

第二讲 软件开发中的质量保证

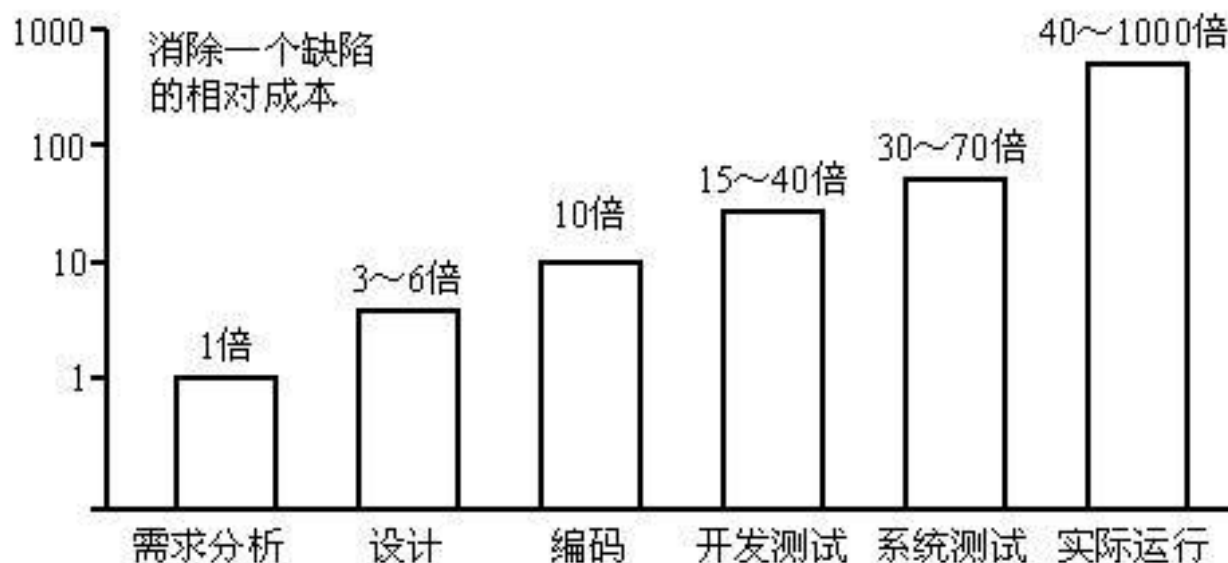
李宁

2019.3

1. 软件评审
2. 高质量的软件需求分析
3. 提高软件设计质量
4. 高质量编程
5. 软件测试之质量
6. 全面质量管理

1. 软件评审 - Why

缺陷发现得越晚纠正费用越高，而软件评审的重要目的就是**通过软件评审尽早的产品中的缺陷，减少大量的后期返工。**



1. 软件评审 - 概述

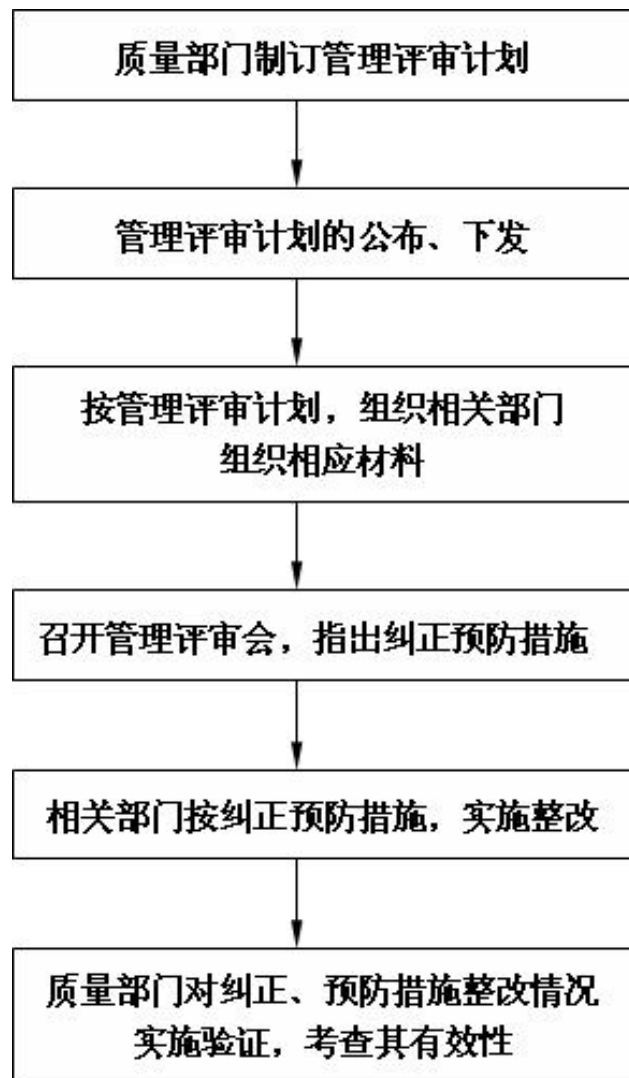
■ 软件评审的角色和职能

- 协调人
- 作者
- 评审员
- 用户代表
- 质量保证代表

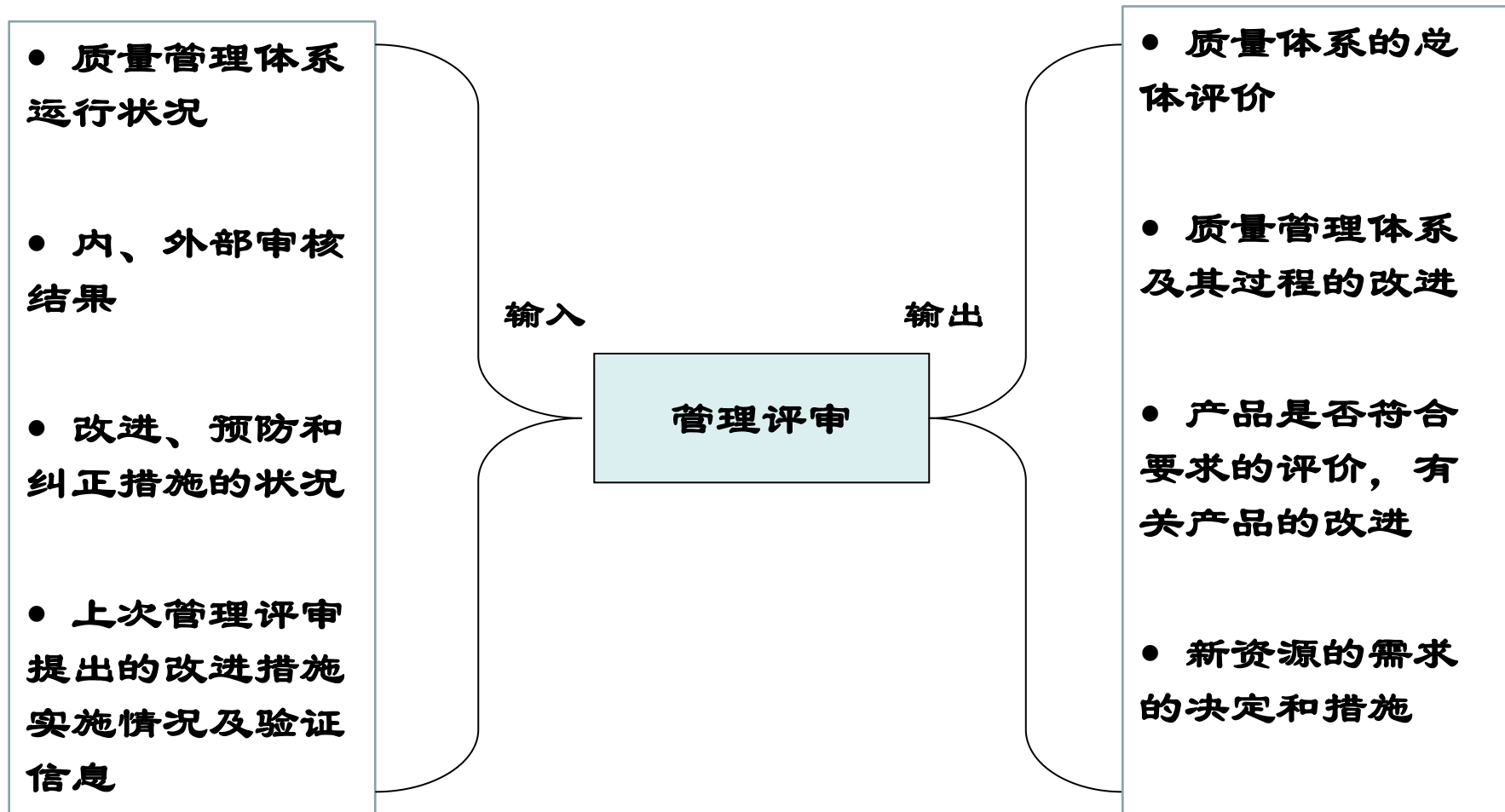
■ 评审内容

- 管理评审
- 技术评审
- 文档评审
- 过程评审

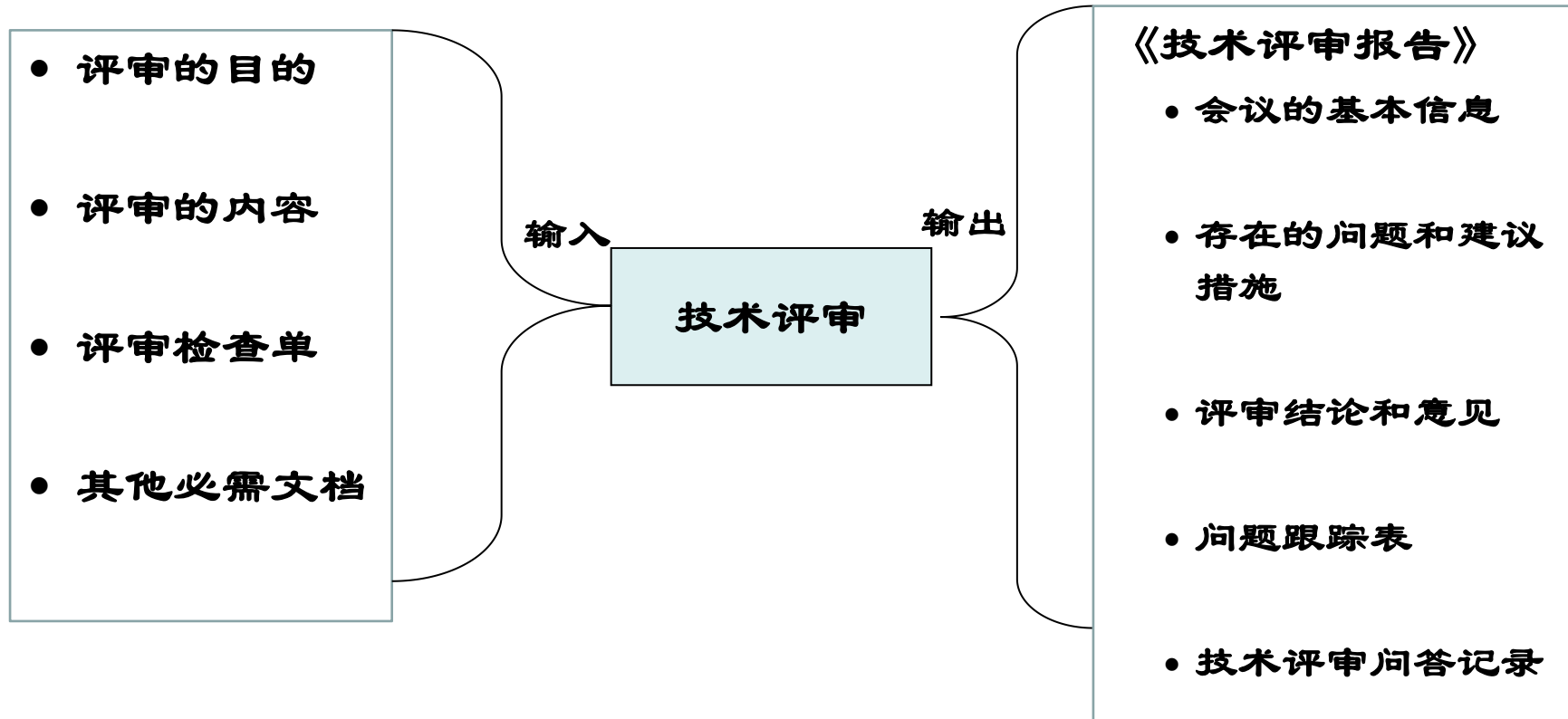
1. 软件评审 - 管理评审



1. 软件评审 - 管理评审



1. 软件评审 - 技术评审



1. 软件评审 - 文档评审

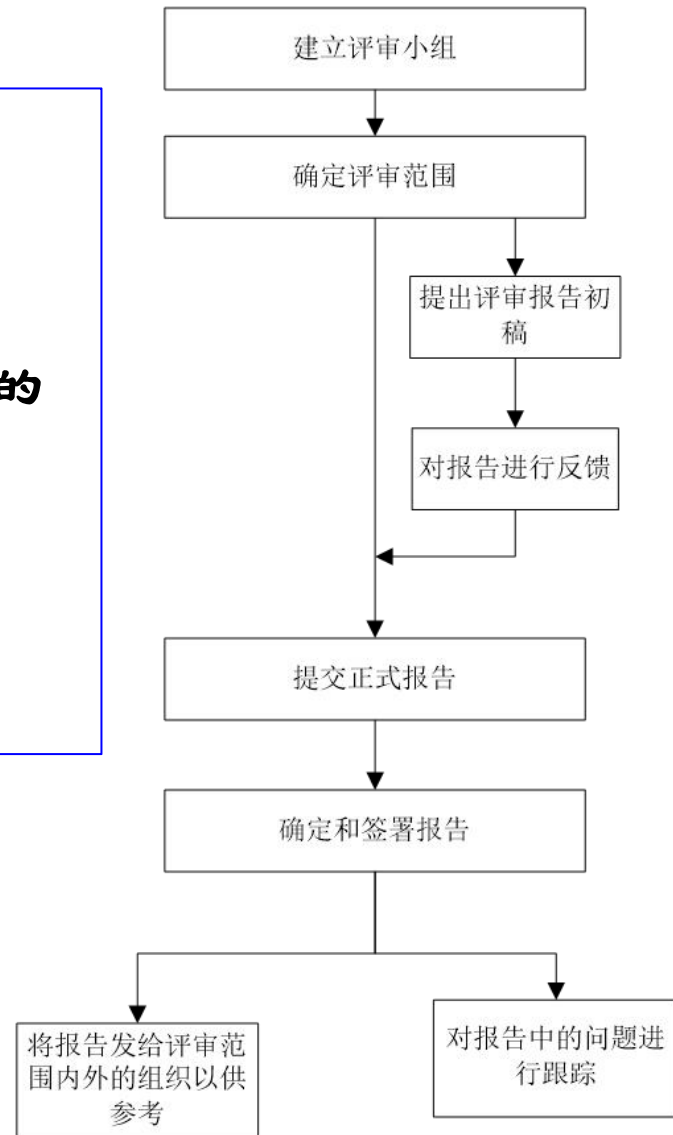
■ 文档评审要点

