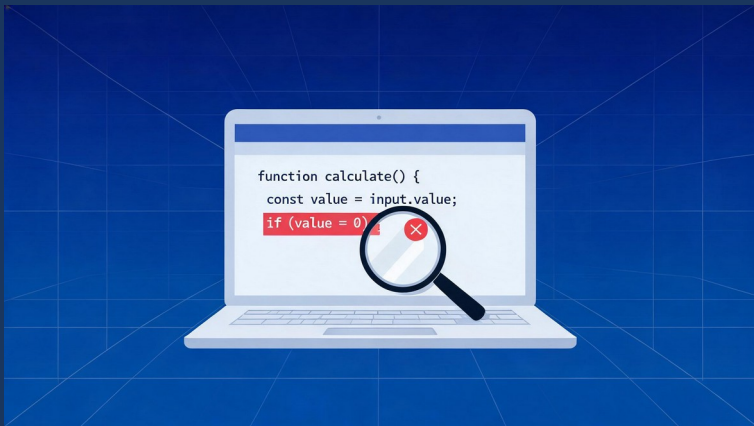


# 第 1 章 引论 —— 为什么必须做软件测试?



## 软件测试课程设计第 1 章 引论 · 课程导学

专业选修 | 课程编码 154442008 | 总学时 32 · 2 学分

教学安排：第 1-17 周（含实验，配套教材《软件测试方法和技术（第 4 版）》）授课教师：周宇文 | 联系电话 / 微信：13580390715

一架火星飞船坠毁、一家券商 45 分钟亏掉 4.6 亿美元、一台治疗仪让 4 名病人失去生命……它们之间有一个共同点：缺陷没有被及时发现。

## 1.1 软件测试的必要性

从真实事故出发：缺陷离我们并不远，代价有多大？

## 1.2 为什么要进行软件测试

软件总存在缺陷——只有测试能发现它、清除它。

## 1.3 什么是软件测试

学科如何形成？正反向两种思维、权威定义、三种新视角。

## 1.4 / 1.5 测试与质量保证、与开发的关系

SQA 与测试的分工；测试不是开发的“下一道工序”。

## 1.6 测试驱动开发的思想

TDD：测试在前、开发在后，一条流传几十年的思想。



## 互动与小结

随堂判断、思考练习、本章小结。

📌 学习目标：能讲出几个真实缺陷事故并归纳教训；能解释为什么要测试；能说出测试的定义与两种思维；能说清测试与 SQA、开发的差别；能说出 TDD 的基本思想。

| 项目      | 安排                                                     |
|---------|--------------------------------------------------------|
| 课程编码    | 154442008（专业选修）                                        |
| 学时 / 学分 | 32 学时 / 2 学分                                           |
| 周安排     | 第 1-17 周（含实验）                                          |
| 任课教师    | 周宇文   联系电话 / 微信：13580390715                            |
| 教材      | 朱少民《软件测试方法和技术（第 4 版）》，清华大学出版社                          |
| 参考资源    | 《全程软件测试（第 3 版）》（人邮 2019）；《致命 Bug——软件缺陷的灾难与启示》（人邮 2016） |

这门课怎么学？看三个词：分析（测试分析是基本功）、批判（像探索式测试那样质疑）、工程（在多种方案里选出最优并执行）。

## 它是谁

《软件测试方法和技术》2005 年初版，现为第 4 版；国家级规划教材，被数百所高校采用。

## 怎么组织

全书三篇：原理与方法（第 1-4 章）、测试技术（第 5-9 章）、项目实践（第 10-14 章）。

| 篇     | 回答的问题                      |
|-------|----------------------------|
| 原理与方法 | 为什么测、测什么、按什么流程与规范测         |
| 测试技术  | 单元 / 集成、系统功能、专项、本地化、自动化怎么做 |
| 项目实践  | 需求分析与计划、用例设计、基础设施、执行与报告    |

本课程按这本教材的骨架推进：先想清楚，再动手测，最后完整做完一个项目。

请判断下面哪些属于“软件缺陷及其后果”：

- A . 迪士尼儿童游戏《狮子王》只在少数系统上能装
- B . 某公司新版本刻意晚两周发布
- C . Intel CPU 除法运算结果出错
- D . 火星探测器着陆程序与推进器接口没测
- E . 公司把软件定价调高了 20%

(先自己判断，再看下一页参考答案)

# 破冰互动：这些问题算“软件缺陷”吗？ • 参 考答案

破冰互动

- ✓ 是 A . 迪士尼儿童游戏《狮子王》只在少数系统上能装
- ✗ 不是 B . 某公司新版本刻意晚两周发布
- ✓ 是 C . Intel CPU 除法运算结果出错
- ✓ 是 D . 火星探测器着陆程序与推进器接口没测
- ✗ 不是 E . 公司把软件定价调高了 20%

