软件项目管理

课堂教学讲稿 · 第 12 次课　第 7 章 软件项目质量管理

某汽车公司质量管理系统案例 | 第 12 周 · 2 学时（90 分钟） | 7.1 概述 · 7.2 规划质量（七种工具）· 7.3 管理质量 · 7.4 控制质量 · 7.5 配置管理 · 7.6 绩效域 · 7.7 实验 5（课外选做）

课程代码 154431006 | 32 学时 / 2 学分 | 广东金融学院 计算机学院

配套课件：http://111.231.0.226:1234/ | 备课台：http://192.168.31.76:8910/备课台/index.html

打印时间：2026 年 11 月 27 日

《软件项目管理》第 12 次课讲稿（第 12 周 · 90 分钟）——第 7 章 软件项目质量管理

课程：软件项目管理（154431006）｜广东金融学院 计算机学院
范围：**7.1 概述 · 7.2 规划质量 · 7.3 管理质量 · 7.4 控制质量 · 7.5 配置管理 · 7.6 两绩效域 · 7.7 实验 5**
教材：宋莹莹等《软件项目管理与实践》（第 3 版）第 7 章（书页 141–177）｜案例：**某汽车公司质量管理系统**
配套课件：`lecture12/12.1.html` ~ `12.32.html`（**32 页 = 上课 31 页 + ★课后拓展 1 页**）｜在线：http://111.231.0.226:1234/lecture12/index.html
教材实验 5（选做）：用 Jira 提交测试缺陷并完成三态流转，**课外自主实践**

一、教学目标（依教案第 7 章）

维度	目标
知识 K	质量与等级之别；质量成本四类与"总成本最低区"；七种基本质量工具的定义与用途归类；QA/QC 的分工；配置管理与三库基线；度量/交付两绩效域的要点
能力 A	会用帕累托图找"少数重要原因"；能判控制图异常并区分控制界限与规格界限；能识别配置项并说清变更该找谁
素质 V	预防胜于检查——**该花在预防上的钱不能省**；缺陷如实记录、不瞒报（流出的账客户先付）

本讲重点：① 七种工具的**定义与用途归类**；② **帕累托图找重点**；③ QA 与 QC 的分工；④ **控制图判异四条 + 控制界限≠规格界限**；⑤ 配置管理与三库基线。
本讲难点：① 质量与等级容易混（"档次低"不等于"质量差"）；② **判异 ≠ 不合格**（控制界限管过程、规格界限管产品）；③ 基线与普通文件夹的区别。

二、时间分配（教案时间轴 · 共 90 分钟）

环节	时间	对应课件页	说明
案例导入	8	12.1–12.4	四道闸口；"缺陷集中却找不到根因、规格还在变"
7.1 质量管理概述	15	12.5–12.8	质量 vs 等级、质量成本、预防胜于检查、敏捷质量
7.2 规划质量管理	22	12.9–12.14	8 类方法、**七工具速查**、因果图、三图示例、**帕累托**、成果
7.3 管理质量	10	12.15–12.17	QA 面向过程、审计/根本原因/面向 X、成果
7.4 控制质量	10	12.18–12.20	QC 面向验收、**判异四条**、成果
7.5 配置管理	15	12.21–12.24	配置项、三库基线、流程、与变更管理的关系
7.6 两绩效域	5	12.25–12.26	度量域（含度量陷阱）、交付域
练习＋小结	5	12.29–12.31	帕累托当堂算、工具归类、判异判断

⚠️ **容量提示**：
1. **本讲稿为完整版**：32 页口播合计约 **25,600 字**，按正常语速（约 200 字/分）约 **128 分钟**，比 90 分钟课堂**多出约 38 分钟**。页内标注的「(X 分钟)」是**按教案时间轴给各块的课堂建议配额**（块内按字数占比分配，合计 90 分钟），不是口播净耗时——照着标注推进即可，讲到哪算哪，不必追求逐字念完。
2. **可压缩的页**（时间紧时按序砍）：12.9 九类方法只点名称（省 1'）→ 12.16 只讲质量审计与 RCA（省 2'）→ 12.21 配置项只讲识别与标识（省 1'）→ 12.12 三图示例只点用途（省 1'）→ 12.24 关系页一句带过（省 1'）。
3. **一页都不要跳的**：**12.10 七工具速查**（定义与归类，期末必考）、**12.13 帕累托**（计算题）、**12.19 判异四条**（判读题）、**12.22 三库与基线**（配置管理核心）。
4. **★课后拓展 2 页**（12.27 实验 5 任务预览、12.28 做法指引）：**课上可跳过**，合计标注 4 分钟不计入 90 分钟；课后自学 + 自主实践。

三、逐页讲稿

12.1 第 7 章 软件项目质量管理（1 分钟）

口播 · 可照读

上一章我们把钱算到了完工：挣值三个数、四个指标、两套预测口径。可有个问题一直悬着没回答——钱花下去了，东西到底**做对没有**？第 7 章就换一把尺子：不问花了多少，问**符合不符合**。

质量管理的核心目标是一句话：**确保项目过程和交付成果符合明确的质量标准，同时满足干系人的期望，为用户和干系人创造最大价值**。“明确”两个字是重点——标准没写清，后面所有“质量问题”都会退化成口味之争：你说不好用，我说够用了，谁也说服不了谁。

这一章的案例是某汽车公司质量管理系统（教学示例）。系统覆盖五类功能：质量数据采集与分析、生产环节质量控制、缺陷追溯、供应商质量管理、质量审计与报告。但真正要看的不是功能清单，而是车间里的动作——它不是“最后检一次”，而是**四道闸口依次把关**：来料检验、SPC、售后 PPM、8D。这四个词后面会逐个展开，现在先记名字和方向。

系统上线前做内测，暴露出两件事：缺陷集中在少数环节，却**说不清根因**；抽样规格还在被临时改动。所以这一章要回答的不是“测了多少轮”，而是——**缺陷凭什么要等到这一环才被发现**。

带三个问题听课：**“合格”由谁说了算**（标准）；**质量的钱该花在哪一类**（成本）；**几百条缺陷先改哪几类**（工具）。

板书

- 核心目标：过程与成果**符合明确的质量标准** + 满足干系人期望
- 案例：某汽车公司质量管理系统（教学示例）｜采集分析 · 过程控制 · 缺陷追溯 · 供应商质量 · 审计报告
- 四道闸口：**来料检验 → SPC → 售后 PPM → 8D**
- 三个问题：合格谁说了算（标准）→ 钱花在哪一类（成本）→ 先改哪几类（工具）
- 一句话：**质量不是最后一道闸门守出来的，是过程里建出来的**

提问 同一款软件，A 功能很少但从不崩、数一分不差；B 功能齐全但登录偶尔失败、对账偶尔错位。哪个质量高？为什么？（限 1 分钟，先各自写一句）

预设回答

- ✅ 正确方向：得先问“各自的规格写清了没有”。A 满足了自己写清的每条规格，是高质量；B 没达到它承诺的规格，就是低质量——**质量比的是“符合规格的程度”，不是功能多少**。这句话正好是下一节（质量与等级）的引子。
- ⚠️ 部分对：说“B 功能多所以质量高”。方向错在把**等级**当成了质量；功能多是档次高，档次高不等于符合得好。补一句：B 的功能项数高，缺陷率也高，两件事要分开记。
- ❌ 常见错：说“没法比，两个产品不一样”。能比——不比功能多少，比“有没有达到自己声明的规格”；判据是规格，不是同类产品横向排名。

易错点 / 考点 ① 核心目标的两半要都答到——**符合标准**与**满足干系人期望**，只答前一半是半对。② 四道闸口的顺序与方向（供应商端→过程端→客户端→纠正端）不能颠倒。③ 别把“测得多”当成质量好，本章反复打这个点。

过渡 *这一章有 7 个小节、三过程加配置管理，先给一张地图，否则容易学成八块并列的知识点。*

12.2 学习导航：三个过程、一本配置账、两个绩效域（2 分钟）

口播 · 可照读

这一章不是八块并列的知识，而是**一条链**：先定标准，再让过程按标准执行，然后逐件核实交付物，最后把“变化”本身管住。

链条上是四个环节。**规划质量**（7.2）定标准：质量要求与验收口径，成果是质量管理计划、质量测量指标；**管理质量**（7.3）管过程：执行得合不合规，成果是质量报告、测试与评估文件；**控制质量**（7.4）验成果：交付物达不达标，成果是核实的可交付成果、工作绩效信息；**配置管理**（7.5）管变化：改了还追不追溯得到，成果是基线、配置库记录、变更评审结论。

一句话分工：**三过程管“质量怎么来”，配置管理管“变了还追不追得回来”**。两绩效域（度量、交付）在 7.6 回收：度量域问“指标可不可信”，交付域问“交付物有没有被确认”。

三过程之上还有**质量管理支撑体系**三层：**基础设施**（质量文化、开发环境、标准体系）、**方法层次**（开发模型、流程）、**技术层次**（工具成熟度、自动化测试水平）。这三层不动，只靠加班检查，质量抬不上去——这句话请记住，它是本章最容易被忽略的一句。

承前：第 6 章的成本在这里交叉成**质量成本**——预防与评估是提前花的钱，失败成本是没提前花而多花的钱。启后：配置管理先把“基线＋变更”立起来，为第 11 章的整体变更控制埋下伏笔。

先做个小互动：写代码的时候，质量到底靠什么保证？各位先各自写三条，两分钟后再对答案。答案会正好落在这条链上：规划侧是编码规范、接口约定、完成的定义；管理侧是同伴评审、静态检查、构建门禁；控制侧是单元测试、集成测试、回归。一句话——**质量主要产生在写代码的那几分钟，而不是测试的那几天**。

板书

- 一条链：**定标准（7.2）→ 管过程（7.3）→ 验成果（7.4）→ 管变化（7.5）**，两绩效域（7.6）回收
- 过程对照：规划＝面向标准｜管理＝面向过程｜控制＝面向成果｜配置＝面向变化
- 支撑体系三层：**基础设施 · 方法层次 · 技术层次**（不动这三层，加班检查抬不上去）
- 承前：质量成本＝质量与成本的交叉点｜启后：基线＋变更 → 第 11 章
- 一句话：三过程管“质量怎么来”，配置管理管“变了还追不追得回来”

提问 写代码的时候，质量到底靠什么保证？请各写三条，并说明它属于链条上的哪一环。（限 2 分钟）

预设回答

- ✅ 正确方向：能按环归类并说清机制。例如“提交前必须过 CI＋同伴评审”属管理侧——目的是让不合格的提交**进不了主干**；“完成的定义（DoD）逐条可验证”属规划侧——标准没写清，后面全是口味之争。
- ⚠️ 部分对：只写“多测试、多检查、认真写代码”。方向不错，但三条全挤在控制侧——那正是本章要纠正的“质量靠最后一关”的错觉；补两条过程侧的机制（评审清单、门禁）。
- ❌ 常见错：说“靠测试同学把住最后一关”。测试只能把已经产生的缺陷拦下来，拦不住的一路流到客户手里；而且越晚拦越贵，这是 12.7 要算的账。

易错点 / 考点 ① 三过程的**面向对象**要能一一对上：标准／过程／成果，别把“管理质量”和“控制质量”互换。② 配置管理不属于三过程，它管的是“变化”，考试常拿它混在质量三过程里出选择题。③ 支撑体系三层是简答题常考的小点，三层名字要写全。

过渡 *地图挂好了，从案例进场——先看那四道闸口到底在车间里怎么运转。*

12.3 导入案例：四道闸口，一条质量信息链（3 分钟）

口播 · 可照读

先把这个系统的边界说清楚。某汽车公司要建一套质量管理系统（教学示例），目标是把整车从进料、生产到交付售后的质量数据统一采集、统一分析、统一追溯。功能上有五块：质量数据采集与分析、生产环节质量控制、缺陷追溯、供应商质量管理、质量审计与报告。

但真正值得看的是车间里的动作。它不是“最后检一次”，而是**四道闸口依次把关**：来料检验、SPC、售后 PPM、8D。每一道闸口都在产生数据，也都在消耗成本——这两件事同时发生，才让质量管理变成一本可以算的账。

第一道，**来料检验**，管的是供应商端：供应商交来的件合不合规格，收还是不收。案例里的示例数据是：月收 180 批，按 AQL 抽检判退 6 批，约 3.3%。问题在哪？它只判“收/退”，不追供应商的**过程能力**——退回去的那批怎么改、改没改好，系统里没有记录。

第二道，**SPC，统计过程控制**，管的是过程端：过程稳不稳，关键特性是否受控。线束压接工序每 2 小时抽 5 件，25 组子组，控制图上有 2 组越界。问题在哪？越界只记“已处理”，**没查原因**，所以下次还在同一个位置越界。这一句请划下来——它正是后面因果图要解决的问题。

第三道，**售后 PPM**，管的是客户端：客户手里每百万件有多少不良。月交付 12 000 台、售后质量问题 14 起，$PPM = \frac{14}{12000} \times 10^6 \approx 1167$。问题在哪？分母按“台”还是按“件”没有统一口径，两个部门算出来差一大截。

第四道，**8D，八步纠正法**，管的是纠正端：根因是什么、措施有没有效。本月 7 份 8D 报告，其中 5 份指向同一压接工位。问题在哪？措施写“加强培训”，**不可验证**；改完没有回访数据，等于没闭环。

四道闸口串起来，就是一条**质量信息链**：来料检验管输入端、SPC 管过程端、售后 PPM 管客户端、8D 管纠正端。后一道闸口的数据，就是前一道闸口的成绩单。可这条链上，每一环都只在“记录”，没有人做“分析”，于是同一类缺陷会反复出现——这不是系统缺功能，是**信息链断了**。

再补一层视角：这套系统真正要交付的不是四张报表，而是**四件事的闭环**——数据能采上来、异常能被看见、根因能被追到、措施能被回访。前面那四条“问题”，恰好一条断一处：来料只判收退，断在“追过程能力”；SPC 越界只记已处理，断在“追原因”；PPM 口径不一，断在“能不能比”；8D 写加强培训，断在“能不能验”。所以后面每一件工具进来，都要能回答“它接上了哪一处断点”。

板书

- 四道闸口：**来料检验（输入端）→ SPC（过程端）→ 售后 PPM（客户端）→ 8D（纠正端）**
- 来料：月收 180 批、判退 6 批≈3.3%｜只判收/退，**不追过程能力**
- SPC：每 2 小时抽 5 件、25 组子组、**2 组越界**｜只记“已处理”，**不查原因**
- PPM：$PPM = \frac{14}{12000} \times 10^6 \approx 1167$｜**分母口径不统一**
- 8D：7 份报告中 5 份同一压接工位｜措施“加强培训”**不可验证**、无回访
- 一句话：后一道闸口的数据，是前一道闸口的成绩单；**只记录不分析，缺陷必重复**

提问 四道闸口里，哪一道的数据最“贵”？为什么？（限 2 分钟）

预设回答

- ✅ 正确方向：**售后 PPM 最贵**。它统计的是**已经被客户发现**的缺陷——代价不只是修，还包括赔付、SLA 罚金、保修成本、口碑与客户流失。越靠左越便宜：来料检验只损失一次抽检工时，SPC 越界只需停线查因，成本都还在自己厂里。“数据最贵”和“发现最早”是一对反比关系，这就是预防胜于检查的账本基础。
- ⚠️ 部分对：说“8D 最贵，因为要开会要写报告”。8D 重要，但它本身是**纠正**动作，代价取决于前面漏出多少；把它和“客户已发现的缺陷”混在一起，就说不清钱到底花在哪一栏。
- ❌ 常见错：说“来料检验最贵，因为每个月都在抽”。抽检是**评估成本**，是四类里最便宜的提前支出之一；把它当最贵，等于把账算反了，后面质量成本那两本账就立不住。

易错点 / 考点 ① 四道闸口的方向不能颠倒（供应商→过程→客户→纠正），顺序错了整条信息链就断了。② PPM 是**百万件口径**，不是百分比；1167 PPM 换成百分比只有约 0.12%，两个单位不要混。③ SPC 越界只说明过程出现**特殊原因的异常波动**，不等于产品不合格——这条在第 7.4 节判异时还会考。④ 8D 是**八步纠正法**，不是八个部门。

过渡 *四道闸口的账算清了，可内测阶段真正卡住团队的不是闸口缺不缺，而是闸口给出的数据没人接得住——看下一页的六条痛点。*

12.4 导入案例：缺陷集中却找不到根因，规格还在变（2 分钟）

口播 · 可照读

内测阶段，团队遇到的是典型的两头受挤：**缺陷一堆，但说不清根因；标准还没定稳，就开始返工**。表面看是“测试不够”，实际上每一类痛点背后都对应本章的一件工具或一个过程。

第一条：几百条缺陷铺在一张表里，**没有归类、没有排序**。后果是什么？不知道哪几类占大头，整改只能平均用力；改完也不确定有没有效果。对应的武器是——核查表先收集，**帕累托图**再排优先。

第二条：缺陷集中在少数环节，**但没人做根因分析**。同一个工位反复出问题，8D 措施写“加强培训”，下一批照旧。对应的武器是**因果图加连续问为什么**，一直追到可行动的那一点。

第三条：**规格频繁变更**——抽样标准被临时改，代码分支对不上。没有基线，改了什么、谁批的、影响哪里，都追溯不到。这对应**配置管理与基线、变更控制**，也是第 11 章的伏笔。

第四条：测试挤在后期，只覆盖基本功能；边缘场景与设备接口兼容性没验证，缺陷全压到集成阶段爆发。对应**预防胜于检查**：评审与测试前移。第五条：同一指标两个部门算出两个数（PPM 的分母），数据不可比、趋势判断失效，汇报会上先吵口径——对应质量测量指标与数据口径的统一。第六条：代码评审靠“看一眼”，复杂交互逻辑漏检，评审没有清单、没有门禁，覆盖率靠自觉——对应核查表、完成的定义与持续集成门禁。

把这六条合起来看，问题不是“测得太少”，而是**信息没有归拢，变化没有管住**：缺陷没归类，就无法排序；原因没追到底，就无法预防；规格没基线，改一次伤一片。这三件事，恰好对应本章的三组武器：**七种工具、质量成本、配置管理**。

还要看清一件事：这六条不是并列的六个毛病，它们有先后。**归不了类，就排不了序；排不了序，就不知道该先追哪个因；没追到因，整改只能靠加人加班。** 而规格一变，前面归好的类又作废——所以基线这件事要跟归类同步做，否则你今天画出来的帕累托图，下周就过期了。这也是后面章节顺序安排的道理：先有标准，再谈数据，最后才谈改进。

板书

- 六大痛点 → 三组武器：**没归类（帕累托）· 没追因（因果图）· 没基线（配置管理）**
- 归类缺失：几百条缺陷一张表 → 只能平均用力 → **核查表 → 帕累托**
- 根因缺失：8D 写“加强培训”不可验证 → **因果图 + 连问 3～5 层为什么**
- 规格常变：抽样标准被临时改、分支对不上 → **基线 + 变更控制**（伏笔 7.5 与第 11 章）
- 测试挤后期 / 口径不一致 / 评审无清单 → **预防前移 · 指标口径 · 核查表+门禁**
- 一句话：不是测得太少，是**信息没归拢、变化没管住**

提问 如果只允许你先改一环，你改哪一环？要说清理由，不接受“全都重要”。

预设回答

- ✅ 正确方向：先补**记录与归类**（给缺陷表加“缺陷类别”一列、每天打勾，一周就能画帕累托图，几乎零成本），同时把**规格变更关进基线**，再把评审与测试前移。理由按三条判据讲：哪一环损失最大（外部失败先堵）、哪一环最便宜最快见效（归类）、哪一环改完能给别的环节供数据（口径统一）。
- ⚠️ 部分对：说“先加测试、先扩覆盖率”。方向不算错，但顺序有问题——**没有数据就先谈预防，等于拍脑袋；没有基线就先谈整改，改完还会被改回去**。补一句：先有分类，测试才知道该往哪儿加。
- ❌ 常见错：说“先买一套自动化测试平台／先上工具”。工具解决的是执行效率，不解决“哪几类缺陷占大头”和“改了谁批的”这两个信息问题；工具一上，混乱只是跑得更快。

易错点 / 考点 ① 六条痛点的归类要能一一对上工具，考试常给现象让选工具。② “缺陷集中”是**分布问题**（用帕累托），不是“数量问题”（加人加班解决不了）。③ 规格变更属于**配置管理**范畴，不属于控制质量；这条边界在第 12.24 还会再考一次。

过渡 *痛点里第二条最扎人——同一个工位反复出问题，可大家连“质量”这个词指的是什么都没统一。先把最容易混的一对概念拆开。*

12.5 质量与等级：一个问“做对没有”，一个问“做到什么档次”（3 分钟）

口播 · 可照读

讨论质量时最先要拆开的一对概念，就是质量与等级。日常语境里两者被混着用，结果是在谈判桌上把“**加功能**”当成了“**提质量**”，把“**修缺陷**”当成了“**加需求**”。这一混，价钱和工期就全乱了。

一句话分清：**质量是满足需求与规格的程度**，回答“做对没有”；**等级是功能与性能的档次定位**，回答“做到什么水平”。两者量的东西也不一样：质量靠缺陷率、通过率、PPM、可用性来量；等级靠功能项数、并发档位、容量来量。

两个方向的性质完全不同。质量偏低，**是问题**——要返工，或者不予验收；等级偏低，**不是问题**——只要规格写清、双方认账即可。也正因为如此，两条不能互换：**不能用“降等级”掩盖质量问题，也不能用“高等级”替代质量承诺**。

两个典型误判，请一定记住。第一种，“功能少，所以质量差”——这是把等级当质量。第二种，“档次低，所以质量差”——同样是拿档次判质量。所以那句最反直觉的话要敢说出口：**低等级可以是高质量，高等级也可能是低质量**。