- 1) 正确性
- 2) 完整性
- 3) 一致性
- 4) 有效性
- 5) 易测性
- 6) 模块化-系统和文档描述必须深入到模块
- 7) 清晰性
- 8) 可行性
- 9) 可靠性
- 10) 可追溯性

1. 软件评审 - 过程评审

过程评审的目的：

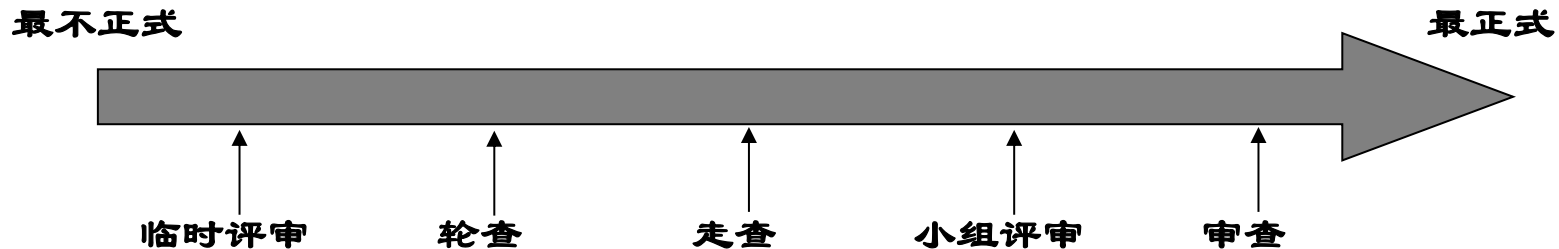
- ❖ 评估主要的质量保证流程
- ❖ 考虑如何处理/解决评审过程中发现的不符合问题
- ❖ 总结和共享好的经验
- ❖ 指出需要进一步完善和改进的地方



1. 软件评审-评审方法和技术

- 临时评审 (Ad hoc review)
- 轮查 (Pass-round)
- 走查 (Walkthrough)
- 小组评审 (Group Review)
- 审查 (Inspection)

“对于最可能产生风险的工作成果，要采用最正式的评审方法。”



1. 软件评审-评审方法和技术

审查、小组评审和走查异同点比较表

角色/职责	审查	小组评审	走查
主持者	评审组长	评审组长或作者	作者
材料陈述者	评审者	评审组长	作者
记录员	是	是	可能
专门的评审角色	是	是	否
检查表	是	是	否
问题跟踪和分析	是	可能	否
产品评估	是	是	否