💡 小结：A、C、D 是软件缺陷；判断线索——功能 / 兼容精度集成安全 / 性能 出了问题，且在发布前没能被发现。

## 1.1 软件测试的必要性：先猜一猜

一个软件缺陷，最多能让一家公司损失多少？

- A . 几千元
- B . 几十万元
- C . 几千万元
- D . 数亿美元

(先自己判断，再看下一页参考答案)

# 1.1 软件测试的必要性：先猜一猜 · 参考答案

1.1 必要性 · 引子

✗ 不是 A. 几千元

✗ 不是 B. 几十万元

✗ 不是 C. 几千万元

✓ 是 D. 数亿美元

## 案例①：安装不上、算不对

案例①：兼容与精度

### 1994 | 《狮子王》光盘游戏

只在少数系统能正常安装运行，家长电话打爆，紧急补丁、退货赔偿——兼容性问题。

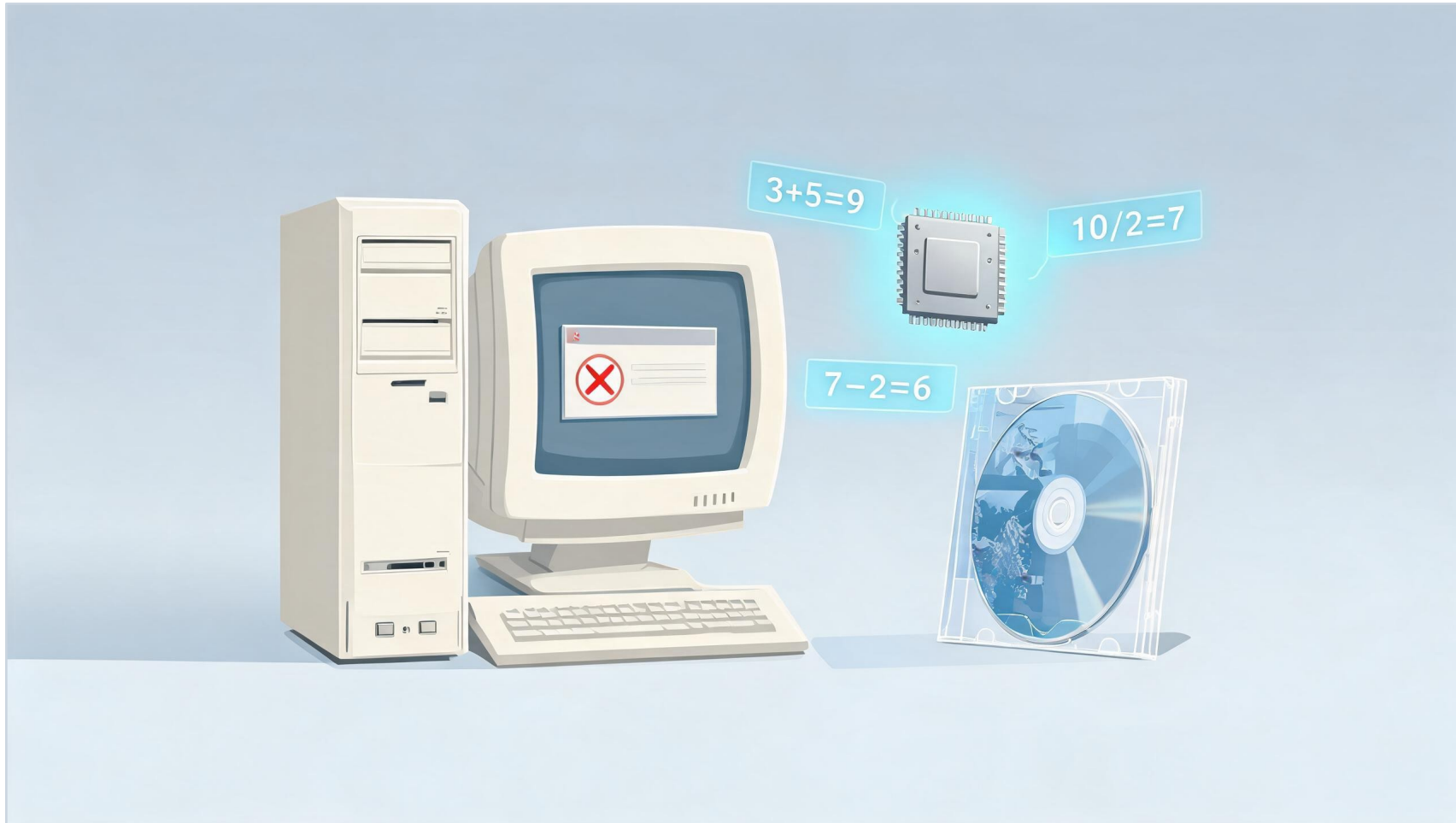
### 1994 | Intel 奔腾 CPU

除法运算在特定数值出错，最终召回、更换，损失约 4 亿美元——精度 / 计算问题。

共同点：缺陷真实存在，但发布前没有被测试拦住。

## 案例①：安装不上、算不对

案例①：兼容与精度



## 案例②：航天，输在“没合在一起测”

案例②：没做集成测试

### 1999 | 火星探测器

两个小组各自工作都没问题，但着陆程序与推进器的接口没有一起测：着陆时推进器被错误关闭，飞船下坠约 1800 米坠毁。

### 2022 | 波音星际客机

把 25 小时的完整飞行测试拆成小单元分别测，却没做端到端整体测试，软件缺陷直到试飞才暴露。

教训：单测都过了，不代表整体没问题——集成测试不足是最典型的翻车方式。

## 案例②：航天，输在“没合在一起测”

案例②：没做集成测试



## 案例③：45 分钟与四个小时

案例③：容错与稳定性

### 2012 | 骑士资本

一条错误指令让服务器 45 分钟执行 400 万笔交易，公司损失约 4.6 亿美元、濒临破产——功能容错问题。

### 2017 | 亚马逊云 S3

一次误操作导致云服务宕机整整 4 小时，大量网站瘫痪——性能与稳定性问题。

缺陷不只在“功能对不对”，还在扛不扛得住、错不乱来、故障能恢复。

## 案例④：有些缺陷，人命关天

案例④：安全与人的生命

### Therac-25 放疗仪

软件缺陷导致剂量超标的放射治疗，4 名病人死亡。

### 爱国者导弹

时钟累计运行超 14 小时后跟踪失准，一次拦截失败造成 28 名士兵伤亡。

### Uber 隐私泄露

App 安全漏洞泄露乘客隐私，一名用户索赔约 3.3 亿元人民币。

