软件质量管理概述（PDF测试岗位课程）

课程讲师：孟林林

课程概述

课程编号：PM106　　**课程名称：**软件质量管理概述　　**课程级别：**初级

基本描述：

本课程提供软件质量管理的基础理论介绍，包括软件质量管理的定义和目标，软件只量管理模型，质量保证与质量控制的概念，常用的质量控制方法、软件质量度量标准、常见的软件质量管理体系。通过学习认识到软件质量管理的重要性。

课程目标：

通过本课程了解软件质量管理的基础概念

了解软件质量管理模型的内容

了解质量保证与质量控制的区别，以及质量控制的方法

了解软件质量的度量标准和指标

了解软件质量管的常用体系，CMMI和ISO9000体系

主要学习内容/要点：

软件质量管理的定义和目标

软件质量管理模型

质量保证与质量控制

软件质量度量标准

常见软件质量管理体系

苏宁软件测试质量度量

目标人群和课程时间：

信息体系初级专业技术人员、序列韩项目经理、产品经理、咨询顾问、IT软件开发工程师、IT软件测试工程师、IT架构师

课程类型、课堂培训

时长：2小时

授课要求和课程特色：

人数要求：20-50

分组要求：无

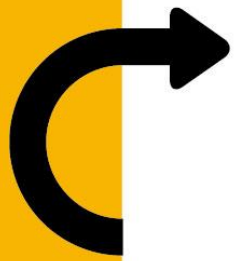
案例要求：无

提前需要学期的内容要求：无

课程目标

完成本课程学习以后，你能够：

- 通过本课程了解软件质量管理的基础概念；
- 了解软件质量管理的模型；
- 了解质量控制与质量保证及关系；
- 了解基本的软件质量度量标准；
- 了解软件质量管理体系：CMMI和ISO9000体系；
- 了解苏宁软件测试管理度量方法



第一部分：软件质量基本概念

第二部分：软件质量管理基本模型

第三部分：软件保证与质量控制&风险识别

第四部分：软件质量度量

第五部分：软件质量体系：ISO9000和CMMI

第六部分：软件测试质量管理

什么是软件质量

词典对质量的定义是：

1、典型的或本质的特征；
2、事物固有的或区别于其他事物的特征或本质；
3、优良或出色的程度。

IEEE及CMM对质量的定义是：

1、一个系统、组件或过程符合特定需求的程序；
2、一个系统、组件或过程符合客户或用户的需求或预期的程度。

思考一个人的“身体质量”——健康程度：如何衡量？
生理健康测量身高、体重、心跳、血压、体温等多方面的因素来判断是否健康；
如果因素都合格，则表明健康；否则表明某方面不健康，需对症下药；

软件质量是许多质量属性的综合体现，各种质量属性反映了软件质量的方方面面，通过改善软件的各种质量属性，从而提高软件的整体质量；

软件质量的属性

- 一般的，软件的质量属性很多，如正确性、精确性、健壮性、可靠性、容错性、性能、易用性、安全性、可扩展性、可复用性、兼容性、可移植性、可测试性、可维护性、灵活性等

- 10个常见软件质量属性，可分为两大类：
  —功能性因素：正确性、健壮性、可靠性
  —非功能性因素：性能、易用性、清晰性、安全性、可扩展性、兼容性、可移植性

- IBM采用CUPRIMDSO衡量软件质量与软件产品的满意度：
  —功能（capability）、使用性（usability）、性能（performance）、可靠性（reliability）、可安全性（installability）、可维护性（maintainability）、文档/信息（documentation）、服务（service）、综合性（overall）

10个软件质量属性及描述

正确性	指软件按照需求正确执行任务的能力。“正确性”的语义涵盖了“精确性”，正确性无疑是第一重要的软件质量属性。
健壮性	指在异常情况下，软件能够正常运行的能力。正确性与健壮性的区别是：前者描述软件在需求范围之内的行为，而后者描述软件在需求范围之外的行为。健壮性有两层含义：一是容错能力，二是恢复能力。
可靠性	是一个与时间相关的属性，指的是在一定环境下，在一定的时间段内，程序不出现故障的概率，因此是一个统计量，通常用平均无故障时间来衡量。软件可靠性问题通常是由于设计中没有预料到的异常和测试中没有暴露的代码缺陷引起的。
性能	通常是指软件的“时间-空间”效率，而不仅是软件的运行速度，人们总希望软件的运行速度高一些，并且占用资源少一些
易用性	只用户使用软件的容易程度。软件的易用性要让用户来评价。
清晰性	清晰意味着工作成果易读、易理解，开发人员只有在自己清晰的时候才能写出让别人易读易理解的程序和文档。
安全性	指防止系统被非法人入侵的能力，既属于技术问题又属于管理问题，一般的，如果黑客为非法入侵花费的代价（考虑时间、费用、风险等因素）高于得到的好处，那么这样的系统就可以认为是安全的。
可扩展性	反映了软件使用“变化”的能力。在软件开发过程中，“变化”是司空见惯的事情，如需求、设计的变化，算法的改进、程序的变化等。可扩展性是系统设计阶段重点考虑的质量属性
兼容性	指两个或两个以上的软件相互交换信息的能力。兼容性的商业规则是：弱者设法与强者兼容，否则无容身之地；强者应当避免被兼容，否则市场将被瓜分。
可移植性	指的是软件不经修改或稍加修改就可以运行于不同软硬件环境（CPU、OS和编译器）的能力，主要体现为代码的可移植性。

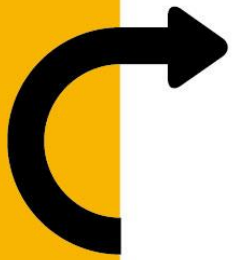
如何看待软件质量属性

- 重视软件质量是应该的，但是并不是“质量越高越好”。只有极少数软件应该追求“零缺陷”，对绝大数软件而言，商业目标决定质量目标；

- 企业的根本目标是为了获取尽可能多的利润，而不是产生完美无缺的产品，必须权衡质量、效率和成本，考虑提高质量所付出的代价；

- 如果某些质量属性并不能产生显著的经济效益，经过评估后，可以忽略他们，把精力放在对经济效益贡献最大的质量要素上，那些才值得下大工夫去改善；

- 对于一个特定的软件，我们首先判断有那些质量要素，然后才能给出提高质量的具体措施。不要一股脑想把所有的质量属性都做好，否则不仅做不好，还可能得不偿失。



第一部分：软件质量基本概念

第二部分：软件质量管理基本模型

第三部分：软件保证与质量控制&风险识别

第四部分：软件质量度量

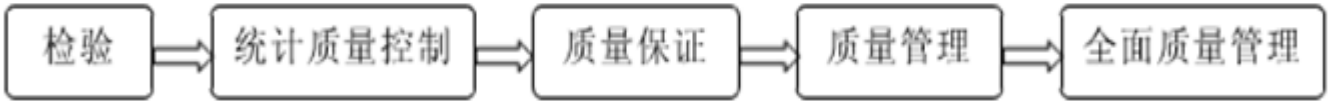
第五部分：软件质量体系：ISO9000和CMMI

第六部分：软件测试质量管理

第二部分 软件质量管理模型

软件质量管理的定义：确定质量方针、目标和职责，并通过质量体系中的质量计划、质量控制、质量保证和质量改进来使其实现的质量管理的所有活动。

软件质量管理的目的：产品监控项目的可交付产品和项目执行的过程，以确保它们符合相关的要求和标准，同时确保不合格项能够按照正确的方法或者预先规定的方式处理。

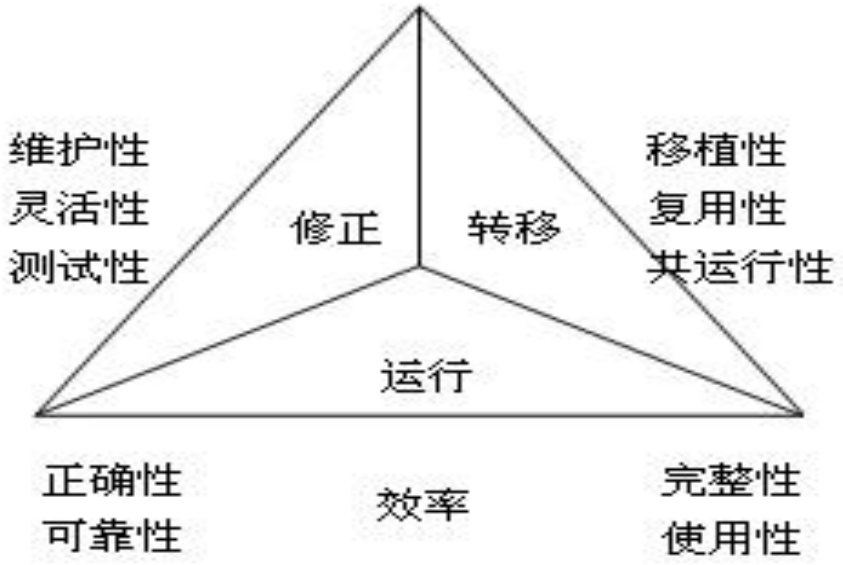


质量管理发展历史阶段

1. 以产品为中心的质量检验和统计质量控制阶段(18世纪 —— 1950年)
2. 以顾客为中心的质量保证阶段(1950年 —— 1987年)
3. 强调持续改进的质量管理阶段(1987年 —— 现在)
4. 全面质量管理阶段(TQM)