评审方法	计划	准备	会议	修正	确认
审查	有	有	有	有	有
小组评审	有	有	有	有	有
走查	是	无	有	有	无

- **缺陷检查表**

它列出了容易出现的**典型错误**，是评审的一个重要组成部分。

- **规则集**

类似于缺陷检查表，通常是业界通用的规范或者企业自定义的各种规则的集合。

- **评审工具的使用**

合理的利用工具，如NASA开发的ARM（自动需求度量）

- **从不同角色理解**

不同的角色对产品/文档的理解是不一样的。

- **场景**

按照用户使用场景对产品/文档进行评审。

1. 软件评审-主要评审点

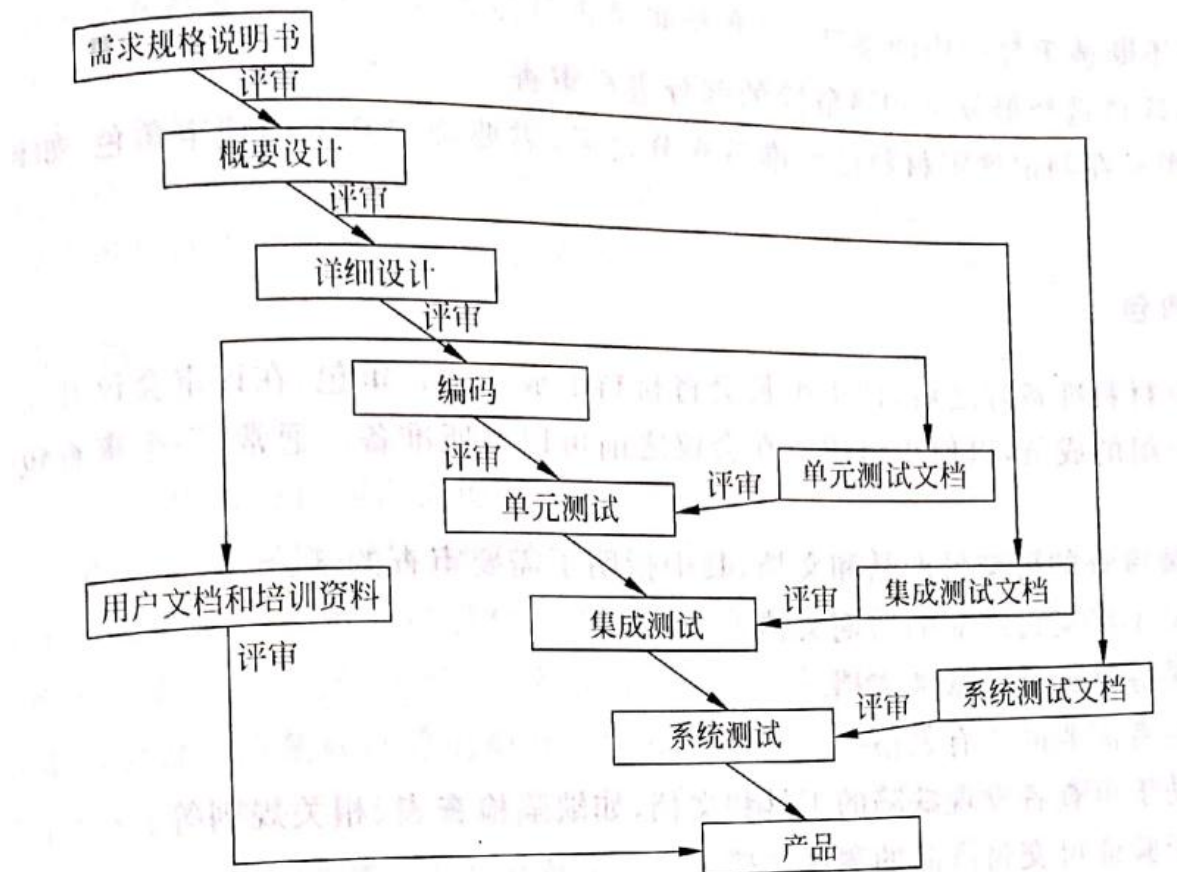


图 9-4 主要评审检查点

1. 软件评审 – 评审会议

- **会议准备**

时间安排、地点、组长以及成员，评审材料，参考材料

- **会议召开过程**

- 1) 由评审员/作者进行演示或说明。
- 2) 评审员会就不清楚或疑惑的地方与作者进行沟通。
- 3) 协调人或记录员在会议过程中完成会议记录。

- **评审决议**

接受、有条件接受、不能接受、未完成

- **评审结束**

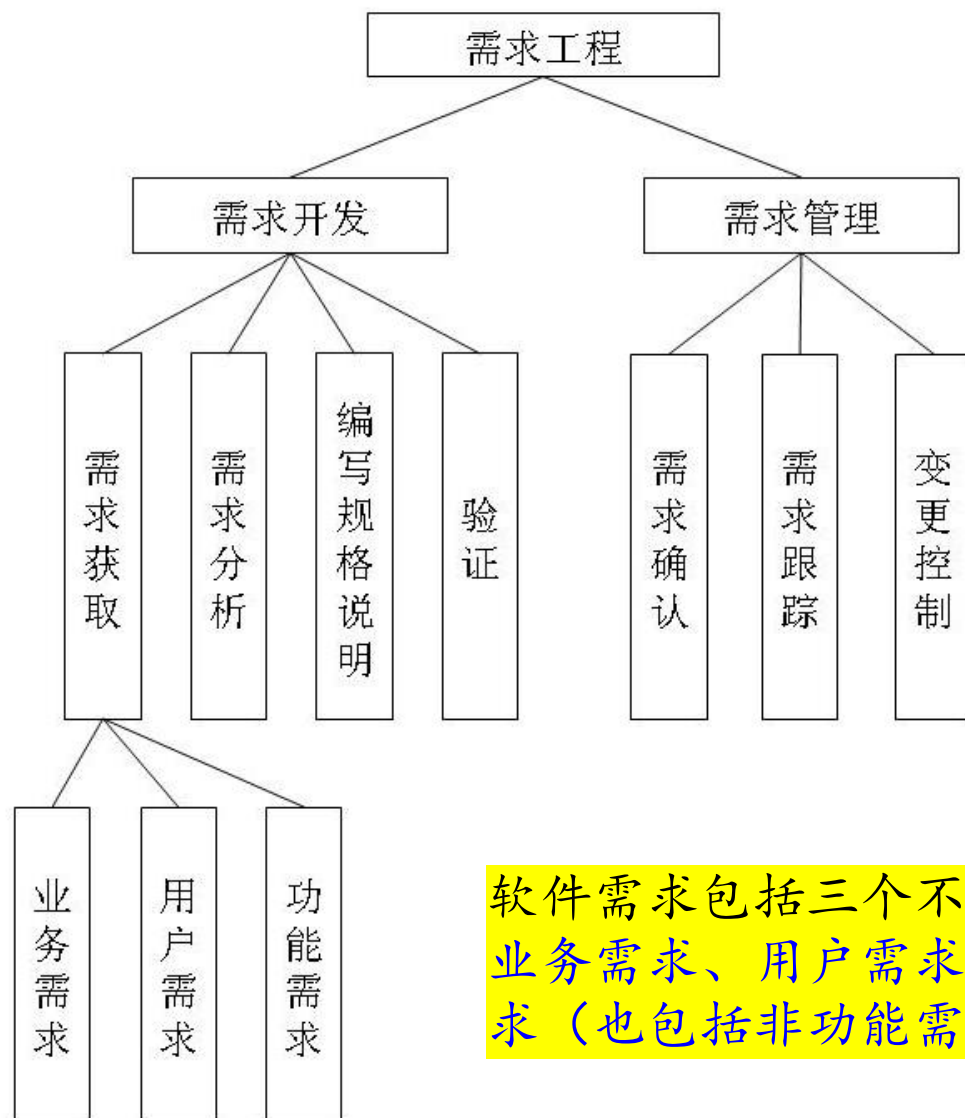
问题列表、会议记录、决议，签名表等。

1. 软件评审 – 评审会议

- **Peer review example (coding review)**
[video](#)
- **Github code review example**
[video](#)

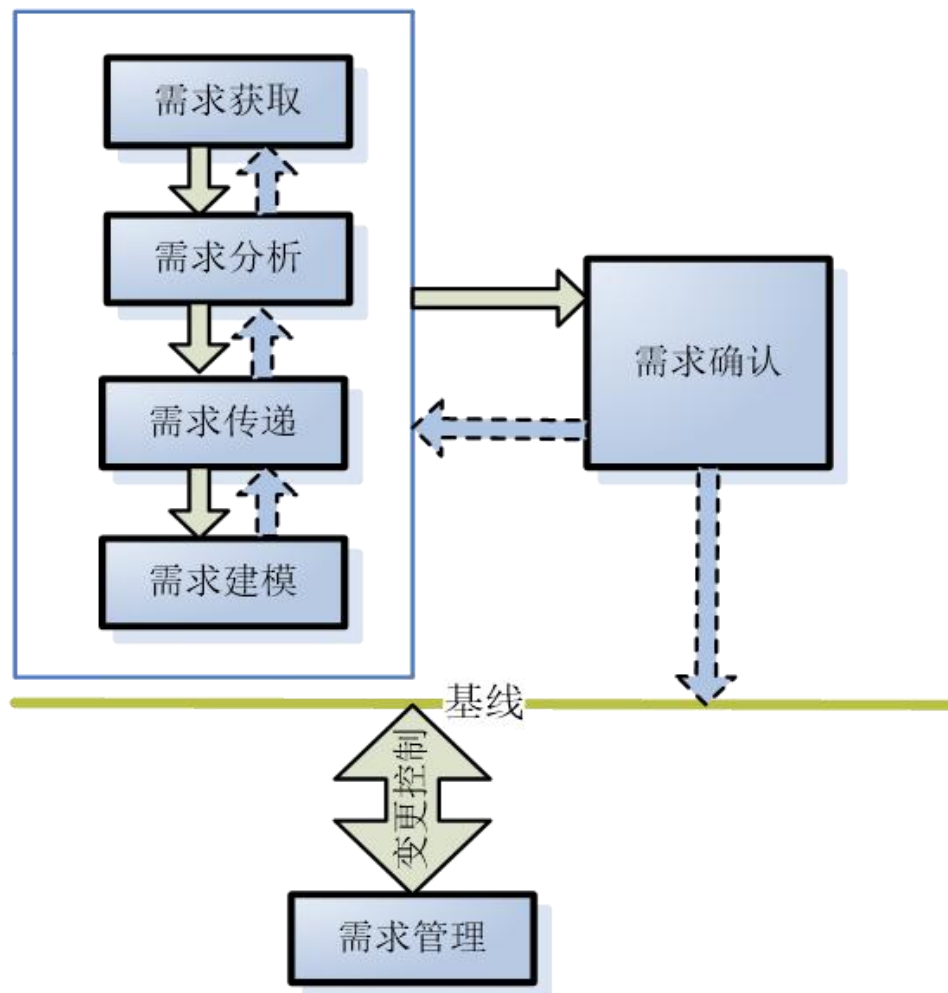
1. 软件评审
2. 高质量的软件需求分析
3. 提高软件设计质量
4. 高质量编程
5. 软件测试之质量
6. 全面质量管理

2. 高质量的软件需求分析



所有与需求直接相关的活动统称为需求工程，需求工程分为了两个部分：**需求开发**和**需求管理**。其中，需求开发又分为了需求获取、需求分析、需求定义和需求验证4个部分，而需求管理则包含了变更控制、版本控制、需求跟踪和需求状态跟踪