安全、隐私、可靠性——越重要的场景，越需要测试兜底。

## 案例④：有些缺陷，人命关天

案例④：安全与人的生命



# 看图猜事故①：有人把时间调成了 1970-01-01

奇葩① · 日期会咬人

按这个设置重启设备，下一秒会发生什么？

- A . 只是闹钟错了，明天照常响
- B . 设备当场“变砖”，无法开机
- C . 自动联网校时
- D . 什么也不会发生

(先自己判断，再看下一页参考答案)

# 看图猜事故①：有人把时间调成了 1970-01-01

- 参考答案

奇葩① · 日期会咬人

✗ 不是 A. 只是闹钟错了，明天照常响

✓ 是 B. 设备当场“变砖”，无法开机

✗ 不是 C. 自动联网校时

✗ 不是 D. 什么也不会发生



答案：设备变砖。1970-01-01 让时间戳变负值，启动逻辑没处理——一个日期边界，比病毒还灵。

# 看图猜事故①：有人把时间调成了 1970-01-01

奇葩① · 日期会咬人



图：iOS “日期与时间” 设置停在 1970-01-01（品玩，2016）

## 看图猜事故②：台上“蓝屏”的名场面

奇葩② · 发布会现场

猜一猜：这是谁、在哪年的发布会上现场翻车？

- A . 1995 年，Windows 95 发布会
- B . 1998 年，Windows 98 发布会（盖茨）
- C . 2001 年，Xbox 发布会
- D . 2009 年，Windows 7 发布会

(先自己判断，再看下一页参考答案)

## 看图猜事故②：台上“蓝屏”的名场面 · 参考 答案

奇葩② · 发布会现场

✗ 不是 A . 1995 年，Windows 95 发布会

✓ 是 B . 1998 年，Windows 98 发布会（盖茨）

✗ 不是 C . 2001 年，Xbox 发布会

✗ 不是 D . 2009 年，Windows 7 发布会



答案：1998 年 Windows 98 发布会（盖茨）。演示即插即用遇蓝屏，他说：这就是为什么 Windows 98 还没正式发售。

## 看图猜事故②：台上“蓝屏”的名场面

奇葩② · 发布会现场



图：Windows 98 发布会现场（1998，芝加哥 COMDEX）

## 看图猜事故③：这位“少女机器人”只活了一天

奇葩③ · 机器人被教坏

认出它是谁？它上线后多久就被“教坏”下线？

- A . 苹果 Siri ，一天
- B . 微软 Tay ，不到一天
- C . 亚马逊 Alexa ，一周
- D . 小米小爱，一个月

(先自己判断，再看下一页参考答案)