举两个例子。基础版记账 App：单账本、手工录入，功能很少，但**零崩溃、账目分毫不差**——它满足了自己写清的每一条规格，这是低等级、高质量。旗舰版：多账本、自动同步样样有，可登录偶尔失败、跨设备对账错位——功能再多也没达到它承诺的规格，这是高等级、低质量。

为什么要费劲分清？因为两件事的处置完全不同。**“加一个报表”是升等级，要谈钱和工期；“修掉这个错账”是提质量，本来就该做到**。混为一谈，就等于用打折换功能：客户以为买的是质量，拿到的是档次。

再补一层：质量不是一个总分，而是**一组带目标的指标**。常见的五个侧面是功能性、可靠性、性能效率、易用性、可维护性。同一个系统可以某项高、某项低——这不是自相矛盾，而是规格本来就这么写的。所以谈质量要指名道姓：说的是哪一项、目标值多少、怎么验。

给一个更贴近各位的判据：一段代码“写得很漂亮”是等级层面的评价，“这段代码在我们约定的规范下通过了评审、覆盖了边界、没引入新的告警”才是质量层面的评价。前者靠眼光，后者靠规格与证据。项目上能拿去验收的，只有后者。

板书

- **质量**＝满足需求与规格的**程度**（做对没有）｜**等级**＝功能与性能的**档次**（做到什么水平）
- 怎么量：质量看缺陷率・通过率・PPM・可用性｜等级看功能项数・并发档位・容量
- 偏低时：质量偏低**是问题**（返工／不予验收）｜等级偏低**不是问题**（规格写清即可）
- 两条红线：**不能用降等级掩盖质量问题**；不能用高等级替代质量承诺
- 例：基础版记账 App 零崩溃＝**低等级・高质量**｜旗舰版功能全但登录失败＝**高等级・低质量**
- 加报表＝升等级（谈钱谈工期）｜修错账＝提质量（本来就该做到）

提问 （本页互动题）互动：下面哪一句是对的？ A．等级越高，说明质量管理做得越好；B．低等级的产品只要符合它自己的规格，就是高质量的；C．质量偏低可以用提高等级来补偿；D．等级与质量是同一件事的两种说法

预设回答

- ✅ 选 B——**低等级的产品只要符合它自己的规格，就是高质量的**。点评：等级是档次定位，质量是符合规格的程度；判据看规格，不看功能多少，也不看同类排名。
- ⚠️ 选 B 但把理由说成“等级低所以要求也低”：方向对了、理由错了——低等级不是**要求更低**，是**规格写得更少更清楚**；要求一点没降，只是范围更窄。这句话说错，后面“低等级可以是高质量”就滑成了“低要求也可以是高质量”。
- ❌ 选 A、C、D：三个选项是同一个错误的三种说法——拿档次判质量。**提高等级解决不了“没达到规格”**，用升等级补偿质量问题，等于把缺陷包进新功能里，账只会更贵。

易错点 / 考点 ① 质量与等级的**定义句**要能各背一句，这是判断题与单选题的高频点。② “低等级可以是高质量”必须能配一个例子说出来，光背结论容易在情境题里翻车。③ “不能用降等级掩盖质量问题”是思政与考点的交叉句，简答题常考。④ 质量是**一组指标**不是总分，别在案例题里写“总体质量良好”就收尾。

过渡 *分清质量与等级只解决了“问什么”，下一个问题是“花多少”——质量这件事，账该怎么算。*

12.6 质量成本：四类支出，两本账（5 分钟）

口播 · 可照读

质量管理最容易吵起来的一句话是“又要马儿跑，又要马儿不吃草”。把质量支出拆成四类之后，争论就从“要不要花钱”变成“**钱花在哪一类**”——后者能算，前者只能吵。

定义先说清：质量成本是**在产品生命周期中，为预防不符合要求、为评价是否符合要求、以及因未达到要求（返工）而发生的所有成本**。注意它把三件事都算进来了：预防、评价、还有没达到要求之后的返工与赔偿。

四类怎么分？先看两本账。**第一本是一致性成本**——为了**预防失败**所投入的费用，包括**预防成本**和**评估成本**。预防成本的例子：培训、流程文档化、设备与工具采购、编码规范、设计评审、持续集成平台；它花在开工前与过程中，目的是“不让它发生”。评估成本的例子：测试、检查、破坏性测试（压测、混沌演练）、代码扫描；它花在交付前，目的是“提前发现”。

第二本是非一致性成本——因为**未能满足质量要求**而发生的费用，包括**内部失败成本**和**外部失败成本**。内部失败成本：返工、重构、废品（废弃代码与返工工时），交付前自己发现，**自己扛**。外部失败成本：客户索赔、保修与热修、SLA 罚金、业务流失，交付后客户发现，**客户先替你付**。

看图 7-1。左框是一致性成本，右框是非一致性成本。这里有个译名要提前打个招呼：图内第一项写的是“预测成本”，它就是**预防成本**，别被名字绕住；它下面挂培训、流程文档化、设备、选择正确的做事时间。左框还有“评价成本”，挂测试、破坏性测试导致的损失、检查。右框里，内部失败成本挂返工、废品；外部失败成本挂责任、保修、业务流失。两个框下面各有一句话：左框是“在项目期间，用于防止失败的费用”，右框是“在项目期间和项目完成后，用于处理失败的费用”——**一个在期间，一个跨到完成后**，这就是两本账最本质的差别。

算总账：$C_{总} = C_{预防} + C_{评估} + C_{内部失败} + C_{外部失败}$。规律要背下来：**预防与评估投入上升，失败成本下降，总质量成本存在一个最低区**。低于这个区，省下的投入会被失败成本加倍吃回去；高于这个区，多花的钱买不回等值损失。所以质量成本不是“越省越好”，也不是“越多越好”，而是**找一个结构最优的点**。

举个数：为省预算砍掉 30% 的测试、少花 40 万元，三个月后线上事故赔付 260 万元。账面看，40 万省下、260 万赔出，净亏 220 万，还只是能算出来的部分。关键不在“该不该测”，而在“测到什么程度”——评估成本的边际收益是递减的，砍到“基本功能跑一遍”，等于把缺陷直接放行到外部失败那一栏。正确动作是**两头一起动**：把一部分评估预算挪到预防，让缺陷在产生的那一小时就被拦住。

软件还有一层容易被漏掉的关联成本：修一个缺陷不只是改代码，还要**验证变更并证实有效、与所有受影响的各方沟通、改掉被这个软件牵连的工作产品与过程**。这些活平时不记在“缺陷修复”那一栏里，但它确实在花预算。而一旦软件已经在客户那里运行、补丁已应用或新版本已发布，这一串成本会大幅上升——这也是外部失败成本最难控的原因：你控制得了自己的排期，控制不了客户环境里的牵连面。

板书

- 定义：生命周期中为**预防**不符合、为**评价**是否符合、因**未达到要求（返工）**发生的全部成本
- 两本账：**一致性成本**＝预防＋评估（防失败）｜**非一致性成本**＝内部失败＋外部失败（处理失败）
- 预防：培训・流程文档化・设备工具・编码规范・设计评审・CI｜评估：测试・检查・破坏性测试・代码扫描
- 内部失败：返工・重构・废品（**自己扛**）｜外部失败：索赔・保修热修・SLA 罚金・业务流失（**客户先付**）
- $C_{总} = C_{预防} + C_{评估} + C_{内部失败} + C_{外部失败}$｜规律：预防评估升 → 失败降 → **总成本有最低区**
- 图 7-1：左框“**预测成本**”即预防成本；“在期间防止失败” vs “期间＋完成后处理失败”

提问 为省预算砍掉 30% 的测试、少花 40 万元，三个月后线上事故赔付 260 万元。砍掉的是哪一类成本？换来的是哪一类？（限 2 分钟）

预设回答

- ✅ 正确方向：砍掉的是**评估成本**，换来的是**外部失败成本**。测试属评估（交付前“提前发现”）；线上事故赔付属外部失败（客户已发现）。净亏 220 万元还只是账面上算得出来的部分，客户流失与团队信誉不在账上，但更贵。
- ⚠️ 部分对：答“换来的是非一致性成本”。归组没错，但没答出“砍掉的是哪一类”——这一问的两半都要答；而且非一致性里要能进一步指出是**外部**失败，因为发现者是客户。
- ❌ 常见错：答“砍掉的是内部失败成本”。内部失败是交付前自己发现的返工与废品，测试被砍恰恰会让缺陷活到交付之后，方向正好反了。也不能答“砍的是预防成本”——测试不是预防，是评估。

易错点 / 考点 ① **内部失败与外部失败的判别标准是“谁先发现”**，不是“损失多大”。② 测试、检查属**评估成本**；培训、规范、设计评审、CI 平台属**预防成本**——这两个最容易装反。③ 图 7-1 里的“**预测成本**”就是预防成本，教材译名差异，答题按教材图注写也认。④ “总成本最低区”两个方向都要说：太低被失败成本吃回去，太高买不回等值损失。

过渡 *最低区为什么偏向“预防多花一点”那一侧？因为同一个缺陷，在不同阶段被发现的代价根本不是一个量级——下一页把这条曲线摊开。*

***【思政落点】**：缺陷流出的账，客户先付。内部失败成本再高，也还在自己账上——返工、重构、废品，都是关起门来消化；一旦流出到客户手里，赔付、罚金、保修、口碑与信任，全都由客户先承担，这是最不该发生的一类成本。所以“如实记录缺陷、不瞒报、不把已知问题留到下一环”不是一句口号，它是把成本从**最贵的那一栏**挪回**最便宜的那一栏**的具体做法。落到同学们的分组项目上：测试没过的功能别写进“已完成”，演示不了的别报“做完了”，缺陷登记表上一条都不要删。*

12.7 预防胜于检查：缺陷越晚发现，代价越不是同一个量级（4 分钟）

口播 · 可照读

同一个缺陷，改在需求阶段是“改一句话”，改在上线后是“改代码＋改数据＋通知客户＋补文档＋重跑回归”。**变的不是工时单价，是牵连面**。这一页讲的就是这条曲线。

看图 7-12：横轴是阶段往后走，纵轴是相对修复代价。曲线在左端很平，往右越来越陡——同一个变更（或同一个缺陷）拖到越靠后的阶段处理，相对代价越高，从 1X 一路抬到约 150X。

分阶段看。**需求阶段**，发现手段是需求评审、原型确认，相对代价 1×，改的只是“一句话、重发一版需求说明”。**设计阶段**，靠设计评审、走查，代价 3×，要改接口、改数据模型、同步多个调用方。**编码阶段**，靠代码评审、静态检查，代价 6×，改代码＋改单测＋重新联调。**测试阶段**，靠系统测试、回归，代价 15×，还要加上定位（复现难）、全量回归与延期。**上线后**，只能靠客户反馈与事故复盘，代价 150×，里面多了赔付、热修、数据订正、信任修复。

为什么越晚越贵？三个原因。第一，**牵连面在放大**：含糊的需求会写进设计、编进代码、写进用例，改它就要连带改掉这一串。第二，**回归风险**：为修一个点动主干，可能带出新缺陷，越靠后改动越大、风险越高。第三，**协调成本**：通知客户、排热修窗口、写事故说明、改培训材料，这些在需求阶段根本不存在。

所以“检查只能筛，不能免”。检查是在缺陷**已产生之后**的拦截：拦截率再高也有漏网的，而且每拦一次都要再付一次评估成本。检查是必要的，但它不是免费的防线，更不是第一道防线。

结论一句话：**钱要花在曲线的左端**——把“多测几轮”的钱挪一部分到“评审＋规范＋门禁”，比在右端救火便宜，而且可重复受益。评审花 20 分钟发现一处需求歧义，与上线后花 3 天修复，差的不是 20 分钟和 3 天，是上面那三个原因叠起来的量级。

这里顺手澄清一个常见的口径混淆：评审属于**预防**一侧（放在开工前与过程中，让它别发生），测试与检查属于**评估**一侧（交付前提前发现）。两者都在曲线左端，但性质不同；判断一道题问的是“预防”还是“评估”，看动作是“改过程”还是“发现问题”。

也别把这句话读成“检查没用”。检查仍然是必要防线：有些缺陷只有在集成起来、跑起来才暴露得出来，评审看不出并发下的时序问题。曲线想说的只是**优先级与配比**——把一部分预算从右端挪到左端，同样的钱能买到更多缺陷拦截。举个量级感受：如果 100 个缺陷里有 40 个能在评审阶段就拦下来，那就等于把 40 次 15 倍甚至 150 倍的账单，换成了 40 次 1 到 3 倍的账单；剩下的 60 个再去测试阶段拦，预算反而更宽裕。

最后落一句可执行的做法：**把“发现缺陷”的责任往前压到产生缺陷的那个人手上**——写需求的人负责让需求可验收，写设计的人负责让接口无歧义，写代码的人负责让提交过门禁。每一环都把已知缺陷留在自己这里，整条链的代价曲线就会整体左移。这也是第 7.3 节“管理质量”要展开的核心主张：面向过程，把缺陷挡在门外。

板书

- 同一个缺陷，**变的不是工时单价，是牵连面**
- 阶段 → 代价：需求 **1×**（改一句话）｜设计 **3×**（改接口与数据模型）｜编码 **6×**（改代码＋单测＋联调）｜测试 **15×**（定位难＋全量回归＋延期）｜上线后 **150×**（赔付＋热修＋数据订正＋信任修复）
- 越晚越贵三原因：**牵连面放大 · 回归风险 · 协调成本**
- 检查只能“**筛**”不能“**免**”：拦截率再高也有漏网，且每拦一次再付一次评估成本
- 钱要花在**曲线左端**：评审＋规范＋门禁 > 右端救火
- 口径：评审偏**预防**、测试与检查属**评估**，两者都在左端

提问 评审时花 20 分钟发现一处需求歧义，与上线后花 3 天修复，差的不只是时间——还差什么？（限 2 分钟）

预设回答

- ✅ 正确方向：还差四样——**牵连面**（要同时改代码、改数据库、改文档、改培训材料）、**回归风险**（修一个点动主干，可能带出新缺陷）、**协调成本**（通知客户、排热修窗口、写事故说明）、**信任**（客户“下次还敢不敢用”的判断，改不回来）。这也正是质量成本曲线最低区偏向“预防多花一点”的原因。
- ⚠️ 部分对：答“还差钱，3 天比 20 分钟贵得多”。账没错，但只说了金额；真正拉开量级的是那四样非金额的东西——尤其信任，它不进任何一张表。
- ❌ 常见错：答“差不多，都是修一个歧义，工作量一样”。这是把“缺陷”当成一个静态对象；同一处歧义在不同阶段承载的牵连面完全不同，这正是 1X 到 150X 的全部含义。

易错点 / 考点 ① 曲线方向的判读要能配阶段说：越靠后越贵，且**非线性**。② 150X 是**量级比喻**，不是严格倍数——简答题写“约百倍量级”即可，别写成精确等式。③ 评审与测试的成本归属（预防／评估）是选择题常设的坑。④ “预防胜于检查”不等于“不要检查”：检查仍是必要防线，只是不该当第一道防线。

过渡 *曲线讲的是“什么时候发现最便宜”，可如果项目本身每两周就交付一次，“什么时候”这个前提就变了——敏捷把这条曲线压成了小时级。*

***【思政落点】**：缺陷不往下传。发现问题的第一时间说出来，比事后解释“为什么没发现”更有价值。把已知缺陷留在自己这一环，多花的只是几小时；让它流到下一环，代价就交给别人承担了——工程上叫“不传给下游”，职业上叫**不甩锅**。写代码的人知道某个边界没处理却不说，测试的人知道某条用例没过却标“通过”，最后付账的一定是客户和整个团队的信誉。同学们做小组项目也一样：接口没调通就说没调通，别在周报里写“基本完成”，那是把 1X 的代价推给了 150X 的那一栏。*

12.8 敏捷质量：把质量做进每个迭代，而不是留到最后（3 分钟）

口播 · 可照读

前面那条 1X 到 150X 的曲线有个隐含前提：项目按阶段推进，缺陷可以在“阶段末”被发现。可如果项目每两周就交付一次，这个前提就变了——敏捷干的正是这件事：**把原本压在项目末尾的一大段质量活动切碎、前移、塞进每个迭代**。

一句话定义：敏捷方法强调**将质量管理与审核步骤贯穿项目整个生命周期，而不是仅在项目结束时才执行**。注意它说的是“贯穿”，不是“取消”——质量活动一条没少，只是密度变了。

四个抓手。**第一个，内建质量**：结对编程、同行评审、自动化测试、持续集成门禁，目标不是“测出多少缺陷”，而是**让不合格的提交进不了主干**。**第二个，完成的定义（DoD）**：团队共同认可的“完成”清单，每条可验证——代码已评审、单测通过、持续集成是绿的、文档已更新、演示通过。没有 DoD，“完成”就只是“我觉得写完了”。**第三个，测试先行（TDD）**：先写一个失败的测试，再写最小实现让它通过，然后重构；测试即规格，缺陷在写代码的那几分钟就被发现。**第四个，小批量＋定期回顾**：聚焦小批量交付，让更多可交付成果在每个迭代得到验证；通过定期回顾、分析问题根源、实施改进措施，并在回顾会议上评估改进效果，持续优化质量管理。

拿一张表对比更清楚。质量计划：传统是一份较厚的文档，评审后归档；敏捷是 DoD 加每迭代的质量目标。缺陷发现时点：传统集中在测试阶段爆发，敏捷在每次提交、每个迭代。验收方式：传统一次性用户验收测试，敏捷迭代演示加增量验收。主要度量：传统看缺陷总数与测试通过率，敏捷看**缺陷逃逸率、缺陷到达模式、构建成功率**——三个指标都在问“缺陷从哪漏出去的”，而不是“测了多少”。

一句话对比：**传统把质量当成一个阶段（测试阶段），敏捷把质量当成一个习惯（每次提交）**。两者共享同一条底层规律——缺陷越早发现越便宜；敏捷只是把“早”压缩到了小时级。

顺便回答一个常被问的：敏捷说“可工作的软件高于详尽的文档”，那还要质量计划吗？要，但形态变了——计划不再是厚文档，而是三样可执行的东西：标准（DoD 与验收标准）、门禁（评审通过、持续集成绿、覆盖率达标才能合并）、指标（逃逸率与构建成功率）。缺了这三样，“敏捷”就退化成“没有计划地赶工”，缺陷一层压一层，最后在发布前堆成一次集中测试——那正好回到了敏捷想避免的老路。

板书

- 定义：**质量管理与审核步骤贯穿整个生命周期**，而非仅在结束时执行（是“贯穿”，不是“取消”）
- 四抓手：**内建质量**（评审 · 单测 · CI 门禁，让不合格提交进不了主干）｜**DoD**（每条可验证）｜**TDD**（测试先行）｜**小批量＋定期回顾**
- 对比：质量计划（厚文档 → DoD＋迭代目标）｜发现时点（测试阶段爆发 → 每次提交）｜验收（一次性 → 迭代演示＋增量）
- 度量换了：缺陷总数 · 通过率 → **缺陷逃逸率 · 缺陷到达模式 · 构建成功率**
- 一句话：传统把质量当**阶段**，敏捷把质量当**习惯**；都把“早”往前压

提问 敏捷强调“可工作的软件高于详尽的文档”，那还要不要质量计划？如果要，它长什么样？（限 2 分钟）

预设回答

- ✅ 正确方向：要，但形态变了——**标准（DoD＋验收标准）＋门禁（评审、CI、覆盖率）＋指标（逃逸率、到达模式、构建成功率）**三样可执行的东西。判据是：能不能被机器或评审当场验证，能就是好计划。
- ⚠️ 部分对：说“要，写在迭代计划里”。方向对，但没说清“写成什么”与“怎么起作用”，成了把厚文档换成薄文档——计划的价值不在厚度，在**可验证**。
- ❌ 常见错：说“不要，敏捷靠团队自组织”。自组织不等于没有标准；没有 DoD 与门禁，“完成”就退化成口头承诺，缺陷会在发布前集中爆发——这恰恰是敏捷想避免的。

易错点 / 考点 ① 敏捷质量的三条原始表述（贯穿生命周期、定期回顾、小批量验证）是简答高频点，别只写“敏捷也重视质量”。② DoD 必须**每条可验证**，写成“代码质量良好”不合格。③ 敏捷的度量指标换成了逃逸率与构建成功率，不要把“缺陷总数”当主指标。④ 敏捷不是取消文档与计划，而是改变形态——这条判断题常反向设坑。

过渡 *说了半天“质量要好”，可“好”是个形容词，验收时没人认。下一页把它翻译成指标。*

12.9 规划质量：把“要好”翻译成“可测量、可验收”（4 分钟）

口播 · 可照读

规划质量管理，就是**识别项目及其可交付成果的质量要求与标准，并书面描述如何证明符合这些要求**。难的不是写下来，而是把模糊的“要好用、要稳定”翻译成别人也能验收的指标。

定义里有两个动词很关键：**识别**和**书面描述**。识别是把干系人嘴里的形容词挖出来；书面描述是把它写成“怎么证明”——没有“怎么证明”，质量要求就只是一句祝愿。

九类方法，各管一段。**标杆对照**回答“同类项目做到什么水平”：与可比项目对比，识别最佳实践，为绩效评估提供依据，标杆可以来自组织内、组织外，甚至跨行业。**头脑风暴**回答“有哪几种做法”：自由发言、创意碰撞，形成候选方案并达成共识。**访谈**回答“没说出口的期望是什么”：与有经验的干系人交流，收集显性与隐性的质量需求，要在信任与保密的环境里做。