软件需求包括三个不同的层次：**业务需求**、**用户需求**和**功能需求**（也包括非功能需求）。

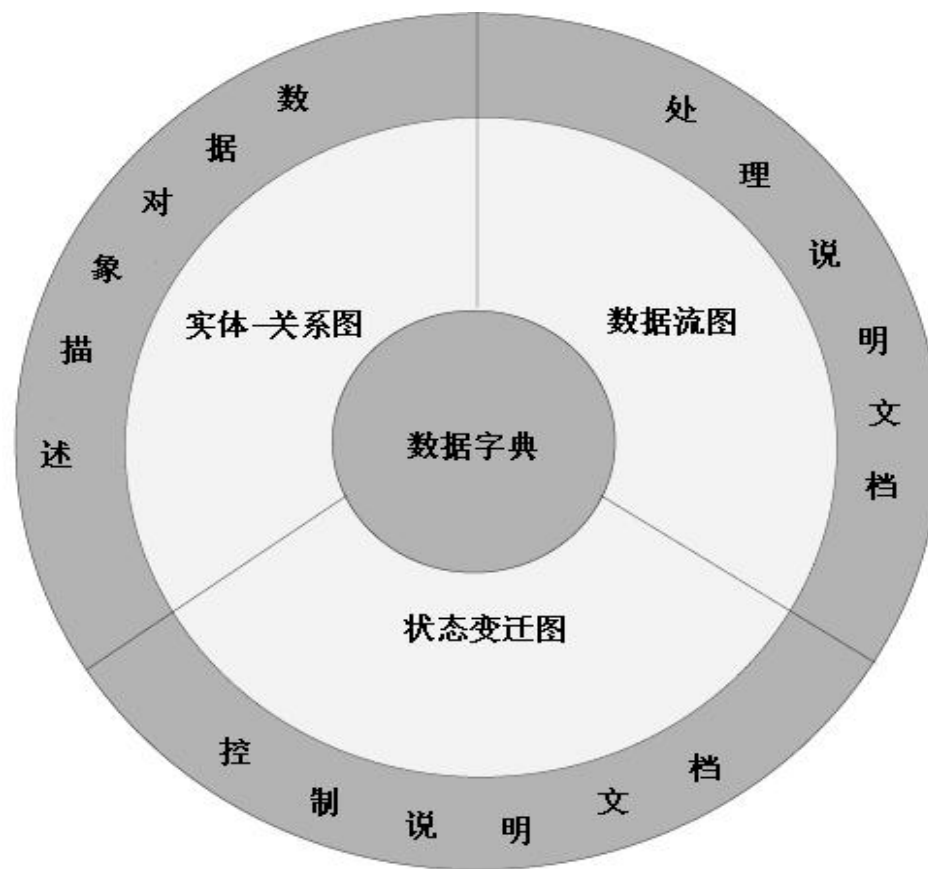


2. 软件需求分析方法

- 需求研讨会
- 头脑风暴
- 用例模型
- 访谈
- 角色扮演
- 原型法



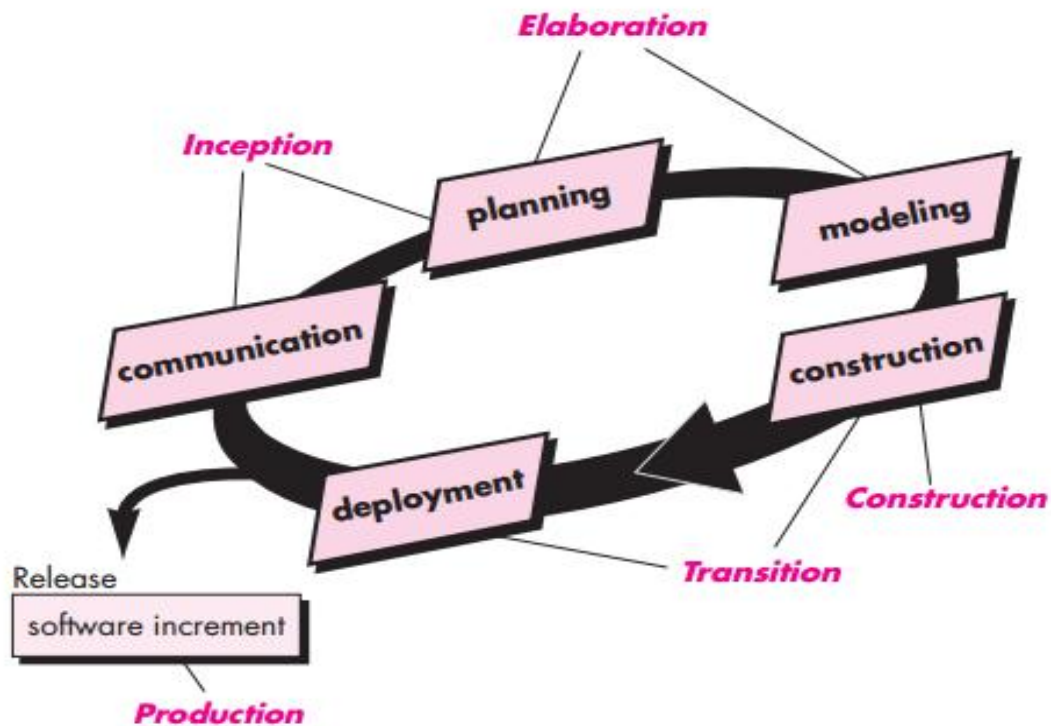
➤ 结构化分析建模



2. 软件需求分析建模

➤ 面向对象分析建模

✓ UP (Unified Process)

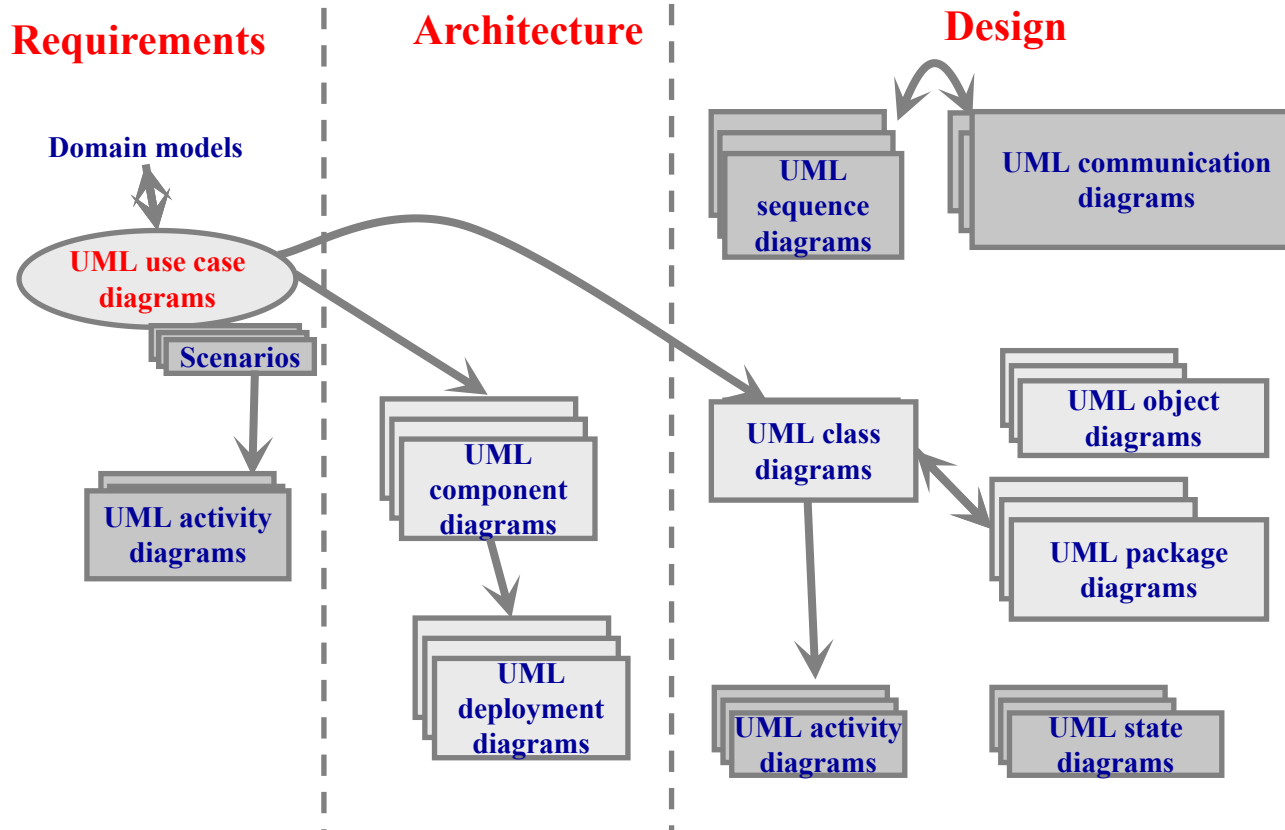
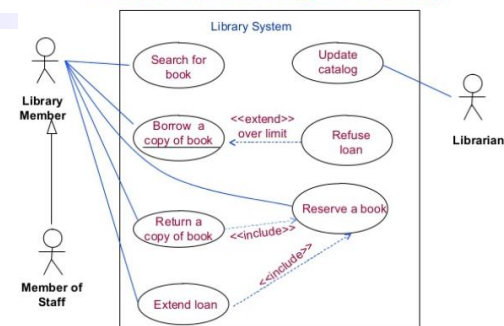


2. 软件需求分析建模

➤ 面向对象分析建模

✓ UML (Unified Modeling Language)

A Simple Use Case Diagram of Library



2. 软件需求分析建模

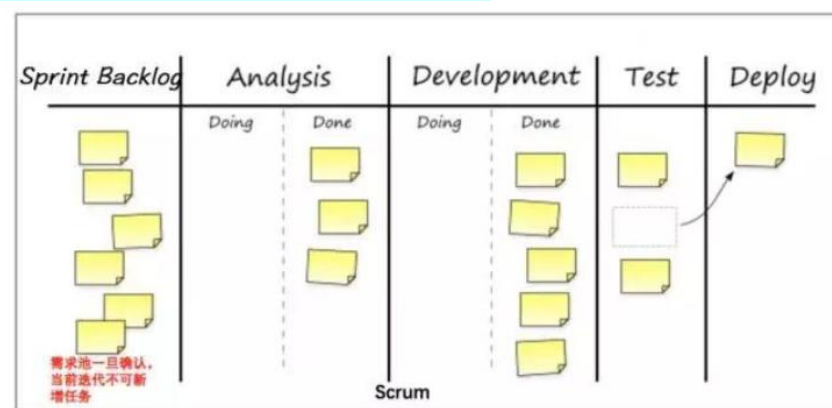
➤ 敏捷建模

- *Individuals and interactions* **over** processes and tools
- *Working software* **over** comprehensive documentation
- *Customer collaboration* **over** contract negotiation
- *Responding to change* **over** following a plan

