



# 软件工程

## 第十三章 软件质量管理

乔立民

**qlm@hit.edu.cn**

**2011年6月6日**

# 主要内容

## 13.1 软件质量管理

### 13.1.1 什么是软件质量

### 13.1.2 软件质量属性和要素

### 13.1.3 全面质量管理

## 13.2 软件变更管理



How the customer explained it



How the Project Leader understood it



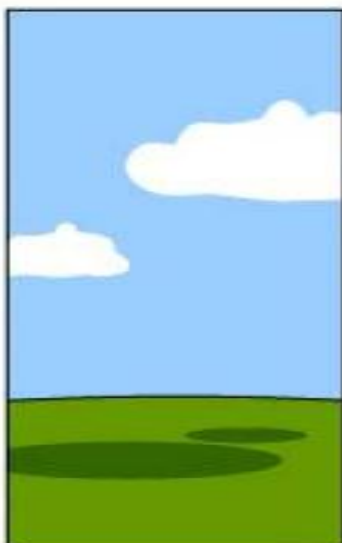
How the Analyst designed it



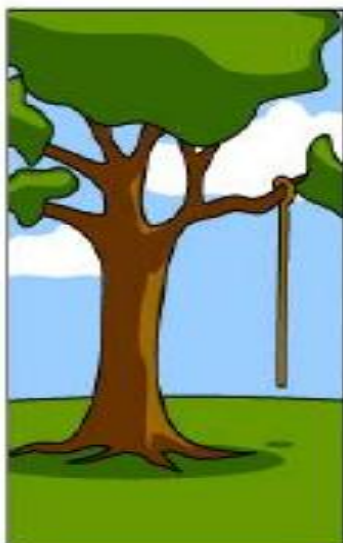
How the Programmer wrote it



How the Business Consultant described it



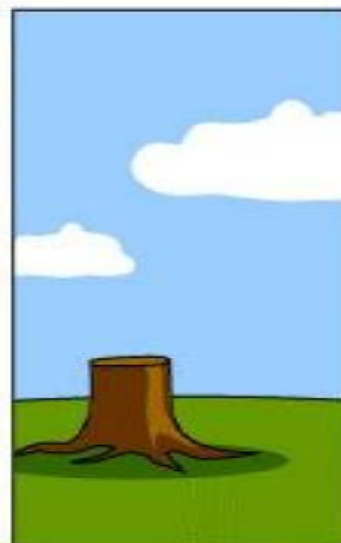
How the project was documented



What operations installed



How the customer was billed



How it was supported



What the customer really needed

# 什么是软件质量？

## ■ 消费者观点

- 适合使用：产品或服务就应该是被期望的那样
- 设计质量：设计的质量特性应包含在产品或服务中
- 用户满意度=合格的产品+好的质量+按预算和进度安排交付

## ■ 生产者观点

- 质量的一致性：确保产品或服务是根据设计制造的
- 利润：确保用最少的成本投入获取最大利益

# 质量的定义

- **质量：某一事物的特征或属性。（美国传统字典）**
  - 可测量的特征——与已知标准可以进行比较，如长度、速度等
  - 软件是一种知识实体，其特征的定义远比物理对象要困难的多

# 软件质量

- 软件质量是对明确陈述的**功能和性能**需求、明确记录的开发标准以及对所有专业化软件开发应具备的**隐含特性的符合度**。
  - 软件需求是质量测量的基础，不符合需求就是没有质量
  - 特定标准定义了一组用以指导软件开发方式的准则。若未能遵守准则，则肯定质量成问题
  - 有一组未被提及的隐式需求（如：对易用性的期望）。若软件符合显示需求，但未能满足其隐式需求，则软件质量仍然值得怀疑

# 软件质量

- 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。人们通过改善软件的各种质量属性，从而提高软件的整体质量（否则无从下手）。
  - 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。
  - 质量属性可分为两大类：“功能性”与“非功能性”，后者有时也称为“能力”（Capability）。

# 主要内容

## 13.1 软件质量管理

### 13.1.1 什么是软件质量

### 13.1.2 软件质量属性和要素

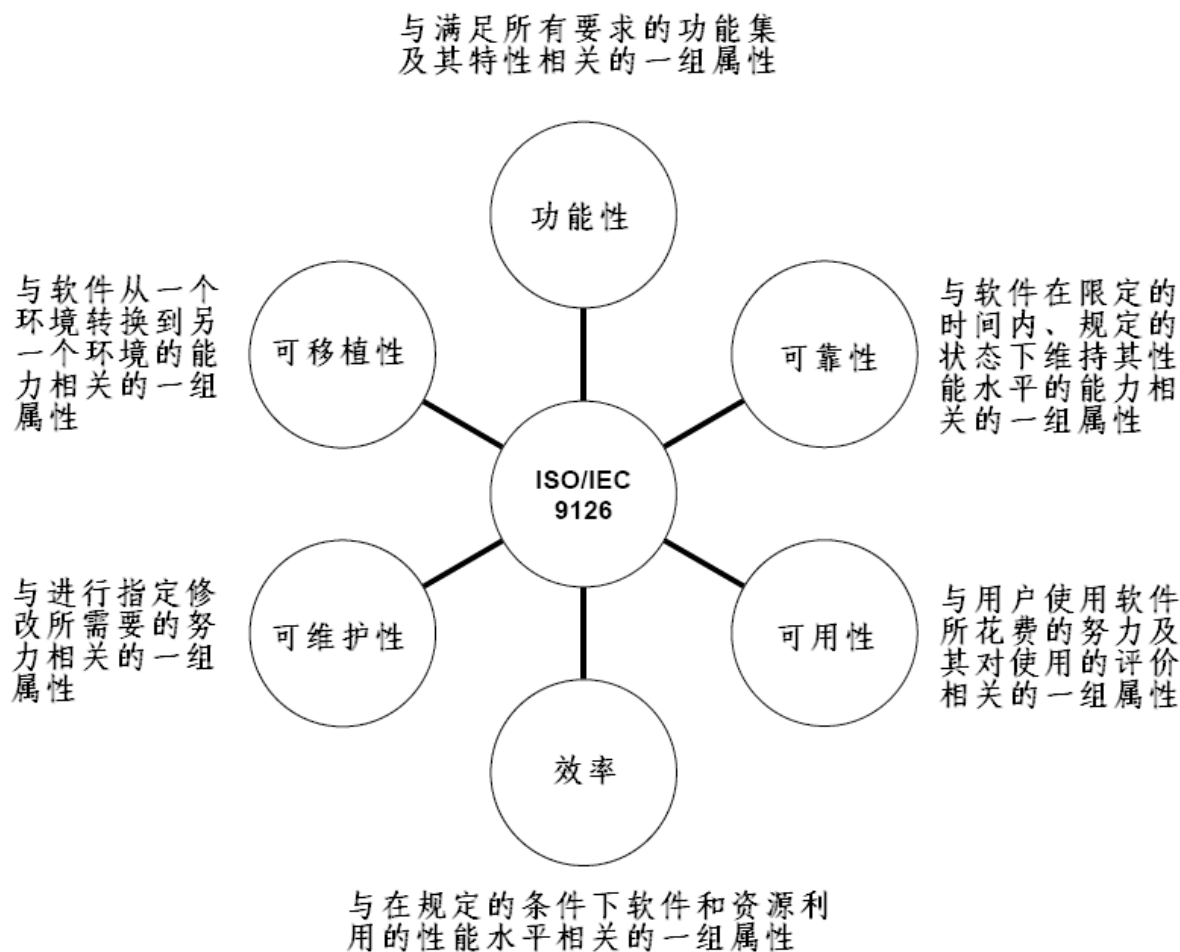
### 13.1.3 全面质量管理

## 13.2 软件变更管理



## 1.软件质量因素

## 国际软件质量标准：ISO/IEC 9126



# ISO/IEC 9126

特征	子特征	简要描述
功能性	精确性	软件准确依照规定条款程度，规定确定了权利、协议的结果或者协议的效果
	依从性	软件符合法定的相关标准、协定、规则或其他类似规定的程度
	互操作性	软件和指定系统进行交互的能力
	安全性	软件阻止对其程序和数据进行未授权访问的能力，未授权的访问可能是有意，也可能是无意的
	适合性	指定任务的相应功能是否存以及功能的适合程度