成本效益分析回答“这笔投入值不值”：比较各方案的成本与预期效益，主要效益包括减少返工、提高生产率、降低成本、提升干系人满意度、增强盈利能力。**质量成本**回答“钱该投在哪一类”：把预防、评估、内部失败、外部失败的分布摆出来，看结构是否失衡。

实验设计回答“哪些因素真正影响结果”：识别影响产品或流程变量的关键因素，据此确定测试的种类与数量、评估其对质量成本的影响。给个数就懂了：假设把测试的缺陷移除率提到 95%，要增加 50% 的测试工作量，但发布后缺陷率只降 5%——这时候维持现有测试水平可能就是更划算的选择。这就是实验设计在做的事：**用数据决定测到什么程度**。

过程流程图回答“哪一步最容易出问题”：显示输入如何在不同步骤转化成输出，常见形式是 SIPOC（供应商—输入—过程—输出—客户），据此估算质量成本、预测可能的质量问题。**逻辑数据模型**回答“口径一致吗”：售后 PPM 的分母是“台”还是“件”、缺陷类别字典怎么定——口径不统一，指标就不可比。**测试与检查的规划**回答“在哪验、验到什么程度”：软件项目里常见 α 测试（内部环境验功能准确性与逻辑完整性）与 β 测试（真实用户环境验性能稳定性与体验）。

共同点一句话：**把“质量要好”变成“可测量、可验收、可比较”**。分工记法：成本效益分析与质量成本管“值不值”，标杆对照管“对标在哪”，头脑风暴与访谈管“听谁的”，实验设计与流程图画“在哪设点”，逻辑数据模型管“口径”，测试与检查的规划管“验到什么程度”。

板书

- 定义：**识别**质量要求与标准 + **书面描述如何证明符合**（没有“怎么证明”，要求就只是祝愿）
- 九类方法：① 标杆对照（对标在哪）② 头脑风暴 ③ 访谈（听谁的）④ 成本效益分析 ⑤ 质量成本（值不值）
- ⑥ **实验设计**（哪些因素影响结果 · 测到什么程度：移除率 95% 需 +50% 工作量、发布后仅降 5% → 可能不值）
- ⑦ 过程流程图／SIPOC（在哪设点 · 估算质量成本 · 预测问题）⑧ **逻辑数据模型**（口径：PPM 分母是台还是件）⑨ 测试与检查的规划（α 内部验功能逻辑／β 真实环境验性能体验）
- 一句话：**把“要好”变成可测量、可验收、可比较**

提问 两个部门报的售后 PPM 相差 10 倍，第一个该查的是什么？（限 1 分钟）

预设回答

- ✅ 正确方向：**先查口径，再查质量**。十有八九是分子分母定义不同——一个用“不良台数／交付台数”，一个用“不良件数／装车件数”，分母差几百倍，PPM 自然对不上。这正是逻辑数据模型要解决的问题：先把实体、属性、口径定下来，指标才有可比性。
- ⚠️ 部分对：答“先查抽样方法或数据来源”。抽样差异会造成偏差，但通常不会造成 10 倍这种量级；先定口径、再查测量、最后才谈现实——顺序错了会白查一周。
- ❌ 常见错：答“先查哪个部门在造假”。一上来就假定有人不老实，会把技术问题变成人事问题；经验规则是：**两个数差一个数量级时，先怀疑定义，再怀疑测量，最后才怀疑现实**。

易错点 / 考点 ① 规划质量管理的定义要写全两个动词（识别＋书面描述），这是简答得分点。② 九类方法要能按“管什么”归类，考试常给场景选方法（例如“两个部门口径不一致”选逻辑数据模型）。③ α 与 β 测试的区别：内部环境验功能与逻辑完整性 vs 真实用户环境验性能与体验，别写反。④ 成本效益分析的五个效益（减少返工、提高生产率、降低成本、提升满意度、增强盈利能力）常作为多选题出现。

过渡 *方法讲了一堆，真正落地要靠七张图——下一页把它们压成一张速查表，一句话一个定义。*

12.10 七种质量工具：一句话定义 + 四个用途组（5 分钟）

口播 · 可照读

七种基本质量工具，也叫 7QC 工具，可以在 PDCA 循环的框架里解决与质量相关的问题。七个名字不好背，但按**用途分组**就好记：谁找原因、谁收数据、谁排优先、谁盯稳定。

第一组，发现原因：因果图、流程图。 因果图又称鱼骨图或石川图，用于直观地展示潜在问题的根本原因及其影响；应用时，**问题陈述放在鱼骨的头部**作为起点，用来追溯问题来源，**回推到可行动的根本原因**。流程图又称过程图，是一种图形化的过程表示方法，用来显示一个或多个输入如何在**不同步骤中转化成输出**；它有助于了解和**估算一个过程的质量成本**（通过工作流的逻辑分支及其相对频率），也有助于项目团队**预测可能发生的质量问题**。

第二组，收集与展示数据：核查表、直方图、散点图。 核查表又称计数表，是一种用于**系统地收集与排列数据**的工具，旨在高效收集与潜在质量问题相关的关键数据；它**特别适用于检查和识别缺陷时记录属性数据**，比如缺陷数量或其后果。直方图是一种**特殊形式的垂直条形图**，用来描述**集中趋势、分散程度和统计分布形状**；请注意这句对比——**与控制图不同，直方图不考虑时间对分布内变化的影响**。散点图又称相关图，标有许多坐标点，用来解释**因变量 Y 相对于自变量 X 的变化**，相关性可以是正相关、负相关或零相关；如果存在相关性，就可以画出一条**回归线**，估算自变量的变化将如何影响因变量。

第三组，找重点排序：帕累托图。 它是一种特殊的垂直条形图，用于**识别造成大多数问题的少数重要原因**，帮助确认问题并对问题排序，变量按**发生频率**排序；横轴上的原因类别作为有效的概率分布，**涵盖了 100% 的可能观察结果**，相对频率逐渐减少，直至以“其他”涵盖未指明的全部其他原因。那句“100%”是它的命门：**它必须把所有观察结果都装进去**，漏掉一类，累计占比就不成立。

第四组，监控过程是否稳定：控制图。 它用来确定一个过程**是否稳定，或者是否具有可预测的绩效**；通常包括**中心线、控制界限和规格界限**。规格上下限是根据协议要求制订的，反映可允许的最大值和最小值，**超出规格界限就可能受处罚**；而**上下控制界限不同于规格界限**——对重复性过程，控制界限通常在过程均值（中心线）正负 3 倍标准差的位置。

这里必须停下来讲清最容易混的一条：**控制界限 ≠ 规格界限**。规格界限来自协议要求，管的是“产品合不合格”；控制界限来自统计计算，管的是“过程稳不稳”。所以，点越出控制界限，只说明过程出现了特殊原因的异常波动，**不等于产品不合格**；反过来，一个过程可能很“受控”，整条带子却压在规格之外——那时过程稳定，产品照样全不合格。这一条第 7.4 节判异时还会再用一次。

板书

- 四组：**发现原因**（因果图 · 流程图）｜**收集与展示**（核查表 · 直方图 · 散点图）｜**找重点排序**（帕累托图）｜**监控稳定**（控制图）
- 因果图：问题陈述放**鱼骨头部**作起点 → 回推到**可行动的根本原因**｜流程图：输入在**不同步骤转化成输出，估算质量成本**（逻辑分支及相对频率）＋**预测质量问题**
- 核查表：**系统收集与排列数据**，特别适用于**检查和识别缺陷时记录属性数据**
- 直方图：描述**集中趋势 · 分散程度 · 分布形状；与控制图不同，不考虑时间对分布内变化的影响**
- 散点图：**Y 相对 X** 的变化（正／负／零相关），有相关性可画**回归线**｜帕累托：少数重要原因，按频率排序，**涵盖 100% 观察结果**
- 控制图：过程**是否稳定 · 绩效是否可预测**；含**中心线 · 控制界限 · 规格界限**；超规格界限**可能受处罚**；**控制界限 ≠ 规格界限**（±3σ）

提问 三问快答——① 已知 500 条缺陷的类别与数量，先整改哪几类？② 想知道某个过程最近有没有失控？③ 想知道一类缺陷背后有多少种可能原因？各用哪张图？（限 2 分钟）

预设回答

- ✅ 正确方向：① **帕累托图**——按频率排序并叠加累计占比；500 条里接线错 320、接口 90，累计约 82%，先整改这两类，这就是“识别造成大多数问题的少数重要原因”。② **控制图**——看有没有点越出控制界限、是否连续 7 点在中心线同侧，判断过程是否稳定、绩效是否可预测。③ **因果图**——问题陈述放鱼头，沿人机料法环测逐类穷举，再回推到可行动的根本原因。
- ⚠️ 部分对：①答“控制图”或②答“直方图”。两张都是条形图，但分工不同：控制图带时间轴、看趋势与稳定性；直方图**不考虑时间**，只看分布形状。这一对混了，判异就会判到分布上去。
- ❌ 常见错：③答“散点图”。散点图看的是**两个变量之间的关系**（X 与 Y），不是“一类问题有几种可能原因”；找原因的图是因果图，散点图是验证原因有没有相关性时才用。

易错点 / 考点 ① **直方图不考虑时间、控制图考虑时间**——这是七工具里最爱考的一条对比。② 帕累托图必须**涵盖 100% 的观察结果**，末尾用“其他”收口；漏项就不成立。③ **控制界限 ≠ 规格界限**：规格界限来自协议（超出可能受处罚），控制界限来自统计（±3σ）。④ 点越出控制界限不等于产品不合格，只说明过程出现特殊原因波动。⑤ 七工具不是七选一，而是一条流水线：**收集 → 展示 → 排序 → 追因 → 监控**。

过渡 *七张图里，只有一张专门用来回答“为什么”——案例里那句“说不清根因”，缺的就是这张图。*

12.11 因果图：把“说不清为什么”拆成人机料法环测（3 分钟）

口播 · 可照读

案例里那句“缺陷集中在少数环节，却说不清根因”，缺的就是这张图。因果图又称鱼骨图或石川图，用于直观地展示潜在问题的根本原因及其影响；应用时，**问题陈述放在鱼骨的头部，作为起点，用来追溯问题来源，回推到可行动的根本原因**。

三步拆开看这个定义。**鱼头**是问题陈述——它是起点，不是结论。**鱼骨**是分类框架：大骨通常按**人、机、料、法、环、测**铺开——人看能力、状态、交接；机看工具、环境、设备是否可靠；料看需求、文档、数据等输入是否合格；法看流程与方法定义了没有、执行了没有；环看外部条件与协作环境；测看测量与检验手段本身靠不靠得住。**鱼尾**才是你要的东西：一条或几条**可行动的根本原因**。

落到软件项目上举例。人：新人未培训就上主线、评审走形式。机：构建服务器不稳定、环境不一致、接口限流。料：需求含糊、接口文档缺字段、测试数据不真实。法：无编码规范、评审无清单、缺陷分类标准不统一。环：工期压得过紧、跨时区协作、多团队改同一模块。测：只压测正常链路、没做弱网与限流场景、指标口径不统一。

图 7-3 这张示例有个细节值得说：它在人机料法环测之外，还另列了**时间**与**能源**两类。这说明**类别可以增减**，按行业与场景调整没问题；但有一条纪律——**别整类漏掉**。整类漏掉，等于有一整片原因你从来没去看过。

四步做法。**第一步，写好鱼头**：一句话、可观测，比如“线束压接工位近三批不良率翻倍”，不写“质量不好”。**第二步，画主骨、分大骨**：按人机料法环测铺开，宁可先多列一类，也别整类漏掉。**第三步，头脑风暴挂原因**：先尽量多列，这个阶段**只记录不评判**；再连续问 3 到 5 层“为什么”，直到落到**可行动**的那一点——“培训不足”不可行动，“新人首次提交未做代码评审”可行动。**第四步，用数据确认**：拿核查表与帕累托图验证哪条原因真的在起作用，只改被验证过的根因，改完回访数据。

三条纪律收口：**问题陈述要具体到可观测；原因要写到可行动；根因必须用数据确认**。鱼骨图负责“穷举可能性”，帕累托图负责“排掉不可能”，两张图配合，才是从“猜原因”走到“改根因”。

板书

- 定义：直观展示潜在问题的根本原因及其影响；**问题陈述放鱼骨头部作起点 → 追溯来源 → 回推到可行动的根本原因**
- 大骨：**人 · 机 · 料 · 法 · 环 · 测**（图 7-3 另列时间、能源 → **类别可增减，整类不可漏**）
- 软件版示例：人（未培训上主线 · 评审走形式）机（构建不稳 · 环境不一致）料（需求含糊 · 文档缺字段）法（无规范 · 无清单）环（工期压紧 · 多团队改同模块）测（只压正常链路）
- 四步：**写鱼头（可观测）→ 分大骨 → 挂原因（只记录不评判 · 连问 3～5 层为什么）→ 数据确认**
- 三条纪律：**具体到可观测 · 写到可行动 · 用数据确认**
- 配合：鱼骨穷举可能性，**帕累托排掉不可能**

提问 把“接口偶尔超时”挂到哪根骨上？只挂一根够不够？（限 2 分钟）

预设回答

- ✅ 正确方向：不够，通常会同时挂上三到四根骨——**机**（第三方限流、网络抖动、连接池过小）、**料**（接口文档没写清超时与重试约定）、**法**（没有统一的超时与重试规范）、**测**（压测只跑正常链路，没做弱网与限流场景）。下一步用连续追问收敛：为什么超时→下游两秒才返回→查询没走索引→数据量涨了十倍没人复查执行计划；追到这里才是可行动的根因。
- ⚠️ 部分对：只挂“机”（网络与服务器）。方向没错，但漏了“料”和“法”——很多超时的根因不在设备，而在**约定没写清、规范没统一**；只改机器，问题会换个场景再来一次。
- ❌ 常见错：挂“人”——写成“开发人员经验不足”。这正是不行动的典型：既不可验证，也无法落地。要往下追到“新人首次提交未做代码评审”这种能被检查、能被拦住的具体行为。

易错点 / 考点 ① 鱼头放的是**问题陈述**（现象），不是结论，更不是“质量不好”这种不可观测的说法。② 大骨六类要能默写：人机料法环测；图 7-3 多列时间与能源，属于类别的可增减，不算错。③ 原因必须写到**可行动**——“培训不足”“意识不强”都不可行动，是简答题的扣分点。④ 因果图只负责穷举，**根因要靠数据确认**（核查表＋帕累托），别把鱼骨图当结论用。

过渡 *七张图现在有了一张“找原因”的，剩下几张怎么把缺陷变成看得懂的形状——下一页开始。*

12.12 核查表、直方图、散点图：把缺陷变成看得懂的形状（4 分钟）

口播 · 可照读

上一页把因果图讲完，我们手里有了一张“该往哪儿查”的地图。可地图要能走，得先有事实。这一页的三张图——**核查表、直方图、散点图**——合起来只解决一件事：**把缺陷变成看得懂的形状**。三者的用途都归在同一组里：**收集与展示数据**。记法不对，什么都看不出来；画法不对，还会得出相反的结论。

先说**核查表**，又称计数表，是一种用于**系统地收集与排列数据**的工具，特别适用于**检查和识别缺陷时记录属性数据**——比如缺陷数量、缺陷的后果。它的价值不在表格本身，而在“**预先定义的结构化表单**”：字段先定死，换谁来填都是同一套口径。

看这张表。十二月八日到九日，四行记录：用户登录模块功能错误 8 条、数据导入模块数据完整性问题 3 条、报表模块页面显示问题 4 条、支付模块性能问题 3 条，**合计 18 条**，其中严重 6 条、一般 12 条。请注意这张表的设计：日期、模块、缺陷类别、严重、一般、合计——**每一个字段都回答一个具体问题**。要是当初只写一句“这两天问题挺多”，后面什么都做不了。

第二张，**直方图**，一种特殊形式的垂直条形图，用来描述**集中趋势、分散程度和统计分布形状**。柱形代表某种属性，柱高是发生次数。这回看的不是缺陷条数，是“未及时提交代码”的原因：代码修复 22 次、团队协作 8 次、需求变更 30 次、设备故障 3 次、代码审查延迟 16 次，合计 **79 次**。最粗那根柱是**需求变更 30 次**，占 $\frac{30}{79} \approx 38.0\%$。

这里有一条必须记牢的区别：**与控制图不同，直方图不考虑时间对分布内变化的影响**。它只回答“分布成什么样”，不回答“随时间有没有异常”。所以直方图的本事是**识别过程失效**：某类原因反复出现，就说明那个环节的流程有问题——比如构建频繁失败，追下去常常是构建没有正确自动化，或者手工构建的核对单不全。

第三张，**散点图**，又称相关图，标出许多坐标点 (x, y)，解释**因变量 y 相对于自变量 x 的变化**。纵轴是程序员的水平，横轴是每个模块的差错数，三十多个点沿着一条自左上到右下的斜带分布，中间再画一条**回归线**：水平越高，每模块差错数越少——这就是**负相关**。另外两种是**正相关**和**零相关**；只有存在相关性，才画得出回归线。

但散点图只说明**相关**，不直接说明**因果**。“水平高的程序员差错少”是真的，可这不等于“换个高手来就必然少犯错”——也许是高手被派去做简单模块。看到斜带，先问一句“有没有第三个变量在背后同时推动这两件事”。

课堂这道题：直方图显示缺陷集中在“需求变更”一类，**能直接说明过程已经失控吗**？不能。失控与否要看**时间顺序**，那是控制图的活；直方图给你的是“**该优先查哪一类原因**”。三张图的分工，一句话：**核查表记事实，直方图看形状，散点图看关系。**

板书 三图归口＝**收集与展示数据** ｜ **核查表**（计数表）：系统收集与排列数据，**特别适用于记录属性数据**；示例 12-08～12-09 共 **18 条**（严重 6／一般 12）——价值在**字段先定死** ｜ **直方图**：描述**集中趋势 · 分散程度 · 分布形状**，柱高＝次数；示例**未及时提交代码 79 次**，需求变更 30 次＝$30/79 \approx 38.0\%$；⚠️ **不考虑时间对分布内变化的影响** ｜ **散点图**（相关图）：(x, y) 点＋**回归线**，正相关／负相关／零相关；示例**水平↑→差错↓**＝负相关 ｜ 三处看错：①用直方图判失控 ②把相关当因果 ③字段各写各的 ｜ 分工：**记事实 → 看形状 → 看关系**

提问 核查表上“严重”和“一般”分两列记，只是为了统计方便吗？如果只记一个“合计”列，后面哪一步会做不下去？

预设回答

- 答“不是，分开记是为了后面按严重程度排优先级”：✅ 对。属性数据之所以要细分维度，就是因为**不同维度的处置方式不一样**——一条严重缺陷和四条显示问题，绝不能同等对待。帕累托图能按“条数”排，也能按“后果严重度”排，前提就是核查表当初把两列分开了。
- 答“就是为了统计方便，合计列够用”：⚠️ 半对。合计列能回答“多少”，回答不了“多要紧”。等到了要定整改顺序的时候，就得回去重新翻原始记录——那正是核查表想省掉的返工。
- 答“严重缺陷直接升级处理，不进核查表”：❌ 典型错。严重缺陷恰恰最该进表：它是后面做**缺陷到达模式**、做**根因分析**最关键的样本，漏记等于把最贵的数据丢了。

易错点 / 考点 ① **直方图不看时间**——这是它与控制图唯一但决定性的区别，判断题几乎年年设。② **散点图只证相关、不证因果**，简答题常要求“举一个相关但非因果的例子”。③ 核查表是**属性数据**的载体，字段必须**预先定义**；事后补记的表不算结构化收集。④ 直方图柱高是**次数**不是占比，占比要自己除总数（79）。

过渡 *79 次这个总数先记住——下一页那根最粗的柱子会被搬到最左边，再叠一条累计曲线，图的名字就叫帕累托。*

12.13 帕累托图示例：500 条缺陷，先改累计 82% 的那两类（3 分钟）

口播 · 可照读

缺陷一多，最怕“每样都改一点”。每样都改一点，等于每样都没改透，下个版本还是同样的坑。所以这一页要解决的是一个排序问题：**先改哪一类**。

帕累托图是一种特殊的垂直条形图，用于识别造成大多数问题的少数重要原因，帮助确认问题并对问题排序。它描述的变量**按发生频率排序**；横轴上的原因类别作为有效的概率分布，**涵盖 100% 的可能观察结果**；每个类别的相对频率逐个递减，**直到用“其他”来涵盖未指明的全部其他原因**。记住这句：**“其他”是兜底，不是垃圾箱**——它存在的意义，就是让这张图的横轴真的覆盖住 100%。

下面是一组示例数据，某模块共 **500 条**缺陷：**接线错 320 条、接口 90 条、文档 40 条、其他 50 条**。先算单项占比：接线错 $\frac{320}{500} = 64.0\%$，接口 $\frac{90}{500} = 18.0\%$，文档 $\frac{40}{500} = 8.0\%$，其他 $\frac{50}{500} = 10.0\%$，四项相加正好 100%。

再算累计，公式是 $\text{累计占比}_k = \frac{\sum_{i=1}^{k} d_i}{D}$，这里 $D = 500$。第一类：$\frac{320}{500} = \mathbf{64.0\%}$；前两类：$\frac{320+90}{500} = \frac{410}{500} = \mathbf{82.0\%}$；前三类：$\frac{450}{500} = \mathbf{90.0\%}$；加到最后：$\frac{500}{500} = \mathbf{100\%}$。**64.0、82.0、90.0、100**，这四个数请写在笔记本上。

结论就是标题那句话：**前两类（接线错 320 ＋ 接口 90）累计 82.0%，优先整改这两类**。为什么是两类而不是一类？只改接线错一类，覆盖 64.0%，剩下 36% 的缺陷还在往外出；把接口一起算进来，82.0% 的缺陷被罩住，而整改对象只从一类变成两类——这就是“**关键少数**”的分量。