典型实践方法：

- Scrum
- Kanban看板
- Pair programming

推荐：中国大学MOOC
南京大学：DevOps导论
第三讲：敏捷软件开发



➤ 需求评审 – 需求说明书的标准

- ✓ 正确性
- ✓ 完备性
- ✓ 易理解性
- ✓ 一致性
- ✓ 可行性
- ✓ 健壮性
- ✓ 易修改性
- ✓ 易测试性和可修改性
- ✓ 易追溯性
- ✓ 兼容性

评审方法：

1. 分层次评审

- 目标性评审
- 功能性评审
- 操作性评审

2. 分阶段评审

➤ 需求管理

需求的标识

〈需求类型〉〈需求#〉

需求类型可以是：

F=功能需求，D=数据需求，B=行为需求，

I=接口需求；O=输出需求。

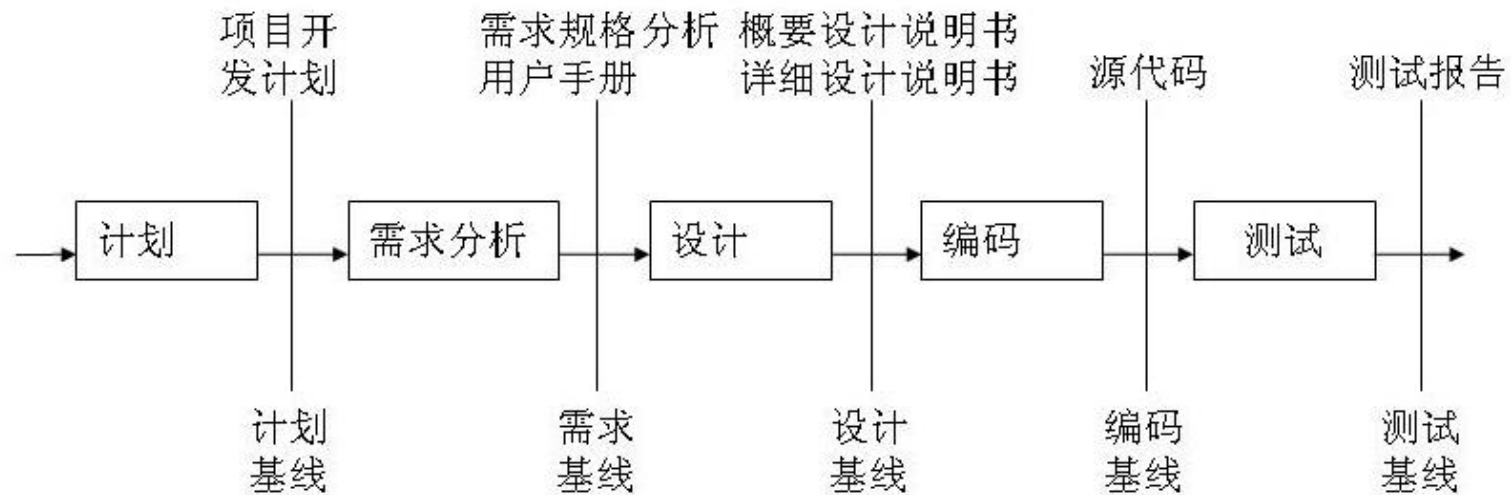


例：需求标识为F03的需求表示编号为3的功能需求。

➤ 需求管理

基线

- ◆ 计划基线
- ◆ 需求基线
- ◆ 设计基线



1. 软件评审
2. 高质量的软件需求分析
3. 提高软件设计质量
4. 高质量编程
5. 软件测试之质量
6. 全面质量管理

3. 提高软件设计质量

□ 软件设计质量考察指标

- 设计结果的稳定性
- 设计的清晰性
- 设计合理性
- 系统的模块结构所显示的宽度、深度等
- 模块间松耦合，模块内聚性高
- 给出的系统设计是否满足软件需求
- 可测试性和可追溯性
- 所要设计的系统在整个项目软件中的地位、作用
- 对各种需求项是否都进行了相应的设计分析

3. 提高软件设计质量

□ 考虑因素

- 体系结构
- 模块与接口
- 界面设计
- 数据库设计
- 设计模式

3. 提高软件设计质量

□ 界面设计：



<https://www.bilibili.com/video/BV1Q741157ve?p=74> (tsinghua university SE)

3. 提高软件设计质量

□ 界面设计评估 KLM模型：

Measure how long will a user take to perform a particular operation

✓ Keystroke-Level Model

- Notations
- Interface Timings
- Calculation Rules
- An Example

✓ The fastest possible interface?

KLM: <https://www.bilibili.com/video/BV1Q741157ve?p=75> (tsinghua university SE)

3. 提高软件设计质量

- H (move hand to GID)
- HP (point to the desired radio button)
- HPK(click the radio button)
- HPKH (move hand back to keyboard)
- HPKHKKKK (type 4 characters, e.g: 23.5)
- HPKHKKKKK (tap Enter)
- Rule 0: HMPMKHMKMKMKMKMK
(Insert M before K,P)
- Rule 1、2、4: HMPKHMKKKKMK

Temperature Converter

Choose which conversion is desired, then type the temperature and press Enter

☒ Convert F to C
☐ Convert C to F

→

Task: C->F
Init: keyboard

‣ $\Sigma = 0.4 + 1.35 + 1.1 + 0.2 + 0.4 + 1.35 + 4 \times (0.2) + 1.35 + 0.2 = 7.15$

‣ : MKKKMK = 3.7 Task: F->C

$(7.15 + 3.7) / 2 = 5.4 \text{ second}$

<https://www.bilibili.com/video/BV1Q741157ve?p=75> (tsinghua university SE)

3. 提高软件设计质量

□ 界面设计评估 Fitts law

$T = a + b \log_2(D/S + 1)$ describes the time taken to hit a screen target:

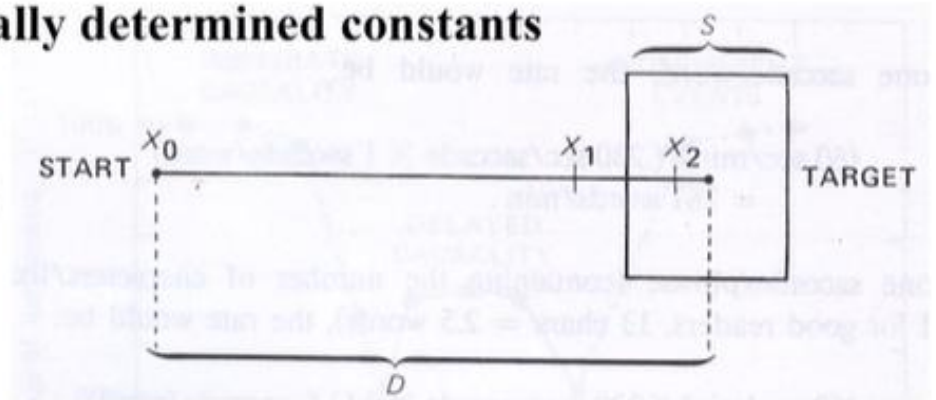
where: **a** and **b** are empirically determined constants

a=50, **b**=150

T is movement time

D is Distance

S is Size of target



⇒ targets as large as possible

distances as small as possible

<http://fww.few.vu.nl/hci/interactive/fitts/>



KLM: <https://www.bilibili.com/video/BV1Q741157ve?p=75> (tsinghua university SE)

3. 提高软件设计质量

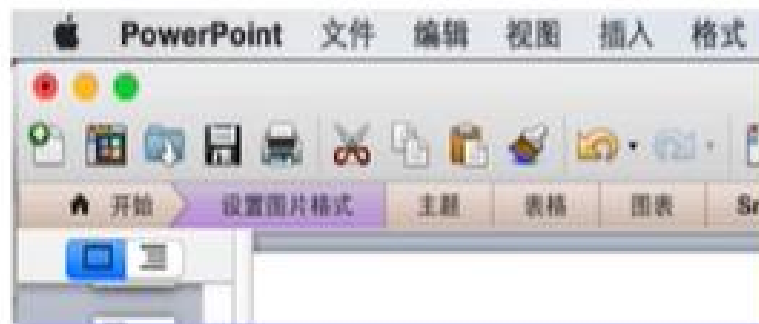
□ 界面设计评估 Fitts law

$$T = a + b \log_2(D/S + 1)$$

s: the size of target



s=5



s=50

$$50 + 150 \log_2(80/50 + 1) = 256 \text{ms}$$

$$50 + 150 \log_2(80/5 + 1) = 663 \text{ms}$$

✓ Fitts' Law based GUI enhancement

- Decreasing D
- Increasing S
- Decreasing D and Increasing S

<https://www.bilibili.com/video/BV1Q741157ve?p=76>

1. 软件评审
2. 高质量的软件需求分析
3. 提高软件设计质量
4. 高质量编程
5. 软件测试之质量
6. 全面质量管理

□ 编码规范

Table of Contents 目录

1.	范围.....	4
2.	规范性引用文件.....	4
3.	术语和定义.....	4
4.	排版规范.....	5
4.1.	规则.....	5
4.2.	建议.....	7
5.	注释规范.....	8
5.1.	规则.....	8
5.2.	建议.....	13
6.	命名规范.....	16
6.1.	规则.....	16
6.2.	建议.....	17
7.	编码规范.....	19
7.1.	规则.....	19
7.2.	建议.....	22
8.	JTEST规范.....	24
8.1.	规则.....	24
8.2.	建议.....	25

□ 规范

- 代码
- 注释
- 命名
- 规则
- 测试

4.1.4. *不允许把多个短语句写在一行中，即一行只写一条语句

示例：如下例子不符合规范。

```
LogFilename now = null;      LogFilename that = null;
```

应如下书写：

```
LogFilename now = null;  
LogFilename that = null;
```

7.1.7. 自己抛出的异常必须要填写详细的描述信息。

说明：便于问题定位。

示例：

```
throw new IOException("Writing data error! Data: " + data.toString());
```

命名 - 匈牙利命名规则对照表

关键字首	数据类型 ↓
c	char ↓
by	BYTE(无符号字节) ↓
n	short ↓
i	int ↓
<u>x,y</u>	<u>int</u> 分别用作 x 和 y 座标 ↓
<u>cx,cy</u>	<u>int</u> 分别用作 x 和 y 长度,c 代表"计数器" ↓
b 或 f	BOOL(<u>int</u>),f 代表"标志" ↓
w	WORD(无符号短整型) ↓
l	LONG(有符号长整型) ↓
<u>dw</u>	DWORD(无符号长整型) ↓
fn	function(函数) ↓
s	string(字符串) ↓
<u>sz</u>	以数组值为 0 结尾的字串 ↓
h	句柄 ↓
p	指针 ↓

4. 高质量编程-代码审查

- **静态分析**

借助工具 SourceMeter、pclint等

- **代码走查**

人模拟计算机运行测试用例，重点是逻辑与计算

- **代码评审**

开评审会等

4. 高质量编程-代码规则

➤ Language

- Python (Pep8)
- Java
- C#
- ...