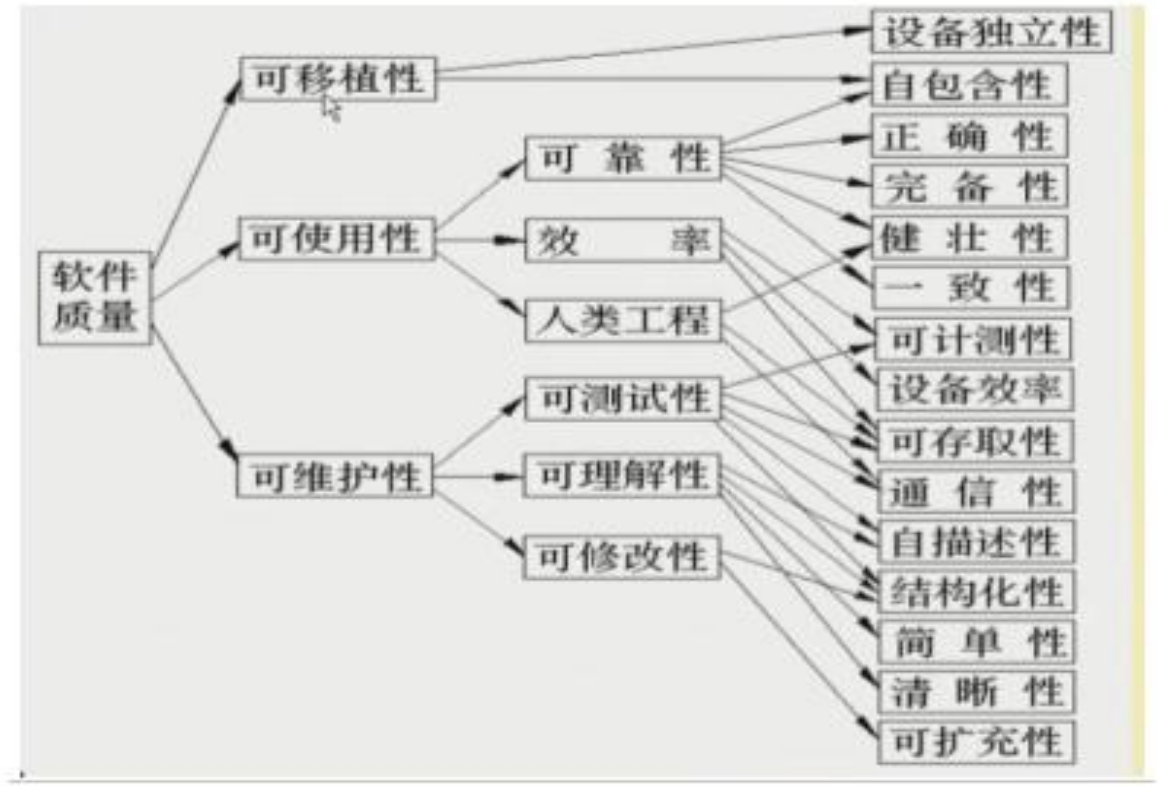
第二部分 软件质量管理模型

McCall软件质量模型：
1977年Walters 和McCall提出了软件质量层次模型与度量。
McCall认为软件的质量模型应该包括 产品的修正、产品的转移，产品的运行。
产品的修正又包括 可维护性、可测试性、灵活性等子特点。
产品的转移包括 可移植性、可复用性、互连性等。
产品的运行包括 正确性、可靠性、效率、可使用性和完整性



第二部分 软件质量管理模型

Boehm 软件质量模型：试图通过一系列的属性的指标来量化软件质量。Boehm 的质量模型包含了 McCall模型中没有的硬件属性。类似于McCall的质量模型，采用层级的质量模型结构，包括高层属性、中层属性和原始属性。



第二部分 软件质量管理模型

McCall模型-软件产品的质量属性

正 确 性	在预定环境下，软件满足设计规格说明及用户预期目标的程度。它要求软件本身没有错误
可 靠 性	软件按照设计要求，在规定时间和条件下不出故障，持续运行的程度
效 率	为了完成预定功能，软件系统所需的计算机资源的多少
完 整 性	为某一目的而保护数据，避免它受到偶然的或有意的破坏、改动或遗失的能力
可使用性	对于一个软件系统，用户学习、使用软件及为程序准备输入和解释输出所需工作量的大小
可维护性	为满足用户新的要求，或当环境发生了变化，或运行中发现了新的错误时，对一个已投入运行的软件进行相应诊断和修改所需工作量的大小
可测试性	测试软件以确保其能够执行预定功能所需工作量的大小
灵 活 性	修改或改进一个已投入运行的软件所需工作量的大小
可移植性	将一个软件系统从一个计算机系统或环境移植到另一个计算机系统或环境中运行时所需工作量的大小
可复用性	一个软件（或软件的部件）能再次用于其他应用（该应用的功能与此软件或软件部件的所完成的功能有关）的程度
互 连 性	又称相互操作性。连接一个软件和其它系统所需工作量的大小。如果这个软件要联网或与其它系统通信或要把其它系统纳入到自己的控制之下，必须有系统间的接口，使之可以联结

第二部分 软件质量管理模型

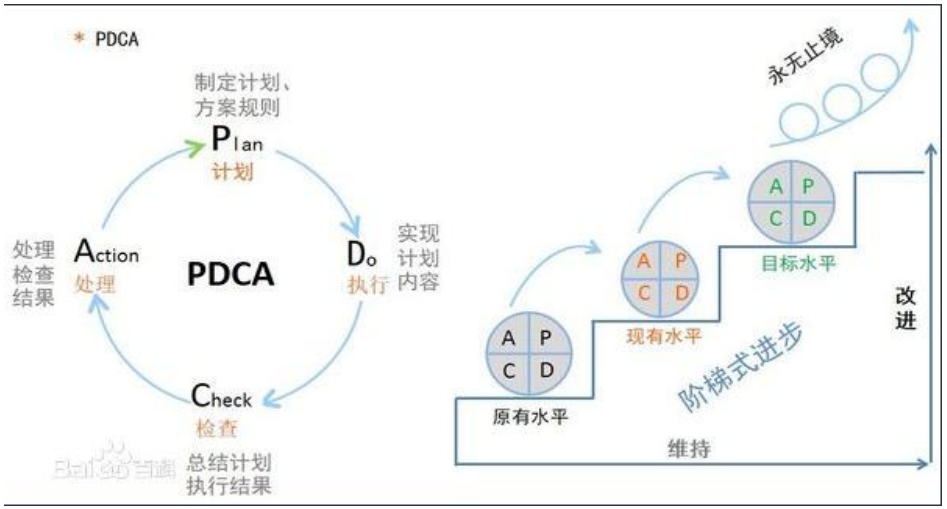
PDCA模型-又名戴明环，是美国质量管理专家休哈特博士首先提出的，它是全面质量管理所应遵循的科学程序。

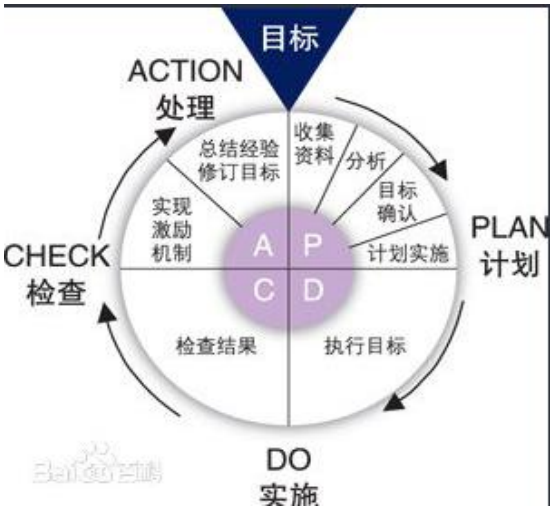
1、P (plan) 计划，包括方针和目标的确定，以及活动规划的制定。

2、D (Do) 执行，根据已知的信息，设计具体的方法、方案和计划布局；再根据设计和布局，进行具体运作，实现计划中的内容。

3、C (check) 检查，总结执行计划的结果，分清对错，找出问题。

4、A (act)修正，对检查的结果进行处理，对成功的经验加以肯定，并予以标准化；对于失败的教训也要总结，引起重视。对于没有解决的问题，应提交给下一个PDCA循环中去解决。

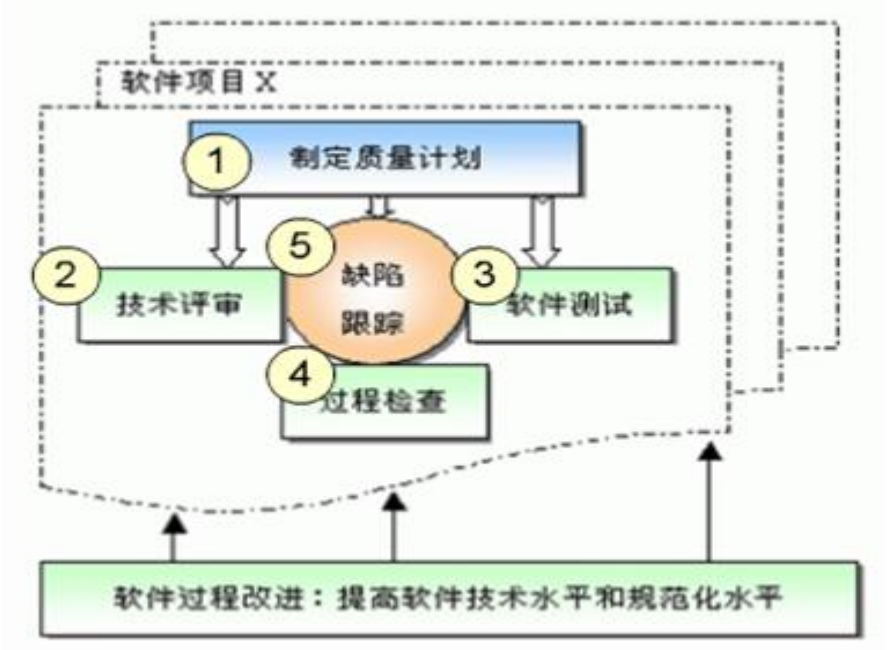




第二部分 软件质量管理模型

全面软件质量管理模型（ TQM ）

下图是全面质量管理的模型，项目中的所有人员几乎都参与质量活动，但介入的程度不同。



第二部分 软件质量管理模型

全面软件质量管理模型（ TQM ）(Total Quality Management)就是指一个组织以质量为中心，以全员参与为基础，目的在于通过顾客满意和本组织所有成员及社会受益而达到长期成功的管理途径。在全面质量管理中，质量这个概念和全部管理目标的实现有关。