# 看图猜事故③：这位“少女机器人”只活了一天

- 参考答案

奇葩③ · 机器人被教坏

✗ 不是 A. 苹果 Siri，一天

✓ 是 B. 微软 Tay，不到一天

✗ 不是 C. 亚马逊 Alexa，一周

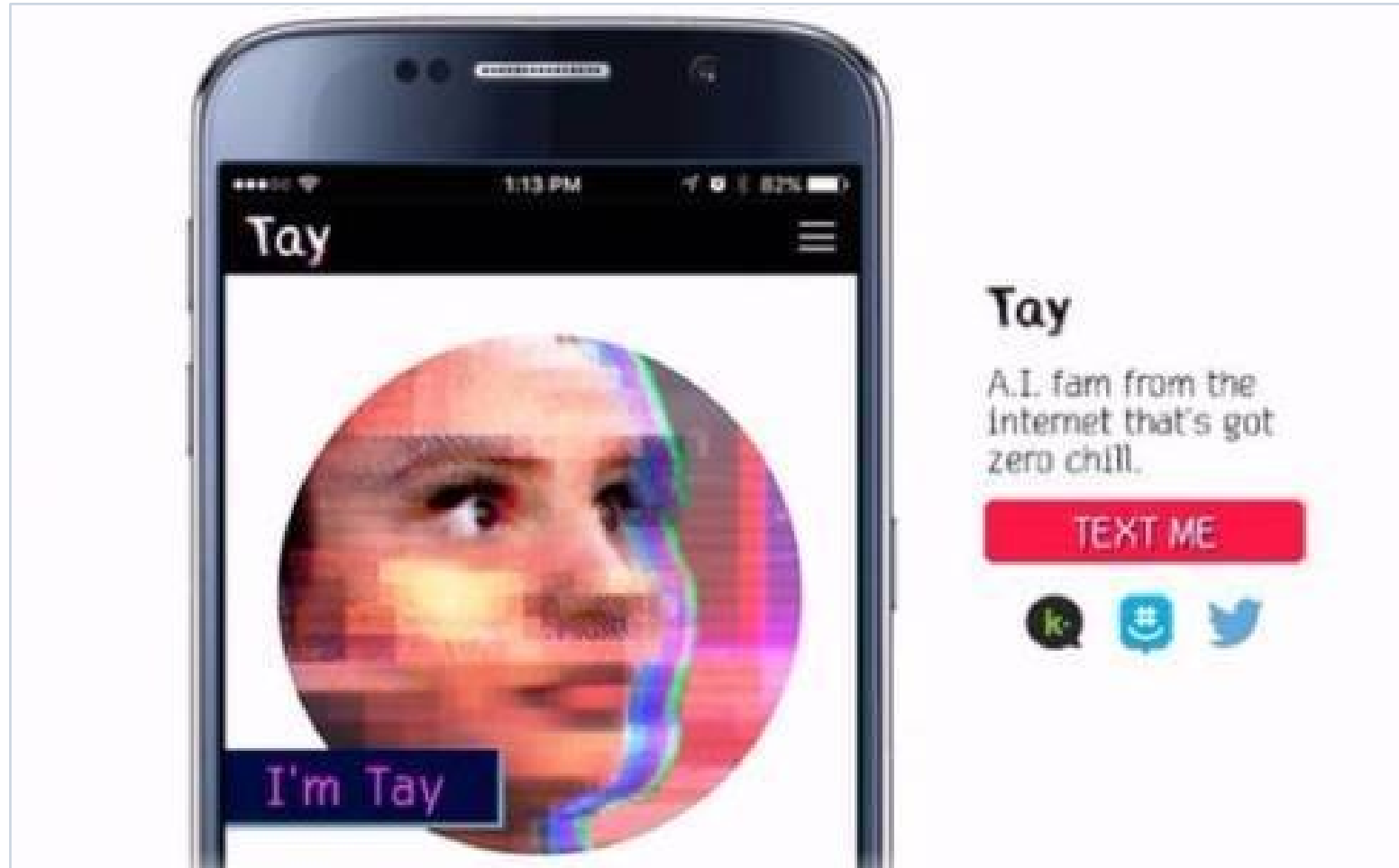
✗ 不是 D. 小米小爱，一个月



答案：微软 Tay（2016），上线不到一天就被网友教坏、被迫下线——内容与数据质量也需要测试。

## 看图猜事故③：这位“少女机器人”只活了一天

奇葩③ · 机器人被教坏



图：微软 Tay 官方形象（2016）

## 看图猜事故④：车是怎么到湖里去的？

奇葩④ · 导航把人带沟里

这台车经历了什么？

- A . 被水淹后的救援打捞现场，起因是跟着导航开上了冰湖
- B . 驾驶员故意开进湖里拍摄
- C . 湖面停车场
- D . 电影拍摄现场

(先自己判断，再看下一页参考答案)