➤ Company

- google
- ...



4. 高质量编程-代码规则

静态检查工具

- Java tools

Findbugs (bytecode)

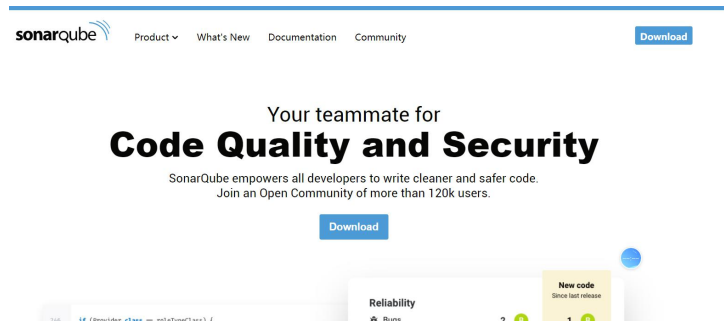
SonarQube

Cobertura

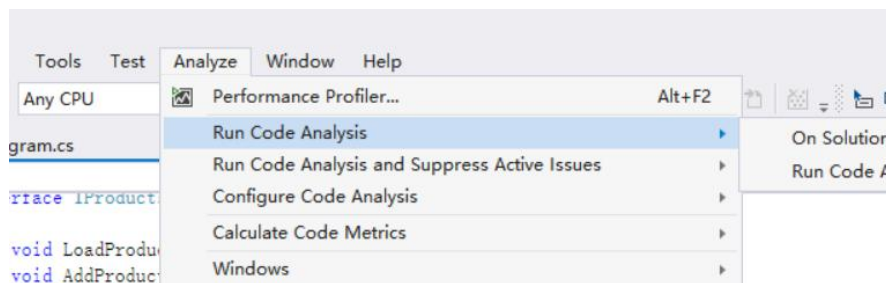
Jacoco



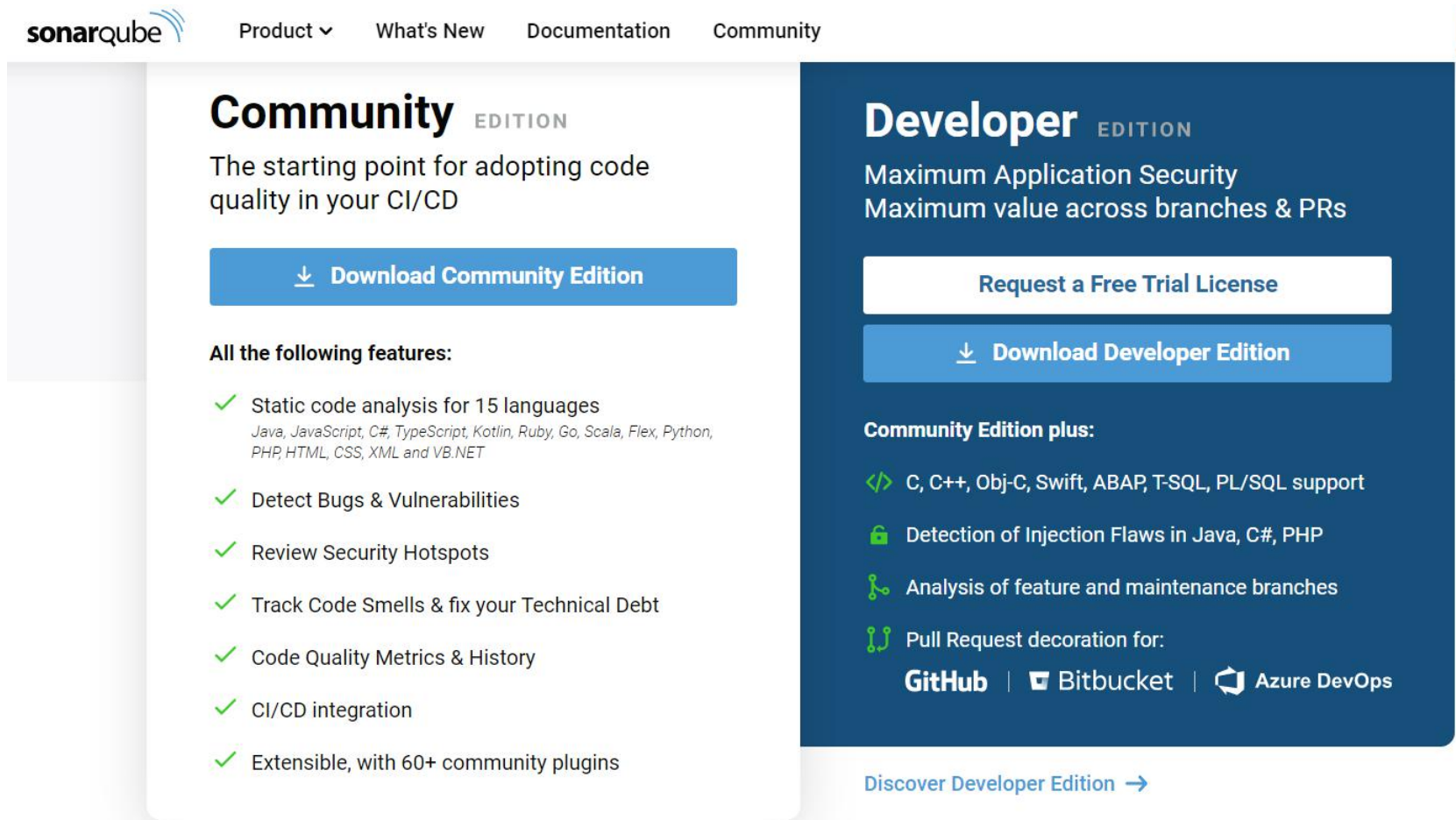
Cobertura



- C++ /C/C#: Static code analysis : Visual Studio



4. 高质量编程-代码规则



The screenshot displays the SonarQube website's download page. At the top, the SonarQube logo is on the left, and navigation links for 'Product', 'What's New', 'Documentation', and 'Community' are on the right. The main content is divided into two columns. The left column, titled 'Community EDITION', describes it as 'The starting point for adopting code quality in your CI/CD' and features a blue button to 'Download Community Edition'. Below this, it lists 'All the following features:' with a checklist of capabilities like static code analysis for 15 languages, bug detection, security hotspots, technical debt tracking, and CI/CD integration. The right column, titled 'Developer EDITION', highlights 'Maximum Application Security' and 'Maximum value across branches & PRs'. It includes a white button to 'Request a Free Trial License' and a blue button to 'Download Developer Edition'. Below these, it lists 'Community Edition plus:' features such as support for various languages, injection flaw detection, branch analysis, and pull request decoration for GitHub, Bitbucket, and Azure DevOps. A link to 'Discover Developer Edition' is at the bottom right.

sonarqube Product ▾ What's New Documentation Community

Community EDITION

The starting point for adopting code quality in your CI/CD

↓ Download Community Edition

All the following features:

- ✓ Static code analysis for 15 languages
Java, JavaScript, C#, TypeScript, Kotlin, Ruby, Go, Scala, Flex, Python, PHP, HTML, CSS, XML and VB.NET
- ✓ Detect Bugs & Vulnerabilities
- ✓ Review Security Hotspots
- ✓ Track Code Smells & fix your Technical Debt
- ✓ Code Quality Metrics & History
- ✓ CI/CD integration
- ✓ Extensible, with 60+ community plugins

Developer EDITION

Maximum Application Security
Maximum value across branches & PRs

Request a Free Trial License

↓ Download Developer Edition

Community Edition plus:

- ✎ C, C++, Obj-C, Swift, ABAP, T-SQL, PL/SQL support
- 🔒 Detection of Injection Flaws in Java, C#, PHP
- 🔗 Analysis of feature and maintenance branches
- 🔗 Pull Request decoration for:
GitHub | **Bitbucket** | **Azure DevOps**

Discover Developer Edition →

<https://www.sonarqube.org/downloads/>

4. 高质量编程-代码性能

- python tools

Profile (python performance check)

```
pylinteg.py x
1  import profile
2
3  # factorial function
4
5
6  def profile_test():
7      ret = 1;
8      for i in range(10):
9          ret = ret * (i+1)
10         print(ret)
11     return ret
12
13
14 if __name__ == "__main__":
15     profile.run("profile_test()")
16
17
```

```
40320
362880
3628800
15 function calls in 0.000 seconds
```