TQM的核心理念：

顾客满意： 顾客即供应所提供产品的接受者，可以是组织内部的，也可以是组织外部的。

附加价值： 用最小的投入获取最大的功能价值，追求组织最大的经营绩效和个人最大的工作绩效。

持续改善： 建立以PDCA为基础的持续改善的管理体系

角色职责：

-谁对软件质量负责？全员负责。任何与软件开发，管理工作相关的人员都对质量产生影响，都要对质量负责。所以人们不要把质量问题全部推给质量人员或测试人员。

-谁对软件质量负最大的责任？谁的权利越大，他所负的质量责任就越大。质量人员是成天与质量打交道的人，但他个人并不对产品质量产生最大影响，所以也不负最大责任。

第二部分 软件质量管理模型

全面软件质量管理—1.制定质量管理计划

质量管理计划就是为了实现质量目标的计划。而质量目标则是由商业目标决定。质量管理计划是全面质量管理的行动纲领。

谁制定质量管理计划？由项目核心成员和质量人员共同协商制定，主要由质量人员起草，由项目经理审批即可。

质量管理计划的主要内容，例：

-1.质量要素分析
-2.质量目标
-3.人员与职责
-4.过程检查计划
-5.技术评审计划
-6.软件测试计划
-7.缺陷跟踪工具
-8.审批意见

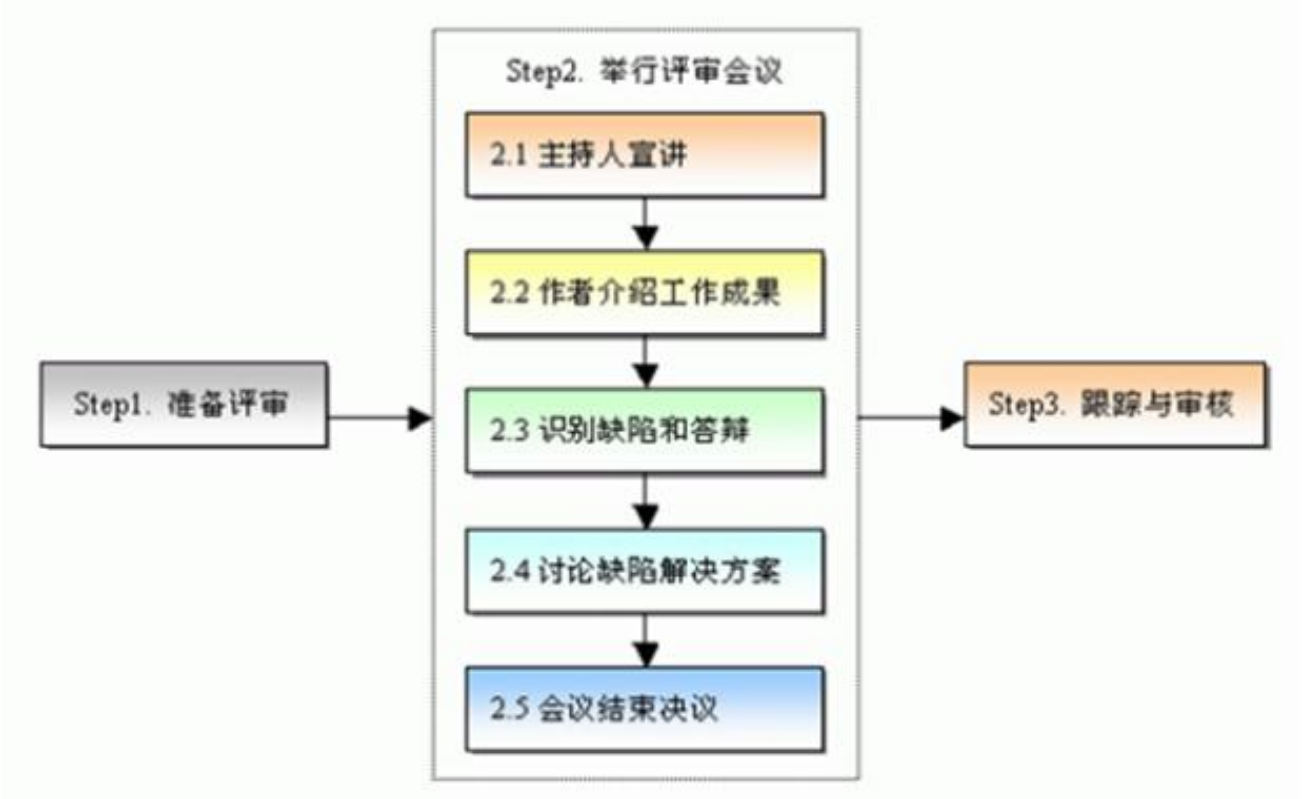
第二部分 软件质量管理模型

全面软件质量管理—2.技术评审

技术评审（Technical Review）的目的是尽早地发现工作成果中的缺陷，并帮助开发人员及时消除缺陷，从而有效的提高产品质量
技术评审防范由IBM倡导，已经被业界广泛采用，被普遍认为是软件开发的最佳实践之一
技术评审的好处有：
-通过消除工作成果的缺陷而提高产品的质量
-技术评审可以在任何开发阶段进行，越早消除缺陷越能降低成本
-开发人员能够得到同行专家的帮助和知道，无疑会增加对工作成果的理解，更好的预防缺陷，一定程度上提高了开发生产率
技术评审有两种基本类型：
-正式技术评审（FTR）：需要举行评审会议，参与人员相对较多
-非正式技术评审（ITR）：形式比较灵活，通常在同伴之间展开，不必举行评审会议，评审人员较少。

全面软件质量管理—2.技术评审

正式技术评审的流程：



第二部分 软件质量管理模型

全面软件质量管理—3. 软件测试

技术评审和软件测试的目的都是为了消除软件的缺陷，两者的主要区别是：

-前者无需运行软件，评审人员和作者把工作摆在桌面上讨论

-后者一定要运行软件来查找缺陷。技术评审在软件测试前进行，尤其在需求开发和系统设计阶段

-相比而言，软件测试的工作量通常要比技术评审大，发现的缺陷也更多

在制定质量计划的时候，已经确定了本项目的主要测试活动、时间和负责人，之后再考虑软件测试的详细计划和测试用例

!强调：质量人员一定要参与软件测试过程，只有这样才能深入的了解软件的质量问题，从而给予项目团队有力的帮助。

第二部分 软件质量管理模型

全面软件质量管理—4. 过程检查

CMM和ISO9000体系所述的软件质量保证，实质就是过程检查，即检查软件项目的“工作过程和工作成果”是否符合既定的规范,符合规范的工作成果不见得就是高质量的，但是明显不符合规范的工作成果十有八九是质量不合格的

-例如，版本控制检查

-例如，组织制定了重要工作成果的文档模板

过程检查的要点是：找出明显不符合规范的工作过程和工作成果，及时指导开发人员纠正问题，切勿吹毛求疵或者在无关痛痒的地方查来查去

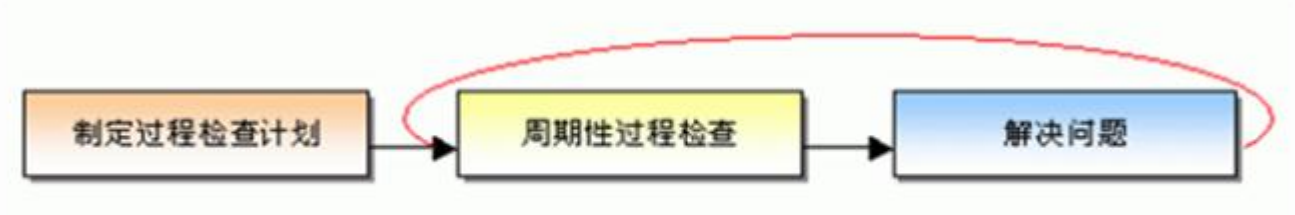
-不少组织的质量人员并没有真正理解过程检查的意义，经常对照规范，查找错别字、标点符号，排版等问题，迷失了方向，这样疲劳且没有效果，而且让开发人员很厌烦

全面软件质量管理—4. 过程检查

过程检查计划的要点是确定主要检查项和检查时间（或频度）

质量人员在执行过程检查的时候，如果发现问题，应该立即记录下来。过程问题也是缺陷，因此最好使用缺陷跟踪工具，有助于提高过程检查的效率

质量人员首先设法在项目内部解决已经发现的质量问题，与项目成员们协商，给出解决措施。在项目内难以解决的质量问题，由上级领导给出解决措施



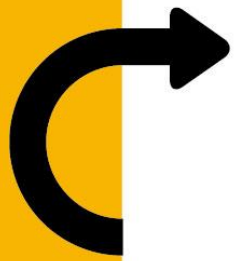
第二部分 软件质量管理模型

全面软件质量管理—5. 缺陷跟踪工具

作用：利用缺陷跟踪工具帮助项目成员记录和跟踪缺陷

缺陷的主要属性：缺陷ID，缺陷类型，缺陷状态，缺陷描述，相关文件，严重性，优先级，报告者，报告日期，接受者，解决方案（建议），解决日期等

缺陷跟踪工具的常见功能：查询缺陷，增加缺陷，修改缺陷，删除缺陷，缺陷统计图，缺陷趋势图，统计分析，邮件功能等



第一部分：软件质量基本概念

第二部分：软件质量管理基本模型

第三部分：软件保证与质量控制&风险识别

第四部分：软件质量度量

第五部分：软件质量体系：ISO9000和CMMI

第六部分：软件测试质量管理

软件质量保证（Quality Assurance）

软件质量保证是向管理层保证拟定出的标准、步骤、实践和方法能够正确地被所有项目所采用。通过对软件产品和活动进行评审和审计来验证软件是合乎标准的。

主要内容：

- 指定科学、合理、可行的质量标准
- 建立项目质量保证体系
- 开展有计划的质量改进活动

质量保证的主要工作：

依据	工具和方法	结果
项目质量计划 项目质量计划的时机执行情况 项目质量工作说	项目质量计划的方法 质量审计 实现规划 质量活动分解 质量保证体系	质量改进与提高的建议

软件质量保证（Quality Assurance）

- 质量保证的依据

① 质量计划：根本依据
② 质量计划的实际执行情况
③ 质量工作说明

- 质量保证的工具和方法

① 质量计划的工具和方法
② 质量审核：结构化审查
③ 事先规划：预先指定规范措施
④ 质量活动分解：逐层分解，使之容易控制
⑤ 质量体系保证：如设立质量保证部门