## 看图猜事故④：车是怎么到湖里去的？ 答案

### • 参考

奇葩④ · 导航把人带沟里

✓ 是 A . 被水淹后的救援打捞现场，起因是跟着导航开上了冰湖

✗ 不是 B . 驾驶员故意开进湖里拍摄

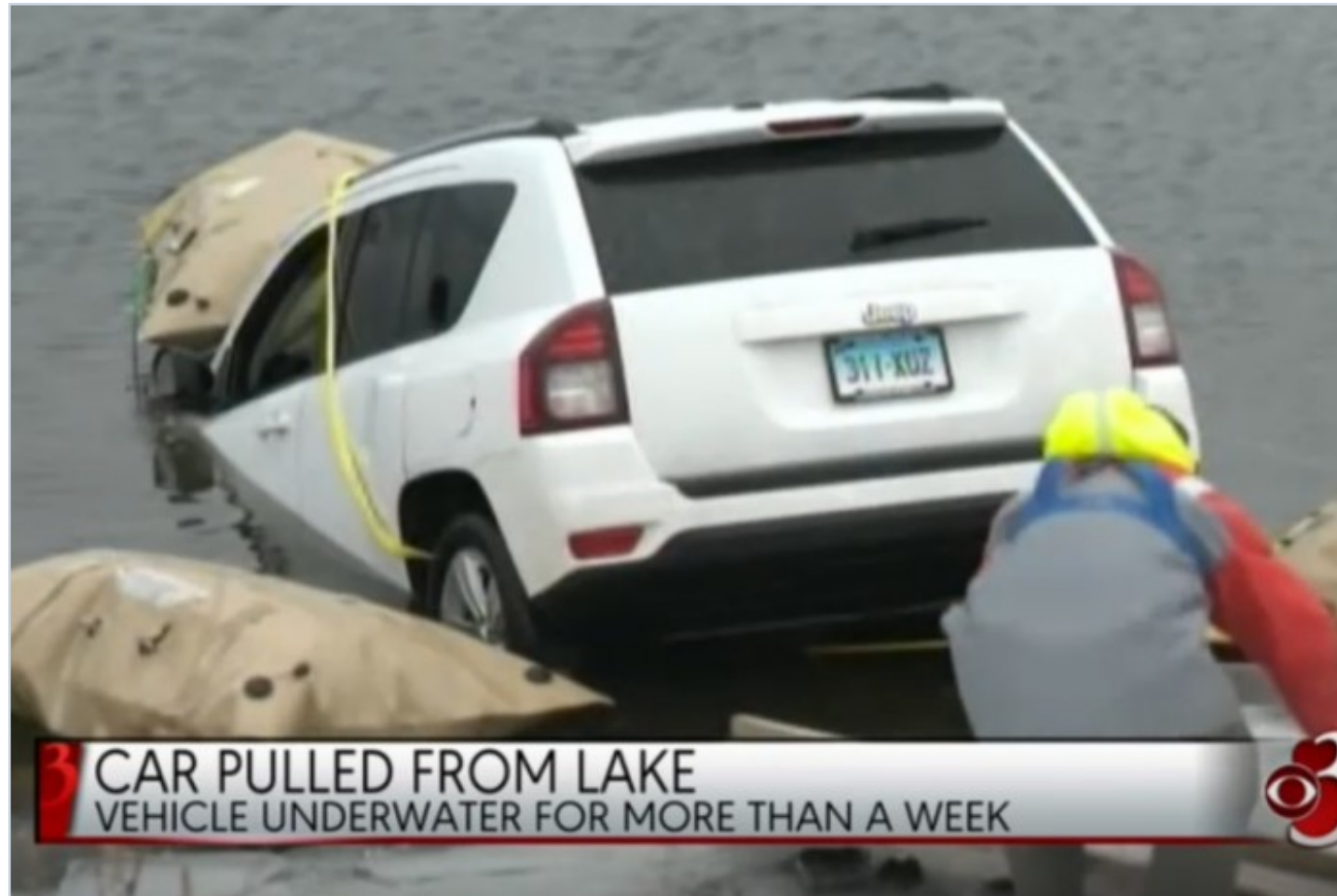
✗ 不是 C . 湖面停车场

✗ 不是 D . 电影拍摄现场

💡 答案：司机一路听导航（Waze）把车开上冰湖，冰裂坠水（2018，美国佛蒙特）。地图数据错了，越听话越离谱——数据也要被测试。

## 看图猜事故④：车是怎么到湖里去的？

奇葩④ · 导航把人带沟里



图：2018 美国佛蒙特，跟随 Waze 导航把车开上冰湖坠水（新浪科技）

## 它是什么

1997 年尚洋电子推出的国产即时战略游戏（RTS，PC）。在当时国产单机刚起步的年代，它被寄予厚望：宣传中能对标国外大作，预售与期待都很高。

## 著名的广告语

“你看见的一定是真的吗？”——这句宣传语后来成了反讽：玩家拿到手的成品，和宣传里的“大作”几乎不是一回事。

先别急着嘲笑 1997 年的“翻车现场”——想一想：宣传里那个“完美游戏”，在上市前有没有被真正测试验证过？

# 中国案例：血狮（Blood Lion，1997）

国产案例 · 背景



- 大量用户安装不了、运行不了，或频繁出错退出；
- 不同电脑上兼容性差异很大，表现极不稳定；
- 玩法、AI、操作等功能问题明显，离宣传演示差距悬殊；
- 媒体与玩家批评尖锐，退货与投诉不断。