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	0.000	0.000	0.000	0.000	:0(exec)
10	0.000	0.000	0.000	0.000	:0(print)
1	0.000	0.000	0.000	0.000	:0(setprofile)
1	0.000	0.000	0.000	0.000	<string>:1(<module>)
1	0.000	0.000	0.000	0.000	profile:0(profile_test())
0	0.000		0.000		profile:0(profiler)
1	0.000	0.000	0.000	0.000	pylinteg.py:6(profile_test)

ncalls: the number of calling

tottime: total time

percall: $\text{totaltime}/\text{ncalls}$

cumtime: cumulated execution time

percall: $\text{cumtime}/\text{ncalls}$

1. 软件评审
2. 高质量的软件需求分析
3. 提高软件设计质量
4. 高质量编程
5. 软件测试之质量
6. 全面质量管理

5. 软件测试之质量

■ 软件测试与软件质量保证

	测试	SQA
主要对象	软件（工作）产品	开发流程
特性	技术工作	管理性工作
焦点	事后检查	预防
范围	软件研发部门	全组织、跨部门

■ 软件测试的目标

软件测试的目标，就是为了更快、更早地将软件产品或软件系统中所存在的各种问题找出来，并促进程序员尽快地解决这些问题，最终及时地向客户提供一个高质量的软件产品

软件测试是为了发现错误而执行程序的过程

- 1) 一个好的测试能够在第一时间发现程序中存在的错误。
- 2) 一个好的测试是发现了至今尚未发现的错误的测试。

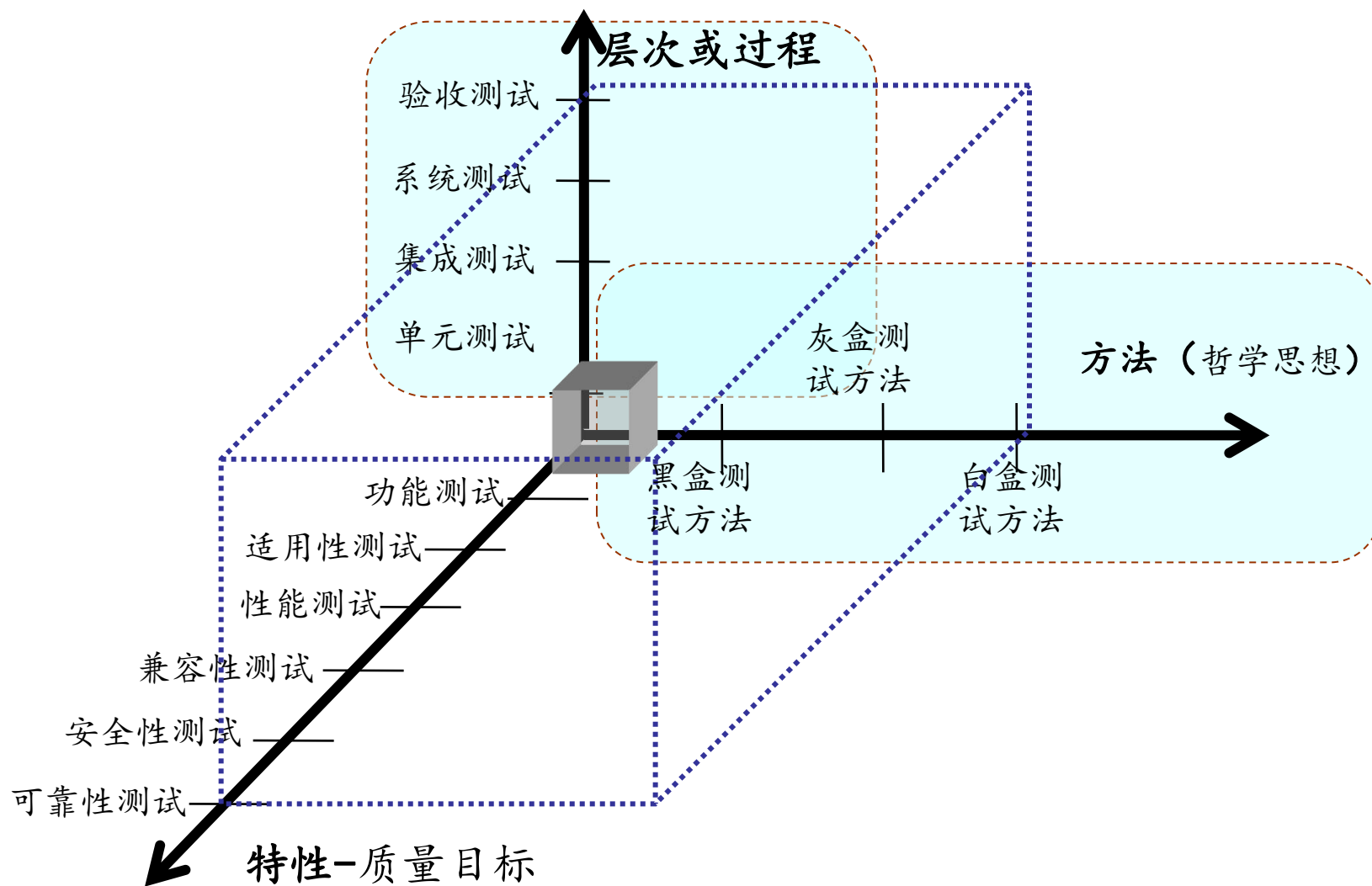


■ 软件测试的原则

1. 所有测试的标准都是建立在用户需求之上。
2. 软件测试必须基于“质量第一”的思想去开展各项工作。
3. 事先定义好产品的质量标准。
4. 软件项目一启动，软件测试也就是开始。应当把“尽早和不断地测试”作为测试人员的座右铭。
5. 穷举测试是不可能的。
6. 第三方进行测试会更客观，更有效。。
7. 软件测试计划是做好软件测试工作的前提。
8. 测试用例是设计出来的，不是写出来的。
9. 不可将测试用例置之度外，排除随意性。
10. 对发现错误较多的程序段，应进行更深入的测试。

5. 软件测试之质量

■ 软件测试的三维空间



5. 软件测试之质量

■ 软件测试方法的辩证统一

□ 白盒测试方法 vs. 黑盒测试方法

□ 静态测试 vs. 动态测试

□ 手工测试 vs. 自动化测试

□ 有计划测试 vs. 随机测试

□ 新功能测试 vs. 回归测试



■ 验证和确认缺一不可

□ **Verification**: Are we building the product right ?

是否正确地构造了软件？

即是否正确地做事，验证开发过程是否遵守已定义好的内容 **验证产品满足规格设计说明书的一致性。**

□ **Validation**: Are we building the right product?

是否构造了正是用户所需要的软件？

即是否正在做正确的事。验证产品所实现的功能是否满足 **用户的需求。**

■ 测试用例设计方法的综合运用

■ 白盒设计方法又分为逻辑覆盖法和基本路径覆盖法，或者分为语句覆盖、判定覆盖、条件覆盖方法

■ 黑盒设计方法分为等价类划分法、边界值划分法、错误推测法、因果图法等。

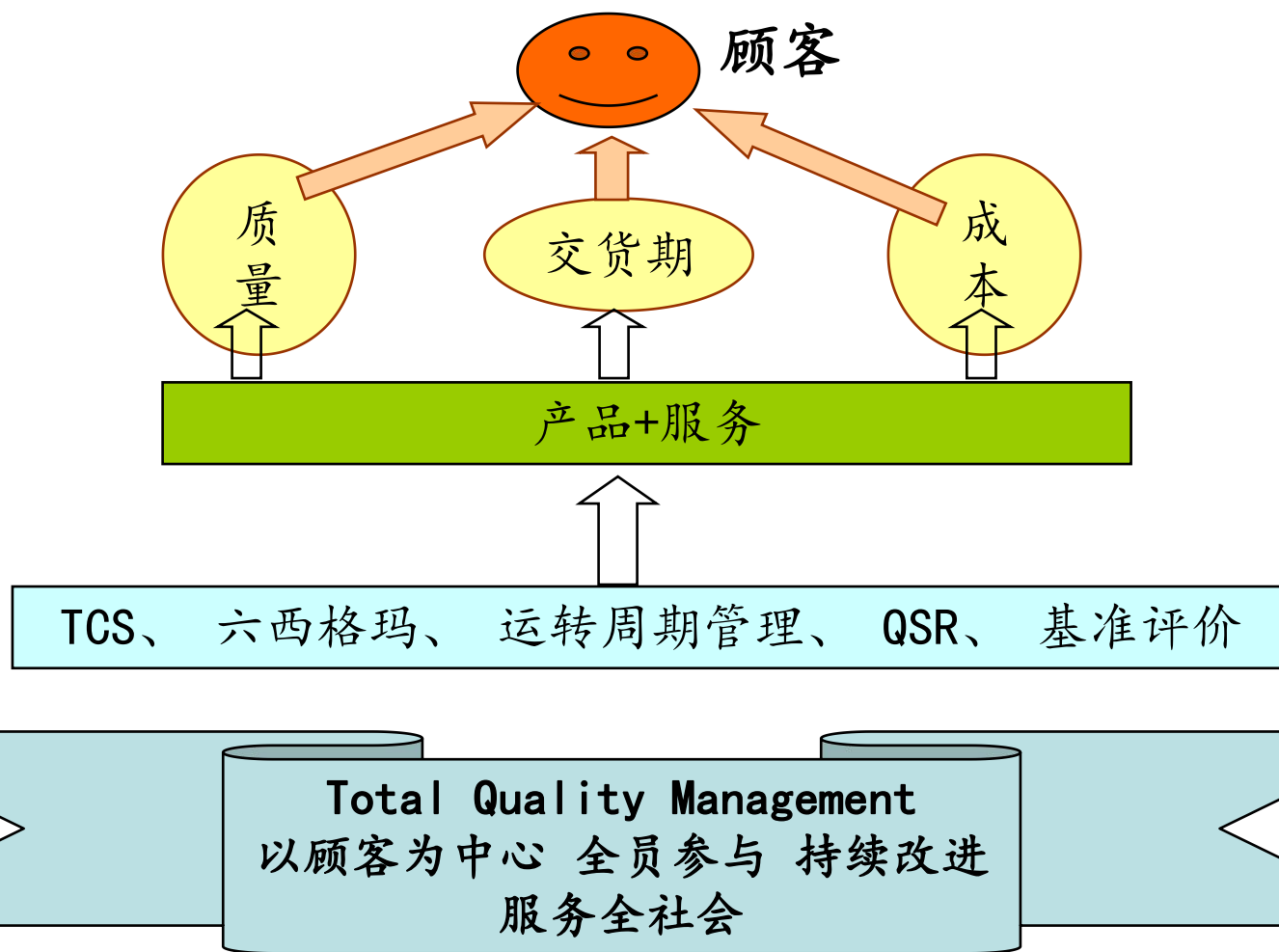
在实际测试用例设计过程中，不仅根据需要、场合单独使用这些方法，常常综合运用多个方法，使测试用例的设计更为有效。

1. 软件评审
2. 高质量的软件需求分析
3. 提高软件设计质量
4. 高质量编程
5. 软件测试之质量
6. 全面质量管理

TQM (Total Quality Management) 就是全面的、全过程的、全员的和科学的质量管理的指导思想。

“一个组织以质量为中心，以全员参与为基础，目的在于通过让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。”