- 质量保证的结果

① 目前存在的问题及后果
② 原因分析
③ 改进或提高目标
④ 改进方法和步骤
⑤ 改进或提高的成果确认方法

软件质量保证（Quality Assurance）

- 质量控制是指检测并记录执行质量活动的结果，以保证绝大多数的缺陷可以在开发过程中被发现
- 主要内容：

① 度量项目质量的实际情况

② 与质量标准进行比较

③ 识别存在的质量问题和偏差

④ 记录质量问题

- 质量控制的主要工作：

依据	工具和方法	结果
项目质量计划 项目质量工作说明 项目质量计划的实际执行情况 质量检查表	因果图 控制图 帕累托图 趋势图 散点图	项目质量改进 验收决定 返工 项目调整

软件质量保证（Quality Assurance）

- 质量控制的依据

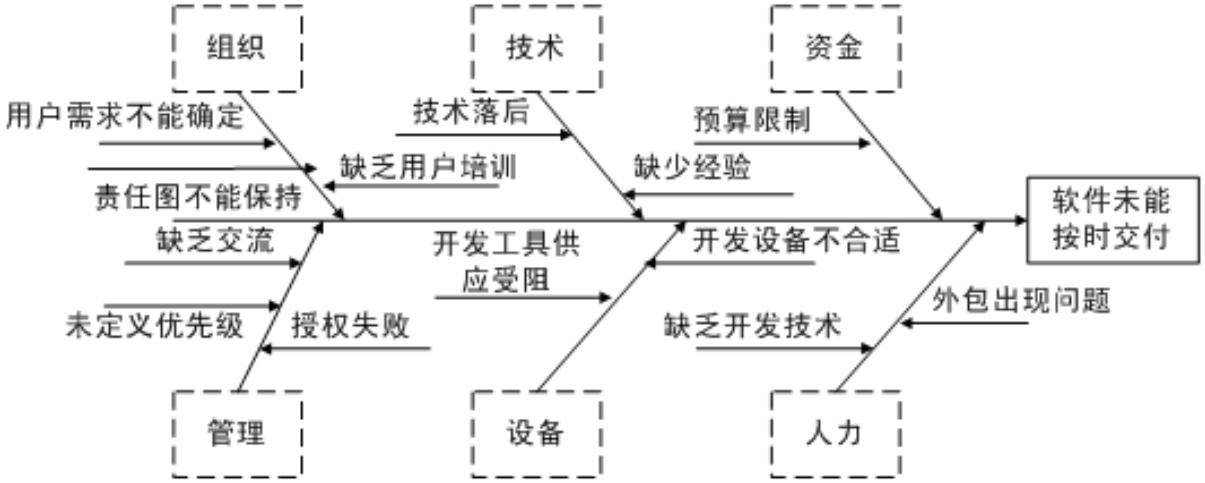
① 质量计划和质量工作说明
② 质量计划的实际执行情况
③ 质量检查表

- 质量控制的工具和方法

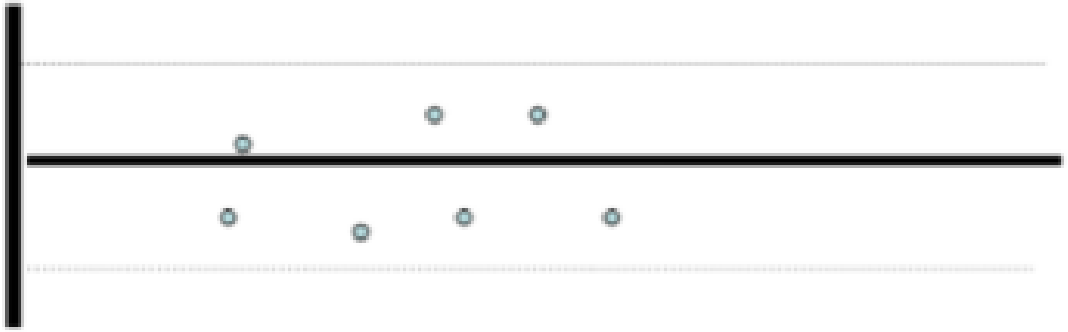
① 因果图
② 控制图
③ 帕累托图
④ 趋势图
⑤ 散点图
⑥ 调查表

软件质量控制的工具和方法

1、因果图：又称石川图或鱼骨图，直观的显示各种因素如何与潜在问题或结果相联系



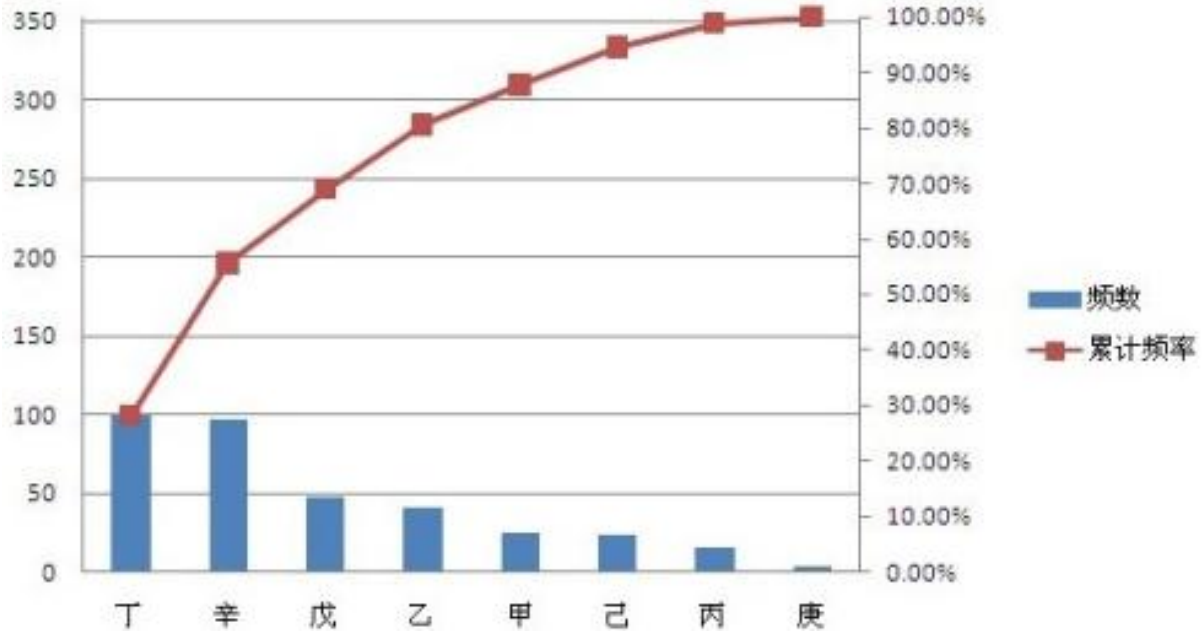
2、控制图：通过描述各样本的质量特征所在的区域来进行质量控制的方法，以判断项目的质量是否处于控制中



软件质量控制的工具和方法

3、帕累托图：依据关键的少数和次要的多数原理——帕累托80/20效率法则，帕累托图又称排列图，是一种寻找影响质量主次因素的方法

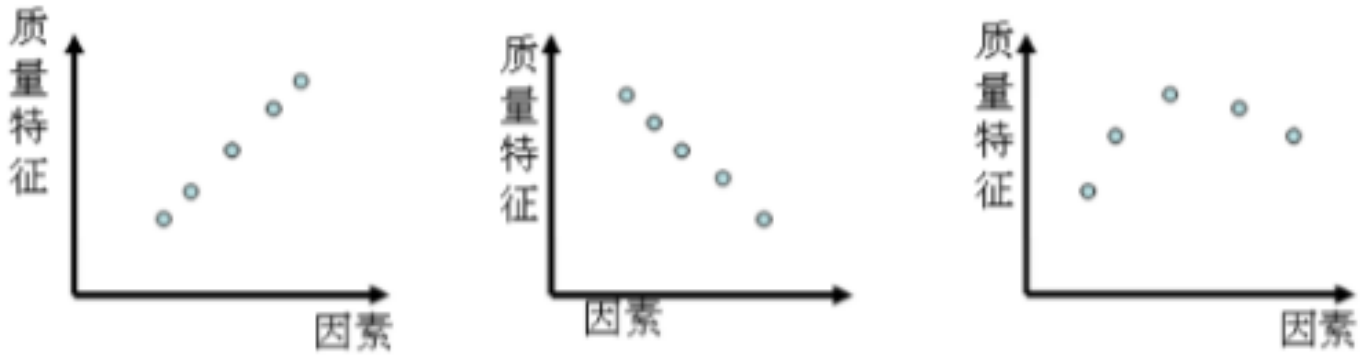
- 左侧纵坐标表示质量因素发生频数，右侧的纵坐标表示质量因素的频率
- 横坐标表示影响质量因素，按其影响程度的大小从左向右依次排序



4、趋势图：相当于没有界限的控制图，用来反映某种变化的历史和模式，趋势图是一种线形图，可以显示随时时间推移的过程趋势，过程变化，或者过程的恶化和改进情况

软件质量控制的工具和方法

5、散点图：是分析、判断、研究两个相对应的变量之间是否存在相关关系并明确相关程度的方法。散布图有一个横坐标X和纵坐标Y构成，根据测得数据画出坐标点，进行相关性分析



6、调查表：统计调查分析表，用图表、表格统计项目执行过程中发生的各种数据，并对统计数据进行粗略的分析。这种方法能将收集的数据，通过表格的形式列出来，使数据呈现的问题系统、简明化，故便于分析使用