# ISO/IEC 9126

可靠性	成熟性	因软件缺陷而导致的故障频率程度
	容错性	软件在故障或者外界违反其指定接口的情况下维持其指定性能水平的能力
	可恢复性	软件在故障后重建其性能水平、恢复其受影响数据的能力、时间和精力
	依从性	同上

# ISO/IEC 9126

易用性	可理解性	用户认可软件的逻辑概念和其适用性需要花费的精力
	可学习性	用户为了学会使用软件需要花费的精力
	可操作性	用户执行软件操作和控制软件操作需要花费的精力
	吸引性	软件吸引用户的能力
	依从性	同上

# ISO/IEC 9126

效率	时间行为	执行功能时的响应时间、处理时间和吞吐速度
	资源行为	执行功能时使用资源的数量和时间
	依从性	同上

# ISO/IEC 9126

可维护性	可分析性	诊断软件中的缺陷、故障的原因或者识别待修改部分需要花费的精力
	可改变性	进行功能修改、缺陷剔除或者应付环境改变需要花费的精力
	稳定性	因修改导致未预料结果的风险程度
	可测试性	确认已修改软件需要花费的精力
	依从性	同上

# ISO/IEC 9126

可移植性	适应性	不需采用额外的活动或手段就能适应不同指定环境的能力
	可安装性	在指定的环境中安装软件需要花费的精力
	共存性	在公共环境中同分享公共资源的其他独立软件共存的能力
	可替换性	在另一个指定软件的环境下，替换该指定软件的能力和需要花费的精力
	依从性	同上

## 2. 软件质量要素

### ■ 软件质量要素

#### — 什么是软件质量要素？

- 从技术角度讲，对软件整体质量影响最大的那些质量属性才是质量要素
- 从商业角度讲，客户最关心的、能成为卖点的质量属性才是质量要素

#### — 对于一个特定的软件而言，我们首先判断什么是质量要素，才能给出提高质量的具体措施，而不是一股脑地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。

#### — 如果某些质量属性并不能产生显著的经济效益，我们可以忽略它们，把精力用在対经济效益贡献最大的质量要素上。简而言之，只有质量要素才值得开发人员下功夫去改善。



# 商业目标决定质量目标！

## ■ 教科书的片面观点

- 大凡软件工程教科书为了强调质量的重要性，总是要举一些历史上发生过的重大软件质量事故，例如航天飞机爆炸、核电站失事、爱国者导弹发生故障等等。这些事故的确不是危言耸听，给人们敲响了质量的警钟。
- 学术界总是喜欢宣扬质量至上的理念，而忽视企业的商业利益，将质量目标凌驾于商业目标之上。
- 重视软件质量是应该的，但是“质量越高越好”并不是普适的真理。只有极少数软件应该追求“零缺陷”，对绝大多数软件而言，**商业目标决定了质量目标，而不该把质量目标凌驾于商业目标之上。**

## ■ 严格系统对质量的要求

- 航空航天等系统对质量要求极高，任何缺陷都有可能导致机毁人亡，所以人们不惜一切代价去消除缺陷。在发射航天器之前，只要发现任何异常，就会立即取消发射指令，直到异常被消除为止。

# 商业目标决定质量目标！

## ■ 商业目标决定质量目标

- 上述严格系统毕竟是少数，绝大多数普通软件的缺陷并不会造成机毁人亡这样的重大损失，否则没有人敢从事软件开发了。在日常工作中，我们接触过的软件几乎都是有缺陷的，即便是软件业老大Microsoft，它的软件产品也经常出错甚至导致死机，人们骂几句后还会照样使用有缺陷的软件。
- 企业的根本目标是为了获取尽可能多的利润，而不是生产完美无缺的产品。如果企业销售出去的软件的质量比较差，轻则挨骂，重则被退货甚至被索赔，因此为了提高用户对产品的满意度，企业必须提高产品的质量。但是企业不可能为了追求完美的质量而不惜一切代价，当企业为提高质量所付出的代价超过销售收益时，这个产品已经没有商业价值了，还不如不开发。
- 企业必须权衡质量、效率和成本，产品质量太低了或者太高了，都不利于企业获取利润。企业理想的质量目标不是“零缺陷”，而是恰好让广大用户满意，并且将提高质量所付出的代价控制在预算之内。

### 3. 质量管理活动

- 质量计划

- 为特定的项目选择合适的程序和标准，并按要求进行修改、调整
- 事前计划，确定目标

- 质量控制（QC）

- 为了保证每一件工作产品都能满足对它的需求而在整个软件过程中所运用的一系列审查、评审和测试。
- 事后检查

- 质量保证（QA）

- 为了保证软件高质量而必需的“有计划的、系统化的行动模式”。
- 过程控制

- 质量管理应与项目管理分离，以确保其独立性

## 4.质量成本的构成

### ■ 预防成本

- 质量计划
- 正式技术评审
- 测试设施
- 培训

### ■ 鉴定成本

- 审查
- 设备校准和维护
- 测试

### ■ 失效成本

- 内部失效成本
  - 返工
  - 修复
  - 错误分析
- 外部失效成本
  - 投诉解决
  - 退换产品
  - 服务支持
  - 保修工作

# 主要内容

## 13.1 软件质量管理

### 13.1.1 什么是软件质量

### 13.1.2 软件质量属性和要素

### 13.1.3 全面质量管理

## 13.2 软件变更管理

# 全面质量管理

## ■ 郎中治病的故事

- 质量的死对头是缺陷（defect, bug...），缺陷是混在产品中的人们不喜欢、不想要的东西，它对产品没有好处只有坏处。缺陷越多质量越低，缺陷越少质量越高，提高软件质量的基本手段是消除软件缺陷。

- 中国郎中看病的故事

在中国古代，有一家三兄弟全是郎中。其中老三是名医，人们问他：“你们兄弟三人谁的医术最高？”

他回答说：“我常用猛药给病危者医治，偶尔有些病危者被我救活，于是我的医术远近闻名并成了名医。我二哥通常在人们刚刚生病的时候马上就治愈他们，临近村庄的人说他是好郎中。我大哥不外出治病，他深知人们生病的原因，所以能够预防家里人生病，他的医术只有我们家里才知道。”

- 郎中三兄弟是三种治病方式的代言人。

# 全面质量管理

## ■ 消除软件缺陷的三种方式

- 老大治病的方式最高明，如果人们能够预防生病的话，那么没病就用不着看医生了。
  - 提高软件质量最好的办法是：**在开发过程中有效地防止工作成果产生缺陷，将高质量内建于开发过程之中。主要措施是“不断地提高技术水平，不断地提高规范化水平”，其实就是练内功，通称为“软件过程改进”。**
- 即使一个人严守养生之道，身体状况良好，但总是会意外地得病的，得了病就要去看医生。老二治病的方式就是医院的模式，病人越早看病，就越早治好，治病的代价就越低。
  - 同理，在开发软件的时候，即使人们的技术水平很高，并且严格遵守规范，但是人非机器，总是会犯错误的，因此无法完全避免软件中的缺陷。
  - **当工作成果刚刚产生时马上进行质量检查，及时找出并消除工作成果中的缺陷。这种方式效果比较好，人们一般都能学会。最常用的方法是技术评审、软件测试和过程检查，已经被企业广泛采用并取得了成效。**
- 老三治病的方式代价最高，只能是不得已而为之。可在现实之中，大多数软件企业采用老三的方式来对付质量问题。典型现象是：在软件交付之前，没有及时消除缺陷。当软件交付给用户后，用着用着就出错了，赶紧请开发者来补救。可笑的是，当软件系统在用户那里出故障了，那些现场补救成功的人倒成了英雄，好心用户甚至还寄来感谢信。

# 最佳的软件开发实践（IBM）

在商业运作中已经证明这是一种能够解决软件开发过程中跟本问题的方法：

- 迭代开发
- 需求的管理
- 应用基于构件的架构
- 可视化软件建模
- 持续质量验证
- 控制软件变更



# 1.全面质量管理

- 事先预防的思想，“缺陷越早发现越早修改越经济”的原则
- 重视质量保证（**SQA**）工作，检查开发和管理活动是否与已定的过程策略、标准和流程一致，检查工作产品是否遵循模板规定的内容和格式

