为一，是你们让我一次次体会到生命的意义在于怀着感恩的心尽情绽放。

天空中没有留下鸟的痕迹，但它已飞过；

水面上没有留下船的痕迹，但它已驶过；

生命中没有留下服务的芬芳，但我已经尽情绽放过……

参考资料

[i] 汤姆.彼得斯，《The Pursuit of Wow! - 追求卓越个人版》, 2006 年12 月，中信出版社

[ii] 高建华，《笑着离开惠普》， 2006 年2 月， 商务印书馆

[iii] 宗萨蒋扬钦哲仁波切，《正见：佛陀的证悟》, 2007 年1 月，中国友谊出版公司

[iv] 彼得.布洛克，《完美咨询》，中国劳动社会保障出版社

[v] 张瑞敏，《张瑞敏谈商录》，2005 年1 月，哈尔滨出版社

[vi] 彼得.德鲁克，《21 世纪的管理与挑战》，2006 年1 月，机械工业出版社

[vii] 洛伊佐斯·赫拉克莱厄斯、约亨·维尔茨、尼汀·潘加卡，《展翅高飞》，2006 年3 月，中
国人民大学出版社

[viii] 托马斯·弗里德曼，《世界是平的》，2006 年10 月，湖南科学技术出版社

[ix] 克里希那穆提，《谋生之道》，九州出版社

[x] 霍华德·舒尔茨多利·琼斯·扬，《将心注入》， 2006 月1 月，浙江人民出版社

[xi] 一行禅师，《一行禅师释佛》，2005 年8 月，中国长安出版社

第四部分：思步翻译

Large-Scale Work—Part I: The Organization

Author:WATTS S. HUMPHREY

Translator: PTG

With this column, I start a new series of articles about the issues faced by large-scale development projects. Since large-scale development is an enormous subject, I plan to touch only on those topics that I think are particularly interesting or

especially important. I currently plan to cover two problems. First, large software projects are almost universally troubled, and second, all large-scale systems-development projects of almost every kind now involve large amounts of software. Unfortunately, this implies that almost all kinds of large-scale development projects will be troubled unless we can devise a better way to develop the software parts of these large-scale systems. The increasing need for large-scale system-development projects raises many questions and presents a significant challenge to those of us in the development business.

大规模的工作 —— 第一部分：组织

作者：WATTS S. HUMPHREY

翻译：思步网 PineTree Group

在这个专栏，我开始一个新系列的文章，内容是关于大规模开发项目所面临的问题。由于大规模开发是一个很大的课题，我只计划涉及那些我认为比较感兴趣或特别重要的话题。首先，绝大多数软件项目都几乎遇到了麻烦；其次，几乎所有每种大规模系统开发项目类型都涉及了大量的软件。不幸的是，这就暗示了几乎所有的大规模开发项目都将遇到麻烦，除非我们能给出一个更好的方法用于开发这些大规模系统的软件部分。随着大规模系统开发项目需求的增长，已经出现了很多问题，同时也在开发领域内对我们所有人提出了一个重大的挑战。

Emergent Properties of Systems

系统的关键属性

Perhaps the greatest single problem with large-scale system development concerns what are called the emergent properties of these systems. These are those properties of the entire system that are not embodied in any of the system's parts. Examples are system security, safety, and performance. While individual components of a system can contribute to safety, security, and performance problems, no component by itself can generally be relied on to make a system safe, secure, or high performing.

也许在大规模系统开发中最大的问题在于什么叫做系统的关键属性。这些属性并没有体现在整个系统的任何一个部分上。例如，系统的可靠性，安全性和性能。虽然系统的单个部分能够解决安全性，可靠性和性能问题，但它自身并不能依赖哪个部分就可以使系统安

全，可靠或者说有更完美的性能。

The reason that emergent properties are a problem for large-scale systems is related to the way in which we develop these systems. As projects get larger, we structure the overall job into subsystems, then structure the subsystems into products, and refine the products even further into components and possibly even into modules or parts. The objective of this refinement process is to arrive at a series of "bite-sized projects" that development teams or individual developers can design and develop.