这张图两种看法：**柱形看单类多少，折线看累计占比**。柱形告诉你“谁最多”，折线告诉你“**改到第几类为止性价比最高**”。两条一起看，才叫帕累托；只看柱形，那是直方图。

原理上是**帕累托原则**：80% 的问题通常源自 20% 的原因，所以也叫“**80/20 原则**”；它的学名叫 **ABC 重点管理法**——A 类是累计到 80% 左右的那几类，先管；B 类其次；C 类最后。软件里怎么用？按组件统计缺陷数，缺陷扎堆的组件要安排资深成员做设计评审和代码评审；也可以拿它分析配置管理数据，**频繁变更的组件**往往说明代码结构该重构了，或者缺陷修复的策略本身有问题。

最后一句话要留着：**帕累托不是一锤子买卖**。这一轮改完接线错和接口，下一轮把数据重新采一遍、重新排一次——原来的“文档 40 条”可能就变成新的第一个柱子。这就是 PDCA 里那个 A，改完要回头看。

板书 **帕累托图**＝特殊的垂直条形图｜用途：识别**造成大多数问题的少数重要原因**＋确认与排序｜按**发生频率**降序｜横轴类别**涵盖 100% 观察结果**，“其他”兜底｜示例（$D=500$）：接线错 **320**｜接口 **90**｜文档 **40**｜其他 **50**｜单项占比 **64.0／18.0／8.0／10.0%**｜累计 **64.0／82.0／90.0／100%**（$\frac{320}{500},\frac{410}{500},\frac{450}{500},\frac{500}{500}$）｜**结论：前两类累计 82% 优先整改**｜两种看法：**柱形＝单类多少，折线＝累计占比**｜80/20 原则＝**ABC 重点管理法**｜改完**重采数据再切一次**

提问 如果某一类缺陷只有 5 条，但每一条都会导致产线停线，帕累托图按“发生频率”排序会把它排在最后——这时还能照图上的顺序改吗？

预设回答

- 答“不能，要按后果严重度重排一次”：✅ 到位。定义的原话是“按类别排列条形，**以测量频率或后果**”——频率和后果是两个可选的度量口径。停线 5 次这种后果，就该用后果当纵轴再画一张，两张对照着定顺序。
- 答“能，帕累托就是按频率排的”：⚠️ 只记住了半句。按频率排是常规做法，但图的目的是“**识别少数重要原因**”，重要性不一定等于次数，把口径固定死反而违背了工具的初衷。
- 答“5 条太少，不构成少数重要原因”：❌ 错在把“重要”等同于“多”。少数重要原因的关键词是**重要**；5 条停线缺陷造成的损失可能超过 320 条接线错。

易错点 / 考点 ① **单项占比与累计占比必须分开写**——考试里最常见的失分是把 82% 写成接口一类的占比（接口是 18.0%）。② 分母恒为总数 $D=500$，不是“前一类条数”。③ “涵盖 100% 的观察结果”是定义原话，靠“其他”兜底实现，选择题常考“其他类能否省略”。④ 帕累托**既能按频率排、也能按后果排**，别把频率当成唯一口径。⑤ 与直方图的区别要能一句话说出：**直方图不排序、不累计**。

过渡 *工具层面到这儿基本齐了，可规划质量这个过程，最后总要交出几样能被别人拿去用的东西——下一页看成果。*

12.14 规划质量的成果：一份计划、一把尺子（3 分钟）

口播・可照读

7.2 的产出不是“等检查完再说”，而是**先把怎么防、怎么测、按什么标准算合格写清楚**。落地成两件东西：**质量管理计划**写“办法”，**质量测量指标**写“尺子”。

先说**质量管理计划**。它写七件事：①采用的质量标准；②质量目标；③质量相关的角色与职责；④需做质量审查的可交付成果与过程；⑤规划的质量控制与管理活动；⑥使用的质量工具；⑦不符合项处理、纠正措施、持续改进等主要程序。七条背不下来没关系，记住它回答的是**“这个项目打算怎么管质量”**——标准是谁定的、谁负责、查什么、用什么查、查出问题怎么办。

再说**质量测量指标**。它的定义是关键：描述项目或产品的属性，并**明确在控制质量过程中验证符合程度的方法**。请注意后半句——“验证符合程度的方法”，它把指标和 7.4 直接焊在了一起。一条合格的指标要有四样东西：**口径、阈值、采集时点、责任人**。

看这张表，全是示例：**测试需求覆盖率**，口径是“已覆盖需求的测试用例数／需求总数”，目标 ≥95%；**遗留缺陷密度**，发布前未关闭缺陷数／千行代码，目标 ≤3 个；**缺陷平均修复时长**，从提交到关闭的平均天数，目标 ≤5 天；**严重缺陷逃逸数**，上线后由客户发现的严重缺陷数，目标 0 个；**客户满意度**，验收问卷百分制，目标 ≥90 分。每一条都能算、能判、能追。

反过来看三个反例：**“界面要友好”“性能要好”“代码要规范”**——都**不是**质量测量指标。没有口径、没有阈值，判不了合格；到了控制质量阶段，根本核不了。指标写成这样，7.4 就只能靠感觉验收。

还有一件容易被漏掉的成果：**过程改进计划**。它写过程边界、过程配置、**过程测量指标与控制界限**、绩效改进目标。注意“控制界限”这四个字——规划阶段就要给每个过程测量指标定一条界，比如“每 20 份提案中打回不超过 1 份”。定不出界，后面 12.19 那张控制图就没有线可画。

最后提醒一句：**软件规模测量不等于代码行数**。行数会因语言和编码风格而异，不能准确反映质量；良好的注释和模块化设计反而可能让行数变多而更好维护。要衡量规模，用需求数、功能点、特性数、用户故事或用例数更靠谱。

板书

- 2 成果＝**一份计划＋一把尺子**｜**质量管理计划七项**：质量标准・质量目标・角色职责・审查对象・质控与质量活动・质量工具・不符合项与改进程序｜**质量测量指标**：描述属性＋**明确验证符合程度的方法**；四要素＝**口径・阈值・采集时点・责任人**｜示例：覆盖率 ≥95%｜遗留缺陷密度 ≤3/千行｜修复时长 ≤5 天｜严重缺陷逃逸 0｜满意度 ≥90｜反例：**“友好”“要好”“要规范”**——无口径无阈值，判不了｜另产出**过程改进计划**：过程边界・过程配置・**过程测量指标与控制界限**・改进目标｜⚠️ 规模测量别只数代码行

提问 （本页互动题）选择下面哪一条可以直接写进质量管理计划，当作质量测量指标？　A．系统要稳定可靠，不能老出问题；B．每月因故障造成的计划外停机时间 ≤ 30 分钟；C．代码要写得漂亮，注释要写全

预设回答

- ✅ 选 B——**每月因故障造成的计划外停机时间 ≤ 30 分钟**。点评：有口径（计划外停机时间／月）、有阈值（≤ 30 分钟）、可采集可判定，到了 7.4 控制质量能直接比对；补上采集时点与责任人，四要素就齐了。
- ⚠️ 选 B 但只说“因为它带了个数字”：方向对、理由浅。点评：**加个数字不等于指标**——“稳定性 ≥1”“响应快于 1 秒”都能带数字，却算不出、判不了；判分看的是口径、阈值、采集时点、责任人四样。
- ❌ 选 A 或 C：“稳定可靠、不能老出问题”“写得漂亮、注释写全”——没有口径、没有阈值，判不了合格，等于把难题推到验收那天。

易错点 / 考点 ① **质量测量指标必须可验证**——这是它与“质量目标口号”的分界线，简答题高频。② 质量管理计划是**项目管理计划的组成部分**，不是独立文件。③ **过程改进计划里的“控制界限”**是 12.19 的前置，别当成两件事。④ 代码行数作为规模度量有系统性偏差，选择题常设“行数越多质量越差”这类陷阱。

过渡 *计划有了、尺子有了，执行阶段谁负责保证大家真的照尺子做？这就是 7.3——管理质量。*

12.15 7.3 管理质量：面向过程，把缺陷挡在门外（3 分钟）

口播 · 可照读

这一页是本讲的第一个分水岭，请把标题里的四个字圈起来：**面向过程**。

管理质量是把组织的质量政策转化为具体的质量计划、标准和流程，并通过过程改进、质量审计等手段，确保项目活动符合质量要求。它的核心就是那句老话：**预防胜于检查**——提前优化过程、规范操作，而不是指望后期测试把缺陷捞回来。所以它管的对象是**过程**：过程稳，成果才稳。

看这条三段式的链子。**输入**：质量管理计划；项目文件，包括质量控制测量结果、质量测量指标、风险报告、经验教训登记册；以及组织过程资产里的质量管理体系。**方法与技术**：七种基本质量工具、核对单、根本原因分析、**质量审计**、**过程分析**、**面向 X 的设计**、问题解决、质量改进。**成果**：质量报告、测试与评估文件、变更请求，以及更新的项目管理计划与项目文件。

注意这条链子的气质：它没有一个动作是在“判定这一件合格还是不合格”。它全部的动作都在问另一件事——**为什么缺陷会反复出现，产生缺陷的条件能不能拿掉**。

所以这两栏一定要分清。**管理质量，也就是质量保证，事前预防＋过程优化**：聚焦优化过程，流程改进、标准与流程制定、过程审计；动作是审计、过程分析、改流程。**控制质量，事后验证**：聚焦检查成果，测试、验收、判合格与否；动作是检查、测试、抽样、比对标准，回答“这一件成果到底行不行”。一句话记法：**QA 问“方法对不对”，QC 问“这一件行不行”**。

管理质量有四个关键作用：①确保质量管理活动与组织质量政策、项目质量计划保持一致；②识别质量管理过程中的低效环节，推动流程优化；③降低因质量问题导致的返工成本，提升干系人信心；④**为控制质量提供明确标准**，形成“规划—执行—检查—改进”的闭环。第四条最容易被忽略：QA 不是在 QC 之外另开一摊，它给 QC 定标准。

补一个很好用的思路：**个体软件过程**。工程师记录自己的时间与缺陷数据，并且**在编译与单元测试之前先做代码评审**。经 PSP 训练的工程师，在代码评审阶段平均每小时发现 **2.96** 个缺陷，而在单元测试阶段只有 **2.21** 个。同样的缺陷，早一步发现，代价小一截——这不是口号，是同一批人身上的两组数。

课堂这道题：团队每周都在加班修缺陷、测试覆盖率也很高，**能说明管理质量做得好吗**？恰恰相反。这更像**过程在持续产出缺陷**的信号——缺陷多到要靠加班和全量回归兜住。测试做得再勤，也只是在漏水的桶下接水。

板书

- 3 **管理质量（质量保证）＝把组织质量政策转化为具体的质量计划 · 标准 · 流程** ｜ 核心：**预防胜于检查**；对象＝**过程**（过程稳，成果才稳） ｜ 链：计划＋项目文件（**质量控制测量结果** · 质量测量指标 · 风险报告 · 经验教训）→ 七工具 · 核对单 · **RCA · 质量审计 · 过程分析 · DFX** · 问题解决 · 质量改进 → **质量报告 · 测试与评估文件** · 变更请求 · 更新的计划与文件 ｜ **QA 面向过程／QC 面向成果**：QA 审计 · 过程分析 · 改流程；QC 检查 · 测试 · 抽样 · 比对 ｜ 记法：**QA 问方法对不对，QC 问这一件行不行** ｜ 四作用：对齐政策 · 优化流程 · 降返工 · **为 QC 定标准** ｜ PSP：评审 **2.96**／时 > 单测 **2.21**／时

提问 同一个团队，质量管理计划写得很漂亮，审计也做了，可缺陷还是每月往外冒。按 7.3 的思路，接下来该查什么？

预设回答

- 答“查流程：缺陷为什么反复出现在同几个环节，评审是不是形同虚设、变更有没有受控”：✅ 对了方向。再补一层：审计“做了”不等于审计“有效”——要看审计发现的**纠正与预防措施**有没有真的落到流程里，没落地的审计只是一份文档。
- 答“加大测试力度，多跑几轮回归”：⚠️ 这是在 QC 侧加水，桶还是漏的。测试能挡住一部分流出，但改变不了缺陷的产出率。
- 答“换人，或者加人”：❌ 把过程问题当人力问题。人换了，流程照旧，缺陷会跟着新来的人重演——这也正是 PSP 强调“记录自己的缺陷数据、改自己的过程”的原因。

易错点 / 考点 ① **QA 面向过程、QC 面向成果**——这一页的分界线是本章最核心的对照，简答与选择题都绕不开。② 质量审计属于**管理质量**的方法，不是控制质量的方法；统计抽样属于**控制质量**。③ 管理质量的成果是**文件**（质量报告、测试与评估文件），控制质量的成果是**成果与信息**（核实的可交付成果、质量控制测量结果、工作绩效信息）——两栏一混就全错。④ “测试覆盖率高”不是 QA 做得好，是 QC 做得勤。

过渡 *知道要改过程，可过程在哪儿？怎么改？下一页是五个能直接上手的动作。*

12.16 管理质量的方法：从根本原因到质量改进（4 分钟）

口播 · 可照读

管理质量说起来抽象，落到动作上是五件事：**挖根因、看别人怎么做、看自己的流程哪里打结、在设计阶段就把质量装进去、用循环把改进固化下来**。

第一，根本原因分析。它是一种系统化分析技术，旨在确定引发偏差、缺陷或风险的根本原因；不仅识别表面现象，还深挖深层次成因。有个例子特别典型：用户抱怨功能运行缓慢，表面原因是代码效率低，**深层原因是需求分析阶段忽略了性能需求**。于是两手一起做——优化代码，同时改进需求分析流程，系统性能上去了，类似问题也不再重演。请注意“**一项根本原因可能引发多个偏差**”，所以消除根因的价值是“一改一大片”；反过来，只改表面现象，就是按下葫芦浮起瓢。

第二，质量审计。定义的原话是：通过**结构化且独立**的方式，评估项目活动是否符合组织和项目的政策、流程及程序。关键在“独立”两个字——通常由**未直接参与项目**的独立团队执行，比如组织内部的审计部门、项目管理办公室，或者外部专业审计机构。五个目标：识别良好实践与最佳实践；发现违规做法与不足；分享行业内的优秀实践；改进过程执行；积累经验教训。执行方式两种：**事先安排**的审计，放在关键节点或阶段结束时，比如测试阶段结束后评估覆盖率是否达标；**随机抽查**，不提前通知，查日常活动的合规性，比如随机抽查代码是否合规。

我自己最想强调这一句：**审计的核心在于后续问题的纠正与预防**。纠正措施解决当前发现的问题——需求文档不一致，就修订并重新评审；预防措施防止同类问题重复发生——引入自动化变更管理工具，让所有变更被记录和追踪。**发现问题只算一半，措施落到流程里才算完。**

第三，过程分析。沿流程一步步走，找出瓶颈、冗余步骤、返工环节和非增值活动，并为过程设定测量指标与控制界限。它的输出正是 12.14 里那份过程改进计划的内容——过程边界、测量指标、改进目标。

第四，面向 X 的设计。一组设计原则，针对特定方面优化设计。X 可以代表可靠性、可维护性、装配性、制造成本、服务性、可用性、安全性和质量等多个维度。软件里最常见的是**面向性能的设计**：电商平台优化数据库查询、加缓存，响应时间下来了，转化率和满意度跟着上去。要点是**在设计阶段就把目标变成设计约束和检查项**，而不是等测出来慢再回头改架构。

第五，质量改进方法，两件家伙。**PDCA 循环**：计划、执行、检查、行动四个阶段滚动推进——计划阶段按质量数据分析问题、制订改进计划；执行阶段落实措施；检查阶段评估结果与目标对比；行动阶段固化有效做法、为下一轮做准备。**六西格玛**：按定义、测量、分析、改进、控制推进，目标是**每百万件产品的缺陷数小于 3.4 个**。注意这个数的量级——3.4 个／百万，不是 3.4%。

课堂这道题：项目组自查时发现测试用例漏了边界值，自己补上并在周报里记了一笔，**这算质量审计吗**？不算。审计要求**独立于被审计活动、结构化评估、并形成可跟踪的纠正与预防措施**。自查发现问题是好事，性质上属于团队自检。另外那个陷阱选项也要破一下：审计**不是只能由外部第三方做**，PMO 完全可以。

板书 五方法 | ① **根本原因分析**：系统化找根因，**一项根因引发多个偏差**；例：慢→表面代码效率低，根因＝需求阶段漏了性能需求→**改代码＋改需求分析流程** ② **质量审计**：**结构化＋独立**（审计部门・PMO・外部机构，**不能自己审自己**）；五目标：良好实践・违规不足・分享实践・改进执行・积累教训；方式：**事先安排／随机抽查**；核心＝**后续纠正与预防** ③ **过程分析**：瓶颈・冗余・返工・非增值；输出**过程测量指标与控制界限** ④ **面向 X 的设计**：可靠性・可维护性・安全性・性能・可用性・成本→**设计约束＋检查项** ⑤ **质量改进**：**PDCA** 四阶段；**六西格玛 3.4 个／百万**

提问 （本页互动题）选择项目组自查时发现测试用例漏了边界值，自己补上并在周报里记了一笔。这算质量审计吗？ A．算：发现了问题就是审计；B．不算：审计须独立于被审计活动，且要有结构化评估与可跟踪的纠正措施；C．不算：审计只能由外部第三方机构做，PMO 也不行

预设回答

- ✅ 选 B（不算）——**审计＝结构化＋独立**，而且必须形成可跟踪的纠正与预防措施。点评：团队自查发现漏了边界值、补上并记一笔，是有价值的自检，但缺了“独立于被审计活动”这一条，性质上不是审计。
- ⚠️ 选 C（不算，但理由是“只能外部第三方做”）：结论对、理由错。点评：审计由**未直接参与项目**的独立团队执行，组织内的审计部门、**PMO**、外部机构都可以；“独立”指与活动无关，不是“必须是外人”。
- ❌ 选 A（算：发现问题就是审计）：发现问题是审计的常见结果，不是它的定义。点评：自查补用例属即时纠正，缺结构化评估，也没有落到流程层面的预防措施——下一轮还会漏同样的边界值。

易错点 / 考点 ① 质量审计必须**独立于被审计活动**，且**PMO 也可以做**（不必是外部第三方），这两个方向都会考。② 审计的落点是**纠正＋预防**，只写“发现问题”不算完整。③ **过程分析**要输出测量指标与控制界限，和 12.14 的过程改进计划是同一套东西。④ **六西格玛的 3.4 是“每百万件 3.4 个”**，别写成 3.4%。⑤ RCA 的例子要能复述：表面是代码慢，根因是需求阶段漏了性能需求。

过渡 *五个动作做完，总要交出能被干系人看懂、能被下一轮接着用的东西——12.17 就看这三样成果。*

12.17 管理质量的成果：质量报告与测试评估文件（3 分钟）

口播・可照读

7.3 的成果要能**拿给干系人看、拿给下一轮改**：一份说清“问题—影响—建议—行动”的**质量报告**，一套能支撑验收的**测试与评估文件**，再加上把“过程好不好”变成数的一组**过程质量度量**；此外还有变更请求与更新的计划、文件。

先说**质量报告**。它是以图形、数据或定性描述形式呈现的文档，帮团队识别问题、采取纠正措施。一份合格的质量报告写四类内容：**质量管理问题、问题影响、改进建议、纠正措施与行动计划**。看这张示例表，案例是教务管理系统的选课模块：问题——选课模块加载超过 5 秒、部分时段显示错误、需求与设计文档版本不一致；影响——学生没能按时选课、教务人员不满、文档不一致导致理解偏差并延误修复；建议——过程上优化渲染与查询，产品上异步加载加缓存，文档上统一平台加版本控制；行动——重构前端、修正数据同步逻辑、启用文档版本控制、安排 100% 回归测试。请注意这四栏的因果关系：**问题不带影响，没人给资源；影响不带建议，等于抱怨；建议不带行动和责任人，就是空话。**

第二样，**测试与评估文件**：测试计划、测试用例、质量核对单、需求跟踪矩阵。它们的用处是证明“每条需求都被测过、每个结论都有依据”，同时也是 7.4 控制质量的重要输入。需求跟踪矩阵在这里特别关键——它逐条列出需求与测试的对应关系，谁没被覆盖一眼就看出来。

第三样，**过程质量度量**，把“过程好不好”变成数，四个常用指标：过程缺陷密度、整体缺陷清除率、阶段性缺陷清除率、缺陷到达模式。前两个有公式。

过程缺陷密度 $\mathrm{DIPF}=\dfrac{D}{S}$：$D$ 是某阶段或整个项目被发现的缺陷数，S 是被测产品的规模。示例数据里，需求阶段 5 个缺陷／200 条需求＝0.025；设计阶段 3／50＝0.06；编码阶段 10／100000 行＝0.0001；测试阶段 7／1000 条用例＝0.007。这四个数**不能横向比大小**——分母的单位都不一样，一个按条需求、一个按个模块、一个按行代码。它只适合**同一阶段跨版本纵向比**。

整体缺陷清除率 $\mathrm{DRE}=\dfrac{D_1}{D}$：$D_1$ 是开发阶段发现的缺陷数，D 是总缺陷数。示例里总计 33／41＝**80.49%**。这个数说明什么？说明**两成缺陷是发布后才被用户替你发现的**。业界平均整体缺陷清除率大约 80%，一些重视质量的公司，主流产品可以做到 95%。

还有一个反直觉的坑：**DIPF 过低不等于质量好**。可能是产品确实好、缺陷难找；也可能是**方法不当、能力不足，压根没把缺陷找出来**。前者值得庆祝，后者是定时炸弹——判别办法是查流程与能力，看是不是评审走过场、测试覆盖率虚高。