6. 全面质量管理体系



6. 全面质量管理 — 6西格玛

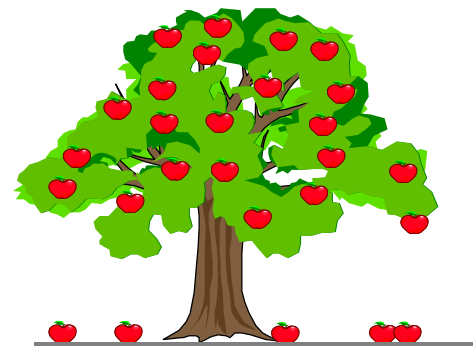
传统质量

- ☐ 注重产品质量
- ☐ 不注重使用数据作出决定。
- ☐ 注重检验
- ☐ 使用一些工具



头痛医头 脚痛医脚

六西格玛



- ☐ 注重流程质量
- ☐ 结构化的改进
- ☐ 数据为决定的依据
- ☐ 结构化的闭环
- ☐ 一次做好的理念

解决根本原因

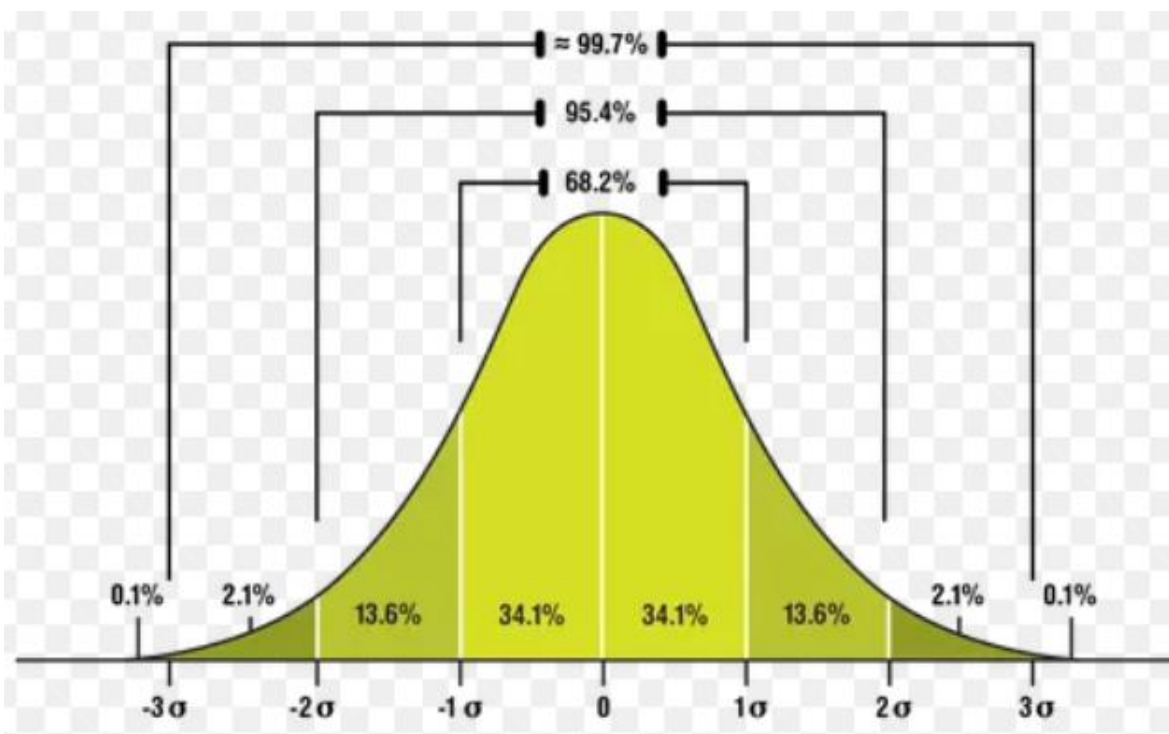
- “六西格玛”一词来源于6个标准偏差, **每百万个操作发生3.4个偏差 (缺陷)**, 它意味着非常高的质量标准。
- 该方法是20世纪80年代末首先在美国摩托罗拉公司发展起来的一种新型管理方式。六西格玛管理是一种近乎完美的管理策略。
- 6西格玛方法学主要的核心步骤 (DMAIC) :
 - ✓ 界定 (define)
 - ✓ 衡量 (measure)
 - ✓ 分析 (analyze)
 - ✓ 改善 (improve)
 - ✓ 控制 (control)

6. 全面质量管理 — 6西格玛

符号 σ (Sigma) 是希腊字母，在统计学中称为均方差。

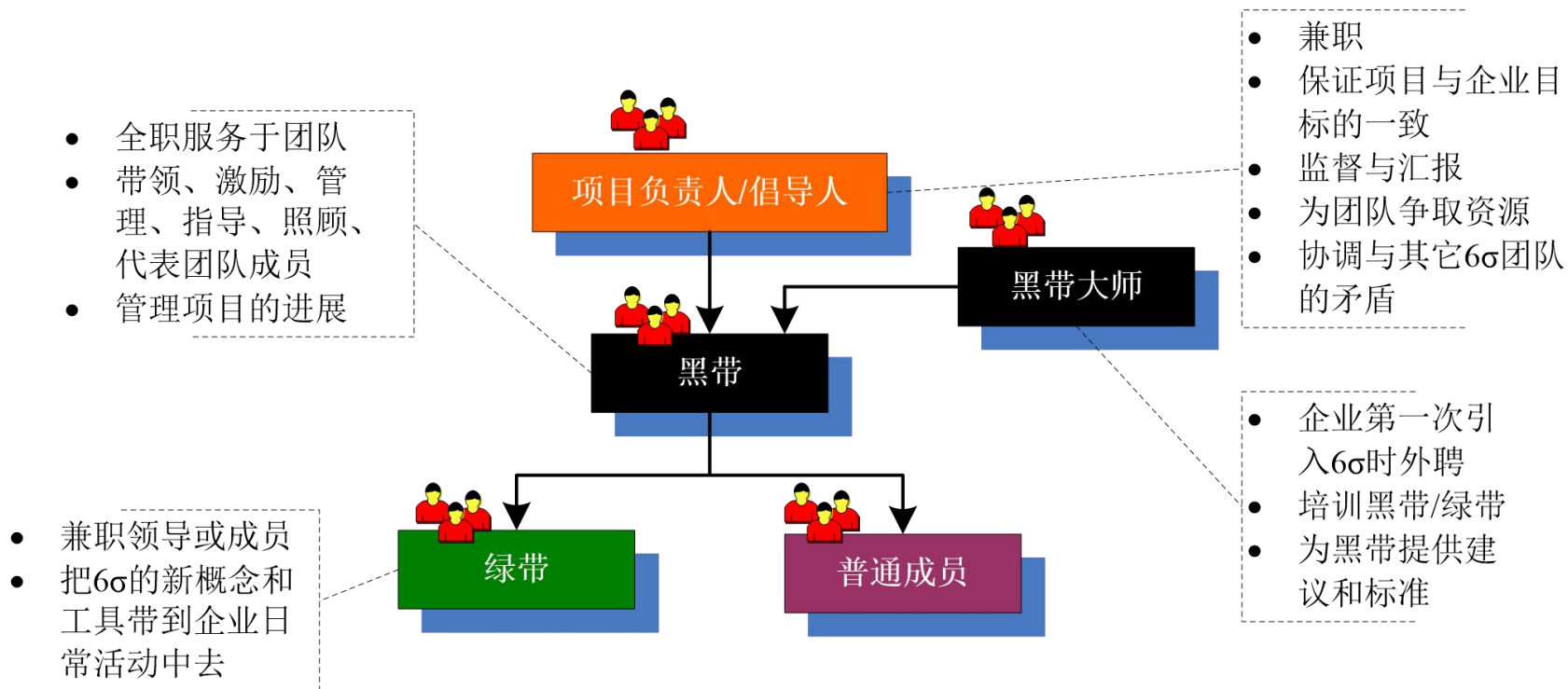
$$\sigma^2 = \frac{\sum (X - \mu)^2}{N}$$

其中：X为样本观测值， μ 是平均值，N为样本容量



σ 值	正品率 (%) 1- (失误次数/ 百万次操作)	DPMO值	以印刷错误为例	以钟表误差为例
1	30.9	690000	一本书平均每页170个错字	每世纪 31.75年
2	69.2	308000	一本书平均每页25个错字	每世纪 4.5年
3	93.3	66800	一本书平均每页1.5个错字	每世纪 3.5个月
4	99.4	6210	一本书平均每30页1个错字	每世纪 2.5天
5	99.98	230	一套百科全书只有1个错字	每世纪30 分钟
6	99.9997	3.4	一个小型图书馆的藏书 中只有1个错字	每世纪6 秒钟

□ 人员组织结构



■ 一个五个阶段的改进步骤DMAIC:

- ✓ 界定 (define)
- ✓ 衡量 (measure)
- ✓ 分析 (analyze)
- ✓ 改善 (improve)
- ✓ 控制 (control)

1. 通过与客户交流的方法来定义客户需求、可交付的产品及项目目标
2. 测量现有的过程及其产品，以确定当前的质量状况（收集缺陷度量信息）
3. 分析缺陷度量信息，并挑选出重要的少数原因
4. 通过消除缺陷根本原因的方式来改进过程
5. 控制过程以保证以后的工作不会再引入缺陷原因

6. 全面质量管理 - 6西格玛

■ 6西格玛质量管理工具

定量分析工具	检查表	直方图	散布图	流程图
	运行图	因果图	排列图	Pareto图
定性分析工具	亲和图	网络图	矩阵图	雷达图
	关联图	箱线图	树图	过程决策程序图 (PDPC)
方法	质量功能展开 (QFD)		头脑风暴法	水平对比法

1. 软件评审
2. 高质量的软件需求分析
3. 提高软件设计质量
4. 高质量编程
5. 软件测试之质量
6. 全面质量管理

理解软件开发过程中各个阶段可以实施的
质量保证措施， 特别注意**软件评审**的相关
技术。



业精于勤，荒于嬉！