关键属性是大规模系统的一个问题的原因,在于它与我们开发这些系统的方式有关。当项目变得很大时，我们把所有工作分解到子系统，然后从子系统分解到产品，进一步从产品分解到组件,甚至可能到模块或者部件。 这个逐步分解过程的目的是为了产生一系列的“字节级规模项目”,使得开发小组或单独的开发人员能设计和开发。

This refinement process can be effective as long as the interfaces among the system's parts are well defined and the parts are sufficiently independent that they can be independently developed. Unfortunately, the nature of emergent properties is that they depend on the cooperative behavior of many, if not all, of a system's parts. This would not be a problem if the system's overall design could completely and precisely specify the properties required of all of the components. For large-scale systems, however, this is rarely possible.

这个逐步分解过程可以很有效，只要在系统的部件之间的接口是明确定义的，并且部件是足够独立的,可以被单独地开发。不幸地是，关键属性的本质是他们依赖于许多的系统部件的协作，而不是所有的。如果系统的整体设计能够完全地和精确地指定全部组件的属性，这也不会是问题。然而对于大规模系统，这几乎是不可能的。

While people have always handled big jobs by breaking them into numerous smaller jobs, this can cause problems when the jobs' parts have interdependencies. System performance, for example, has always been a problem, but we have generally been able to overpower it. That is, the raw power of our technology has often provided the desired performance levels even when the system structure contains many inefficiencies and delays.

当人们在处理大的工作时通常都是将它分成好几个较小的工作，然而当这些较小的工作之间存在相互依存关系时就会产生问题。举个例子来说，系统性能经常就是个问题，但是我们经常都能够克服它。那就是说，技术本身的力量往往提供了许多理想的性能水平，即使是在系统结构包含了许多低效率和延误的时候也是如此。

As the scale of our systems increases, and the emergent properties become increasingly important, we now face two difficult problems. First, the structural complexity of our large organizations makes the development process less efficient. Since large-scale systems are generally developed by large and complex organizations, and since these large organizations generally distribute large projects across multiple organizational units and locations, these large projects tend also to have complex structures. This added complexity both complicates the work and takes added resources and time.

当系统规模增加，关键属性变得非常重要时，我们就会面临两个困难的问题。第一，我们庞大组织的结构复杂性使开发过程效率降低。由于大规模系统通常是由大且复杂的组织来开发的，这些负责大型项目的组织普遍都是跨越多个组织单元和地域，且大型项目也都具有复杂的结构。这些都增加了工作的复杂性和增大了资源和时间投入。

The second problem is that, as the new set of emergent properties becomes more important, we can no longer rely on technology to overpower the design problem. Security, for example, is not something we can solve with a brute-force design. Security problems often result from subtle combinations of conditions that interact to produce an insecure situation. What is worse, these problems are rarely detectable at the module or part levels.

第二个问题是，随着一系列关键属性越来越重要，我们也不能再依赖技术克服设计上存在的问题。举个例子来说，安全性就不是我们能依靠强大的设计可以解决的问题。安全问题常常由许多细小的条件组合导致，这些条件组合相互影响，可以产生一种不安全的场景。更糟糕的是，这些问题几乎很少能在模块或部件一级检测到。

A War Story

战争故事

Some years ago, while I still worked at IBM, I was made director of programming. My organization had about 4,000 software professionals in 15 laboratories and 6 countries. While this group was responsible for developing all of the software to support IBM's products, about half of the group was working on one big system: OS/360. These people were highly capable and motivated, but the OS/360 schedule had slipped three times during the past year and a half, and no one believed any of our dates. Since IBM was starting to ship the 360 hardware without the software, the lack of a believable schedule was a major marketing problem.

几年前，我还在 IBM 工作的时候，我是个开发主管。我的组织有近 4000 个专业软件人员分布在 15 个实验室和 6 个国家。当这些小组负责开发支持 IBM 的产品的所有软件时，小组中半数的人在做一个大系统：OS/360。这些人能力很强而且有很高的积极性，但是 OS/360 的进度计划在过去的一年半时间内延迟了 3 次，没有人相信我们的进度。IBM 正准备对没有软件的 360 硬件发货，然而缺乏可信的计划是一个主要的市场问题。

Marketing argued that the customers would be willing to run their existing system software temporarily with the emulators provided with the 360 systems, but that they would not order many new systems until they had a believable plan for OS/360 software support. We needed a schedule right away, but it had to be one that we would meet without fail. If we had another schedule slip, no one would believe us again.