其余三样成果别漏：**变更请求**（走整体变更控制）、**更新的项目管理计划**（含质量管理计划与各基准）、**更新的项目文件**（问题日志、经验教训登记册、风险登记册）。

板书

- 3 成果 | ① **质量报告**：**问题 → 影响 → 改进建议 → 纠正措施与行动计划**（示例：选课加载 >5 秒・时段显示错・文档版本不一致 → 学生未按时选课／教务不满／延误修复 → 优化渲染查询・异步加载＋缓存・统一平台＋版本控制 → 重构前端・修正同步・100% 回归）② **测试与评估文件**：测试计划・用例・质量核对单・**需求跟踪矩阵**（也是 7.4 的输入）③ **过程质量度量**：$\mathrm{DIPF}=\frac{D}{S}$（0.025／0.06／0.0001／0.007——**单位不同，不可横比**）；$\mathrm{DRE}=\frac{D_1}{D}$（示例 $33/41=80.49\%$；业界 ≈80%，优秀 ⩾95%）④ 变更请求 ⑤ 更新的计划 ⑥ 更新的文件 | ⚠️ **DIPF 低 ≠ 质量好**

提问 某模块整体缺陷清除率只有 60%，团队的解释是“我们测试很严格，问题都提前发现了”。这个解释站得住吗？

预设回答

- 答“站不住：清除率的分母是总缺陷数，60% 意味着四成缺陷是发布后才暴露的”：✅ 准确。可以再补一句改进方向：不是再跑几轮测试，而是把**评审与测试前移**，让需求和设计阶段就把缺陷挡住。
- 答“站得住，测得严就说明质量意识好”：⚠️ 把“测得严”和“测得早”混了。清除率统计的是**缺陷在哪个阶段被清掉**，发布后才清掉的，代价最高。
- 答“60% 说明还有 40% 的缺陷没修”：❌ 口径错。不是“没修”，是**“发布后才发现、才修”**——已经流出去了，客户先替你付了账。

易错点 / 考点 ① **DRE 的分母是总缺陷数**（开发中发现＋发布后发现），不是“开发中发现的”。② **DIPF 的低值有双解**，回答“说明质量好”是不完整的。③ DIPF 各阶段**单位不同不可横比**，这是本章少见的计算陷阱。④ 质量报告的四栏是**因果链**，缺“影响”或“行动计划”都不合格。⑤ 测试与评估文件同时是 7.3 的成果和 7.4 的输入——这句话判断题常考。

过渡

- *3 一直在管过程，可客户最终只看一件事：交出去的这批东西，到底合不合格。这就是 7.4，控制质量。*

12.18 7.4 控制质量：面向验收，用数据判合格（4 分钟）

口播 · 可照读

7.3 面向过程，7.4 面向成果——标题里“**面向验收**”四个字，就是这一页的立场。

控制质量是监督并记录质量活动的执行结果，以便评估绩效并推荐必要变更的过程，确保项目输出完整、正确、满足客户期望。它的主要作用有两个：一是**核实可交付成果和工作是否符合主要干系人的质量要求，为最终验收提供依据**；二是确认项目输出达到预期目标，符合所有适用的标准、要求、法规和规范。**该过程贯穿整个项目生命周期**——不是阶段末才做一次。

它的核心动作是三个词：**测量、控制、报告**。具体说，就是把可交付成果和各种需求逐条比对——协议、方针、标准、计划、需求和期望，比出偏差、推荐改进措施。在敏捷项目里，每次迭代都有测试、演示和回顾评审；内部的质量控制由团队成员做，结对编程、同行评审、功能测试；外部的质量控制常在特性级和发布级做独立测试。

看这张六维对照表，这是本页的主干。**关注对象**：QA 是过程、做法与方法，QC 是成果、可交付成果与产品。**回答的问题**：QA 问“方法对不对、会不会再出同类问题”，QC 问“这一批成果是否符合标准与验收要求”。**主要动作**：QA 是质量审计、过程分析、根本原因分析、面向 X 的设计；QC 是检查与评审、测试与产品评估、统计抽样、绩效审查。**时点**：QA 贯穿全周期偏事前事中；QC 贯穿全周期偏事中与交付前。**主要成果**：QA 交质量报告、测试与评估文件、变更请求；QC 交质量控制测量结果、核实的可交付成果、工作绩效信息。**常见误区**：QA 侧是把“测试做全”当成质量保证的全部；QC 侧是把“过程受控”当成“产品合格”。

四类方法，一类一类过。① **检查与评审**：对工作产品做审查，验证是否符合书面标准，并用测量数据记录。形式有走查、同行评审、演示、可用性测试；走查是非正式评审，通过共同讨论查逻辑一致性和符合性。软件早期阶段——需求和设计文档阶段——走查和检查尤其有效。有一句话要背下来：**检查被认为是识别软件及文档缺陷与疏漏最有效的方法之一**。而且检查不只发现缺陷，还能**确认已修复的缺陷**。

② **测试／产品评估**：结构化、有组织的活动，按项目需求提供关于被测产品的客观信息，目标是发现错误、缺陷、漏洞和不合规问题。类型有单元测试、集成测试、黑盒测试、白盒测试、接口测试、回归测试。**TDD** 先写测试再写代码，靠自动化工具（JUnit、TestNG、Selenium 这类）保证执行一致、可重复。开发早期还可以用脚手架模拟未完成部分的输入输出，提前做集成或回归。

③ **统计抽样**：从目标总体中取一部分样本检查分析，用样本推断总体。示例：从 110 个模块里按 10% 抽检 **11 个**；另一组示例是登录模块 30 个抽 3 个（合格）、导入模块 50 个抽 5 个（发现缺陷 1 处）、报表模块 30 个抽 3 个（合格）。要点有两个：**抽样的频率和规模要写进质量管理计划**，且样本要**有代表性**——抽得不代表，结论就是错的。

④ **绩效审查**：对实际结果进行测量、比较和分析，评估是否满足质量管理计划里的质量测量指标。看这张表：功能实现率目标 ≥95%、实际 90%，**未达标**，措施是加强需求管理；代码缺陷率目标 ≤3、实际 3.5，**未达标**，措施是增加代码评审；用户满意度目标 ≥90 分、实际 92 分，**达标**，措施是继续保持。三行一对照，你就明白 12.14 为什么非要指标带阈值——**没有目标值，这张表一个字都填不出来**。

课堂这道题：哪一项属于控制质量，而不是管理质量？**按测试用例对支付模块做回归测试，并记录通过率与缺陷**——这是对具体成果做测试并记录结果，属于核实成果符合要求。另外两个选项，“审计团队是否按规程执行代码评审”查的是做法是否被遵守，“修改测试流程减少漏测”改的是流程，都归管理质量。

板书

- 4 **控制质量＝监督并记录执行结果→评估绩效→推荐必要变更**；作用：**核实成果符合质量要求，为最终验收提供依据；贯穿全生命周期** ｜ 核心：**测量・控制・报告**（与协议・标准・计划・需求逐条比对） ｜ **QA／QC 六维对照**：对象（过程／成果）・问题（方法对不对／这一批行不行）・动作（审计・过程分析／检查・测试・抽样・审查）・时点（事前事中／事中与交付前）・成果（报告文件／测量结果・核实成果・绩效信息）・误区（测试＝QA／受控＝合格） ｜ **四类方法**：①**检查与评审**（走查・同行评审・演示・可用性测试；**识别缺陷最有效之一**，也能确认已修复）②**测试／产品评估**（单元・集成・黑/白盒・接口・回归；TDD）③**统计抽样**（110 抽 10%→**11 个**；30→3・50→5・30→3；比例写进计划）④**绩效审查**（90% vs ≥95% 未达・3.5 vs ≤3 未达・92 vs ≥90 达标）

提问 （本页互动题）选择以下哪一项属于控制质量，而不是管理质量？ A．审计团队是否按规程执行代码评审；B．按测试用例对支付模块做回归测试，并记录通过率与缺陷；C．修改测试流程，减少边界值场景被漏测的情况

预设回答

- ✅ 选 B——**按测试用例对支付模块做回归测试，并记录通过率与缺陷**。点评：对象是**成果**，动作是测试与产品评估，产出是测量结果，问的是“这一批成果合不合要求”，正是控制质量。
- ⚠️ 选 C（改测试流程、减少漏测）：动作看着像质量控制，对象却是**过程**——“让流程以后不再漏”属过程分析与过程改进，归管理质量。点评：判断只看一句——**动手改的是成果，还是做法。**
- ❌ 选 A（审计团队是否按规程执行代码评审）：查的是“做法有没有被遵守”，属质量审计，归管理质量。点评：**审计属 QA、统计抽样属 QC**，这两项是全章最容易被互换的一对。

易错点 / 考点 ① **QA 面向过程、QC 面向成果**，六维对照表要能默写其中三维以上。② 质量**审计**属管理质量，**统计抽样**属控制质量——两个最容易被互换。③ 检查**既能发现缺陷，也能确认已修复的缺陷**，这个双重作用常被漏答。④ 抽样结论必须**带上抽样比例与方案**，不能表述为全检结论。⑤ QC **贯穿整个开发周期**，不是只在阶段末做。

过渡 *四类方法里最容易被考倒的，是那张画着五条横线的图——控制图。下一页面它，还要破一个全章最容易错的观念。*

12.19 控制图判异：四条规则，与一个必须分清的区别（4 分钟）

口播 · 可照读

这一页是全章最硬的骨头，请把笔放下、眼睛抬起来，跟着图走。

控制图用来确定一个过程是否稳定，或者是否具有可预测的绩效。 图上通常有三样东西：**中心线**，也就是过程均值；**控制界限**，对重复性过程取均值 ± 3 倍标准差，$UCL=\mu+3\sigma$、$LCL=\mu-3\sigma$；**规格界限**，按协议要求制订的规格上下限，反映可允许的最大值和最小值。注意这三样东西的来源完全不同：中心线和控制界限是**算出来的**，来自过程自己的数据；规格界限是**定出来的**，来自协议和客户要求。

先看图。自上而下五条横线：最上面粗实线是**规格上限**，往下点线是**控制上限**，中间实线是**中心线**，再往下点线是**控制下限**，最下面粗实线是**规格下限**。横轴是样品 1 到样品 10。现在看点：样品 2 是全图最低点，落在控制下限上方一点点，没越界；样品 6 是全图最高点，**越过了控制上限**，可它离规格上限还有一段距离。这一处细节，就是等一下要讲的那个区别的实物证据。

判异四条，一条一条来。① **有点越出控制界限**：图里就是样品 6。这是**特殊原因造成的突变**，要立即排查——换了料、换了设备、环境变了、有人临时改了参数。

② **连续 7 个点落在均值上方，或连续 7 个点落在均值下方**：这叫**均值偏移**。某一个点偏高可能是偶然，连着 7 个点都在同一侧，就不是运气了——换人、换工具、来料变了、需求变了。

③ **连续 7 个点持续上升或持续下降**：这是**系统性趋势**。趋势最阴险，因为它每一个点可能都在界限内，可方向一直没变——缺陷在累积、环境在劣化、某项指标在慢慢漂。

④ **明显非随机模式**：周期性起伏、成簇、贴着界限反复跳动。这说明过程里藏着还没被识别的规律。

四条记法：**越界、同侧、趋势、花样**。任何一条命中，结论都只有一句：**过程出现了非随机波动，该去查特殊原因了。**

现在讲那个必须分清的区别，两条，都很容易错。

第一条：判异不等于不合格。 判异说的是**过程**出了异常信号，不是**产品**超了规格。回看图里样品 6——它越了控制上限，可离规格上限还有距离，**这一件产品本身是合格的**，它只是在大声提醒你“过程里有特殊原因，去查”。反过来也要说清：某次越界如果是设备故障造成的孤立事件，隔离掉那一件、修好设备，后面的产品照样合格。

第二条：控制界限不等于规格界限。 这两条线的来源、管的对象、后果全都不同。控制界限管**过程稳不稳**，用 $\pm 3\sigma$ 算出来，越界要找特殊原因；规格界限管**产品合不合格**，按协议定出来，**超出规格界限就可能受处罚**。定义里那句“上下控制界限不同于规格界限”，是这一章最容易被考、也最容易出错的一句话。

举两个反例把这件事钉死。**反例一**：某过程很稳，十个点全在控制界限内，可它的**中心线本身就落在规格界限之外**。这时过程完全受控、波动可预测，但**每一件产品都不合格**——过程稳和产品合格，是两件事。**反例二**：规格界限定得很宽，比如允许误差 ± 10 个单位，而过程抖动很剧烈，于是每个点都在规格界限内、**件件合格**，可控制图上连续 7 点同侧、还越了界——**过程已经失控**，照这样下去出事只是时间问题。两个反例的方向相反，结论同一个：**受控 ≠ 合格，失控 ≠ 不合格。**

再补一层统计直觉：$\pm 3\sigma$ 这个口径，正常波动被误判为异常的概率大约是千分之三。所以判异是**信号**，不是**判决**——它通知你去查，不是通知你去报废。查完发现是特殊原因，处理它；查完发现是虚惊，记录它。这就是“基于计算出的控制界限发现须采取纠正措施的检查点”这句话的意思。

最后说软件里怎么用。控制图不只用来看批量生产：**成本与进度偏差、产量、范围变更频率**，都能当输出变量画上去；软件项目里更常见的是每周缺陷数、构建时长、接口响应时间、每月线上故障次数。画上去之后，你就能回答一个很值钱的问题：**这个过程的偏差，是否处在可接受的控制范围之内？**

板书 **控制图＝确定过程是否稳定／是否具有可预测的绩效** ｜ 三样东西：**中心线**（过程均值，算出来）· **控制界限** $UCL=\mu+3\sigma$、$LCL=\mu-3\sigma$（算出来）· **规格界限**（按协议定出来，**超出可能受处罚**） ｜ **判异四条**：①**点越出控制界限**（样品 6）②**连续 7 点同侧**（均值偏移）③**连续 7 点持续升／降**（系统性趋势）④**明显非随机模式**（周期・成簇・贴界跳动） ｜ 记法：**越界・同侧・趋势・花样** ｜ **★判异 ≠ 不合格**：判异说的是过程，产品仍可能合格 ｜ **★控制界限 ≠ 规格界限**：一个管过程稳不稳，一个管产品合不合格 ｜ 反例一：**中心线落在规格界限之外** → 全在界内但件件不合格 ｜ 反例二：**规格很宽而过程剧烈抖动** → 件件合格但连续 7 点同侧＝失控

提问 （本页互动题）某过程控制图上所有点都在控制界限内，因此产品一定合格——这句话对吗？
选项：A．对：过程受控，成果就必然合格 B．错：受控只说明过程稳定可预测；数值仍可能整体超出规格界限 C．错：因为控制图只能用于制造业，不能用于软件

预设回答

- ✅ 选 B——这是判断题，结论是“错”。受控只说明过程**稳定、波动可预测**，不说明数值**达到规格**；若中心线本身就压在规格界限之外，过程可以稳稳地一直产出不合格品，正是本页的反例一。
- ⚠️ 选 B 但只答“因为判异不等于不合格”：方向对、判据没落下——这句话的根子错在**把控制界限当成了规格界限**。控制界限由 $UCL=\mu+3\sigma$、$LCL=\mu-3\sigma$ **算出来**，管过程稳不稳；规格界限由**协议定出来**，管产品合不合格，两条线的来源、对象、后果都不同。
- ❌ 选 A：把“受控”当成了“达标”，一句话错了两处——控制界限不是按客户要求画的那两条线，而且全程没越界也照样可能件件超规格。这正是本页要破的误区，也是六维对照表里 QC 侧的“常见误区”。
- ❌ 选 C：结论（错）碰对了，理由完全站不住。控制图不只用来看批量生产：成本与进度偏差、产量、范围变更频率都能当输出变量画上去，软件项目里每周缺陷数、构建时长、接口响应时间、线上故障次数都在用。

易错点 / 考点 ① **判异四条**要能完整默写，尤其“**连续 7 点**”这个数字，写成 5 点、6 点都算错。② **判异 ≠ 不合格、控制界限 ≠ 规格界限**——本章最易错，判断题几乎必出，而且常反过来考“超出规格界限就是过程失控”（错，那是产品不合格的问题）。③ **中心线不是目标值**，它是过程实际的平均水平；中心线偏离目标值，本身就是该改进的信号。④ 规格界限由**协议**决定，控制界限由**统计计算**决定，来源不能混。⑤ 图 7-7 里样品 6 越控制上限但仍在规格上限之内，这个细节是解释“判异≠不合格”的最好证据，答题时可以直接引用。

过渡 *判完、核完，手上要留下能交给下一道关的东西——12.20 就数这六样成果。*

12.20 控制质量的成果：核实的可交付成果与工作绩效信息（2 分钟）

口播 · 可照读

控制质量做完，手上有六样东西。其中两样最关键：**核实的可交付成果**证明“做对了”，**工作绩效信息**说明“过程怎么样、下一步怎么调”。

一样一样过。① **质量控制测量结果**：按质量管理计划规定的格式，记录缺陷数、测试覆盖率、代码复杂度、缺陷修复率等，汇入质量报告，供过程分析用。② **核实的可交付成果**：经测试与评审确认符合质量要求的成果。它的去向很关键——交给**确认范围**，由客户正式验收。③ **工作绩效信息**：包含需求实现情况、拒绝原因、需返工的部分、纠正措施建议、已核实的可交付成果、质量测量指标的最新状态、过程调整需求；进绩效报告，支撑决策。④ **变更请求**：纠正措施、预防措施、缺陷补救引发的变更，走整体变更控制。⑤ **更新的项目管理计划**：主要是质量管理计划。⑥ **更新的项目文件**：测试与评估文件、问题日志、经验教训登记册、风险登记册，最终沉淀为组织过程资产。

这里有一条流水线，请记住顺序：**控制质量 → 确认范围 → 交付**。同一件成果要过两道关：先在**内部**判“对不对、符不符合质量要求”，再由**外部**判“要不要、客户接不接受”。**顺序一般不能颠倒**——先让客户看一件还没内部核实过的东西，等于把返工摆到客户面前。

那这两道关到底差在哪？四点。**第一**，控制质量看**正确性与符合质量要求**，确认范围看**客户是否接受**。**第二**，控制质量**一般先于**确认范围，但也可以同时进行。**第三**，确认范围**一般在阶段末**进行，控制质量**贯穿整个开发周期**。**第四**，控制质量是**内部检查**，由组织相应质量管理部门实施；确认范围由**外部干系人**——客户或发起人——验收。

一句话收束这一页：**核实是内判，验收是外判；内判不过，别急着外判。**

板书

- 4 六成果 ｜ ①**质量控制测量结果**（缺陷数・覆盖率・复杂度・修复率→汇入质量报告）②**核实的可交付成果**（经测试评审确认→**交给确认范围**）③**工作绩效信息**（需求实现・拒绝原因・返工部分・纠正建议・指标最新状态・过程调整需求）④**变更请求**（走整体变更控制）⑤更新的项目管理计划 ⑥更新的项目文件 ｜ 流水线：**控制质量（内判对不对）→ 确认范围（外判要不要）→ 交付** ｜ 与确认范围四点区别：**看什么（正确性／接受）・先后（一般先／可同时）・时点（贯穿周期／阶段末）・谁做（内部质量部门／外部干系人）** ｜ 金句：**内判不过，别急着外判**

提问 客户在验收会上签字通过了，这份签过字的成果，属于控制质量的成果还是确认范围的成果？

预设回答

- 答“属于确认范围的成果；控制质量交出去的是‘核实的可交付成果’”：✅ 准确。两个名字很像，区别就在“谁判的”：内部核实过的叫核实的可交付成果，客户接受了的叫验收的可交付成果。
- 答“都是，反正都在质量这条线上”：⚠️ 太笼统。两个过程的成果不能混记，考试里这就是两个不同的名词。
- 答“属于控制质量，因为验收也是检查的一种”：❌ 把“检查”当成了判据。检查是 QC 的方法，验收是确认范围的动作；判据是**谁来做**——内部质量部门还是外部干系人。

易错点 / 考点 ① **核实的可交付成果 ≠ 验收的可交付成果**，前者是 QC 的输出、确认范围的输入。② 控制质量**一般先于**确认范围，但**可以同时进行**——两个方向都可能出判断题。③ **工作绩效信息**要与“工作绩效数据”“工作绩效报告”区分：数据是原始记录，信息是加工后的比对结果，报告是汇编呈现。④ 变更请求一律**走整体变更控制**，不能就地改。

过渡 *质量这条线管的是“东西对不对”。可项目里还有另一条线，管的是“东西有没有被人悄悄改掉”——那就是配置管理。*

***【思政落点】**（宁返工，不放行）：发现缺陷时最“省事”的做法，往往是把判定标准解释得松一点，把问题推到下一环节去。可外部失败的成本最高——上线后的一次故障，代价可能是返工的几十倍，买单的是用户和团队的信誉。如实记录、如实判定、该返工就返工，是质量岗位上最硬的一条底线。落到同学们的分组项目上：测试记录里发现了问题就写进去，不要把“已知问题”从用例里删掉来让通过率好看；那一行被删掉的记录，迟早会在演示那天以另一种方式回来。*

12.21 7.5 配置管理：先把配置项认全、编号（5 分钟）

口播 · 可照读

讲完质量控制，质量这条主线告一段落。可项目里还有一类问题，和质量问题长得不一样，破坏力却一点不小——**东西被人改掉了，却没人知道改了什么、谁改的、为什么改**。这就是配置管理要管的事。