## 2.软件质量保证活动

- 为项目准备**SQA**计划
- 参与开发项目的软件过程描述
- 评审各项软件工程活动，以验证其是否符合定义的软件过程
- 审核指定的软件工作产品，以验证其是否符合定义的软件过程中的相应部分
- 确保软件工作及工作产品中出现的偏差已文档化，并且按照文档化的规程进行了处理
- 记录所有不符合的部分，并报告给高层管理者

## 2.全面软件质量管理：质量管理计划

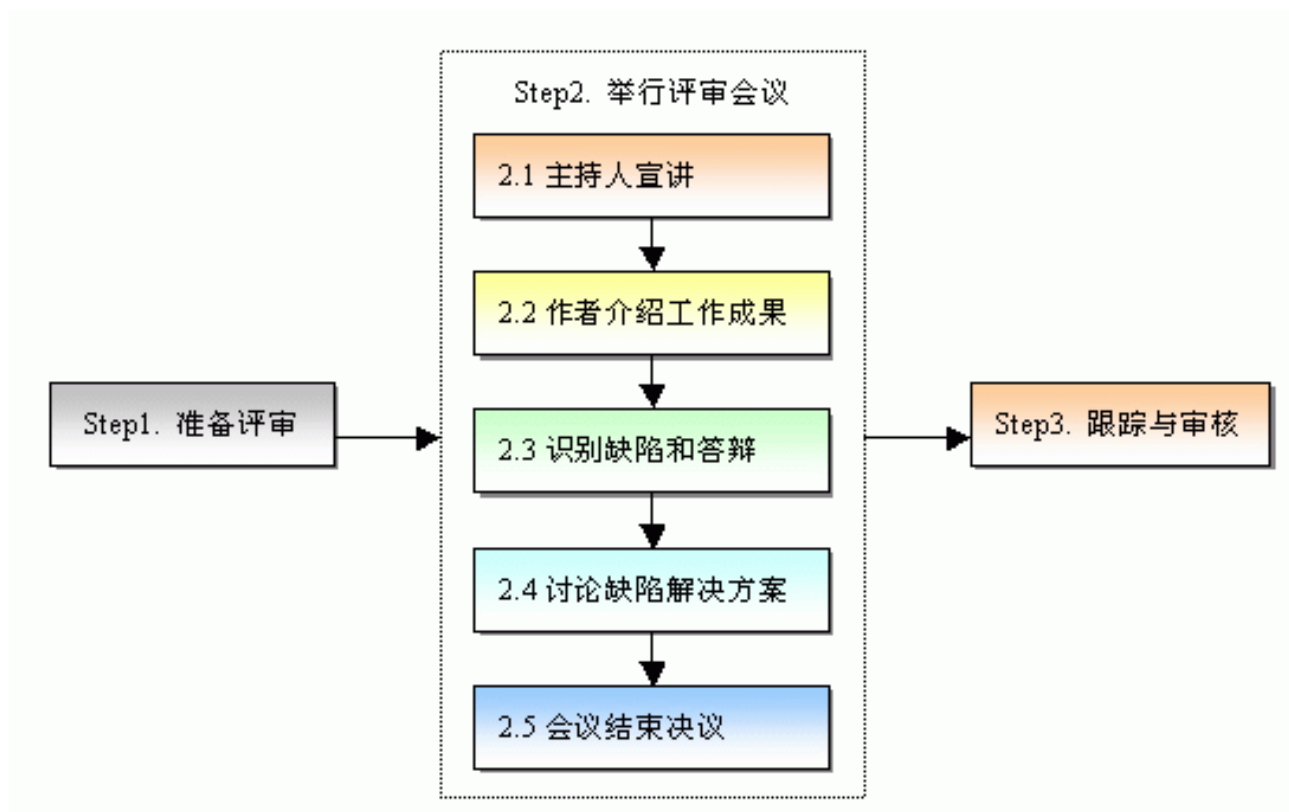
- 质量管理计划就是为了实现质量目标的计划。而质量目标则是由商业目标决定的。开发软件产品的最终目的是为了赚钱，所以人们为提高软件质量所付出的代价是有上限的，项目负责人当然希望代价越低越好。**质量管理计划是全面质量管理的行动纲领。**
- 谁制定质量管理计划？由项目核心成员和质量人员共同协商制定，主要由质量人员起草，由项目经理审批即可。
- 质量管理计划的主要内容：
  - 1. 质量要素分析
  - 2. 质量目标
  - 3. 人员与职责
  - 4. 过程检查计划
  - 5. 技术评审计划
  - 6. 软件测试计划
  - 7. 缺陷跟踪工具
  - 8. 审批意见

### 3.全面软件质量管理：软件评审

- 软件评审时软件过程中的“过滤器”
- 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，从而有效地提高产品的质量
- 软件评审包括“正式技术评审”、“走查”、“审查”、“轮查”等
  - 发现软件的任何一种表示形式中的功能、逻辑或实现上的错误
  - 验证评审中的软件是否满足其需求
  - 保证软件的表示符合预先定义的标准
  - 得到以统一的方式开发的软件
  - 使项目更易于管理
- 技术评审的主要好处有：
  - 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执行，不必等到软件可以运行之际，越早消除缺陷就越能降低开发成本；
  - 开发人员能够及时地得到同行专家的帮助和指导，无疑会加深对工作成果的理解，更好地预防缺陷，一定程度上提高了开发生产率。

### 3. 全面软件质量管理：技术评审

- 正式技术评审的流程



### 3. 软件评审与测试区别

#### ■ 观点

- 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成果摆放在桌面上讨论；
  - 而后者一定要运行软件来查找缺陷。技术评审在软件测试之前执行，尤其是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比技术评审的大，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，只有这样他才能深入地了解软件的质量问题，而且给予开发小组强有力地帮助。

## 4. 全面软件质量管理：质量保证，过程检查

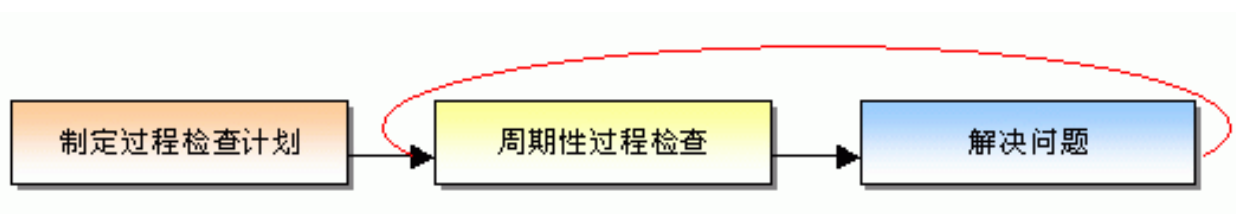
### ■ 观点

- CMM和ISO9001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
  - 再例如，机构制定了重要工作成果的文档模板（例如需求规格说明书、设计报告等），要求开发人员写的文档尽可能符合模板。如果质量人员发现开发人员写的文档与机构的模板差异非常大，那么就要搞清楚究竟是模板不合适？还是开发人员偷工减料？
- 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。
  - 不少机构的质量人员并没有真正理解过程检查的意义，老是对照规范，查找错别字、标点符号、排版格式等问题，迷失了方向，这样只有疲劳没有功劳，而且让开发人员很厌烦。
  - 对于中小型项目而言，过程检查工作由质量人员一个人负责就够了，约占其20%的工作量，让质量人员抽出更多的时间从事技术评审和软件测试工作。

## 4. 全面软件质量管理：过程检查

### ■ 流程

- 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。





## 5. 全面软件质量管理：缺陷跟踪工具

### ■ 概念

- 如果没有缺陷跟踪工具的话，人们只好用纸张或文件去记录缺陷，不仅变更缺陷信息很麻烦，而且难以共享信息。缺陷跟踪工具就是帮助项目成员记录和跟踪缺陷用的，一般都有数据库支持，可以在局域网内运行。
- Internet上有许多缺陷跟踪工具，大家可以免费下载使用。由于缺陷跟踪工具仅仅是一种辅助性的工具，我们没有必要太在乎该软件的功能，只要用起来方便就行。
- 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。
- 缺陷跟踪工具的常见功能：查询缺陷, 添加缺陷, 修改缺陷, 删除, 缺陷分类图, 缺陷趋势图, Email