市场部门争论的是客户是否愿意暂时在竞争者提供的 360 系统上运行已经存在的系统软件，并且他们不会定购更多新的系统直到他们有一个可信的 OS/360 软件支持计划。我们需要立刻制定一个进度计划，但是这个进度计划必须不会失败。如果我们再一次计划延迟的话，就没人会再相信我们了。

Before this job, I had run several software projects and also managed some large hardware projects. Therefore, I had learned that it is practically impossible to produce a reliable schedule without first producing a detailed plan for the work. Since the several thousand developers in my group had never before made detailed plans, they didn't know how to do it. What was worse, they didn't believe that planning was important. They knew how to code and test, so that is what they did. Without plans to guide them, the

projects were pretty chaotic, and nobody had time to do anything but code and test.

在参与到这次工作之前，我曾经管理过许多软件项目和几个大型的硬件项目。因此，我了解到，没有预先为某项工作做详细的计划是不可能产生一个可靠的进度计划。在我组里的有许多开发者之前都从来没有制定过详细的计划，他们不知道该怎么去做。更糟糕的是，他们不相信计划的重要性。他们认为自己该做的事情就是知道如何编码和测试。没有一个计划来指引他们，项目就变得异常的混乱，并且所有的人除了编码和测试外没有时间做其任何事。

I had to do something to make planning important. So, to get everybody's attention, I established a new operating procedure: Without a detailed plan, no one could announce or ship any software product. I even threatened to cut the budget for any project that did not have a plan. I gave the groups 60 days to produce plans and to review them with me personally. This made planning a crisis. When I got the plans, I reviewed them all and made the developers defend them. We lengthened many of the schedules but never cut a single one. In fact, I even added a 90-day cushion to every committed date.

我必须要做一些事情让他们觉得计划的重要性。因此，为了引起每个人的关注，我制定了一个新的运作规定：没有一个详细的计划，任何人都不能公布或发布任何软件产品。我甚至威胁他们如果不做计划的话就削减项目预算。我给了每个小组 60 天的时间去做计划，并且每个人和我都要对此进行评审。这形成了编制计划的危机感。当我拿到计划时，我评审了所有的计划，并且让开发人员解释他们的计划。我延长了很多进度计划但没有删减掉任何一个。事实上，我甚至在每个承诺的交付日期之后都增加了 90 天的缓冲时间。

This worked. We did not miss a single date for the next two and a half years. Soon, however, we started to eat up my 90-day cushion. We had not appreciated the consequences of a multi-release delivery plan. Every schedule slip for every release had a cumulative effect on every subsequent release. After about two years, these small schedule slips finally added up to my 90-day cushion. However, by then everybody believed our dates, so the subsequent minor schedule adjustment was not a problem. But after that, we no longer committed delivery dates to customers that were more than about a year out. By then, we were meeting all of our delivery dates, and even beating

the hardware.

开始工作了。在之后的两年半的时间里我们任何一天都没有松懈。然而，很快，我们便开始消耗那 90 天的缓冲时间。多版本控制计划的结果并不令我们满意。每次发布导致进度计划的推迟都累积影响接下来的发布。在大约两年的时间里，这些小的进度延迟累积的时间合计达到了 90 天的缓冲时间。不过，那时每个人都还记得我们约定的日期，所以随后的微小的进度调整没有成为问题。但在那之后的一年多的时间内，我们不敢再给客户承诺交付日期。此后，我们都会开会讨论我们的交货日期，甚至对于硬件也一样。

The project-planning procedure worked extremely well. While it imposed a demanding planning discipline on the development groups, they continued producing plans even after I moved on to another job. Unfortunately, with the mechanism we established for plan management, it was easy to add further procedures, so many managers did. Over time, this simple plan-management system became a bureaucratic jungle. The initial planning controls we established had been relatively simple and saved IBM billions of dollars. However, with the added controls, projects could no longer respond quickly to special customer needs, and bureaucracy became a big company problem.

