1.所有的测试都应追溯到用户**需求**

软件缺陷最多的地方是**软件需求规格说明书**（软件需求定义），而不是程序代码，规格说明书的错误占总错误的54%，设计方面占24%，代码占15%

做法：

测试的第一任务是需求分析

测试需求分析要做好

时刻提醒自己要考虑用户需求

制造缺陷的罪魁祸首不是程序员

做好需求评审

审查所作内容是否符合用户的需求

测试应该是与软件开发或维护并行进行的一个过程

3.早做测试计划

软件测试不仅是测试执行

应该在测试工作真正开始前较长时间就进行测试计划

4.穷尽测试不可能&软件测试有风险

测试的输入量太大，时间不够

做法：

如果决定不去测所有情况，那么我们就面临很大的风险

如果时间不够，建议使用风险分析，确定测试的重点和优先级，控制测试的开销

风险分析需要判断技能， 常识，和感觉（类似于开车，什么情况容易加塞）

5.测试工作的Good-enough原则

既不要做过多测试，也不做不充分测试

解决方法：通过需求分析和风险分析（时间费用资源），找到测试重点。指定最低测试通过标准和测试内容，然后具体问题具体分析（这个需要熟能生巧）

6.Pareto法则应用于软件测试

pareto法则由意大利经济学家pareto提出，又称作28效率法则

一般情况下80%的缺陷聚集在20%的关键核心业务模块里面

在分析设计实现阶段的复审和测试工作能够发现和避免80%的缺陷，而系统测试能找到剩余缺陷的80%，最后4%的缺陷可能只有用户在大范围，长时间使用之后才会暴露出来

使用方法：

做好测试需求分析，分清测试计划，

尽早测试

持续测试

7 尽可能的分阶段测试

单元测试->集成测试-》系统测试-》验收测试

8 为了达到最佳的测试结果，应该由独立的第三方来构建测试

不能让写功能代码的人进行测试

因为：

程序员从来不承认自己的程序有问题

程序员测试自己的编码是一件很糟糕的事情，但是让他们测试别人的代码却又是最好的测试人员

编码人员的思维有局限性，测试人员就是来发现这些局限性的错误

多数程序员没有严格的职业训练

程序员没有良好的BUG追踪和回归测试的习惯

9. 测试旨在发现存在的缺陷

软件测试可以报告软件缺陷存在，却不能报告软件缺陷不存在

即使在测试过程中未能发现软件失效，也不能保证被测软件不存在问题

10 为了保证测试的有效性和高效性，测试必须是破坏性，系统化的

充分，有效，系统的测试可以减少软件中未被发现缺陷的可能性

测试既要验证软件的正确性，更要通过破坏软件，发现软件的不确定性（超数域参数）

11.缺陷有群集效应，找到的缺陷越多，说明软件隐含的缺陷越多

应该在发现缺陷的地方继续寻找

12 杀虫剂怪事（缺陷的线性收敛）

用于描述软件测试越多，其对测试的免疫力越强的现象

程序员对于测试人员的“惯用伎俩”可以做到主动规避。

该表自己的测试技术，用新的测试方法，测试技术发现问题

13 并非所有的软件缺陷都要修复

没有足够的时间

不算真正的软件缺陷

修复的风险太大

不值得修复

14,使用木桶原理

木桶原理在软件产品生产方面就是全面质量管理的概念

团队合作

15.前进两步，后退一步

缺陷修复以后，可能以20%到50%引入新的缺陷

每次修复之后必须要进行确认测试和回归测试

16 软件测试是一个迭代的过程

无论项目采用什么开发模式，测试人员总是一个版本接着一个版本地测试，其测试活动总是迭代向前的

17 测试需要遵循标准

国际标准，如ISO，CMM，IEEE

国家标准： GB，GB/T

行业标准：移动 1+0.1 = 2 打电话超了一点也是一分钟

公司标准：

用户规定

思路决定测试，思路要有条理性

具体问题具体分析，测试要灵活

无责任心不成测试

测试不能靠猜测，需求应该根据代码逻辑来设计测试用例，以实际需求为依据