质量保证与质量控制的关系

- 相同点：

—二者均是质量管理活动的关键内容，互相之间有交互重叠

—均在整体的质量计划的指导下执行

- 不同点：

—质量保证的重要目的是寻求根源，查找影响质量因素的原因，然后改进

—质量控制的重要目的是讲实际情况对比质量标准，跟踪记录差距

风险识别

- 风险的定义（PMP中的定义）：

—风险是指与项目相关的若干不确定性事件或条件，一旦发生，将会对项目目标的实现产生正面或负面的影响。

- 风险管理的目的：

—风险管理的目的是在风险发生前，界定出潜在的问题，以便在产品或项目的生命周期中规划风险处理活动，并于必要时启动之，从而将不利于完成目标的影响降低。

- 风险识别过程：

—风险识别过程是将不确定性事项转变为明确的风险陈述。

—风险三要素判断：不确定性、未发生、对目标的达成有影响。

- 可能的参与人员：

—项目组成员、机构项目管理或专门的风险管理人员、技术专家、客户、最终用户、项目组外有经验的项目经理、其他干系人等。

风险识别

- 风险的特点：

—风险存在的客观性和普遍性

—风险的多变性

—风险的时间性

—风险发生具有偶然性和必然性

- 风险的基本因素：

—风险的基本因素

—风险信号（征兆）：指在风险发生之前的提示信号。

—发生概率：风险发生的几率。

—风险影响：风险发生后对项目产生的影响。

—风险级别：根据风险发生的概率以及风险影响程度将风险划分为高、中、低不同的等级。

风险识别

- 风险的识别流程

—是一个在整个项目过程中多次反复的过程。

风险识别-风险的识别技术

一、信息收集法

1、头脑风暴法（Brain Storm）：通过思维高度活跃,打破常规的思维方式产生大量创造性设想的方法。该方法最常用，当所有项目组成员或人数众多的成员参与项目风险识别时使用。

2、Delphi法：征得专家的意见之后，进行整理、归纳、统计，再反馈给各专家，再次征求意见，再集中，再反馈，直至得到稳定的意见。该方法用户专家对项目风险进行识别时使用。

3、面谈法：与有经验的项目经理、行业专家以及相关方代表进行面谈，通过他们的经验为项目风险识别提供帮助。

4、SWOT法（道斯矩阵）：将与项目相关的各种主要优势因素（Strengths）、弱点因素（Weaknesses）、机会因素（Opportunities）、威胁因素（Threaths），通过调查罗列，并按照矩阵形式排列，运用系统分析方法，将这些因素加以分析，得出结论。通常用于项目风险管理小组对风险的识别。

		内部因素	
		优势	劣势
外部因素	机会	优势与机会相匹配 SO	劣势与机会相匹配 WO
	威胁	优势与威胁相匹配 ST	劣势与威胁相匹配 WT

风险识别-风险的识别技术

二、图表分析法

1、因果图（鱼刺图/石川图）:用于对影响结果的全部因素进行分析。适用于各种角色的人对项目风险进行识别。

2、项目流程图:通过寻找项目各组成部分之间的相互联系和依赖关系，发现其中的风险。适合项目风险管理小组使用。

3、相关图:通过变量与结果之间的关系来确定影响结果的因素，从而识别其中的风险。适合项目风险小组使用。

三、风险条目检查表法

检查表中已经根据软件项目的特点列出了潜在的风险，由适用人员逐项评估。使用于项目经理和项目风险管理小组对项目进行风险识别。

四、假设条件分析法

对于项目计划制定过程中设定的主要假设条件进行评估，从中寻找出存在的风险因素。适用于项目经理和项目风险管理小组对风险进行识别。

风险识别的输出:

项目风险列表，并纳入组织的“项目风险知识库”。

第一部分：软件质量基本概念

第二部分：软件质量管理基本模型

第三部分：软件保证与质量控制&风险识别

第四部分：软件质量度量

第五部分：软件质量体系：ISO9000和CMMI

第六部分：软件测试质量管理

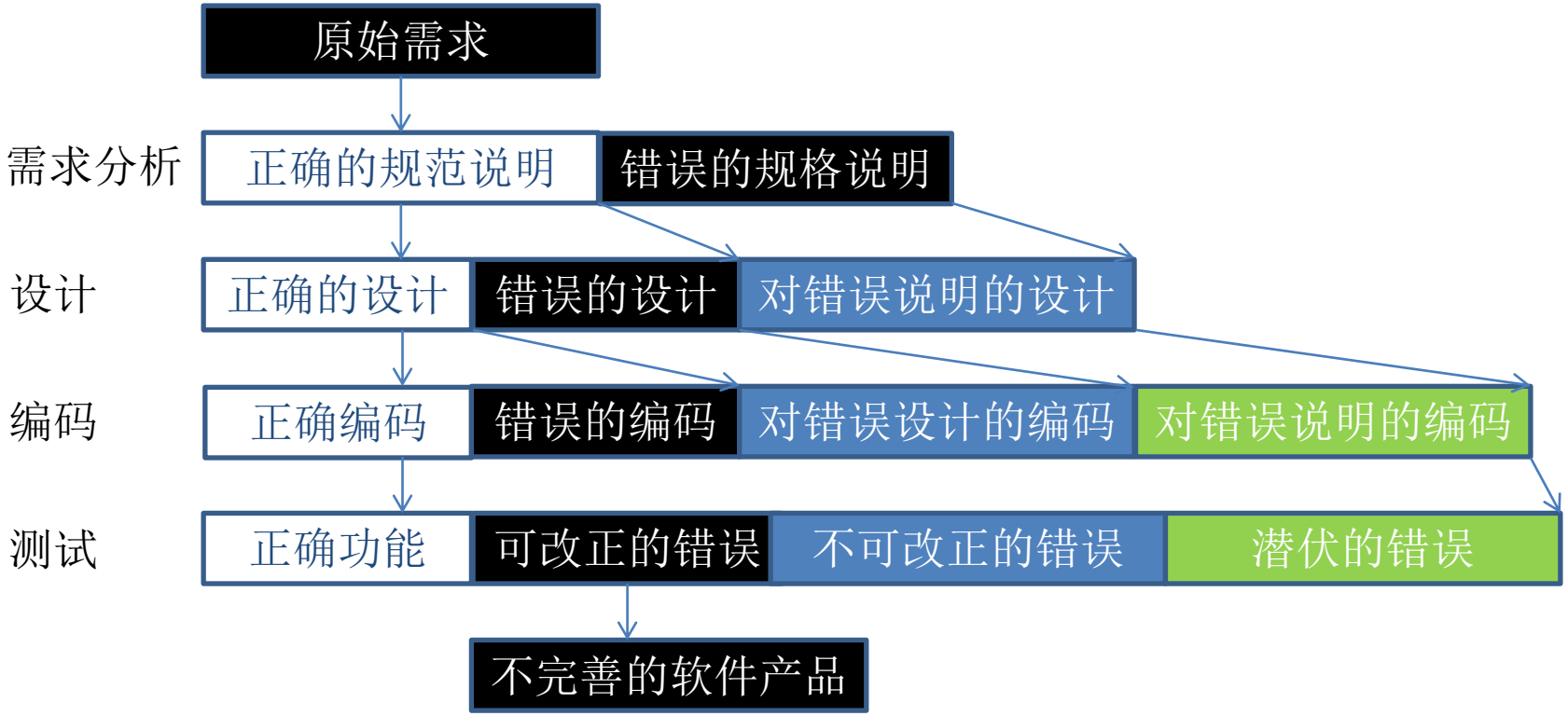
第四部分 软件质量度量

软件质量度量介绍

软件质量度量的目的：
度量是管理的根基
度量是对软件产品进行范围广泛的测试类，给出一个系统或过程的某个给定属性的定量测量

通常，质量度量包括：
软件产品质量的度量----测量各个质量指标，评估最终产品整体质量
--产品质量因素和质量水平
--用户问题：来自用户角度的缺陷或非缺陷问题
--客户满意度：非常满意/满意/一般/不满意/非常不满意
过程中质量的度量----衡量过程效率/有效性，持续改进过程，提高软件生产力
--缺陷密度、缺陷移除率、缺陷分布、缺陷修复率等
--代码行LOC、代码行缺陷率、功能点缺陷率维护的度量
--平均修复时间MTTR

软件产品缺陷的来源



缺陷的“积累和放大”效应

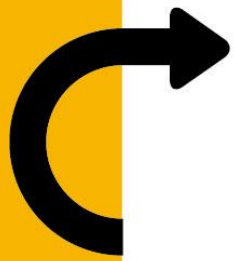
第四部分 软件质量度量

常见的软件质量度量指标

序号	度量指标	定义	参考基准
1	已交付产品的质量	已交付产品的缺陷数/功能点（FP） 已交付产品的缺陷数=验收缺陷+维护缺陷	0.00-0.094个/FP （平均0.021）
		已交付产品的缺陷/KLOC	0.2-0.5个/KLOC
		已交付产品的缺陷/人时	0.00-0.012个/人时
2	缺陷引入率	总的生命期缺陷引入率=缺陷总数/人时	0.00-0.015个/人时 （平均0.052）
		总的生命期缺陷引入率=缺陷总数/FP	
3	缺陷清除效率	整体缺陷排除效率=（1-验收测试缺陷数/总缺陷数）100%	78~100% （平均94%）
		阶段缺陷清除率=本阶段清除的缺陷/（阶段入口处已存在的缺陷+本阶段注入的缺陷）100%	

第四部分 软件质量度量

序号	度量指标	定义	参考基准
4	质量成本	（评审工作量+返工工作量+缺陷修改工作量+测试计划准备工作量+测试准备工作量+测试执行工作量+培训工作量+质量保证工作量）/实际工作量	30-40%
5	进度计划符合度	实际进度与计划进度的对比	80%的项目在原定进度计划的10%以内
6	工作量的分布	按工作性质划分占总体的占比	
7	缺陷引入的分布	按缺陷引入的时间、功能区域划分	
8	生产率	考察队成品的产出率	
9	缺陷密度（每千行代码缺陷数）	缺陷数/新增千行代码量（defects / KLOC）	CMM1级 11.95‰，CMM2级 5.52‰，CMM3级 2.39‰，CMM4级 0.92‰，CMM5级 0.32‰