## 6. 全面软件质量管理： 人员

### ■ 角色职责

- 谁对软件质量负责？是全员负责。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出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的影响，所以也不负最大的责任。
  - 质量人员的主要职责：
    - （1）负责制定质量计划（很重要但是工作量比较少）；
    - （2）负责过程检查（类似于CMM中的质量保证），约占个人工作量的20%；
    - （3）参与技术评审，约占个人工作量的30%；
    - （4）参与软件测试，约占个人工作量的30%；
    - （5）参与软件过程改进（面向整个机构），约占个人工作量的20%；
- \* 上述工作量的比例仅供参考，在实际应用时必须根据项目的特征而定。

# 讨论：有了质量保证是否能够保证软件质量？

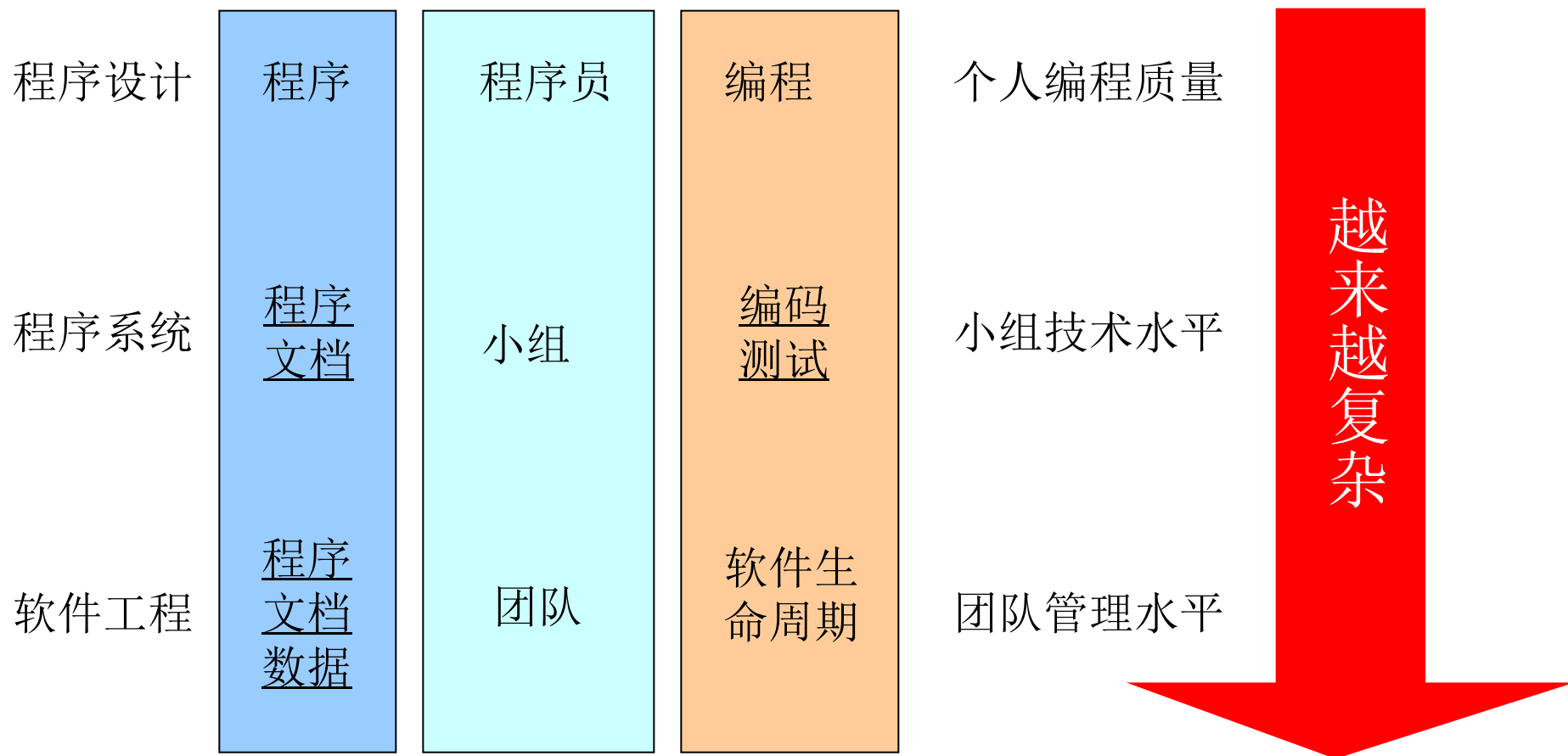
- **CMM、ISO9000**质量规范及标准
- 软件项目的特点？

# 主要内容

**13.1 软件质量管理**

**13.2 软件变更管理**

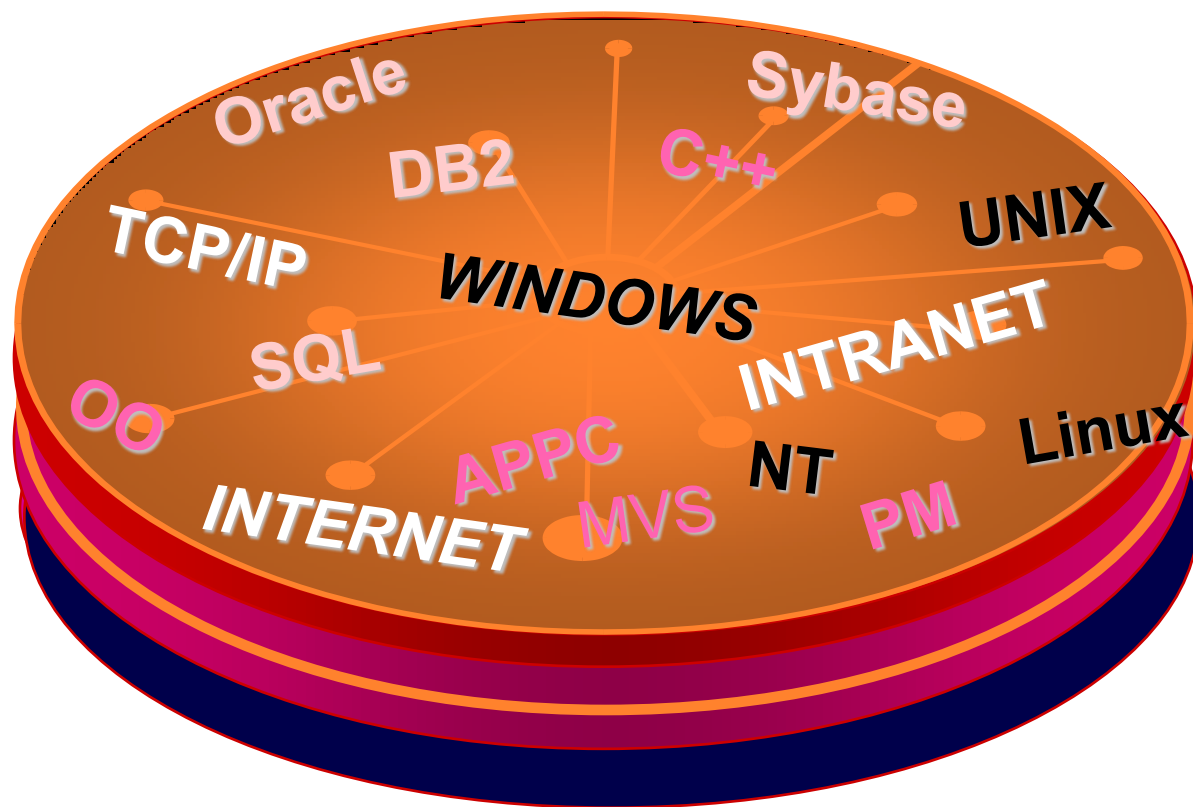
# 1 软件变更的背景



# 软件开发过程中面临的困境

- 缺乏对用户需求进行有效的管理和追踪的工具
- 产品升级和维护所必需的程序和文档非常混乱
- 代码可重用性差从而不能对产品进行功能扩充
- 开发过程中的人员流动经常发生
- 软件开发人员之间缺乏必要的交流
- 由于管理不善致使未经测试的软件加入到产品中
- 用户与开发商没有有效的产品交接界面