项目计划程序运作的非常好。同时对于开发小组形成了一个严格的计划政策，甚至在我换了另外一份工作后，他们还持续的制定计划。不幸的是，我们这个为计划管理所建立的机制，很容易进一步添加程序，许多管理者也就是这样做的。随着时间的推移，这个简单的计划管理系统变成了一个官僚性的体制。我们最初建立的计划控制机制相对简单并为 IBM 节省了数亿美元。然而，随着控制的增加，项目不再能针对专门的客户需求做出快速反应，而且官僚作风也成为了公司的一个大问题。

The Tropical Rain Forest

热带雨林

The fundamental problem of scale is illustrated by analogy to the ecological energy balance in a tropical rain forest. In essence, as the forest grows, it develops an increasingly complex structure. As the root system, undergrowth, and canopy grow more complex, it takes an increasing percentage of the ecosystem's available energy just to

sustain the jungle's complexity. Finally, when this complexity consumes all of the available energy, growth stops.

规模的根本问题类似于热带雨林的生态平衡。在本质上，发展一个日益复杂的组织就像森林的成长。根系、矮丛林、树冠层等结合体，它捕获生态系统中有用的能量来维持其成长。直到最后消耗完所有可利用的能量，生长也就停止了。

The implication for both projects and organizations is that, as they grow, their structure gets progressively more complex, and this increasingly complex structure makes it harder and harder for the developers to do productive work. Finally, at some point, the organization gets so big and so complex that the development groups can no longer get their work done in an orderly, timely, and productive way. Since this is a drastic condition, it is important to understand the mechanisms that cause it.

这暗示了对于项目和组织而言，就像雨林的生长一样，他们的组织变得日益复杂，并且这个日益复杂的组织使开发者的开发工作变得越来越困难。最终，在某一时刻，组织变得如此庞大、复杂，以至于开发小组无法再以有序、及时和高效的方式完成工作。由于这是一个巨变的环境，所以理解它的起因机制就非常重要。

Organizational Growth

组织的成长

In principle, organizations grow because there is more work to do than the current staff can handle. However, this problem is usually more than just a question of volume. As the scale increases, responsibilities are subdivided and issues that could once be handled informally must be handled by specialized groups. So, in scaling up the organization, we subdivide responsibilities into progressively smaller and less meaningful business elements. Tasks that could once be handled informally by the projects themselves are addressed by specialized staffs. Now, each staff has the sole job of ensuring that each project does this one aspect of its job according to the rules. Furthermore, since each staff's responsibility is far removed from business concerns, normal business-based or marketing-based arguments are rarely effective. The staffs' seemingly arbitrary goals and procedures must either be obeyed or overruled.

原则上，组织的成长是由于有超过当前员工能够处理并需要完成的更多工作而引起的。然而，问题通常是一连串的。随着规模的增长，职责被进一步细分和发布，而此前的发布都是由专门的小组非正式处理的。所以，当处于组织的发展阶段，我们会划分职责到日益增多的很小并且有很少作用的业务部门里。曾经被各个项目组非正式处理的任务被分配给专门的人员。现在，每个人员都确保有独立的工作，保证每个项目组根据这个规则完成整个工作的一部分。此外，由于每个人员的职责远离了商业利益，基于业务或基于市场的普通争辩就变得无效了。人员表面上随意的目标和过程确必须被服从或被驳回。

This growth process generally happens almost accidentally. A problem comes up, such as a missed schedule, and management decides that future similar problems must be prevented. So they establish a special procedure and group to concentrate on that one problem. In my case, this was a cost-estimating and planning function that required a plan from every project. Each new special procedure and group is like scar tissue and each added bit of scar tissue contributes to the inflexibility of the organization and makes it harder for the developers to do their work. Example staffs are pricing, scheduling, configuration management, system testing, quality assurance, security, and many others.

这种成长过程通常偶然地发生。当一个问题出现时，例如进度偏差，管理者必须采取措施来预防未来发生类似的问题。所以他们建立专门的过程和小组关注这个问题。在我的案例里，每个项目都需要一个计划来执行他们的成本估算和计划职责。一个新的专门的过程和小组就像拥有伤疤的组织，每一个新增的小伤疤都带给组织不确定性，使组织越来越困难地去开展他们的工作。例如员工进行的估计、进度安排、配置管理、系统测试、质量保证、安全和其它活动等等。

The Enemy

敌人

Since these specialized staffs are all designed to monitor and control the projects, the projects view them as the enemy. So, if they can get away with it, the projects simply ignore the staffs. To guard against this, management establishes a review procedure where the staffs have the authority to sign off at key project milestones. If the project

team does not do its job properly, the staff does not let the project proceed. What is most insidious about this situation is that the staffs are all enforcing rules that management and most objective observers would agree are needed. However, because of the tangle of approvals and sign-offs, the process consumes a great deal of the project's energy. Also, once the staffs have power, they often use that power to push pet objectives that are not strictly in their official charter.