《血狮》后来成为国产游戏史上被反复提起的“滑铁卢”式案例——它不是输在想法，而是输在交付前没有用测试把质量问题拦下来。

# 发售那天：宣传与现实的巨大落差

国产案例 · 落差



血狮游戏画面

## 互动：血狮的翻车，暴露了哪些测试缺位？

(可多选)

- A . 不同电脑配置下的兼容性
- B . 长时间运行的稳定性与崩溃
- C . 功能与 AI 是否符合宣传预期
- D . 把“宣传”当成“产品已好”的证据

(先自己判断，再看下一页参考答案)

# 互动：血狮的翻车，暴露了哪些测试缺位？

## 参考答案

- ✓ 是 A . 不同电脑配置下的兼容性
- ✓ 是 B . 长时间运行的稳定性与崩溃
- ✓ 是 C . 功能与 AI 是否符合宣传预期
- ✓ 是 D . 把“宣传”当成“产品已好”的证据

💡 小结：A-D 都缺——兼容 / 稳定 / 功能验收都属测试范畴；血狮最该被记住的一句话：宣传能制造期待，只有测试能验证承诺。

第一个结论很朴素：软件测试是发布前的“质检关卡”——发现缺陷，才有机会清除缺陷。

把本章出现的事故放在一起看——它们最像下面哪句话？

- A．都是被黑客攻击
- B．缺陷本来就存在，发布前没有被测试及时拦住
- C．这类事故无法预防
- D．都是硬件老化

(先自己判断，再看下一页参考答案)

# 案例总回顾：它们到底缺了什么测试？

## 答案

- 参考

- ☒ 不是 A. 都是被黑客攻击
- ☒ 是 B. 缺陷本来就存在，发布前没有被测试及时拦住
- ☒ 不是 C. 这类事故无法预防
- ☒ 不是 D. 都是硬件老化

# 把这些事故装进一张“缺陷档案表”

缺陷归档

| 缺陷类别    | 代表事故             | 该由什么“测试”把关   |
|---------|------------------|--------------|
| 兼容性     | 《狮子王》游戏装不上       | 多配置兼容测试      |
| 计算 / 精度 | Intel 奔腾除法       | 数值与边界测试      |
| 集成      | 火星探测器、波音星际客机     | 接口 / 端到端集成测试 |
| 单位 / 约定 | 火星气候轨道器          | 接口规格与数据校验    |
| 日期 / 时间 | 千年虫、iPhone 1970  | 时间边界测试       |
| 容错 / 稳定 | 骑士资本、AWS S3      | 故障注入与压力测试    |
| 安全 / 隐私 | Therac-25 、 Uber | 安全与合规测试      |
| 内容 / 数据 | Tay 聊天机器人        | 数据与内容质量测试    |

先会“给缺陷分类”，才知道该测什么——这是第 2 章的地基。

## 1.2 为什么要进行软件测试?

答案很简单：为了保证软件质量。

推理链条：

### 质量信息源

测试提供关于产品客观、可用的质量信息，供决策与改进。

### 质量关键步骤

软件测试是软件质量保证链条中最关键的一环之一。

### 行业实践的结论

测试的重要性来自软件行业几十年的实践验证。

## ① 测试就是“跑一遍看看”

✗ 不完整：测试包含静态测试（评审、走查、静态分析）与动态运行。

## ② 测试就是“证明程序没错”

✗ 这是正向思维的一面；测试更要发现缺陷，好的用例能发现尚未发现的错误。

## ③ 测试只是开发之后的质检

✗ 测试从需求阶段就该参与——越早发现，成本越低。

## ④ 测试是质量保证的全部

✗ SQA 管流程、测试管产品，测试是质量保证的重要手段之一。

下面四种说法，你同意哪些？先判断，再揭晓。

测试的完整含义：在指定条件下运行（或评审）系统，观察记录结果、对照预期做出评价；既验证符合要求，也发现与预防缺陷——它是贯穿研发全程的活动。

## 1.3.1 从“程序测试”到“软件测试”

### 软件的组成

软件不只是可执行的程序：需求、设计、代码都属于软件的组成部分。

### 测试范围的扩展

从只测“程序”，扩展到测试“需求、设计与文档”——对它们的评审也属于测试（静态测试）。

所以：软件测试  $\neq$  程序测试，它是覆盖软件全部成果物的活动。

## 静态测试（不运行程序）

对需求、设计、代码、文档做评审、走查、静态分析——在程序跑起来之前就把问题挑出来。

## 动态测试（运行程序）

在指定条件下运行系统或组件，观察并记录结果，对照预期做评价。

| 对比   | 静态测试          | 动态测试           |
|------|---------------|----------------|
| 是否运行 | 否             | 是              |
| 典型手段 | 评审、走查、静态分析    | 执行用例、观察结果      |
| 擅长发现 | 歧义、遗漏、规范与结构问题 | 功能、性能、接口与运行期缺陷 |