软件配置管理是一套管理软件开发、维护及其中间产品的方法和规则，是提高软件质量的重要工具。它的核心目标是：在软件生命周期的每个阶段，系统地控制配置的修改，**确保配置的完整性和可追溯性**。请注意它管的不只是中间软件产品——代码、文档、脚本、执行文件、安装包——还包括**生成这些产品的控制信息**，甚至连开发工具，比如编辑器、编译器、CASE 工具，也可以纳入配置管理，以保障开发环境的稳定和一致。

被它管控的工作产品叫**软件配置项**，是受配置管理控制的工作产品集合，**包括计算机程序、文档和数据**，贯穿软件生命周期的各个阶段。举几个典型例子：源代码、设计文档、测试计划、用户手册、数据库脚本、接口文档。

定义之外，有个说法特别好记——配置管理像软件开发里的“**位置管理**”，它回答四个问题：**我是谁、我为何在这里、我为何是我、我属于哪里**。“我是谁”是标识，“我为何在这里”是它属于哪个基线、哪个阶段，“我为何是我”是它为什么是这个版本，“我属于哪里”是它在配置库里的位置。四个问题问清了，一个配置项才算被真正管住。

接下来是这一页最关键的一个类比：**配置项可以看成类，版本可以看成实例**。看这张图：上面一个框是“**需求规格**”，标注为**配置项类**；从它往下分出三个框——**需求规格 V1.1、V1.2、V1.3**，标注为**配置项实例**。三个版本**彼此不同，但具有相关性**：它们都是从最原始的那个配置项演变出来的。这一条直接决定了两件事：第一，旧版本**不能随手删**，它是回溯和定位问题的依据；第二，版本号不是装饰，它回答“我为何是我”。

那么“配置项识别”到底要做哪五件事？① **明确范围**：界定哪些工作产品纳入管控，既包括交付产品，也包括开发支持工具。② **统一规则**：制订配置项的命名规则、组织结构，明确变更管理流程。③ **分配标识**：为每个配置项分配**唯一标识符和版本号**——一个标识符只对应一个配置项，改了才升版本。④ **定义基线**：确定经技术复审的正式版本。⑤ **结构化存储**：引入配置管理工具，以清晰的目录结构把配置项存进配置库，而不是散在个人电脑和聊天记录里。

看这张标识示例表，五类配置项各有各的前缀：需求类是需求规格说明书和需求跟踪矩阵，标识如 **REQ-2026-001 V1.2**；WBS 与计划类是 WBS 词典、进度计划、质量管理计划，如 **WBS-3.2.1 V1.0**；代码类是源代码、构建脚本、第三方库清单，如 **SRC-PAY-014 V2.3**；测试类是测试用例、测试数据、自动化脚本，如 **TC-PAY-021 V1.1**；文档与交付类是用户手册、安装说明、发布包，如 **DOC-UG-003 V1.0**。这套写法的价值在于：**看到一个编号，就知道它是什么类、属于哪个模块、第几版**。

最后是识别这件事的目的和时机。目的是**明确系统中各组件及其相互关系**，定义配置项的结构和管理方式，确保跟踪与管理，为后续的控制、审计打底。时机更关键：**规则要在项目初期就定下来**。事后补建，代价极高——一个跑了半年的项目要回头给几百个文档和代码补编号、理版本，基本上等于重做一遍，而且这半年里已经被覆盖掉的修改，永远找不回来了。

课堂这道题：同一份需求规格的 V1.1、V1.2、V1.3，在配置管理里算什么？**同一个配置项的三个实例，也就是三个不同版本，相关但可区分**。不能说成三个互不相干的配置项——那样就丢了“它们同源”这条最重要的信息；更不能说“只留最新的 V1.3，旧的删掉”——旧版本是回溯的依据，删了就没法回答“V1.2 当时为什么这么写”。

板书 **SCM＝管理软件开发·维护及其中间产品的方法和规则**；目标：各阶段系统控制配置修改，确保**完整性＋可追溯性** ｜ 管理对象：代码·文档·脚本·执行文件·安装包＋**生成这些产品的控制信息＋开发工具**（编辑器·编译器·CASE） ｜ **SCI**＝受控的工作产品集合，**包括程序·文档·数据**（源代码·设计文档·测试计划·用户手册·数据库脚本·接口文档） ｜ 四问：**我是谁·我为何在这里·我为何是我·我属于哪里** ｜ 图 7-10：**配置项＝类，版本＝实例**（需求规格 → V1.1／V1.2／V1.3，**相关但可区分，旧版不能删**） ｜ 五项任务：**明确范围·统一规则·分配标识·定义基线·结构化存储** ｜ 标识示例：**REQ-2026-001 V1.2／WBS-3.2.1 V1.0／SRC-PAY-014 V2.3／TC-PAY-021 V1.1／DOC-UG-003 V1.0** ｜ 时机：**项目初期定规则，事后补建代价极高**

提问 （本页互动题）同一份需求规格的 V1.1、V1.2、V1.3 三个版本，在配置管理里算什么？
选项：A．三个互不相干的配置项，各自独立管理 B．同一个配置项的三个实例（不同版本），相关但可区分 C．只需保留最新的 V1.3，旧版本可以直接删掉

预设回答

- ✅ 选 B——**配置项是类，版本是实例**：V1.1／V1.2／V1.3 同属“需求规格”这个配置项类，是它演化出的三个实例，**彼此不同、又具相关性**；靠唯一标识符加版本号（如 REQ-2026-001 V1.2）区分，也靠它回答“我为何是我”。
- ⚠️ 选 B 但把理由落到“三个文件都还在、没丢就算管住了”：方向对了、判据偏了——**堆在同一个文件夹里不等于纳入配置管理**，还缺三样：类别唯一编号（如 REQ-2026-001）、版本升级的触发规则、配置库里的结构化位置。文件名写成“需求规格-最终版.doc”更答不上“谁改的、为什么改”。
- ❌ 选 A：把同源关系丢了。三个版本若“互不相干”，就没人能回答“当前引用的是哪一版、上一版改了什么”——配置管理最值钱的那条追溯链，断在第一步；顺带也就丢掉了“类—实例”这个本章的送分类比。
- ❌ 选 C：删旧版是最贵的一种“整洁”。旧版本是回溯与定位问题的依据，把 V1.1、V1.2 删掉，等于把变更历史连同排障线索一起删掉；识别规则要在**项目初期**定，事后补建代价极高。

易错点 / 考点 ① **配置项是类、版本是实例**——这个类比是本章的送分点，也是“旧版本能否删除”的判据。② **配置管理的管理对象包含开发工具**，选择题常设“编译器不属于配置管理对象”这类陷阱。③ 五项任务里**“分配唯一标识符和版本号”与“定义基线”是两件事**，不要合成一条。④ **识别规则必须在项目初期制定**，事后补建的代价是简答题的常见落点。⑤ 配置管理**不等于备份**：备份只保留副本，配置管理还要管版本、变更、基线与依赖关系。

过渡 *配置项认全了、编号也定了，接下来要回答两个问题：这些东西**存在哪里**、按什么规矩**改动**——下一段就是三库与基线。*

12.22 配置库三库与基线：开发库 · 受控库 · 产品库（3 分钟）

口播·可照读

上一页把配置项认全、也编好了号，这一页要回答三个更硬的问题：这些东西放在哪儿、谁能改、改完算哪一版。教材给的答案是**配置库**——存放配置项及其相关信息的工具。请留意“相关信息”这四个字：它存的不仅是文件本身，还有**版本、修改历史、依赖关系**。所以配置库不只是一个网盘，它能回答五个问题：当前系统用的是哪个版本、某个配置项改过几次、每次是谁改的、什么时候改的、不同配置项之间有哪些依赖。它的管理对象是**半成品、阶段性产品和最终产品**——从写到一半的模块，到交付给客户的安装包，全都进来。

配置库分三种，分工的依据只有一条：**能不能随手改**。第一种**开发库**，又称动态库，是开发人员的个人工作区，放正在开发或修改的模块与文档，改得频繁，所以**不施严格的配置控制**，由开发人员自行管理。第二种**受控库**，又称主库，放当前的基线版本及变更，实行**严格的版本控制**；每个开发阶段完成，工作产品存入受控库，形成**新的基线**。第三种**产品库**，又称静态库，放已经完成并通过测试的最终产品，包含**所有发布版本**的存档，严格受控，供部署或安装使用。三间房，越往后越严，越往后越靠近客户。

再说基线，这是本节最该背下来的一句定义：**基线是经过技术复审的配置项集合，在软件生命周期某一阶段被正式接受，成为项目开发活动的起始点和参考点**。拆成三块记：**技术复审、正式接受、起始点与参考点**。复审的目的，是确保配置项在被接受之前的**正确性和完整性**；成为基线之后，它就是后续开发的出发点，也是后面所有比对的参照物。它还有四个作用：有效控制需求、预算、进度和设计方案的变动；提供开发过程中的**特定检查点**；确保交付项中有一致的子集，也就是系统一致性与状态一致性；作为**控制点**，确保正式发布的系统符合质量标准。

落到案例上，想一想某汽车公司那套质量管理系统：来料检验的抽样规则文档、SPC 控制图的判异参数、售后 PPM 的统计口径、8D 报告的模板——这些都是配置项。它们的基线定了，检验员每天按的就是这一版；基线没定，两个人手里的抽样表可能不是同一份，同一批来料能检出一高一低两个结论。

最后看图 7-11。它画的是三个库之间的一条闭环，四个动作：**复制、检出、检入、更新**。这四个动作的先后顺序，就是下一页要讲的流程；先记住一条直觉——产品库是“存档”，受控库是“大家认可的那一版”，开发库是“我的工作台”。**改得快靠工作台，改得稳靠受控库。**

板书 **配置库**＝存配置项＋**版本·修改历史·依赖关系**（管理对象：半成品·阶段性产品·最终产品） ｜ **开发库**（动态库）个人工作区·改得频繁·**不施严格配置控制** ｜ **受控库**（主库）**当前基线及变更**·严格版本控制·阶段完成→**新基线** ｜ **产品库**（静态库）**通过测试的最终产品**·含所有发布版本·旧版不删 ｜ **基线＝技术复审**通过 ＋ 生命周期某阶段**正式接受** ＋ **起始点与参考点** ｜ 四个作用：控变动 · 检查点 · 一致子集 · 发布控制点 ｜ 图 7-11 四动作：**复制 → 检出 → 检入 → 更新**

提问 新同事说“我先在受控库里把这个函数改了，回头再补流程”——问题出在哪？（限 1 分钟）

预设回答

- ✅ 正确方向：受控库是**基线所在地**，直接改等于在“大家认可的那一版”上悄悄动手。具体三条后果：别人的检出与锁定对不上版本，两人一提交就互相覆盖；变更没有记录与验证，缺陷出现时查不到是哪次改动引入的；事实上基线已变却没有新版本号，测试与文档仍对着旧口径。正确路径是**检出**到开发库改 → 一致性检查与基本测试 → **检入**受控库 → 评审后**更新**版本。
- ⚠️ 部分对：答“应该在开发库改，受控库不让动”。方向对了一半——受控库不是“不让动”，而是**动要走申请、要留版本**；小改也要走，只是评审规模可以小。补一句：真正决定严不严的是**这一版在不在基线上**，不是改动大小。
- ❌ 常见错：答“先改了再补流程也一样，反正结果相同”。结果不同：补流程补得出**内容**，补不出**当时那一版**。版本号一旦断档，回溯就失去了锚点，配置审计也无从证明这次改动经过授权。

易错点 / 考点 ① 三库的别名要能对上：开发库＝动态库、受控库＝主库、产品库＝静态库，选择题常拿别名与定位错配来设错项。② **受控库存的是“当前基线及变更”**，产品库存的是“通过测试的最终产品且含所有发布版本”——两者的区别是“在改的/已发布的”，不是“重要的/次要的”。③ 基线定义的三要素缺一不可，尤其**“经过技术复审”和“被正式接受”**：自己拍一版存档不叫基线。④ “开发库不施严格配置控制”不等于“开发库没有版本”，个人工作区照样有本地版本，只是不纳入受控流程。⑤ 旧版本**不删除**——删除旧版本是案例题里的经典错误，回溯和问题定位全靠它。

过渡 *三间房分清了，可东西怎么在三间房里流转、谁批准每一次进出？这就是流程要回答的问题——而且不做流程，会出三个很具体的麻烦。*

12.23 配置管理流程：未经管理的变更，会出三个问题（4 分钟）

口播 · 可照读

流程不是把简单的事做复杂，而是因为不做它，三件麻烦一定会来。教材列得很干脆：第一，**版本冲突**——多名开发人员修改同一模块，如果没有版本管理，修改可能互相覆盖；第二，**问题追踪困难**——变更没有被记录或验证，问题出现时无法定位；第三，**低效协作**——开发人员缺乏统一的变更机制，导致代码管理混乱。教材还给了一个很短的情景：开发人员甲和乙各自修复不同的代码问题，但未经变更管理直接提交，结果**乙的修改覆盖了甲的修改，最终两个问题都没有解决**。请注意这句话的落点——不是“有一个人白干了”，而是“两个问题都没解决”，因为被覆盖掉的那份修复，连证据都找不回来了。

这三条麻烦对应着流程里的三道关口。**第一道，版本冲突靠“锁定”解决**：检出的时候，被检出的代码段就被锁定，甲检出模块 A，乙就检不出 A，冲突从机制上被消除，而不是靠两个人打招呼。**第二道，追踪困难靠“记录加验证”解决**：检入时做一致性检查和基本测试，通过才解除锁定；改了什么、谁改的、什么时候改的，全在版本历史里。**第三道，低效协作靠“统一入口”解决**：所有变更走同一条路，谁手上的版本都对得上，联调不用再猜版本。

流程本身六步，一步一步看。① **识别配置项**：定范围、定命名、定编号。② **建库定基线**：三库就位，基线立起来。③ **提交变更**：写清理由与影响，不是“我想改”。④ **CCB 评审**：批准、驳回或者缓办。⑤ **更新版本**：检入并形成新基线。⑥ **配置审计**：查完整性、一致性、可追溯性。这六步是闭环——审计发现的问题回到第①步，连配置项清单与命名规则本身也要维护。

六步怎么落到图 7-11 那四个动作上？看这张对照。第①②步对应**复制**：把待升级的基线版本，比如 V2.1，从产品库取出放进受控库；受控库作中间库，产品库保持稳定不动，这一手是在**保护产品库的完整性**。第③步对应**检出**：从受控库检出到个人开发库去改，同时锁定。第④⑤步对应**检入**：改完检回受控库，做一致性检查与基本测试，通过后解锁。最后一步对应**更新**：把受控库里的新基线存入产品库，**生成新版本号**，比如 V2.2，而 V2.1 **不删除**，留着回溯与定位。

讲一个案例里的场景。汽车质量系统的售后 PPM 报表，口径要从“按批次统计”改成“按车型统计”。这是一次正经的变更：先提变更请求写清理由和影响——报表、看板、历史数据、客户月报全都要跟着动；CCB 评审时间的是“这个改动会不会让验收标准失效”；批准后从受控库检出报表模板与统计脚本，在开发库改；自测通过检入受控库；再更新到产品库，版本号从 V1.4 升到 V1.5。整个过程留下的是：一条变更记录、一个新版本号、一份审计结论。反过来，如果谁顺手在生产环境里把口径改了，客户下个月拿到的月报就和上个月不是一套算法——这就是“问题追踪困难”在真实业务里的样子。

板书 **三问题**① **版本冲突**（多人改同一模块 · 互相覆盖）② **问题追踪困难**（变更无记录/验证 · 无法定位）③ **低效协作**（无统一机制 · 代码管理混乱） | 示例：甲、乙直接提交 → 乙覆盖甲 → **两个问题都没解决** | **六步**：识别配置项 → 建库定基线 → 提交变更 → **CCB 评审** → 更新版本 → 配置审计（闭环回第①步） | 对应图 7-11：①②**复制**（V2.1 入受控库 · 产品库不动）→ ③**检出**（**锁定**）→ ④⑤**检入**（一致性检查＋基本测试 · **解锁**）→ ⑥**更新**（新版本号 V2.2 · 旧版不删）

提问 改动已经在开发库改完、自测也过了，能不能不经过 CCB 直接检入受控库？（限 1 分钟）

预设回答

- ✅ 正确方向：**检入是技术动作，CCB 是授权动作**，两件事不能互相顶替。检入只证明“改完的东西能通过一致性检查与基本测试”，不证明“这个改动该不该做、做到什么程度”。规范顺序是：**CCB 批准 → 检出动手 → 检入受控库形成新基线 → 确认后更新到产品库**。
- ⚠️ 部分对：答“小团队可以简化，负责人点头就行”。可以简化形式，不能简化留痕：小团队里角色常兼任，但记录上**“谁批准”与“谁执行”必须分开写**，否则审计时无法证明这次改动经过授权。
- ❌ 常见错：答“自测通过就说明质量没问题，可以直接进”。自测只验证了改动本身，没验证**影响面**：这次改动会不会让别的模块、验收标准、历史数据出问题，只有影响分析能回答；而影响分析正是评审环节的内容。

易错点 / 考点 ① 三个问题要与三道机制对上：版本冲突→**锁定**，追踪困难→**记录与验证**，低效协作→**统一入口**，简答题常要求“问题—措施”配对。② 图 7-11 的顺序不能记成“检出→复制”：**先复制（把基线搬进受控库），才谈检出**，顺序反了，等于在旧存档上直接开工。③ **检入做的是“一致性检查与基本测试”**，不是“完整回归测试”，别把两者混为一谈。④ 更新时**新版本号要生成、旧版本要保留**，这一对动作在案例题里几乎必考。⑤ “CCB 驳回”也是正常结果，不等于流程失败；缓办也是决策之一。

过渡 *流程走通了，还有一条边界必须划清：范围管理、变更管理、配置管理，三套东西都在管“变”，到底谁定、谁批、谁记账？*

12.24 配置管理与范围管理 · 变更管理：谁定、谁批、谁记账（3 分钟）

口播 · 可照读

三套管理放在一起最容易糊，因为它们都在说“变”。给它们各派一个问句就清楚了：**范围管理**回答“做什么、验收到什么标准”，**变更管理**回答“这次改动批不批、影响多大”，**配置管理**回答“现在是什么版本、谁改的、能不能追溯”。三句话，对应三个不同的动作：**定、批、记账**。

先看范围管理。它的产出是范围基准、WBS 与验收标准——而质量需求恰恰就写在验收标准里。缺了它会怎样？说不清范围，缺陷与变更都无从判断“该不该”：客户说“这不是我要的”，你翻不出当初约定了什么；团队说“这个改动很小”，你说不出它有没有越过范围边界。**范围是判断的尺子，尺子没刻线，量什么都是猜。**

再看变更管理。它管的是“批不批”：变更请求进来，先做影响分析——对范围、进度、成本、质量各有什么影响，然后由 CCB 决策。缺了它会怎样？改动随意进出，基准形同虚设。这一套在第 11 章项目整合管理里会展开成全项目统一的整合变更控制，本章只需要记住它的三个动作：**提请求、做影响分析、由 CCB 决策**。

最后是配置管理，它管的是“记账”：配置项怎么标识、放在哪个库、当前是什么版本、谁在什么时候改过、能不能审计。缺了它会怎样？你不知道自己在改哪一版，也无法证明改过什么。**变更有决定权，配置管理有记录权——两权分开，才可能互相制约。**

支撑这套账的是几个角色，别记混。**CCB，变更控制委员会**，有的组织叫配置管理委员会 CMB：评估变更在技术、逻辑、策略、经济和组织上的影响，分析对进度与预算的影响，最终**决定批准、驳回或缓办**；CMB 还负责监督与指导整个项目的配置管理工作。**配置管理负责人**，也叫配置经理：管理与决策整个生命周期中的配置活动——定规则、定基线、定审计节奏。**配置管理员**：实施配置管理的主要活动，建库与授权、执行检出检入、生成版本、组织配置审计。**配置项负责人**：确保所负责配置项的**准确性与真实性**，内容对不对、版本说明全不全。

这四层角色为什么要分开？因为它们互相是对方的检查。配置管理员能改库，但不能自己批准改动；配置项负责人最懂内容，但不能自己给自己发版本号；CCB 能批准，但批完必须由配置管理员落到版本上。案例里那条质量信息链——来料检验、SPC、售后 PPM、8D——每一环的报表口径都属于配置项，口径变更走 CCB，版本由配置管理员维护，数据真实性由配置项负责人担保。**一条链上三个人的职责不重叠，质量数据才敢拿去对外报。**

板书 **范围管理**答“做什么 · 验收到什么标准” → 范围基准 · WBS · 验收标准 → 缺了：**说不出“该不该”** | **变更管理**答“批不批 · 影响多大” → 变更请求 · 影响分析 · **CCB 决策**（第 11 章展开） → 缺了：**基准形同虚设** | **配置管理**答“什么版本 · 谁改的 · 能否追溯” → CI 标识 · 三库 · 基线 · 版本号 · 配置审计 → 缺了：**不知道在改哪一版** | 三个动作：**定 / 批 / 记账** | 角色四层：**CCB（CMB）批 · 配置经理定规则 · 配置管理员执行 · 配置项负责人保真实**

提问 需求方说“就改一个字段，不用走流程了吧”——这句话的问题在哪？（限 1 分钟）

预设回答

- ✅ 正确方向：**变更的大小不等于影响的大小**。一个字段可能牵动接口、报表、历史数据迁移和验收标准。更要紧的是三件事：没有变更请求就没有影响评估，谁也不知道它动了范围、进度还是成本；没有 CCB 决策就没有授权留痕，审计时无法证明这次改动由谁批准；基线事实上变了却没升版本，之后所有人都在“名义 V1.2、实际 V1.2 加”的版本上工作。
- ⚠️ 部分对：答“要走流程，但可以走简易流程”。对，且这才是实务答案：**流程可以分级**——影响小的走简化评审，影响大的上 CCB；但“分级”不等于“免检”，判断分级的那一步（影响分析）不能省。
- ❌ 常见错：答“字段级改动属于维护，配置管理不管”。维护恰恰是配置管理最忙的地方：改得越频繁，越需要版本与记录；把维护划到管理之外，等于让最活跃的改动全部脱管。