由于这些专业的人员都是在监测和控制项目，项目似乎也把他们视为敌人。因此，如果专业人员能够摆脱项目，这些项目也就会完全忽略这些专业人员。为了防止这种情况的发生，管理者就建立了一个审查程序，在审查程序中，专业人员有权在关键项目里程碑点上停止项目活动。如果项目团队没有完全的完成相应工作，专业人员就不让项目继续进行。这种情况最关键的是所有专业人员都在执行规则，而这些规则是管理者和客观的观察者需要认同的。然而，由于混乱的的批准和停止活动，过程消耗了大量的项目精力。并且，一旦员工拥有权力，他们往往使用权力推行在正式章程中没有确认的“个人偏爱的”目标。

While this review and approval strategy generally guarantees that the projects pay attention to the staffs, the projects must also surmount an increasingly complex bureaucratic tangle of approvals just to do their work. This problem is bad enough when all projects were similar but, as projects get larger and more complex, they, too, become increasingly specialized. They then must use custom-tailored processes and procedures that are almost impossible to establish in a large and complex organizational jungle. Because of the energy consumed in getting bureaucratic approvals, the projects generally have to follow a process that doesn't precisely fit their needs. Now, just like a tropical rain forest, an increasing proportion of the organization's resources are devoted to delaying or stopping projects. Because of the enormous energy required to fight the bureaucracy, the projects have progressively less energy left to design and build their products. That is when rapid growth stops.

虽然这些审核和批准策略一般情况下保证了项目对专业人员的关注，但是项目应该克服批准工作所带来的日益复杂的官僚作风。当项目类似时，这个问题就变得很糟糕，如果项目变得更大更复杂，项目也会相应增加专业化程度。那时他们必须使用自定义的过程和规程，而这些过程和规程在庞大而复杂的组织结构中几乎不可能建立。由于项目在这种官

僚审批中消耗精力，因此项目一般都遵循了一个不是很符合他们需要的过程。现在，就像是一个典型的热带雨林，组织资源的增长比例延迟甚至停止了项目。由于大量能源用于对抗官僚作风，设计和构造产品就相应的减少了项目能源。那也就是快速增长停止的时候了。

While there is no magic solution to the problems of project scale, the situation is not hopeless, and there are useful steps we can take. Subsequent columns outline the most promising avenues for addressing these problems. I will also outline some principles to consider when working on large-scale development projects, some of the consequences of scale-related development problems, and some strategies for solving them.

虽然对于项目规模问题没有什么神奇的解决方案，但也不是没有一点希望，我们还可以采取一些有用的措施。后续的栏目提及解决这些问题的最有效的方法。我也会论述工作在大规模开发项目上应该考虑的一些原则，一些与规模开发问题相关的一些后果，和一些解决它们的策略。

Acknowledgements

感谢

In writing papers and columns, I make a practice of asking associates to review early drafts. For this column, I particularly appreciate the helpful comments and suggestions of Dan Burton, Julia Mullaney, Bill Peterson, Marsha Pomeroy-Huff, and Mark Sebern.

在写论文和专栏时，我经常回顾早期相关的草稿。对于这个专栏，我要特别感谢 Dan Burton ,Julia Mullaney, Bill Peterson, Marsha Pomeroy-Huff 和 Mark Sebern 给我提出的有用的意见和建议。

In closing, an invitation to readers

结束语，对读者的邀请

In this particular series of columns, I discuss some of the development issues related to large-scale projects. Since this is an enormous subject, I cannot hope to be comprehensive but I do want to address the issues you feel strongly about. So, if there are aspects of this subject that you feel are particularly important and would like covered,

please drop me a note with your comments, questions, or suggestions. Better yet, include a war story or brief anecdote that illustrates your ideas. I will read your notes and consider them when planning future columns.

Thanks for your attention and please stay tuned in.