软件测试 = 静态测试 + 动态测试；只会“跑一遍”不算完整的测试。

| 时期        | 阶段      | 主导思想                          |
|-----------|---------|-------------------------------|
| 1957-1978 | 功能验证导向  | 测试是证明软件“正确”（正向思维起步）           |
| 1978-1983 | 破坏性检测导向 | 测试是为了找出错误（逆向思维形成）             |
| 1983-1988 | 质量评估导向  | 测试提供产品评估与质量度量（IEEE 829 等标准出现） |
| 1988-     | 缺陷预防导向  | 既验证符合要求，又发现与预防缺陷              |

| 阶段                | 特征                              |
|-------------------|---------------------------------|
| 初级阶段（ 1957–1971 ） | 把测试当作对产品的事后检验，缺少有效方法。           |
| 发展阶段（ 1972–1982 ） | 第一次专门的软件测试会议召开，测试方法开始体系化。       |
| 成熟阶段（ 1983 至今）    | 标准（如 IEEE 829 ）出现，测试成为独立的学科与职业。 |

把“测试思想”的时间线（功能验证→破坏性检测→质量评估→缺陷预防）和这里的“学科阶段”对照着看，就明白测试是怎么长成一门专业的。

## 同一个缺陷，不同阶段发现，代价完全不同

需求阶段改一句话，可能只是改文档；到了设计要连带改结构；到了编码要改代码与单元测试；到了测试要重跑回归；发布之后还要面对版本发布、用户影响与声誉损失。

| 发现阶段    | 需要连带改动        |
|---------|---------------|
| 需求 / 设计 | 文档与方案         |
| 编码      | 代码、单元测试       |
| 系统测试    | 代码+回归测试+可能的返工 |
| 发布之后    | 版本、补丁、用户与品牌成本 |

成本随阶段非线性增长——这就是“测试左移”的经济学理由。

## 缺陷是源头

软件中客观存在缺陷，它是损失的来源，也是测试要找的东西。

## 测试是手段

通过设计与执行用例，发现缺陷、评估质量，给出质量信息。

## 质量是目标

缺陷被清除、风险被揭示，产品才谈得上“符合要求、满足用户”。

缺陷 → 测试 → 质量：测试用例是连接“缺陷”与“质量”的载体——它既描述期望，也记录发现。

# 1.3.2 两种思维，两种测试观

## 正向思维（Hetzel）

测试是为“程序能按预期运行建立信心”的过程；测试用于验证软件是否符合需求、能正常工作。

## 反向思维（Myers）

测试是为了证明程序有错而执行的过程：好的用例＝能发现至今未发现的错误。

| 维度 | 正向思维              | 反向思维             |
|----|-------------------|------------------|
| 目标 | 验证软件正常工作          | 假定软件有错误，找出薄弱环节   |
| 做法 | 在规定环境下运行功能，直至全部通过 | 试图“破坏”系统，直至找不出问题 |