# 开发环境的复杂性



多操作系统  
多开发工具  
网络化  
团队方式  
异地开发

# 缺乏管理所造成的问题

软件生产达不到规模化

缺少有效的通信机制



成员间缺少沟通

人员流动



## 2.软件变更管理

- 变更管理，通常叫做软件配置管理（**SCM**），是贯穿于整个软件过程的活动
- 软件配置(**software configuration**): 由在软件工程过程中产生的所有信息项构成，它可以看作该软件的具体形态(软件配置项)在某一时刻的瞬间影像。



## 2. 软件变更管理

- “配置管理能够系统的处理变更，从而使得软件系统可以随时保持其完整性，因此称为‘变更控制’，可以用来评估提出的变更请求，跟踪变更，并保存系统在不同时间的状态。”

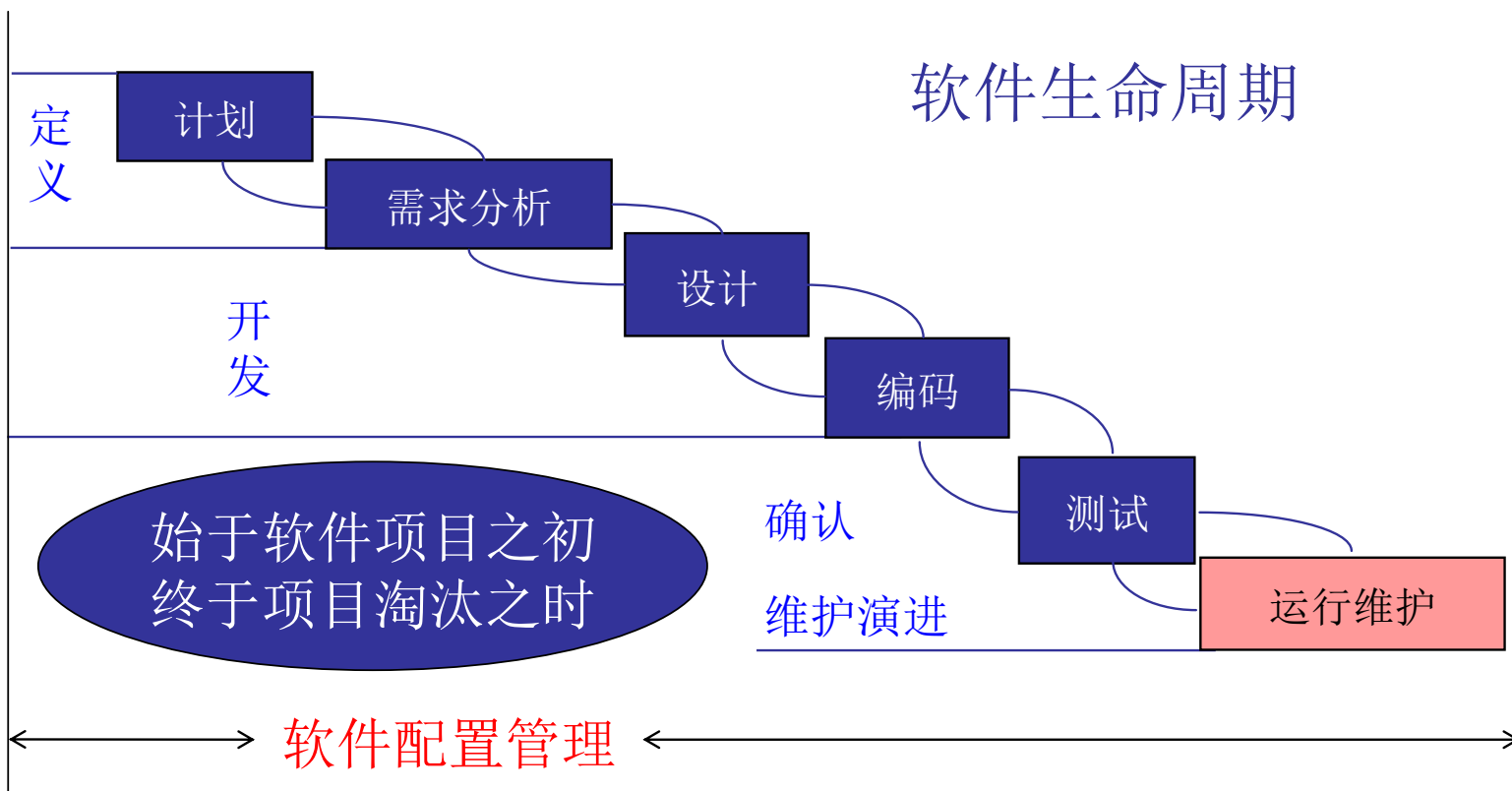
——Steve McConnell

《代码大全》

- 目标
  - 标识变更
  - 控制变更
  - 保证正确地实现变更
  - 向那些利害关系人员报告变更

# 软件配置管理的特点

- **SCM**贯穿整个软件生命周期与软件工程过程



### 3 SCM的角色与职责

- **项目经理(Project Manager, PM)**

- 制定项目的组织结构和配置管理策略;
- 批准、发布配置管理计划;
- 决定项目起始基线和软件开发工作里程碑;
- 接受并审阅配置控制委员会的报告。

- **配置控制委员会(Configuration Control Board, CCB)**

- 批准配置项的标志, 以及软件基线的建立;
- 制定访问控制策略;
- 建立、更改基线的设置, 审核变更申请;
- 根据配置管理员的报告决定相应的对策。

# SCM的角色与职责

## ■ 配置管理员(Configuration Management Officer, CMO)

- 软件配置管理工具的日常管理与维护;
- 提交配置管理计划;
- 各配置项的管理与维护;
- 执行版本控制和变更控制方案;
- 完成配置审计并提交报告;
- 对开发人员进行相关的培训;
- 识别开发过程中存在的问题并制定解决方案

# SCM的角色与职责

- **系统集成成员(System Integration Officer, SIO)**

- 负责生成和管理项目的内部和外部发布版本，包括集成修改、构建系统、完成对版本的日常维护、建立外部发布版本等

- **开发人员(Developer, DEV)**

- 根据项目组织确定的配置管理计划和相关规定，按照配置管理工具的使用模型来完成开发任务

## 4 SCM的基本元素

- 配置项(**Configuration Item, CI**)
- 基线(**Baseline**)
- 配置管理数据库(**CMDB**)
- 备件库(**Definitive Hardware Store, DHS**)
- 最终软件库(**Definitive Software Library, DSL**)

# (1) 配置项

- 软件过程的输出信息可以分为三个主要类别：

- 计算机程序(源代码和可执行程序)
- 描述计算机程序的文档(针对技术开发者和用户)
- 数据(包含在程序内部或外部)