易错点 / 考点 ① 三套管理的**问句**要能一口报出，简答题常要求“分别说明三者关系”。② **CCB 管决策、配置管理员管执行**，两者不能由同一个人在同一件事上兼任并只留一份记录。③ 配置管理的对象是**工作产品**，不是“代码仓库”——需求、文档、测试用例、脚本、编译工具都可能成为配置项。④ “范围管理不管变更”是错的：范围基准是变更影响分析的**基准输入**，两者是输入与决策的关系，不是互不相干。

过渡 *配置这本账记的是“东西变成了什么”，还有一本账记的是“数字说明什么”——接下来两页看两个绩效域，先看度量。*
***【思政落点】**（按程序办事，规矩不是麻烦）：这条边界值得记一辈子——**谁定的、谁批的、谁记的，三件事不能挤到一个人身上**。真正的专业不是把流程绕过去，而是把该走的程序走完，还让它在十分钟内走完。*

12.25 度量绩效域：选指标 → 收集 → 趋势 → 决策（3 分钟）

口播 · 可照读

第 6 章讲过度量绩效域的基准绩效、资源、价值与预测型指标；放到质量管理里，还要多盯两个对象：**可交付成果**和**交付过程**。度量的价值不在报表好看，而在**能不能改变某个人的动作**——一个指标如果谁看了都不改做法，它就只是装饰。

一个有效的度量循环四步：**选指标、收集、看趋势、做决策**。选指标要过两道筛：一是有没有**关键绩效指标**的分工——**提前指标**预测趋势，帮你在还来得及的时候调整方向；**滞后指标**反映已经完成的成果或事件，比如进度偏差和成本偏差。二是够不够 **SMART**：具体、可衡量、可实现、相关且及时。收集这一步最容易注水：口径不统一，两个人报的“缺陷数”能差一倍；所以**定义、计算式、采集频率**要先写死。看趋势不能只看单点，要**对比基线、连看几期**。做决策才是终点：纠偏、改进、或者提变更。

指标分四类，一类一类认。**缺陷相关指标**：缺陷的来源、识别出的缺陷数量、已解决的数量——它衡量产品质量，也衡量修复效率。**绩效度量指标**：与系统运行相关的物理或功能属性，比如软件的准确度、系统容量、可靠性、效率。**技术绩效度量指标**：评估系统组件是否符合预定技术要求，比如接口响应达标率、兼容性通过项。**交付度量指标**：主要与正在进行的工作有关，多用于敏捷项目——在制品、提前期、周期时间、队列大小、批量大小、过程效率。

然后是最该记住的一页：**度量陷阱**。教材列了六条，考试最认前两条。**霍桑效应**：当某一事物被度量时，其行为可能会发生变化。举个最贴近的例子——如果只看“可交付成果的数量”，团队就会专注于增加输出，而忽视满足客户需求、提升客户满意度这个真正的目标。你度量什么，就会得到什么，但得到的可能只是被度量那一面。**虚荣指标**：那些对决策没有实际帮助的度量标准。网站的新访客数量看着很漂亮，但相比页面访问量，它提供不了多少洞察；用虚荣指标，浪费的是资源和精力。

另外四条也点一下，别在选择题上栽跟头：**士气低落**——目标过于苛刻或不可能实现，会打击团队积极性；**误用度量指标**——过分关注不重要的指标，或者为了短期指标牺牲长期目标；**确认偏见**——人倾向于关注与自己已有观点相符的信息；**相关性与因果关系混淆**——进度落后与预算超支常常同时发生，但不代表前者导致了后者，两者可能都被同一个原因带出来，比如估算不准、变更管理不足、风险管理不到位。

回到案例。汽车质量系统里，如果只考核“每月关闭缺陷数”，团队就会倾向于把难改的缺陷拆成几条、快速关掉小问题；数字漂亮，售后 PPM 一点没降。改法是把指标配成一对：**提前指标**看未关闭缺陷的增速与缺陷到达模式的斜率，**滞后指标**看售后 PPM 与一次通过率——一个指方向，一个验结果，中间用趋势连着看。

板书 **四步**：选指标（**KPI：提前／滞后 + SMART**）→ 收集（口径统一 · 如实登记）→ 看趋势（对比基线 · 连看几期）→ 做决策（纠偏 · 改进 · 提变更） ｜ 四类指标：**缺陷相关**（来源 · 发现数 · 已解决数）· **绩效度量**（准确度 · 容量 · 可靠性 · 效率）· **技术绩效**（是否符合预定技术要求）· **交付度量**（在制品 · 提前期 · 周期时间 · 队列大小 · 批量大小 · 过程效率） ｜ **陷阱六条**：**霍桑效应**（被度量则行为改变）· **虚荣指标**（对决策无帮助）· 士气低落 · 误用 · 确认偏见 · **相关≠因果**

提问 如果只能给客户报 3 个质量指标，你选哪 3 个？为什么？（限 1 分钟）

预设回答

- ✅ 正确方向：一个稳妥组合是**缺陷密度**（每千行代码缺陷数，看“做得怎么样”）、**一次通过率**（看“交付得怎么样”）、**售后 PPM**（百万件不良率，看“用户感受怎么样”）。三者都可算、可比、能跨项目对照，分别覆盖过程侧、交付侧与客户侧；再加一个提前指标（未关闭缺陷的增速）就更完整——它预测趋势，其余三个都是滞后指标。
- ⚠️ 部分对：答“缺陷总数、测试用例数、Bug 关闭率”。能算，但**测试用例数接近虚荣指标**：它只说明投入了多少，不说明质量到了什么水平；用例写得再多，覆盖不到关键路径也没有意义。补一句：指标要能回答“变了以后谁的动作会变”。
- ❌ 常见错：答“把能报的都报上去，客户自己看”。指标越多越没人看，这叫**误用度量指标**；报三个能联动的，比报十五个各自为政的有用得多。

易错点 / 考点 ① **提前指标与滞后指标**的区分是选择题高频点：能预测的是提前，已发生的是滞后，进度偏差与成本偏差属于滞后。② **霍桑效应**的内核是“被度量本身改变了行为”，不是“数据造假”，两者别混。③ **虚荣指标**的判据是“对决策有没有帮助”，不是“数字大不大”。④ 相关性与因果混淆这一条常与“缺陷多与代码规模同增”一起考，答案是**相关不等于因果**。⑤ SMART 的中文五要素要能报全。

过渡 *度量回答“数字说明什么”，交付回答“东西交出去算不算成”——而且它给了一句很硬的话：交付不只是范围和需求。*

12.26 交付绩效域：按验收标准核实，再与干系人确认（2 分钟）

口播 · 可照读

这一页有一句话，请原样记住：**交付不仅仅只是范围和需求。范围和需求聚焦于需要交付的内容，质量聚焦于需要达到的绩效水平。** 同一件交付物要过两把尺子：**符合规格**，也就是核实的可交付成果；**干系人认可**，也就是验收的可交付成果。两把尺子都过，才叫交付。

两把尺子的差别，用一张表看最清楚。**聚焦点**不同：范围与需求问“要交付什么”，质量问“要达到什么绩效水平”。**写在哪儿**不同：范围基准、WBS、需求文件写“内容”；质量需求则可能反映在**完成标准、完成的定义、工作说明书或需求文件中的质量条款**里。**谁来判**不同：内容由干系人确认，绩效由测试与检查来核实、由过程审计来保证。**成本视角**也不同：与质量相关的成本由项目所在组织承担，所以在项目管理中必须**权衡质量要求与实现质量所需成本**之间的平衡——不是越高越好，是合适才好。

为什么前期质量这么要紧？因为**缺陷会累积、会放大**。设计和开发工作通常基于前期的需求和范围展开，前期工作若有缺陷，后面的工作会把缺陷带着走。所以**发现缺陷的时间越晚，修复的成本越高**；同样，前期工作完成得越多，变更的成本也相应增加。这条曲线教材画在图 7-12：需求阶段约 1X，设计约 5X，构建 20X，测试 50X，到生产阶段能到 150X。**积极主动地开展质量工作，就是在躲开这条曲线陡起来的那一段。**

那么交付绩效域怎么检查？教材给的办法很实在：**用质量问题来衡量**——用户反馈中的缺陷报告数量、系统崩溃次数、功能异常问题的频率。这些与质量相关的问题数量，既反映用户对交付成果的满意度，也是评估项目质量的重要指标。请注意这几个数的性质：它们都是**从使用现场长出来的**，不靠自己填表，所以最难粉饰。

再看它和上一节 7.4 的衔接。控制质量产出**核实的可交付成果**——通过检查与测试，证明它符合规格；确认范围才产出**验收的可交付成果**——干系人按验收标准点头。顺序不能倒：先核实、后确认。现实里最常见的错位，是拿“测试全过”当“验收通过”——测试证明的是“按我们写的规格做对了”，验收问的是“这是不是当初说好的东西”。

落到案例。汽车质量系统的报表模块，单元测试全绿，但客户验收时提出“月报要能按车型下钻”。测试没错，规格里也没写——这就是**范围与需求的缺口**，不是质量的缺口。处理它只有一条正路：提变更、做影响分析、CCB 决策、改基线，然后重新核实、重新确认。绕过去直接上线，欠下的就是外部失败成本，最后连信任一起赔。

板书 **交付 = 范围／需求（交什么）+ 质量（达到什么绩效水平）** ｜ 两把尺子：**核实**（测试・检查→**核实的可交付成果**）→ **确认**（干系人按验收标准→**验收的可交付成果**），**顺序不可倒** ｜ 质量需求写在：完成标准 ・ **完成的定义（DoD）** ・ 工作说明书 ・ 需求文件的质量条款 ｜ 成本权衡：质量成本由**组织承担**，要求与成本要平衡 ｜ 变更成本曲线：**1X → 5X → 20X → 50X → 150X**（需求→设计→构建→测试→生产） ｜ 检查口径：**缺陷报告数 ・ 崩溃次数 ・ 功能异常频率**

提问 某模块单元测试全部通过，能不能直接宣布“交付完成、可以上线”？

预设回答

- ✅ 正确方向：不能。测试通过只说明**符合规格**，交付还要按**验收标准**与干系人确认；两道关不能互相顶替。补充一句判据：验收标准是当初与干系人一起写下的那几条，不是测试报告里自拟的通过条件。
- ⚠️ 部分对：答“要客户签字才算交付”。签字是形式，实质是**按验收标准逐条确认**；只签字不逐条对照，签下来的是一份没有依据的认可，出问题时双方都说不清依据在哪。
- ❌ 常见错：答“项目经理同意就能上线”。上线是交付决策，不是个人判断：它要有核实记录、验收确认、以及必要时的变更批准；缺任何一项，风险都留给使用者承担。

易错点 / 考点 ① **核实 ≠ 确认**：核实对内（符合规格），确认对外（干系人认可）；对应的两个成果名称常被互换着考。② 那句原话要能背：**范围和需求聚焦内容，质量聚焦绩效水平**。③ 变更成本曲线五个倍数与阶段的对应关系要能报出，尤其 **1X 在需求、150X 在生产**。④ “质量越高越好”是错的：要与实现成本权衡，这与质量成本曲线的最低区是同一个道理。⑤ 交付域的检查口径是**现场数据**（缺陷报告、崩溃次数、异常频率），不是内部工时。

过渡 *两本账都记完了，接下来把方法交到手上——实验 5 和课堂练习，练的是同一件事：用数据说话。*

***【思政落点】**（用户中心，宁返工不放行）：外部失败成本最高——缺陷流到用户手里，赔的是钱，更是信任。“差不多能用了”从来不是验收标准；验收标准是当初和干系人一起写下的那几条，写下来了就要做到。*

12.27 实验 5 任务预览：缺陷三态流转 + 四项任务（2 分钟）

口播 · 可照读

本页为课后拓展，课上可跳过。实验 5 是**课外自主实践（选做）**，不计学时学分、不单独计分，成果作为课堂讨论与作业的素材；愿意做的同学把这一页记下来，回去自己走一遍。

它要练的东西不复杂：用 Jira 提交测试缺陷，走完 **待办 → 已修改 → 已完成** 三态流转，再对缺陷做统计分析。教材用教务管理系统做例子，缺陷单写的是“平时成绩与期末成绩无法加权计算”。界面操作这里只做**随堂投影演示（跟随观看，课后可自行复现）**，课后用电子表格或看板记同样的三态，结论一样成立。

三态为什么要单设“已修改”？因为**待办是测试提的，已修改是开发改完的，已完成是测试回归通过的**——三个人、三个动作、三个状态，谁也不能替谁点下一步。缺了中间这一态，“改完了”和“验过了”就挤在一起，回归测试这一步会被悄悄跳过。

缺陷单要写四要素：**预置条件、重现步骤、实际结果、期望结果**。缺任何一条，开发都得回来问你一遍；四条齐了，一份缺陷单就能独立复现。

实验有四项任务：一页纸的**质量计划要点**，含三项目标、至少三项测量指标、七工具选配、审计与检查安排；**核查表加因果图**各一张，鱼骨从人、机、料、法、环、测六个方向追原因；**缺陷数据分析**，按类型与模块统计频数、占比、累计占比，给出整改优先级；**配置记录**，建配置项清单并写一次完整的变更记录。四项里最容易做成“抄概念”的是第一项，判据只有一条：**写的东西能不能被检查**——指标有没有定义、计算式、目标值、采集频率。

板书（★课后拓展）：**课外自主实践（选做）**・不计学时学分・不单独计分 ｜ 三态：**待办**（测试提）→ **已修改**（开发改完）→ **已完成**（测试回归通过） ｜ 缺陷四要素：**预置条件 ・ 重现步骤 ・ 实际结果 ・ 期望结果** ｜ 四项任务：**质量计划要点 ・ 核查表＋因果图 ・ 缺陷数据分析（帕累托） ・ 配置记录（CI 清单＋一次变更）** ｜ 判据：**写得能被检查**

提问 四项任务里，哪一项最容易做成“抄一遍教材概念”？

预设回答

- ✅ 正确方向：第一项，质量计划要点。写成“提高质量、加强测试、确保客户满意”就等于没写——**判据是能不能被检查**：每条指标要有定义、计算式、目标值与采集频率；每类缺陷要有工具与责任人；每次变更要能查到版本号与批准记录。
- ⚠️ 部分对：答“配置记录最容易照搬格式”。格式可以照搬，内容不行：清单里的编号、版本、责任人、所在库必须对应自己项目里的真实文件；一次变更记录要有“因为什么改、谁批准、改后什么版本”三行实事。
- ❌ 常见错：答“帕累托图最难，画不出来”。图能画，难的是**结论**：只要数据是自己记的，累计占比一算，结论就落在一句“先改哪两类、谁改、用什么验证”上，反而最容易做出真东西。

易错点 / 考点 ① 三态的**提交人不同**：待办由测试提、已修改由开发置、已完成由测试回归后置，这条顺序是考点。② 实验定位是**课外自主实践（选做）**，不计学时学分、不单独计分，别当成计分任务去准备。③ 缺陷四要素缺一不可，尤其**预置条件**最常被漏。④ 本页只做任务预览，细节回去看下一页指引。

过渡 *任务清单有了，回去怎么动手？下一页给四步做法、两张图的画法，还有文件命名。*

12.28 实验 5 做法指引：自拟数据 · 电子表格绘图 · 文件命名（2 分钟）

口播 · 可照读

这一页是做法，四步走完就能交东西：**攒数据、统计、画图、下结论**。实验仍是**课外自主实践（选做）**，不计学时学分、不单独计分；绘图用电子表格，随堂投影演示（跟随观看，课后可自行复现）。

第一步攒数据：自拟 **30 到 50 条**缺陷，字段统一成“编号、发现日期、模块、类型、严重度、状态”。类型别开太多，四类就够——接线配置错、接口、文档、其他。数据自拟没有关系，**关键是口径一致**：同样一条缺陷，两次归类要落在同一类。

第二步统计：按类型、按模块各数一遍频数，再算占比。这一步用数据透视表最快，也最不容易错。

第三步画图，两张。**帕累托图五步**：按类型统计频数；按频数**降序**排列；加“占比”与“累计占比”两列；柱形画频数、折线画累计占比，两者用**次坐标轴**，百分比轴设成 0 到 100；最后在 80 处画一条参考线，线左侧的类型就是优先整改对象。**控制图五步**：取 10 到 12 个批次的同一指标，比如每周缺陷数或一次通过率；算均值 $\bar{x} = \frac{1}{n}\sum x_i$ 与标准差 σ；画中心线 $\bar{x}$ 与上下控制限 $\bar{x} \pm 3\sigma$；逐点连线；按四条判异规则标注——越界、连续 7 点同侧、连续 7 点持续升或降、明显的非随机模式。

第四步下结论，一句话就够，但要能落地：**先改哪两类、由谁在什么时候改、改完用什么指标验证。** 只报一个百分比，不算结论。

最后是文件命名，这条最不起眼，却直接影响交上来能不能被认出来。统一格式：**学号-实验5-内容-版本.扩展名**，例如 2024xxxx-实验5-缺陷记录-v1.xlsx、2024xxxx-实验5-帕累托图-v1.png。同一实验放同一个文件夹，改版只升 v2、v3，**不写“最终版最终版”**；原始数据另存一份不改，分析在新表上做。数据一改，结论就变，原始记录是唯一能回溯的锚点。

补一句自拟数据的规矩：40 条缺陷，第一类占 45%，能不能直接宣布“关键少数”？不能——45 离 80 还有距离，要看累计到多少才越线；样本小，个别归类变化就能让占比跳几个百分点，结论要写“倾向”；再换一个维度（按模块或按发现阶段）核一遍，两个维度一致才可信。

板书 四步：**攒数据**（30–50 条 · 字段统一）→ **统计**（按类型／模块计数）→ **画图**（帕累托／控制图）→ **下结论**（一句可执行） ｜ 帕累托五步：降序 → 占比＋**累计占比** → 柱＋折线**次坐标轴**（0–100%）→ **80 处参考线** ｜ 控制图五步：取 10–12 批 → $\bar{x}$ 与 σ → 中心线 $\bar{x} \pm 3\sigma$ → 连线 → **四条判异** ｜ 命名：**学号-实验5-内容-版本** · 改版只升 v2／v3 · **原始数据另存不改**

提问 40 条自拟缺陷，第一类 18 条（45%），能据此宣布“关键少数”吗？（限 30 秒）

预设回答

- ✅ 正确方向：不能急着宣布。三条理由：45% 离 80 线还远，得看累计到多少才越线；样本量小，个别记录的归类变化就能让占比跳几个百分点，结论只能写“倾向”；换维度再算一次（按模块或按发现阶段），两个维度一致才可信。
- ⚠️ 部分对：答“可以，45% 已经是最多的那一类”。最多，但不等于“关键少数”——帕累托要的是**累计越过 80%** 的那一组类型，单看第一名没有意义；结论要写成“先改哪两类、谁改、用什么验证”。
- ❌ 常见错：答“数据是自拟的，怎么算都行”。自拟的是数据来源，不是算法：**降序、累计、80 线**这套规矩一条都不能改，改了就不是帕累托，只是一张柱图。

易错点 / 考点 ① **帕累托必须降序**，不排序的累计占比毫无意义。② 控制图的控制限是 $\bar{x} \pm 3\sigma$，**算的是过程自己的波动**，与规格界限无关，别写成规格上下限。③ 次坐标轴与 0–100 的百分比轴是帕累托图的**形式要件**，答辩与作业都会看。④ 文件命名三要素（学号、内容、版本）是提交的硬项，缺一项就可能被判为未交。

过渡 *方法讲完，接下来当堂动手——三道练习，从算到判，十分钟内见分晓。*

12.29 课堂练习：帕累托计算 · 工具归类 · 判断 3 道（1 分钟）

口播 · 可照读

现在动手，三道题，先自己做，做完对答案。第一题帕累托当堂计算：某模块共 **500 条**缺陷，其中**接线配置错 320 条、接口 90 条、文档 40 条、其他 50 条**。要求三件事：按降序算出各类占比与累计占比；判断应当优先整改哪几类；最后校验一遍合计。

算法只有一条路径：先排序，再逐项累加。接线配置错 $\frac{320}{500}=64.0\%$，累计 64.0%；接口 $\frac{90}{500}=18.0\%$，累计 82.0%；文档 $\frac{40}{500}=8.0\%$，累计 90.0%；其他 $\frac{50}{500}=10.0\%$，累计 100%。结论：前两类累计 **82%** 已越过 80 线，**优先整改接线配置错与接口**；其余两类合计 18%，先登记、暂不投专项资源。校验一句：四类占比相加必须等于 100%，对不上就是算错了。

第二题工具归类，四个场景。①“同类接线错误三个月里反复出现，想弄清到底为什么”——首要工具是**因果图**，鱼骨从人、机、料、法、环、测六个方向追，必要时配流程图看步骤，用途归**发现原因**。②“缺陷集中在少数几类，先改哪一类”——**帕累托图**，降序柱加累计折线，用途归**找重点、优先排序**。③“近 12 批的缺陷数上下跳，想判断过程是否稳定”——**控制图**，中心线加减三倍标准差加四条判异，用途归**监控过程**。④“模块代码规模与缺陷数是不是一起变”——**散点图**，可加回归线，但**相关不等于因果**，用途归**收集与展示数据**。