第一部分：软件质量基本概念

第二部分：软件质量管理基本模型

第三部分：软件保证与质量控制&风险识别

第四部分：软件质量度量

第五部分：软件质量体系：ISO9000和CMMI

第六部分：软件测试质量管理

第五部分 软件质量体系：ISO9000和CMMI

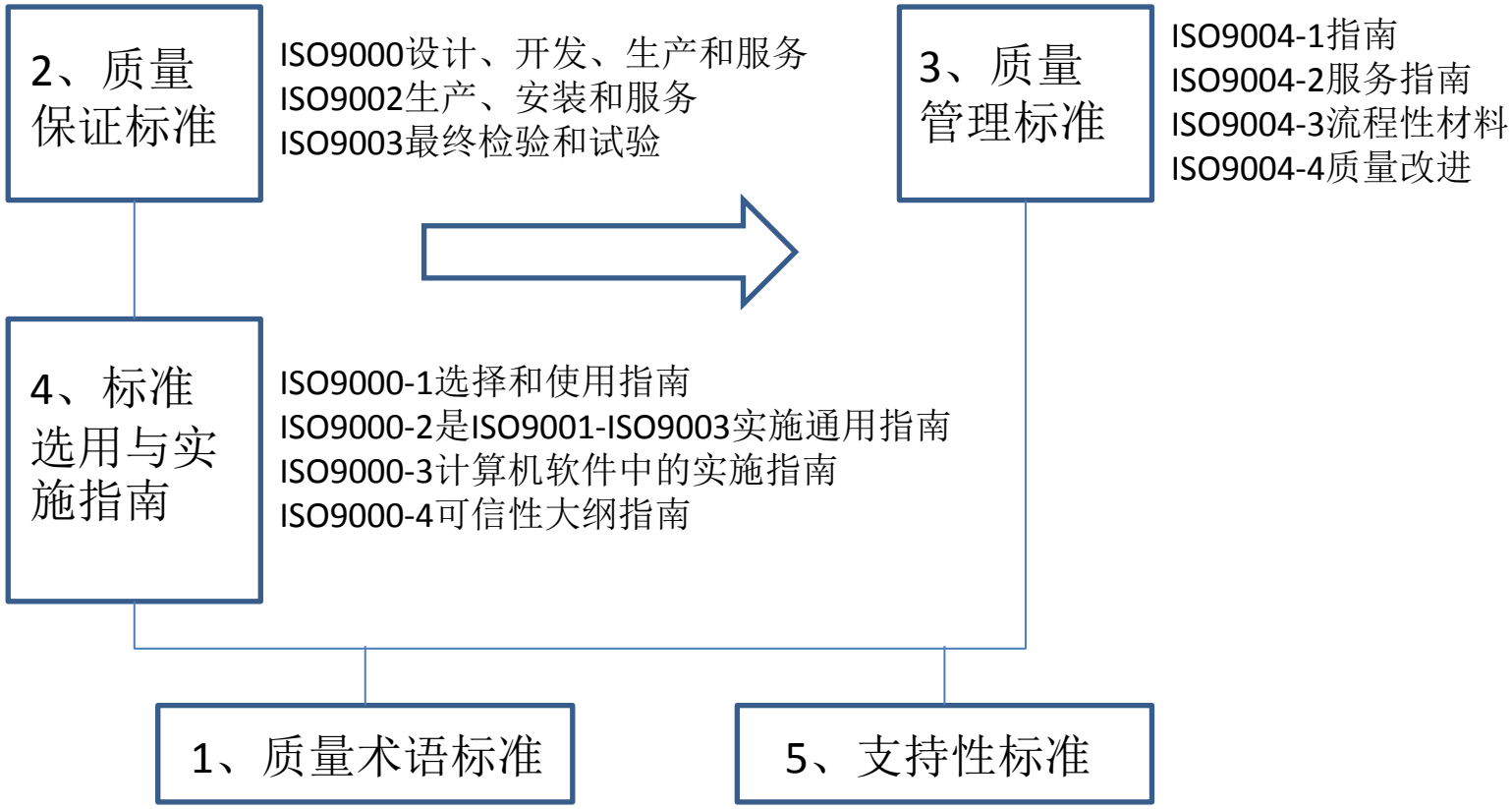
ISO9000体系简介

ISO标准系列，是指一种标准的统称，由TC176质量管理体系技术委员会制定的所有国际标准

ISO9000族标准由5个部分组成：

1、质量术语标准
2、质量保证标准
3、质量管理标准
4、质量管理和质量保证标准的选用和实施指南
5、支持性技术标准

ISO9000体系的标准框架和内容



ISO9000标准系统框架

ISO9000的原则：

1．以顾客为关注焦点

组织应该理解顾客当前的和未来的需求，满足顾客要求并争取超越其期望。①客户永远是对的；②如果客户不对，则执行①

2．领导作用

领导者确立组织统一的宗旨及方向，他们应当创造并保持使员工能充分参与实现组织目标的内部环境。80%质量问题与管理有关，20%与员工有关。

3．全员参与

各级人员都是组织之本，只有他们的充分参与，才能使其才能给组织带来效益。

4．过程方法

将活动和相关的资源作为过程进行管理，可以更高效地取得期望的结果,如流程图方法等。

5．管理的系统方法

将相互关联的过程作为体系加以识别、理解和管理，有助于组织提高实现目标的有效性和效率。木水桶的围板原理。

6．持续改进

持续改进总体业绩应当是组织的一个永恒的目标。PDCA循环。

7．基于事实的决策方法

有效决策是建立在数据和信息分析的基础上。

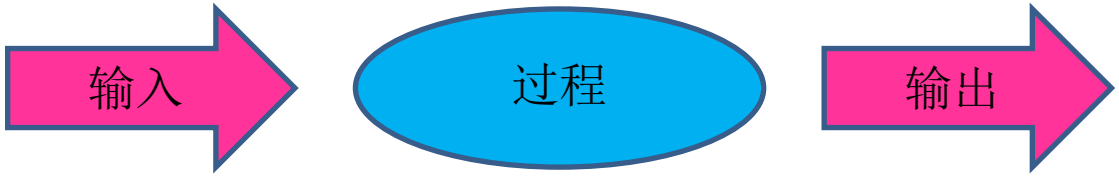
8．互利的供方关系

组织与供方是相互依存的，互利的关系可以增强双方创造价值的能力。

CMMI体系介绍—软件过程改进概述

1、过程的基本概念

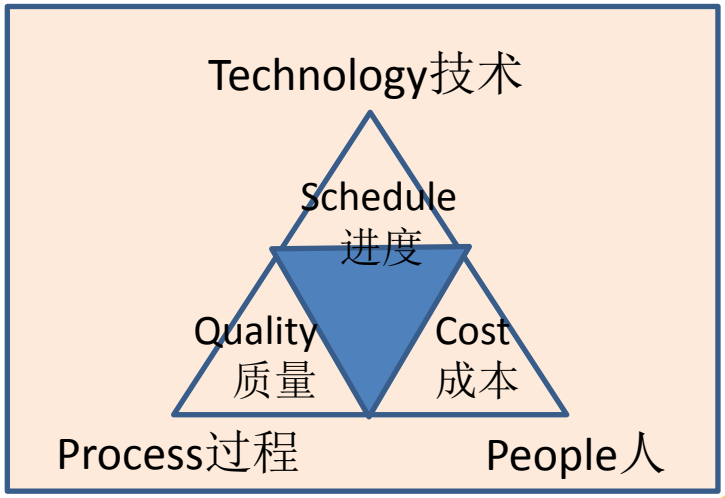
—过程（Process： A process is a set of practices performed to achieve a given purpose）就是指人们根据相应的方法/规程、使用技术/工具等将原材料（输入）转化成用户需要的产品（输出）



—过程的3个基本要素是：人、方法与规程、技术与工具，过程、人和技术是产品开发成本、进度和质量的主要决定因素

—过程与产品存在因果关系，即好的过程才能得到好的产品，而差的过程只会得到差的产品

“产品的质量很大程度上取决于开发和维护该产品的过程的质量”



第五部分 软件质量体系：ISO9000和CMMI

2、什么是软件过程改进

—从20世纪90年代至今，软件过程改进成为软件工程学科的一个主流研究方向，其中CMM和CMMI是该领域举世瞩目的重大成果

—提高软件过程能力的实践通称为软件过程改进（Software Process Improvement）

—软件过程改进的根本目的是：提高质量、提高生产率并且降低开发成本

—主要的软件过程域：

工程类主要过程域：需求开发、系统设计、软件实现、软件测试、软件维护等

管理类主要过程域：项目规划、项目监控、需求管理、质量管理、配置管理等

3、软件过程中改进必须走规范化之路

—提高软件过程能力唯有走“规范化”之路，即“制定适合于本企业的软件过程规范，并按照此规范执行”

—“规范化”不会抑制人们的创造力，相反的，它使得团队可以大规模的复用前人积累的智慧和财富

—业界实施已经证明，走“规范化”之路是“成本最低、见效最快、能持续发展”的软件过程改进方法，任何IT企业（不论大小），都有办法走规范化之路，从而有效的提高软件过程能力

第五部分 软件质量体系：ISO9000和CMMI

CMMI是什么

—CMM（Capability Maturity Model）能力成熟度模型是用于衡量软件过程能力的事实上的标准，同时也是目前软件过程改进最好的参考标准

—美国卡内基-梅隆大学软件工程研究所（SEI）研制

发展简史

—CMM1.0于1991年制定

—CMM1.1于1993年发布，该版本应用最广泛

—CMM2.0草案于1997年制定（未广泛使用）

—到2000年，CMM演变成为CMMI（ Capability Maturity Model Integration），CMM2.0成为CMMI1.0的主要组成部分

—CMMI-SE/SW 1.1（CMMI for System Engineering and Software Engineering）于**2002年1月正式推出**