这些项包含了所有在软件过程中产生的信息，总称为软件配置项(**SCI**)。

- **SCI**是软件全生命周期内受管理和控制的基本单位，大到整个系统，小到某个硬件设备或软件模块。

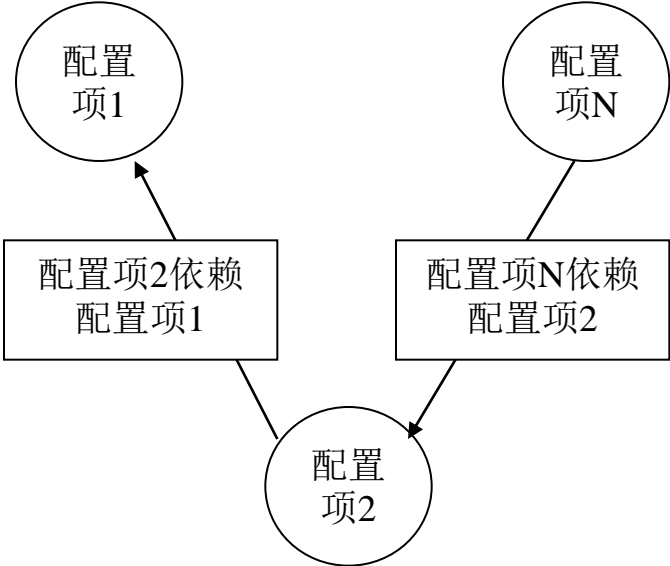


# 配置项

- **SCI**具有唯一的名称标识和多个属性。

- 名称
- 描述
- 类型(模型元素、程序、数据、文档等)
- 项目标识符
- 资源表
- “实现”
- 变更和版本信息

# 配置项之间的依赖关系

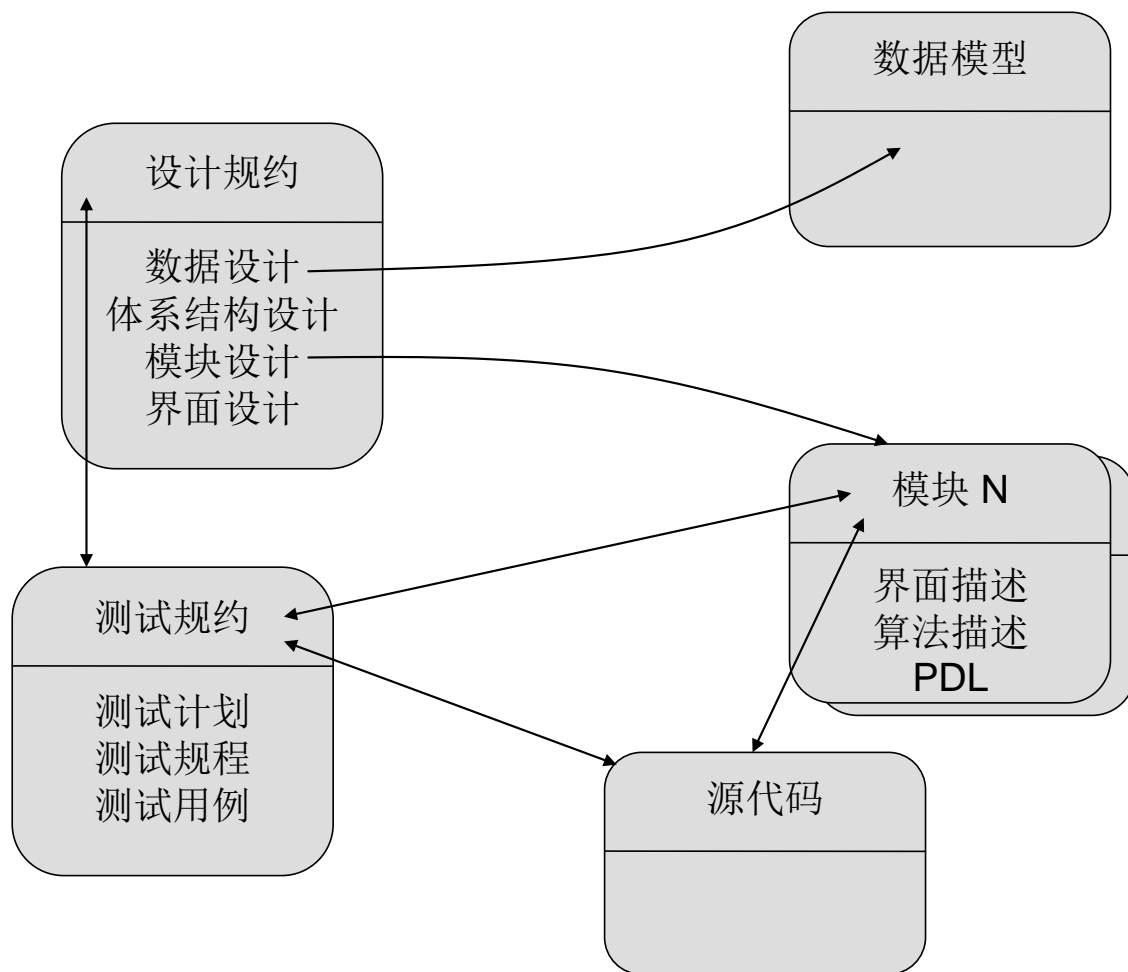


依赖关系	配置项1	配置项2	...	配置项N
配置项1		X		
配置项2				X
...				
配置项N				

# [课堂讨论]配置项之间的关系

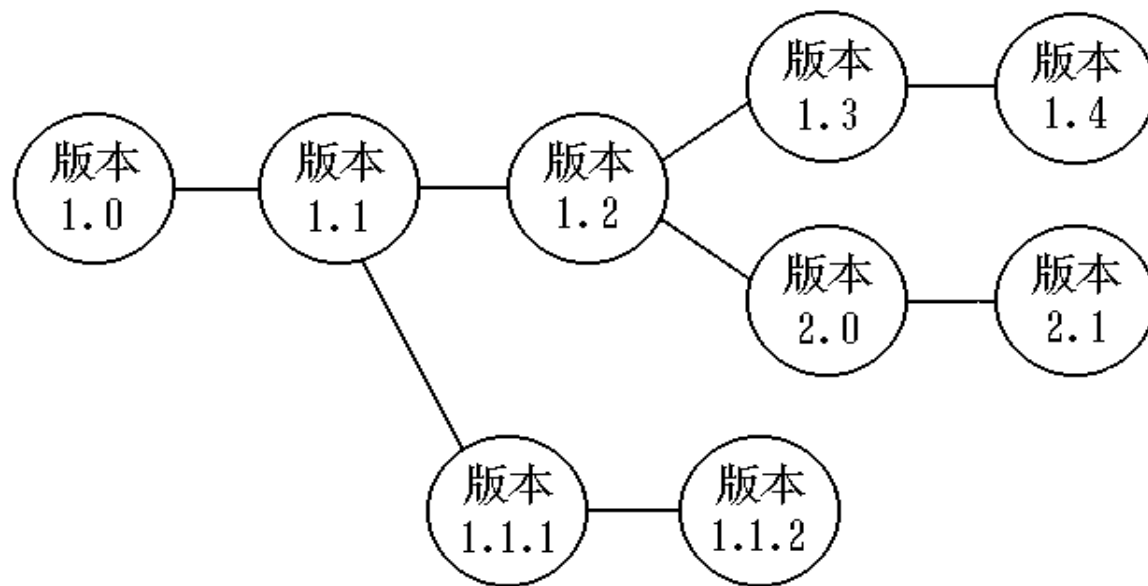
- 配置项之间都可能有哪些类型的依赖关系？
  - 整体-部分关系
    - Data flow diagram (DFD) <part-of> analysis model
    - Analysis model <part-of> requirement specification
  - 关联关系
    - Data model <interrelated> data flow diagram (DFD)
    - Data model <interrelated> test case class m;
  - 还有哪些？