第三题判断三道，请逐条说出理由，不要只报对错。第一道：质量等级越高，质量越好——**错**。质量是满足需求规格的程度，等级是功能多少与技术特性的档次；高等级可能高质量，也可能因过度设计而失败，低等级同样可以高质量。第二道：控制图连续 7 个点落在中心线同一侧，说明过程出现异常波动——**对**，这是四条判异规则之一；但要补一句，**判异只说明过程异常，不等于产品不合格**。第三道：质量保证面向过程，质量控制面向成果——**对**，管理质量做审计与过程改进，控制质量检验交付物、产出核实的可交付成果。

提醒一句答题节奏：计算题**写过程**，归类题**写用途**，判断题**写理由**。三样都写了，才拿得到全部分；只写答案，阅卷时无法判断你是会了还是蒙对了。

板书 第 1 题（500 条）：接线配置错 320 → **64.0%／累计 64.0%**；接口 90 → **18.0%／82.0%**；文档 40 → **8.0%／90.0%**；其他 50 → **10.0%／100%** → **前两类 82% 优先整改**；校验：合计＝100% ｜ 第 2 题：①因果图→**发现原因** ②帕累托图→**找重点排序** ③控制图→**监控过程** ④散点图→**收集与展示**（**相关≠因果**） ｜ 第 3 题：①**错**（等级≠质量）②**对**（判异≠不合格）③**对**（QA 过程／QC 成果）

提问 第 1 题算到累计 82% 就够了吗——如果客户说“其他”那 50 条里有一大半是同一个原因引起的，结论会不会变？

预设回答

- ✅ 正确方向：会变。帕累托的横轴是**分类**，分类一改，频数分布与累计曲线就跟着改；“其他”里若藏着同一原因引起的一批缺陷，把它拆出来单独成类，越过 80 线的那一组可能就从“两类”变成“两类加新类”。所以做帕累托先问一句：**这个分类是按什么口径切的，能不能落到整改动作上**。
- ⚠️ 部分对：答“不会变，总数还是 500 条”。总数确实不变，但累计占比是按类型重新分配的，合并或拆分类别都会改变“谁先越过 80 线”，而这正是结论本身。
- ❌ 常见错：答“‘其他’不用管，反正只占一成”。它占 10%，在四类里排第三；忽略它等于承认有一成缺陷不处理。正确说法是“**先登记、暂不投专项资源**”。

易错点 / 考点 ① **先排序、再逐项累计**：本页口径为 320 → 90 → 40 → 50，累计 64.0%／82.0%／90.0%／100%；最后一类累计必须正好落到 100%，这是整题唯一能自证的校验点。② **80% 是习惯线不是定律**，说“必须正好 80%”是错的，越过 80 的那一组就是关键少数。③ 四个场景与用途组的配对是高频选择题：因果图与流程图→**发现原因**；核查表、直方图、散点图→**收集与展示**；帕累托→**找重点排序**；控制图→**监控稳定**。④ 判断题必须写理由，“判异≠不合格”“QA 对过程、QC 对成果”这两句是必答点。

过渡 *答案对不对是一时的，错在哪个环节才是一世的——下一页把三处最容易错的地方摊开看。*

12.30 答案点评：三处最容易错的地方（1 分钟）

口播 · 可照读

练习题的对错不重要，**错在哪个环节**才重要。三段点评，对应刚才三道题。

第一处，计算：顺序与校验。 错法一，没按降序排就累计——累计占比于是失去意义，画出来的折线忽高忽低，读不出“关键少数”。错法二，忘了校验“四类相加等于 100%”，一步算错后面全错。错法三，只报“前两类合计 82%”，没落到动作。合格答案是一条链：**先排降序、逐项累计、对着 80 线看、写成谁改什么用什么验证**。四步缺一，扣的不是计算分，是分析分。

第二处，归类：别把工具当万能。 最常见的三个错：把**帕累托图**当成“发现原因”的工具——它只排序，原因要靠因果图与流程图去追；把**控制图**当成“合格判定”——它判的是过程稳不稳，不判产品合不合格；把**直方图**当成“看趋势”——它看的是分布形状、集中趋势与分散程度，而且**不考虑时间对分布内变化的影响**，趋势要看时间序列。

第三处，判断：三处概念边界。 质量与等级、判异与不合格、质量保证与质量控制——这三对概念每年都考。给一个记法：**等级是档次，判异是信号，保证管过程**。质量是“做对了没有”，等级是“做到什么水平”；判异是过程给你发的信号，不是判决书；QA 对着过程做审计与改进，QC 对着成果做检验与核实。

再往上走一层，三条加分观察。**第一**，80% 是习惯线不是定律，“二八”只是近似——本例第一类 64% 还没过线，加上第二类才到 82%，所以答案是**两类**，不是“一类”，也不是“前三类”。**第二**，换维度重算，结论可能不同：按模块算与按发现阶段算，冒头的类型可能换人，所以**维度选择本身会影响结论**，实验里必须写清口径。**第三**，占比高不等于该先改——还要看整改成本与风险：接口类改动可能牵动上下游，需要单独评估，不能只按条数排队。

最后一句话留给案例：汽车质量系统里，接线配置错往往是“人能改、改得快、见效快”的那一类，接口问题则要跨团队。**先改第一类、同时为第二类做根因分析**，这就是帕累托给优先级、而不是下命令的意思。

板书 ① **计算**：降序 → 累计 → 对 80 线 → 落到动作（＋合计校验 100%） ｜ ② **归类**：帕累托**只排序**（不找原因）· 控制图**判过程**（不判合格）· 直方图**看分布**（不看趋势） ｜ ③ **边界**：**等级是档次 · 判异是信号 · 保证管过程** ｜ 加分：80% 是**习惯线** · 换维度结论可变 · **成本与风险**也决定先后

提问 只改第一类（接线配置错，占 64%）够不够？为什么？（限 1 分钟）

预设回答

- ✅ 正确方向：不够。累计 64% 还没越过 80 线，第二类仍有 18% 的分量，漏掉它整改效果会打折；但也不能机械地“凑到 80%”——如果第二类整改成本极高，可以先做第一类，同时为第二类做根因分析，把改动排进下一个迭代。帕累托给的是优先级，不是命令。
- ⚠️ 部分对：答“够了，64% 已经是大头”。大头没错，但“大头”不是判据，判据是**累计是否越过 80 线**；这一题的设计就是要让人看到“第一名 64% 仍不够”的情形。
- ❌ 常见错：答“应该四类一起改，全面整改最稳妥”。资源有限时“全面整改”等于没有优先级，正是帕累托要治的病；四类一起上，最可能的结果是哪一类都没改透。

易错点 / 考点 ① 帕累托的用途**只有排序**，把它说成“找根本原因”是典型错项。② **直方图不考虑时间因素**，这是它与控制图最关键的区别，也是判断／选择题的原题点。③ 三对概念要给得出**一句话边界**，而不能只报名字。④ “占比最高就该先改”不成立，还需过成本与风险这一关。⑤ 结论要写成可执行句，**只报百分比不得分**。

过渡 *练到这儿，整章的零件齐了。最后两页把它们装成一台机器——先看小结那条主线。*

***【思政落点】**（数据要真，口径要清）：缺陷分类时把难改的归到“其他”、把已修的重复计数，帕累托照样画得漂亮，结论却是假的。统计口径写清楚、原始记录不改动，是质量工作的底线——**数据可以不好看，但不能不真实**。*

12.31 本章小结：一条主线、七个关键词（2 分钟）

口播 · 可照读

这一章讲完了，收成一句主线：**质量不是检出来的，是规划出来、保证出来、控制出来的**；而配置管理让每一次改动都留下证据。六个环节串起来：**概念、规划质量、管理质量、控制质量、配置管理、两个绩效域**。

第一环概念：质量与等级的分野——质量是满足需求与规格的程度，等级是功能与性能的档次；低等级可以是高质量，高等级也可能因过度设计而失败。质量成本四类：预防与评估是一致性成本，主动花；内部失败与外部失败是非一致性成本，被动赔；规律是预防评估投入上升、失败成本下降，**总质量成本存在最低区**。还有一句“预防胜于检查”——缺陷越晚发现代价越大。

第二环规划质量：把“要好”翻译成“可测量、可验收”，产出质量计划与质量测量指标，方法里最硬的是七种基本质量工具。第三环管理质量，也就是质量保证：面向**过程**，靠根本原因分析、质量审计、面向 X 的设计、质量改进方法，产出质量报告与测试评估文件。第四环控制质量：面向**成果与验收**，靠检查、测试与产品评估、统计抽样、绩效审查，产出**质量控制测量结果、核实的可交付成果、工作绩效信息**。

第五环配置管理：三库、基线、CCB、配置审计——它的作用是把“改了什么”变成可查的记录。第六环两个绩效域：度量回答“指标能不能改变动作”，交付回答“东西交出去算不算成”。

七个关键词请带走。**CoQ，质量成本**：一致性成本是预防加评估，非一致性成本是内部失败加外部失败，总成本有最低点。**QA、QC、QM**：QM 是质量管理，QA 面向过程，QC 面向成果。**SPC，统计过程控制**：用控制图把正常波动与异常波动分开，四条判异——越界、连续 7 点同侧、连续 7 点升或降、明显非随机模式；记住判异不等于不合格，控制界限是加减三倍标准差，与规格界限是两回事。**CI，配置项**：受配置管理控制的工作产品，要唯一标识、按目录结构入库。**CCB，变更控制委员会**：评估变更的技术、逻辑、策略、经济与组织影响，决定批准、驳回或缓办。**DoD，完成的定义**：团队共同认可的“做完”标准，敏捷里用它把内建质量写进每一项工作。**PPM，百万件不良率**：把客户端的抱怨换算成可比较的数字，也就是案例里那道售后闸口。

最后一个回环别忘：控制质量发现的问题，要回到规划与管理质量——**改指标、改过程，而不是只改这一件产品**。只改产品，下一次还会犯；改过程，同类缺陷才会少。

板书 主线：**质量不是检出来的，是规划出来、保证出来、控制出来的**；配置管理让每次改动留下证据 ｜ 六环：概念 → 规划 → 管理（QA）→ 控制（QC）→ 配置管理 → 两绩效域（度量・交付）→ **回环：改指标、改过程** ｜ 七词：**CoQ**（一致性＝预防＋评估；非一致性＝内部＋外部失败；**总成本有最低点**）・**QA／QC／QM**（QM 总称・QA 过程・QC 成果）・**SPC**（四条判异・**判异≠不合格**・$\pm 3\sigma$≠规格界限）・**CI**（唯一标识・入配置库）・**CCB**（批准・驳回・缓办）・**DoD**（做完的标准）・**PPM**（百万件不良率）

提问 这七个关键词里，哪一个最能解释“为什么很多团队测试做得很足，质量还是上不去”？

预设回答

- ✅ 正确方向：**QA／QC／QM 的分工**。测试做得多属于 QC 的范畴，面向成果、事后把关；如果 QA 缺位——过程没有审计、根因没有分析、流程没有改进——同样的缺陷会一批批重新长出来。测试能挡住流出，挡不住复发。
- ⚠️ 部分对：答“CoQ，因为测试太贵”。方向有一半对：评估成本上升而预防投入不足，确实会把总质量成本推高；但“贵”只是表象，**根因仍是过程没改**，只压测试预算只会让缺陷流到外面。
- ❌ 常见错：答“DoD，因为团队没有定义做完”。DoD 重要，但它解决的是“什么叫做完”，不解决“为什么会做错”；把质量问题全推给定义缺失，等于绕开了过程改进这一环。

易错点 / 考点 ① 七个关键词的**一句话内核**要能默写，期末简答与选择都从这里出。② **CoQ 四类归属不能混**：返工、废品、重测属内部失败；保修、投诉、信誉损失属外部失败。③ **控制界限与规格界限是两回事**，前者管过程稳定（$\pm 3\sigma$），后者管产品合格，这一条几乎每年考。④ QA 与 QC 的成果不要互换：质量报告、测试与评估文件属管理质量；质量控制测量结果、核实的可交付成果、工作绩效信息属控制质量。

过渡 *最后几分钟，把作业说清楚，再留一个问题带到第 8 章。*

***【思政落点】**（质量是干出来的，不是写出来的）：把缺陷记录如实填、把变更按流程走、把验收标准当承诺——这三件小事，就是质量体系落在一个人身上的样子。体系写在文件里只是纸，落在动作上才是质量。*

12.32 作业与下章预告：三题作业＋第 8 章资源管理（1 分钟）

口播 · 可照读

最后几分钟，说三件事：作业、开放题、下一章。

作业是章后习题，**选 2 到 4 题**，题号以教材为准，建议三类各约一题：**概念题**考质量与等级的区别、质量成本的构成、为什么说“预防胜于检查”；**工具题**考七种工具的用途归类与帕累托的累计占比计算；**配置管理题**考配置库三库与基线、以及一次变更该走什么流程。要求两条：**每条不超过三句话，各举一个软件场景的例子；写依据，不只写结论**——比如“预防胜于检查”，要写出“缺陷越晚发现，修复成本从 1X 涨到 150X”，而不是只写一句口号。

顺带点三个易错的选择题方向。第一，**质量等级不等于质量**，高等级可能低质量，这是判断题的常客。第二，**配置库三库**的名称与定位要配对：受控库存当前基线及变更，产品库存通过测试的最终产品并含所有发布版本。第三，**度量绩效域要重点关注的内容**里没有“风险”——制订有效的度量指标、度量内容与相应指标、度量陷阱、持续改进才是它的四项。这三处都是送分题，也是丢分题。

开放题一道：为综合小组项目，或自拟一个软件场景，自拟质量测量指标**不少于 5 项**。每项要写清**定义、计算式、目标值、采集频率、采集方式**，并为每项**选配一种工具或图形并说明理由**。成果可以并入课外综合小组项目的质量要点部分，下次课自愿分享。评分只看一件事：**这个数变了，谁的动作会变**——答不出这一句的指标，一律算无效指标。

下章预告：第 8 章 软件项目资源管理，第 13 周。四个板块：资源管理计划——怎么管人、怎么分工；估算与获取资源——要什么角色、从哪来；建设与管理团队——能力、协作、激励；控制资源——用得住、用得值。责任分配矩阵 RAM 与 RACI 会在那里出现，冲突管理与激励理论也会讲。从这一章起，视角从“管好事”转到“管好人”。

留一个问题带过去：**质量指标压得紧、人手又不够，先动人还是先动指标？** 这个问题没有标准答案，但它逼着我们把质量与资源放在同一张桌上算账——下一周我们就从这张桌子开始。

板书 作业＝章后习题**选 2–4 题**（概念 · 工具 · 配置管理各约 1 题）＋**开放题 1 道** | 要求：**每条≤3 句＋一个软件场景 · 写依据不写口号** | 三个易错点：**等级≠质量 · 三库定位配对 · 度量域四项不含风险** | 开放题：指标 **≥5 项**（定义 · 计算式 · 目标值 · 采集频率 · 采集方式）＋**工具选配与理由**；判据：**这个数变了，谁的动作会变** | 第 8 章：资源管理计划 · 估算与获取 · 建设与管理团队（RAM／RACI · 冲突 · 激励）· 控制资源

提问 开放题要求“为每项指标选配一种工具并说明理由”，如果某项指标选不出工具，说明什么？

预设回答

- ✅ 正确方向：说明这项指标很可能**不可采集或不可比较**——工具是数据的出口，选不出出口，多半是定义与计算式没写清，或者它的数据源根本不存在。处理办法：要么改成可采集的替代指标，要么承认它是定性判断、不放进指标表。
- ⚠️ 部分对：答“可以选控制图，什么指标都能画”。控制图要求同一指标按批次连续采集，且要有明确的计量口径；硬套上去只会得到一张没有判异意义的折线，这属于**误用度量指标**。
- ❌ 常见错：答“指标和工具没关系，工具只是装饰”。工具决定这张图能回答什么问题：帕累托答“先改谁”，控制图答“稳不稳”，散点图答“是不是一起变”；工具选错，结论方向就错。

易错点 / 考点 ① 作业是**选 2–4 题**，不是全做，但三类要覆盖到。② 开放题的**五个要素**（定义、计算式、目标值、采集频率、采集方式）缺一项即扣分，这是评分表上的硬项。③ 度量绩效域的四项内容里**没有“风险”**，这一条是选择题原题。④ 下章是**第 8 章 软件项目资源管理**，别记成整合管理或沟通管理。

过渡 *这一章到此收尾。下一周换个角度：质量定下来了，**人从哪来、怎么带**——这就是资源管理要回答的。*

***【思政落点】**（把标准写下来，把承诺做出来）：作业里每一条指标，都要经得起一句追问——**这个数变了，谁的动作会变？** 答得出，指标就是管理工具；答不出，它只是一行好看的数字。*

口播 · 可照读

四、板书设计（建议）

```
┌──────────────────────────┬──────────────────────────┬──────────────────────────┐
│ ① 概念与成本（左）       │ ② 七工具（中，主）       │ ③ 配置与绩效域（右）     │
│                          │                          │                          │
│ 质量＝满足规格的程度     │ 发现原因：因果图 · 流程图 │ 基线＝复审通过 · 正式接受 │
│ 等级＝功能性能的档次     │ 收集展示：核查表 · 直方图 · 散点 │ ＝起始点与参考点 │
│ 低等级可以是高质量       │ 找重点：帕累托（100%）   │ 三库：开发→受控→产品     │
│                          │ 控过程：控制图           │ 变更三问题：版本冲突/    │
│ 成本四类：               │                          │ 追踪困难/低效协作        │
│ 一致性：预防＋评估       │ 判异四条：越界/7点同侧/  │ 流程：识别CI→建库定基线→ │
│ 非一致性：内部＋外部失败 │ 7点升降/非随机           │ 变更→CCB→更新→审计       │
│ 总成本有最低区           │                          │                          │
│ 预防 1X → 检查 150X      │ 判异≠不合格              │ 度量陷阱：霍桑/虚荣      │
│                          │ 控制界限≠规格界限        │ 交付：质量＝达绩效       │
└──────────────────────────┴──────────────────────────┴──────────────────────────┘
```

五、常见学生疑问与应答

学生可能问	应答要点
质量和等级到底怎么分？	一句话：**质量问"做对没有"，等级问"做到什么档次"**。单账本的基础版记账 App，功能少（等级低）但零崩溃、账目分毫不差（质量高）；反过来，功能堆满的 App 天天闪退，就是高等级低质量

学生可能问	应答要点
为什么"低等级可以是高质量"?	因为质量的判据是**有没有满足自己写清的规格与验收标准**。规格写得清楚、做到了，就是高质量；与功能多少无关
质量成本越高越好还是越低越好?	都不对。**追求总成本最低，不是某一项最低**。预防与评估多花一点，失败成本能降很多；但堆到超出最低区，多花的钱买不回等值损失
帕累托图为什么要看"累计 82% 先改"?	因为它是**排序工具**，不是描述工具。少数几类原因往往占大多数缺陷，先改这几类，投入产出比最高。本例前两类占了 82%，先动它们
直方图和控制图都画柱子/点，区别在哪?	关键在**时间**。直方图看分布形状（集中趋势、分散程度），**不考虑时间顺序**；控制图看**随时间变化的稳定性**。前者答"数据长什么样"，后者答"过程稳不稳"
控制图上的点都在界限内，产品就一定合格吗?	**不一定**。控制界限是 ±3σ 的**过程**自然波动范围，规格界限是**产品**允不允许的范围。点全在控制界限内只说明过程稳定可预测，数值仍可能整体偏移、全部落在规格界限之外——那就是"稳定地产出不合格品"
QA 和 QC 到底谁管谁?	**QA 管过程（防），QC 管成果（检）**。QA 通过审计、过程分析、面向 X 的设计让缺陷少产生；QC 通过检查、测试、抽样、绩效审查判断做出来的东西合不合格。一个往回看流程，一个往前看结果
配置库和我电脑上的文件夹有什么区别?	文件夹只存文件；配置库管的是**版本、修改历史、依赖关系**，还能回答"当前是哪个版本、谁改的、什么时候改的"。更关键的是：库里的**基线**一旦确立，改动必须走变更控制，不能随手覆盖
不更新配置库直接改代码会怎样?	三个问题：**版本冲突**（两人改同一模块互相覆盖）、**问题追踪困难**（没记录就定位不到）、**低效协作**（各改各的，合并时扯皮）
霍桑效应、虚荣指标是什么坑?	**霍桑效应**：人知道自己被测量，行为就变了（一被统计，缺陷数突然"变少"，其实是漏报）。**虚荣指标**：看着好看但不驱动决策的指标（比如"累计代码行数"）。指标要能改变某个人的动作才有意义
## 六、作业与下节预告	
课后作业（计入作业 20%，学习通发布）	
1. **概念题**：质量与等级的区别，各举一个软件产品例子。	
2. **七工具归类**：给 6 个场景，各选一种工具并说明理由。	
3. **配置管理流程**：画出"识别 CI → 建库定基线 → 变更 → CCB 评审 → 更新版本 → 配置审计"流程，标出每步谁负责。	
4. **开放题**：为本组项目设计 **≥5 项质量测量指标**，并为每项选配一种工具；可并入综合小组项目 M2 的"质量要点"。	
5. ▲ **课外自主实践（选做，不单独计分）**：完成**教材实验 5**——用 Jira 提交测试缺陷并走完 **待办 → 已修改 → 已完成** 三态流转，整理缺陷统计。	
下节预告：第 13 次课进入**第 8 章 软件项目资源管理**——质量管住了"做得好不好"，接下来管"谁来干、够不够、怎么配合"。会看到 RACI 矩阵（谁负责谁批准）、资源分解结构、资源直方图与过度分配的识别纠正，以及团队建设与激励理论。	
课外阅读（★课后拓展页）：12.27 实验 5 任务预览（课外选做，课后完成并带结论回课堂分享）。	