—2007年8月，CMMI V1.2 正式发布

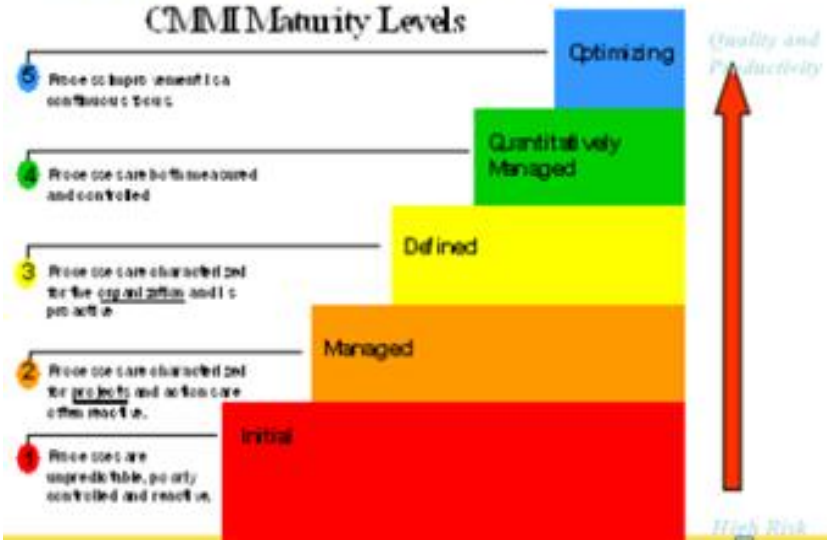
—目前CMMI V1.3版本

CMMI模型简介

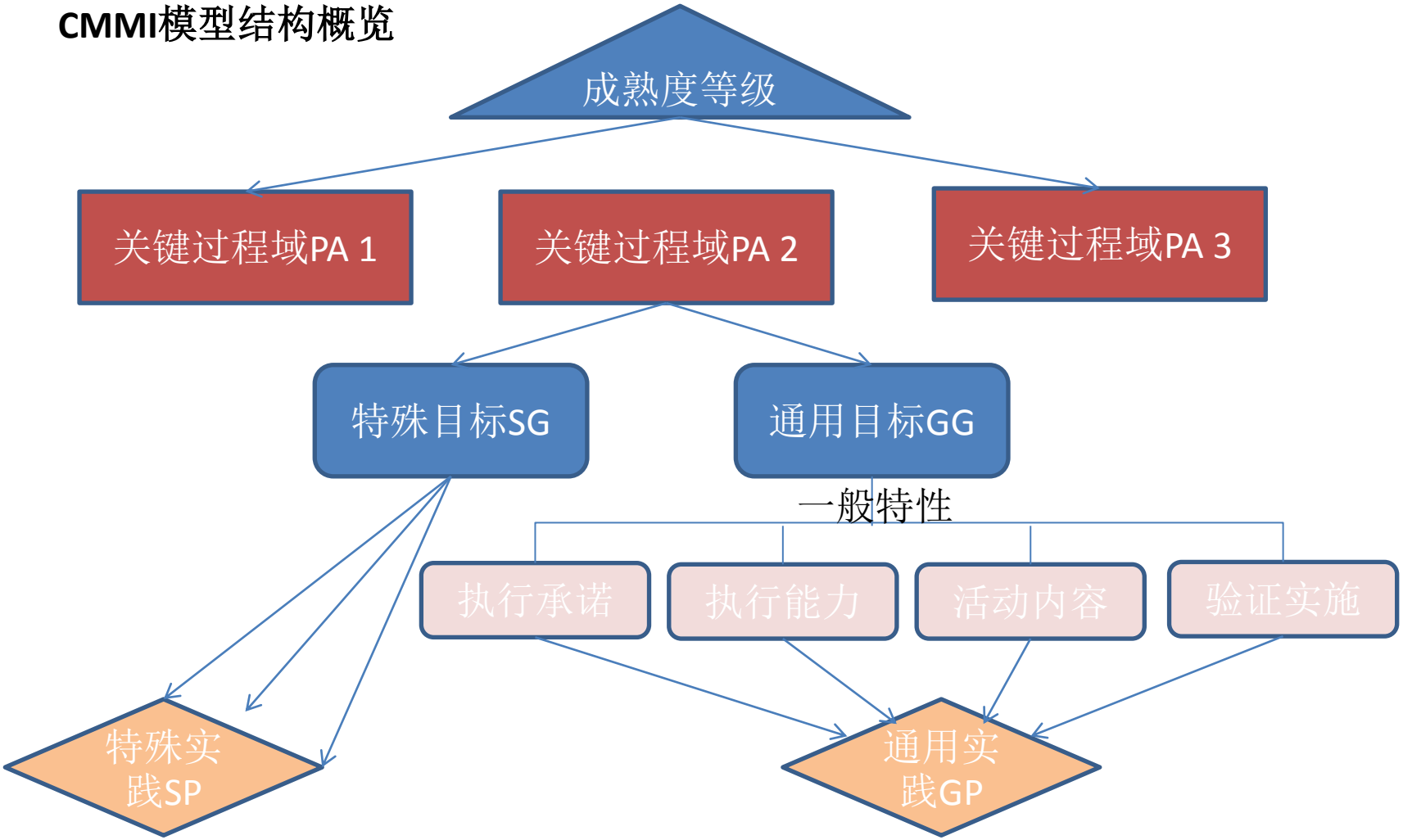
- CMMI目前包括4个过程改进模型
  - 系统工程 (SE)
  - 软件工程 (SW)
  - 集成产品和过程开发 (IPPD)
  - 软件采购 (SS)

- CMMI模型包括：
  - CMMI框架 (CMMI framework)
  - 成熟度模型 (Maturity models)
  - 等级评估方法 (Appraisal methods)
  - 等级评估材料 (Appraisal materials)
  - 培训 (Training)

- CMMI提供了阶段式和连续式两种表示方法



CMMI模型结构概览


成熟度等级
关键过程域PA 1
关键过程域PA 2
关键过程域PA 3
特殊目标SG
通用目标GG
一般特性
执行承诺
执行能力
活动内容
验证实施
特殊实践SP
通用实践GP

定义

—成熟度等级：共分为5级

—关键过程域(Process Area):是指为了达到某个成熟度等级必须要解决的一组问题。它包括了一系列的相关实践活动，通过正确的执行，达到一组重要的目标，从而使得该过程域获得显著改进

—特殊目标（Specific Goals）：描述了针对某一特定过程，为改进该过程而必须完成的特殊的特性

—特殊实践（ Specific Practices）：是指为完成相关的特殊目标而至关重要的活动

—通用目标（Generic Goals）个通用实践（ Generic Practices ）：描述了适合于所有过程改进的制度或策略

每个PA只有一个GG，每个PA都具有想用的GP

每个过程域，都有一些特殊目标和通用目标，通过相应的特殊实践和通用实践来实现这些目标。当一个过程域的所有特殊实践和通用实践都按要求得到实践，就能实现该过程域的目标

第五部分 软件质量体系：ISO9000和CMMI

CMMI模型包括25个关键过程域（PAs），分别对应5个成熟度等级

成熟度等级	关注点	关键过程域
5 – 优化管理级	持续过程改进	◆ Organizational Innovation and Deployment 组织革新与部署 ◆ Causal Analysis and Resolution 原因分析与决策
4 – 量化管理级	量化过程管理	◆ Organizational Process Performance 组织过程性能 ◆ Quantitative Project Management 量化项目管理
3 – 已定义级	过程标准化	◆ Requirements Development 需求开发 ◆ Technical Solution 技术方案 ◆ Product Integration 产品集成 ◆ Verification 验证 ◆ Validation 确认 ◆ Organizational Process Focus 组织过程焦点 ◆ Organizational Process Definition 组织过程定义 ◆ Organizational Training 组织培训 ◆ Integrated Project Management for IPPD 集成产品项目管理 ◆ Risk Management 风险管理 ◆ Integrated Teaming 整合团队 ◆ Integrated Supplier Management 集成供应商管理 ◆ Decision Analysis and Resolution 决策分析与制定 ◆ Organizational Environment for Integration 组织集成环境
2 – 已管理级	基本项目管理	◆ Requirements Management 需求管理 ◆ Project Planning 项目计划 ◆ Project Monitoring and Control 项目监控 ◆ Supplier Agreement Management 供应商协议管理 ◆ Measurement and Analysis 度量及分析 ◆ Process and Product Quality Assurance 过程及产品质量保证 ◆ Configuration Management 配置管理
1 – 初始级		◆ None 无

第五部分 软件质量体系：ISO9000和CMMI

CMMI-SW中成熟度2共有22个目标，成熟度3共有43个目标

关键过程域	SG				GG	
	SG1	SG2	SG3	SG4	GG2	GG3
需求管理	5				10	2
项目计划	4	7	3		10	2
项目监控	7	3			10	2
供应商协议管理	3	4			10	2
度量与分析	4	4			10	2
过程及产品质量保证	2	2			10	2
配置管理	3	2	2		10	2
需求开发	2	3	5			12
技术方案	3	4	2			12
产品集成	3	2	4			12
验证	3	3	2			12
确认	3	3				12
组织过程焦点	3	4				12
组织过程定义	5					12
组织培训	4	3				12
集成产品项目管理	5	3	2	3		12
风险管理	3	2	2			12
整合团队	3	5				12
集成供应商管理	2	3				12
决策分析与制定	6					12
组织集成环境	3	3				12

成熟度 2　成熟度 3　不要求

CMMI与ISO9000体系的关系

相似点

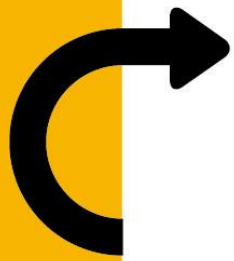
—着眼于质量和过程管理，两者都为了解决同样的问题
—“言所行，行所言”
—强调管理、过程、规范化和文档化

不同点

—CMM模型明确强调持续的过程改进，而ISO9001只要求质量体系的最小保证
—CMM把焦点严格对准软件，而ISO9001的规范适用于更大的范围，包括：硬件、软件、流程性材料和服务

两者之间的联系

—CMM2级与ISO9001强相关
—CMM的每个关键过程域至少按某种解释与ISO9001弱相关



第一部分：软件质量基本概念

第二部分：软件质量管理基本模型

第三部分：软件保证与质量控制&风险识别

第四部分：软件质量度量

第五部分：软件质量体系：ISO9000和CMMI

第六部分：软件测试质量管理

第六部分：软件测试质量管理

软件测试从直观上来讲是对测试对象进行检查、验证，似乎很简单，但实际不然，它是由许多处理环节构成的。根据测试目标、质量控制的要求，它被划分为各类环节，并被设置了不同的准入、准出标准。而这些环节和标准是根据什么来进行管理，如何度量？