# 配置项之间的依赖关系



# 配置项的演变图

- 在对象成为基线以前可能要做多次变更，在成为基线之后也可能需要频繁的变更。
- 对于每一配置项都要建立一个演变图，以记录对象的变更历史。

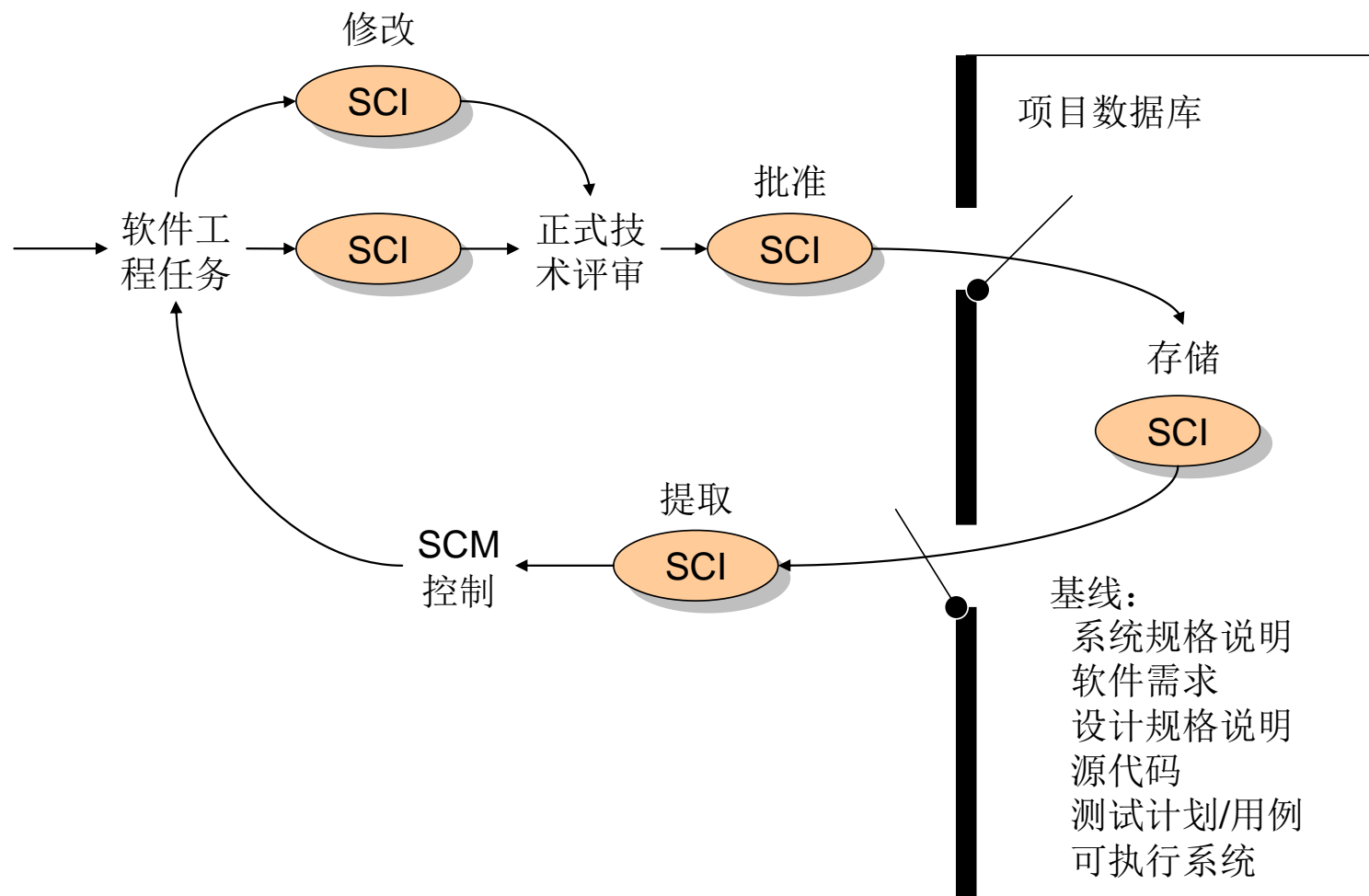


## (2) 基线

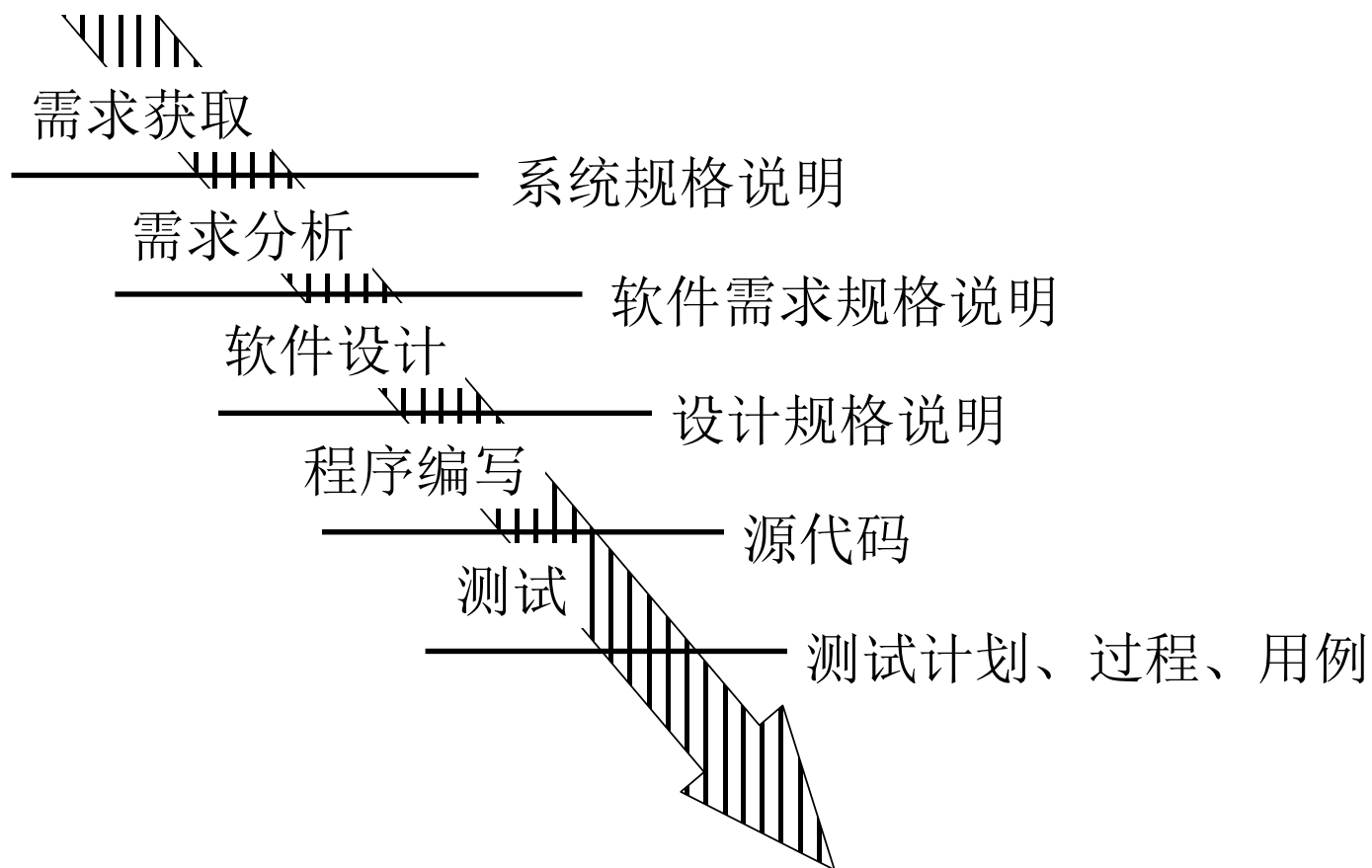
- 基线：已经通过正式评审和批准的软件规格说明或代码，它可以作为进一步开发的基础，并且只有通过正式的变更规程才能修改它。
- 在软件配置项成为基线之前，可以迅速而随意的进行变更；
- 一旦成为基线，变更时需要遵循正式的评审流程才可以变更。
- 因此，基线可看作是软件开发过程中的“里程碑”。