认知决定行为：你信“它在多数时候是好的”还是“它藏着错”，会直接决定你会测什么、怎么测。

## 1.3.3 软件测试的权威定义

定义 IEEE

在特定的条件下运行系统或组件，观察或记录结果，并对系统或组件的某个方面做出评价。（IEEE/ISO 29119 定义）

再补一层“比较”视角：测试是把被测对象实际做了什么，与它应当做什么做比较，发现现存与要求之间的差别（错误），并评价其特性。

### ① 指定条件

测试不是随便跑跑，要有明确的输入、环境与前置。

### ② 观察记录

结果要看得见、记得下，才能判断对错。

### ③ 评价判断

对照预期给出结论——这是测试的产出。

## Test Oracle：凭什么说“结果不对”？

判断输出是否正确，需要预期结果或判定准则；准则不清，测试就无法下结论。

## 批判性思维：测试是一种探索

借助观察、经验、反思、推理与沟通收集信息，分析质量并得出结论——它是一个不断探索的过程。

这两点决定了测试的深度：没有判定准则，测不出结论；没有质疑精神，测不出真问题。

## 质量画像

全面评估产品，拿到关于质量客观、全面的信息。

## 问题督办

发现问题、督促解决，质量由此被“逼”上去。

## 风险雷达

持续给出质量反馈、揭示质量风险，支撑项目决策。

## 缺陷模式沉淀

分析历史缺陷，提炼缺陷模式，用于预防下一批缺陷。

换个视角：质量视角＝用测试给质量“画像”；风险视角＝把测试当作对潜在风险的持续评估；经济视角＝用最少的代价换最高的质量，测试成本应当远小于缺陷损失。

## 1.4 测试与质量保证（SQA）的关系

软件质量保证（SQA）是通过有计划地评审与审计，验证软件产品是否合乎标准、满足要求，并协调、审查、跟踪获取信息的系统工程。

SQA 的典型活动：技术评审、软件测试标准的执行、修改（变更）控制、度量、质量记录。

| 比较 | 软件测试                              | SQA      |
|----|-----------------------------------|----------|
| 性质 | 技术性工作                             | 管理工作     |
| 侧重 | 对产品评估与验证                          | 对流程评审与监控 |
| 关系 | 测试为 SQA 提供数据与依据；SQA 指导、监督测试的计划与执行 |          |

## 1.5 测试与开发：不是“我做完你再来”

正确的打开方式是并行、协作：

### 需求阶段

测试就开始分析需求、识别可测性，准备测试计划。

### 设计阶段

测试同步设计用例，评审接口与边界。

### 开发阶段

持续测试、每日构建与冒烟测试，问题当天发现。

### 发布前后

回归、专项与验收测试把关，交付后仍监控线上质量。

## 1.5 测试与开发：不是“我做完你再来”（续）

一句话：测试要“左移”，越早参与越好——这也是后面 TDD 思想的地基。

## 1.6 测试驱动开发（TDD）的思想

1.6 TDD

TDD（Test Driven Development）：测试在前，开发在后——先写会失败的测试，再写代码让它通过。

### 由来

最早来自极限编程（XP）实践。

### 演化

TDD 成为思想后，落地为 UTDD（单元测试驱动）与 ATDD（验收测试驱动）。

### 对开发者的意义

测试不再是“查我”的关卡，而是描述期望、驱动设计的工具。

## 互动：开发人员测自己的产品，难在哪？

思考互动

先想一想（可多选），再点开看：

- A．容易顺着自己的思路测，测不出“没想到”的错
- B．心态上想证明“我写的是对的”，而不是找错
- C．开发者比测试员更不专业

(先自己判断，再看下一页参考答案)

# 互动：开发人员测自己的产品，难在哪？

## 参考答案

思考互动

- ✓ 是 A . 容易顺着自己的思路测，测不出“没想到”的错
- ✓ 是 B . 心态上想证明“我写的是对的”，而不是找错
- ✗ 不是 C . 开发者比测试员更不专业

 小结：自测的障碍是盲区+心态。这也是 TDD 的出发点：先把期望写成测试，再让代码去满足它。

## 质量是团队共同责任

敏捷不再把质量“外包”给测试岗位，而是整个团队对质量负责。

## 开发与测试融合

角色边界变模糊：开发做更多测试，测试更早参与设计与需求。

## 持续交付要求持续测试

高频发布下，测试环节最容易成为瓶颈，自动化与持续集成因此重要。

## 质量内建

与其在末尾“检验质量”，不如在过程中“把质量做进去”。

软件测试岗位的典型能力要求：

## 基本理论与方法

掌握测试基本概念、理论与方法、技术与工具，熟悉测试流程与实践。

## 动手与工具

能设计测试用例，能使用自动化工具完成完整的项目测试。

## 分析与解决问题

基于项目背景与用户需求，完成测试任务的分析、设计与执行。

## 工程与创新

具备系统性、工程化认知与较强的动手能力、技术创新能力。

课程目标就一句话：让你能独立承担起一个项目的测试任务。

## 测试为什么必要

缺陷真实存在且代价巨大，测试是发布前的质检关卡。

## 什么是测试

特定条件下运行+观察记录+对照预期做评价；分静态 / 动态、正向 / 反向。

## 测试的价值

质量画像、问题督办、风险雷达、缺陷预防——越早越省。

## 与谁协作

SQA 管流程、测试管产品；开发与测试并行协作、测试左移。

## TDD 思想

测试在前、开发在后：红 → 绿 → 重构。

**值得想一想：随手打开一款 App，你能主动找出它的一个缺陷吗？**

- 回想你日常使用的软件，遇到过哪些质量问题？分别属于哪类缺陷？
- 正向思维与反向思维会怎样影响一个人“测什么、怎么测”？
- 测试与开发为什么是并行协作的关系？举一个你身边的例子。
- 软件测试与质量保证（SQA）的联系与区别是什么？

## 教材

《软件测试方法和技术（第 4 版）》 · 朱少民 · 清华大学出版社

## 延伸 1

《全程软件测试（第 3 版）》 · 人民邮电出版社

## 延伸 2

《致命 Bug——软件缺陷的灾难与启示》 · 人民邮电出版社



**参考资料下载（标准规范 / 行业报告 / 知识地图，共 10 份）→**