Watts S. Humphrey

watts@sei.cmu.edu

在这一系列的专栏中，我讨论了与大规模项目相关的开发问题。由于这是一个非常大的课题，我不能期望它是非常全面的，只是我想阐述一些大家非常关注的问题。因此，如果你感觉到有一些很重要但还未描述清楚的问题，请把你的想法、疑问或建议告诉我。最好能够用一个战争的故事或简短的趣闻阐明你的想法。我将认真阅读这些反馈并在后续专栏的时候考虑到它们。

感谢您的关注，请继续留意。

Watts S. Humphrey

watts@sei.cmu.edu

About the Author

关于作者

Watts S. Humphrey founded the Software Process Program at the SEI. He is a fellow of the institute and is a research scientist on its staff. From 1959 to 1986, he was associated with IBM Corporation, where he was director of programming quality and process. His publications include many technical papers and several books. His most recent books are Introduction to the Team Software Process (2000) and Winning With Software: An Executive Strategy (2002). He holds five U.S. patents. He is a member of the Association for Computing Machinery, a fellow of the Institute for Electrical and

Electronics Engineers, and a past member of the Malcolm Baldrige National Quality Award Board of Examiners. He holds a BS in physics from the University of Chicago, an MS in physics from the Illinois Institute of Technology, and an MBA from the University of Chicago.

Watts S. Humphrey 在 SEI 创建了软件过程研究计划。他是 SEI 的一员，从事科学研究工作。从 1959 到 1986 年，他受雇于 IBM 公司，负责管理产品质量与过程。他发表了许多技术论文，并且出版了许多书籍。他最新的书籍是《团队软件过程》（2000）和《软件制胜之道》（2002）。他拥有 5 个美国专利。他是美国计算机协会和电气电子工程师协会的一员，曾经是美国 Malcolm Baldrige 全国质量奖的评审委员会成员。他本科毕业于芝加哥大学物理学，硕士毕业于美国伊利诺理工大学物理学，芝加哥大学 MBA。

第五部分：国内业界行情

声明：国内业界行情部分目前主要介绍国内咨询公司和工具厂商的简介及动态，所有信息均由咨询公司和工具厂商提供，仅供会员参考，思步网不对内容的真实性负责。

后期，国内业界行情部分会发布国内各咨询公司和工具厂商业务动态，以及国内行业动态（包括但不限于薪资水平、行业普遍问题、职业发展路线等）。

1 中科方德软件

中科方德 “基础软件国家工程研究中心”是国家发展改革委员会批复（发改高技[2005]425 号）由中国科学院软件研究所负责组建， 中科方德软件有限公司是基础软件国家工程研究中心的项目法人单位。公司由中国科学院软件研究所联合用友软件股份有限公司、上海和勤软件技术有限公司、深圳因泰克计算机技术有限公司、福建新大陆电脑股份有限公司、无锡滨湖科技创业投资有限责任公司等在国内相关领域具有技术优势、市场优势、资金与管理优势的科研单位和企业出资组建，注册资金 5200 万。

中科方德软件有限公司，以中国科学院软件研究所基础软件国家工程研究中心为后盾，是专门从事基础软件平台系列产品、安全系统软件系列产品和软件工程工具系列产品的研制、开发和销售的高科技公司。中科方德软件有限公司分别面向政府、军队/军工、航空航天、科研以及大、中、小型企业提供服务。 公司目前研发人员 300 余人，形成以中、青年

科技专家为骨干，年轻的博士、硕士为主要一线研发人员的科技队伍。

中科方德是国内唯一一家获得 SEI PARTNER 的中国机构，是国内所有咨询机构中唯一自身通过 CMMI 4 级评估，具备丰富的理论和实践经验、可以为过程改进提供全面的工具支撑、同时拥有一批具有丰富咨询经验的 CMMI 咨询师和评估师的公司。

中科方德涉及的领域包括软件类、硬件类、嵌入式类以及集成类等，在这些项目和产品开发中，进行了充分的实践和总结；经过多年的积累，推出技术和管理相融合的过程改进全面解决方案，即：**“过程改进全面解决方案” =“CMMI 咨询评估” + “理论结合实践的技术咨询” +“过程管理支撑工具”。**

（1）CMMI 咨询评估，包括 CMMI 过程改进咨询服务，SCAMPI 预评估和正式评估服务。

（2）技术咨询，包括需求获取、需求分析、项目拆分与估算、过程管理、项目管理、质量保证、质量管理、测试管理、配置管理等软件工程方法论的培训服务。

（3）工具支撑，基于 QONE 产品，提供需求管理、项目管理、质量管理、量化管理等方面的解决方案，全面支持 CMMI。