# 基线



# 软件开发各阶段的基线



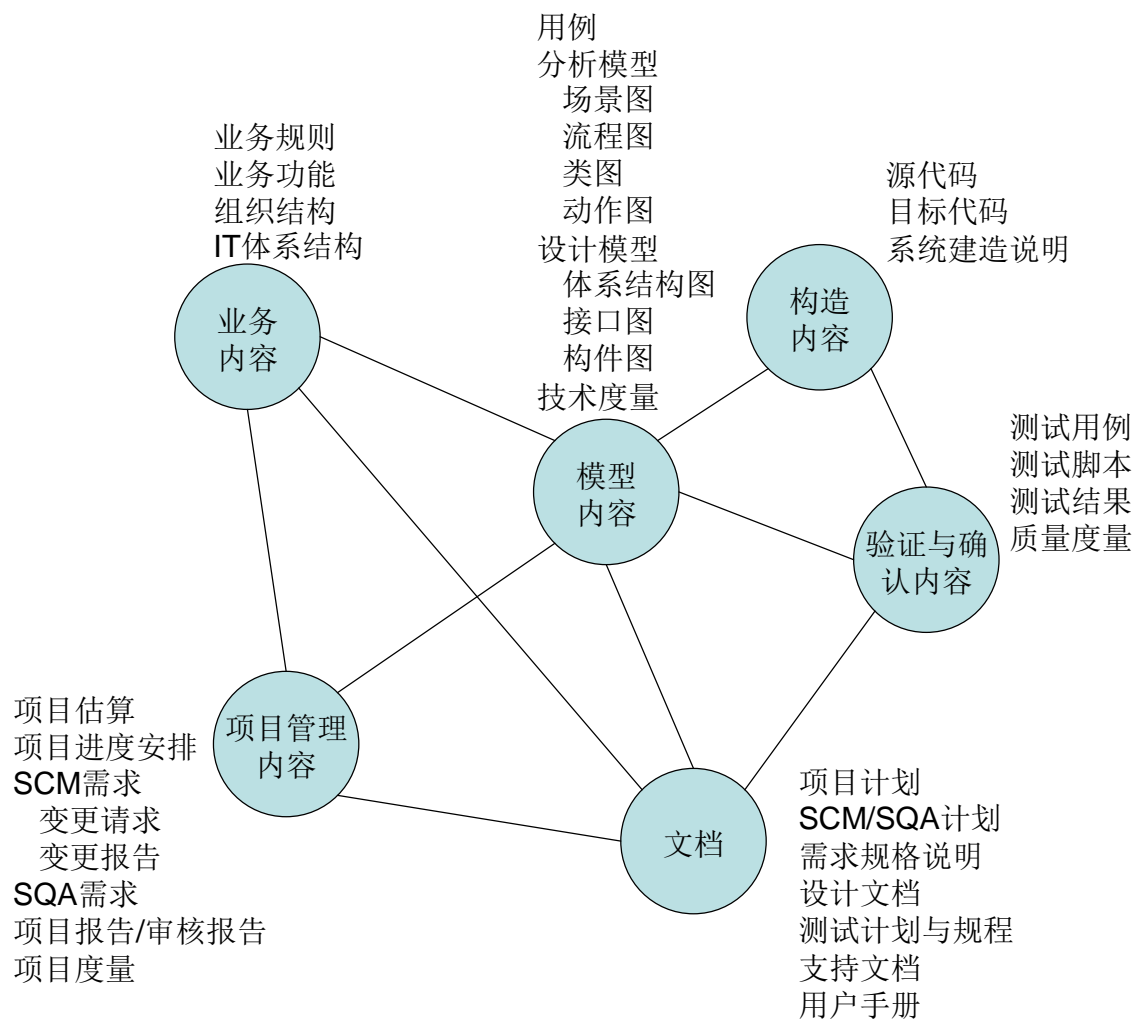


### (3) 配置管理数据库

- 配置管理数据库(CMDB)(也称“SCM中心存储库”), 用于保存于软件相关的所有配置项的信息以及配置项之间关系的数据库。
  - 每个配置项及其版本号
  - 变更可能会影响到的配置项
  - 配置项的变更路线及轨迹
  - 与配置项有关的变更内容
  - 计划升级、替换或弃用的配置项
  - 不同配置项之间的关系



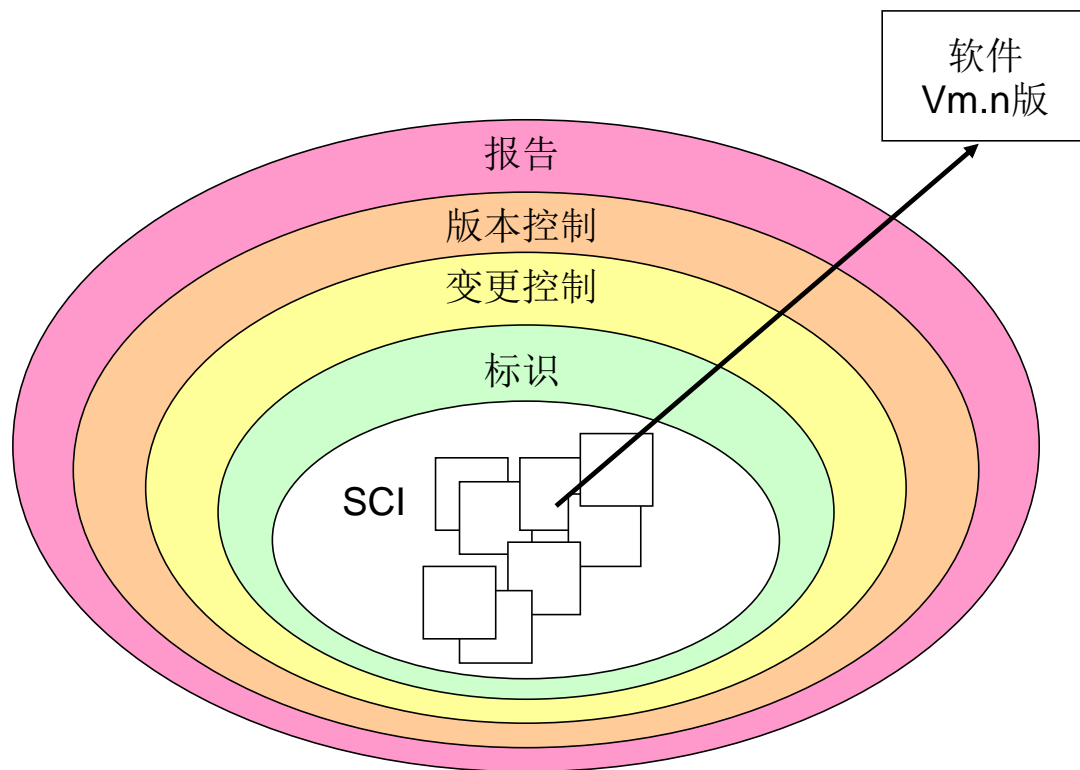
# 配置管理数据库



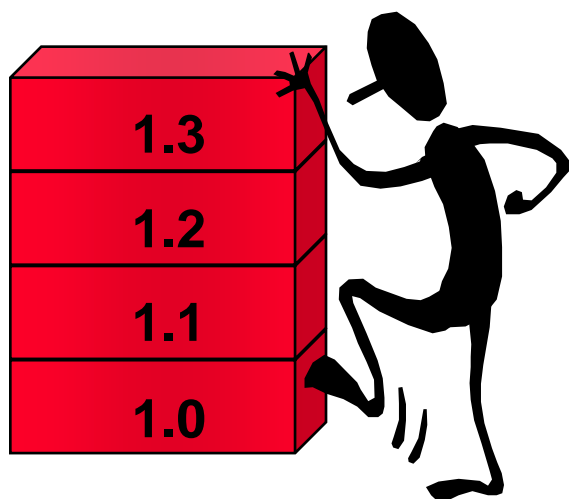
# 配置管理数据库

## ■ CMDB的功能:

- 存储配置项及其之间的关系
- 版本控制
- 相关性跟踪和变更管理
- 需求跟踪
- 配置管理
- 审核跟踪



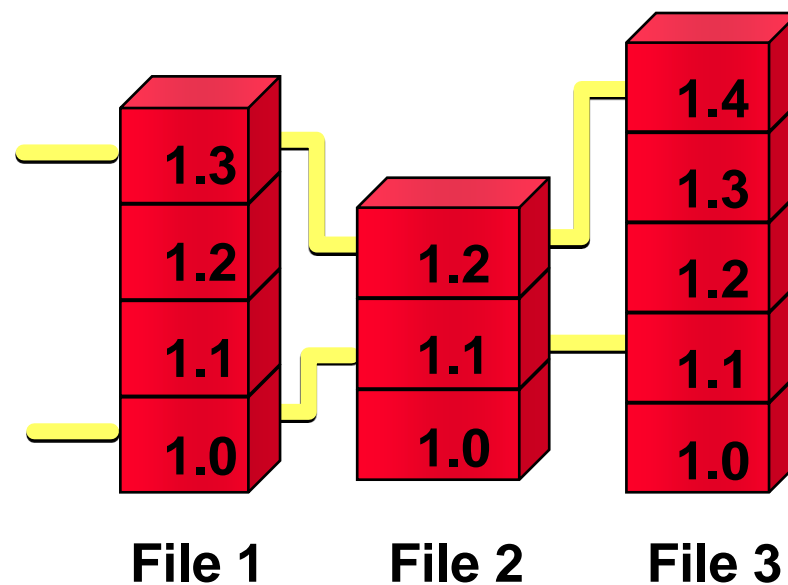
# 强调：版本控制



正式版

Beta 1

Version Labels




## 5 SCM常用工具

- IBM Rational ClearCase
- Microsoft Visual SourceSafe
- CVS (Concurrent Version System)
- **SVN(Subversion)**

# SCM总结

- 软件开发者的典型心理：“嫌麻烦”、“侥幸”
- 而一旦开发过程中出现变更，就会造成更大的麻烦。
- 任何项目成员都要对其工作成果进行配置管理，应当养成良好的习惯。
- 不必付出过多的精力，最低要求是保证重要工作成果不发生混乱。



结束

**2011年6月6日**