下面以苏宁金融集团测试管理中心的项目监控表作为一个例子进行描述：

一、确认管理里程碑阶段：

大体上分为：需求定义，用户体验设计，系统设计，测试计划，测试准备，测试执行（包括准入和准出），测试报告等；

二、确认有效关键项

根据项目的具体情况，确认有效关键项，不是所有项都必要全部执行。

下面列举各阶段关键项：

1、需求定义阶段：编写需求说明书，评审需求说明书，基线需求说明书

2、用户体验设计：编写UI/UE设计说明书，评审UI/UE设计说明书，基线UI/UE设计说明书

3、系统设计：编写系统设计说明书，评审系统设计说明书，基线系统设计说明书

第六部分：软件测试质量管理

4、测试计划

过程阶段	关键工作项	检查点描述	是否执行	计划完成日期	实际完成日期	关键输出项
测试计划	编写测试计划	是否制订了测试阶段的详细计划？	是	XXX	XXX	测试计划_终稿
	评审测试计划	豆芽？邮件？会议？哪种评审方法？是否提交配置库？	是	XXX	XXX	测试计划_评审记录
	基线测试计划		是	XXX	XXX	测试计划_基线报告（邮件）
	编写测试方案	是否定义了测试范围、质量指标、测试环境、测试工具、测试方法？	是	XXX	XXX	测试方案_终稿
	评审测试方案	豆芽？邮件？会议？哪种评审方法？是否提交配置库？	是	XXX	XXX	测试方案_评审记录
	基线测试方案		否	XXX	XXX	测试方案_基线报告（邮件）
	编写RTVM矩阵		是	XXX	XXX	RTVM矩阵_终稿
	评审RTVM矩阵		是	XXX	XXX	RTVM矩阵_评审记录
	基线RTVM矩阵		否	XXX	XXX	RTVM矩阵_基线报告（邮件）

5、测试准备

过程阶段	关键工作项	检查点描述	是否执行	计划完成日期	实际完成日期	关键输出项
测试准备	编写测试案例	1、测试工具/浏览器、测试环境、模块执行前置条件，是否和《测试方案》中内容一致？ 2、是否定义了案例编号、案例名称、案例描述、案例级别、预置条件、测试数据、步骤、步骤描述、预期输出？	是	XXX	XXX	测试案例_终稿
	评审测试案例	豆芽？邮件？会议？哪种评审方法？是否提交配置库？	是	XXX	XXX	测试案例_评审记录
	基线测试案例		否	XXX	XXX	测试案例_基线报告（邮件）
	准备测试数据	项目配置库\01项目测试管理\10需求管理\测试数据，是否全部提供？（包括测试环境、测试工具、测试账号等）	是	XXX	XXX	测试数据

第六部分：软件测试质量管理

6、测试执行

首先确认，开发是否已经冒烟通过，并发送冒烟通过邮件，然后测试进入测试执行阶段（如冒烟不通过则打回）。

过程阶段	关键工作项	检查点描述	是否执行	计划完成日期	实际完成日期	关键输出项
测试执行	冒烟测试	测试人员是否进行了冒烟测试再进行SIT测试？	是	XXX	XXX	冒烟测试_执行记录
	冒烟测试报告	测试人员是否编制冒烟测试报告？	是	XXX	XXX	冒烟测试_测试报告（邮件）
	SIT测试准入检查	是否编制《测试准入标准表》？	是	XXX	XXX	测试准入检查记录
	SIT测试执行	测试人员根据《测试案例》进行集成测试，并记录测试情况。	是	XXX	XXX	SIT第1轮测试_执行记录
	SIT测试准出检查	是否编制《测试准出标准表》？	是	XXX	XXX	测试准出检查记录
	PRE测试准入检查	是否编制《测试准入标准表》？	是	XXX	XXX	测试准入检查记录
	PRE测试	测试人员根据《测试案例》进行预生产测试，并记录测试情况。	是	XXX	XXX	PRE第1轮测试_执行记录
	PRE测试准出检查	是否编制《测试准出标准表》？	是	XXX	XXX	测试准出检查记录
	验收测试	测试人员进行预生产验收测试，并记录测试情况。	否	XXX	XXX	PRE验收测试_执行记录
	封版测试	测试人员进行预生产封版测试，并记录测试情况。	是	XXX	XXX	PRE封版测试_执行记录
	测试日报	1、每天上午由测试经理把昨天的测试情况，汇报给相关人员，是否识别了近期测试中的风险问题？ 2、监控测试进度?进度异常是否有调整？ 3、测试缺陷等级分布、测试问题的解决结果	是	XXX	XXX	测试日报（邮件）

第六部分：软件测试质量管理-缺陷有效跟踪

7、测试报告

过程阶段	关键工作项	检查点描述	是否执行	计划完成日期	实际完成日期	关键输出项
测试报告	功能测试报告	所有测试完成，测试经理编制功能部分的测试报告，提交配置库。	是	XXX	XXX	功能测试报告
	验收测试报告	针对外包项目做验收测试完成后，测试经理编制验收测试报告，提交配置库。	是	XXX	XXX	验收测试报告
	性能测试报告	所有测试完成，测试经理编制性能部分的测试报告，提交配置库。	否	XXX	XXX	性能测试报告

三、确认完关键项之后，根据关键项进行度量数据收集。例如关键项输出的数量，质量，时效性等。例如对于测试案例来说，可以统计测试案例的执行率，需求覆盖度，有效性，案例执行趋势等。

四、数据收集完成之后做测试质量分析和总结。

第六部分：软件测试质量管理-缺陷有效跟踪

缺陷-指软件文档（如软件需求规格说明、设计规格说明等等）或程序代码中存在的数据错误、逻辑错误、内容遗漏以及内容上的不一致性等等。

一、缺陷管理的目标：

1、 确保每个被发现的缺陷都能够被解决；这里解决的意思不一定是被修正，也可能是其他处理方式（例如，在下一个版本中修正或是不修正），总之，对每个被发现的BUG的处理方式必须能够在开发组织中达到一致；

2、 收集缺陷数据并根据缺陷趋势曲线识别测试过程的阶段；决定测试过程是否结束有很多种方式，通过缺陷趋势曲线来确定测试过程是否结束是常用并且较为有效的一种方式。

3、收集缺陷数据并在其上进行数据分析，作为组织的过程财富。

第六部分：软件测试质量管理-缺陷有效跟踪

二、缺陷的处理流程：

整体流程：提交>审核>修复>回归

PS：上述流程存在重复多次的情况，若回归不通过则重新打开继续下一个流程。缺陷的详细流程见对象文档

Microsoft Visio
2003-2010 绘图

第六部分：软件测试质量管理-缺陷有效跟踪

三、合理的的缺陷的描述

1、对缺陷的描述应该包含可追踪信息

每个缺陷有唯一的一个缺陷号，可以根据该编号搜索、根据、查看该缺陷的处理情况。

2、对缺陷的描述应该包含缺陷的基本信息

通常缺陷的基本信息包括缺陷状态、缺陷标题、缺陷严重程度、缺陷紧急程度、缺陷提交人、缺陷提交日期、缺陷解决人、缺陷解决时间、缺陷解决结果、缺陷处理人、缺陷处理最终时间、缺陷处理结果、缺陷确认人、缺陷确认时间、缺陷确认结果等等。

3、对缺陷的描述应该包含缺陷的详细描述；

即是对缺陷的特征应做详细的描述，例如程序代码中的错误，应详细描述错误发生的软硬件环境，相关输入输出数据，出错时程序的状态等等，以方便编码人员进行错误复现和错误定位。又如设计规格说明中的错误，应指明它与软件开发文档（如需求规格说明）中哪些条款相违背，为什么判为违背，都需要描述清楚，以方便设计人员进一步核实。

4、对某些缺陷的描述应该包含必要的附件；

有时，某些缺陷可能无法用语言文字表达清楚，如用户界面出现的一些缺陷，光用语言文字是难以表述清楚。这时，就需要借助于屏幕拷贝等方式来进一步描述缺陷。

第六部分：软件测试质量管理-缺陷有效跟踪

三、缺陷中的角色和相应的职责

角色	描述	工作说明
测试人员	测试执行人员	识别缺陷 记录并提交缺陷 执行复测，检验缺陷是否成功解决 重新打开或关闭缺陷
测试经理	测试经理	审核缺陷的有效性 驳回不完整或描述有误的缺陷 识别重复缺陷和无效缺陷， 提交缺陷给项目经理审核分配 管理缺陷，跟踪缺陷状态
项目经理	项目经理	审核缺陷的有效性 驳回不完整或描述有误的缺陷 识别重复缺陷和无效缺陷， 评估需要延期修复的缺陷， 分发缺陷给开发人员、需求人员或者开发组长分析
修复人员	开发组长	识别重复缺陷或无效缺陷，退回给项目经理 分析缺陷，分发给开发人员修改，或转发给其他团队的适合的受理人 协调开发团队修复缺陷，参与评审或给出缺陷解决方案 对需要延期的缺陷返回给项目经理，给出意见或参与评审
	开发人员	识别重复缺陷或无效缺陷，退回给项目经理 修改程序设计或代码的相关缺陷 修改后的代码或程序自测完成后，提交测试人员复测
	需求人员	识别重复缺陷或无效缺陷，退回给项目经理 解决业务需求相关的缺陷 对需要延期的缺陷提供业务分析和决策

第六部分：软件测试质量管理-缺陷有效跟踪

三、跟进缺陷的注意点

苏宁目前在用JIRA做为缺陷管理工具，在使用该工具做缺陷跟踪时，要注意以下几点：

1、缺陷描述：新建时描述信息要明确，步骤要清晰，具体的测试数据应该在描述中体现；

2、标签：每个版本会有一个问题单的统一标签用于问题单的统计和整体管理；

3、遗留的问题要经过产品，开发，测试三方沟通同意后，交由项目经理（or产品经理）置为遗留，严重以上级别的问题需要明确应对方案并经过中心领导同意；

4、回归不通过的问题单，状态改为重新打开，并备注详细不通过的原因

5、验证通过后解决结果为“已验证关闭”；

6、问题单需要图片，日志等进行辅助说明的，附件上传图片和日志；

7、跟踪开发解决问题，必须要有详细的问题解决描述；

软件测试质量管理度量

课程要点回顾

—软件质量的基本概念
—软件质量管理的定义和目的
—软件质量管理模型：McCall、Boehm、PDCA、全面软件质量管理TQM
—质量保证的目的和内容
—质量控制的目的和内容
—质量控制方法：鱼骨图、帕累托图、控制图、趋势图、散点图
—软件质量度量标准及指标
—常见软件质量体系：CMMI、ISO9000体系
—苏宁金融软件测试质量管理

Thanks!