软件测试方法和技术

培训人：袁媛

培训日期：2021-07-28

课程目标

- 什么是软件测试？
- 软件测试和开发模型二者有什么关联？
- 软件测试的方法有哪些？

什么是软件测试？

Bug：
(1) any small insect；小昆虫；虫子；
(2) an infectious illness that is usually a fairly mild；轻微的传感病；小病；
(3) an enthusiastic interest in sth such as a sport or hobby; 热衷；着迷； the travel bug；
(4) A fault in a machine, especially in a computer system or program；

为什么Bug会用来表示软件缺陷？

1.1软件缺陷/案例

- 1963年美国飞往火星的火箭爆炸，原因是Fortran程序 DO 5 I = 1,3 误写为 DO 5 I = 1.3，损失10000万美元；
- 1967年苏联“联盟一号”宇宙飞船返回时因忽略一个小数点，在进入大气层时，打不开降落伞烧毁；
- 1994年美国迪斯尼公司的狮子王游戏软件bug；
- 1994年12月30日，Tomas博士发现Intel Pentium处理器浮点除法缺陷；
- 2000年，跨世纪“千年虫”问题。

上述所有实例中的软件问题在软件工程或软件测试中都被称为软件缺陷或软件故障。

1.1软件缺陷/定义

(1) 软件未达到产品说明书已经标明的功能;

(2) 软件出现了产品说明书中指明不会出现的错误;

(3) 软件未达到产品说明书中虽未指出但应当达到的目标;

(4) 软件超出了产品说明书中指明的范围;

(5) 软件测试人员认为软件难以理解、不易使用，或者最终用户认为该软件使用效果不良。

1.2软件测试/定义

（1）狭义定义：程序测试是为了发现错误而执行程序的过程。

（2）广义定义：延伸到需求评审、设计审查活动中去。

（3）软件测试是使用人工或自动的手段来运行或测定某个软件系统的过程，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别。

1.2软件测试/目的和意义

（一）软件测试的目的

（1）发现软件缺陷；

（2）发现软件缺陷，尽可能早一些；

（3）发现软件缺陷，尽可能早一些，并确保其得以修复。

（二）软件测试的意义

（1）提高产品的质量；

（2）提高产品的商誉，获得更多的销售机会和市场份额；

（3）降低客户的售后支持成本以及产品维护成本。

测试无法说明错误不存在，只能说明软件错误已出现。

1.2软件测试/对象和特点

（一）软件测试的对象

（1）软件测试不等于程序测试；

（2）软件测试贯穿于软件定义和开发的整个过程；

（3）软件开发过程中所产生的需求规格说明、概要设计规格说明、详细设计规格说明以及源程序都是软件测试的对象。

（二）软件测试的特点

（1）软件测试开销大

按照Boehm的统计，软件测试的开销大约占总成本的30%-50%。微软为打造Windows2000，1700多个开发人员，以及3200个测试人员，开发和测试人员之比约为三比五。

（2）不能进行“穷举”测试

如果不能查看代码内在逻辑，可输入的测试用例是几乎无限的。

例如：程序P有两个整型输入量X和Y，输出量为Z，在32位机上运行。所有的测试数据组合有 2^32 * 2^32，1ms执行1次，共需多少年?

1.3软件测试/原则

（1）所有的测试都应追溯到**用户需求。**

（2）应尽早和不断地测试。

- 由于软件的复杂性和抽象性，在软件生命周期各个阶段都可能产生错误，所以不应把软件测试仅仅看作是软件开发的一个独立阶段，而应当把它贯穿到软件开发的各个阶段去。
- 在需求分析和设计阶段就应开始进行测试工作，编写相应的测试计划及测试设计文档，同时坚持在开发各阶段进行技术评审和验证，才能尽早发现和预防错误，杜绝某些缺陷和错误，提高软件质量，测试工作进行得越早，越有利于提高软件的质量，这是预防性测试的基本原则。

（3）不可能进行详尽的测试。

在有限的时间和资源下进行完全测试，找出软件所有的错误和缺陷是不可能的，软件测试不能无限进行下去，应适时终止。

1.3软件测试/原则

（4）测试仅能表明存在缺陷。

测试只谈论存在缺陷，而不谈论没有缺陷，即软件测试可以降低软件中未发现的缺陷的可能性，**但即使没有发现缺陷，也不是没有问题的证明。**

（5）充分关注缺陷集群现象。

80%的问题出现在20%的模块中，在测试的程序段中，若发现的错误数目多，则残存在其中的错误也多，应当花较多的时间和代价测试那些具有更多错误数目的程序模块。

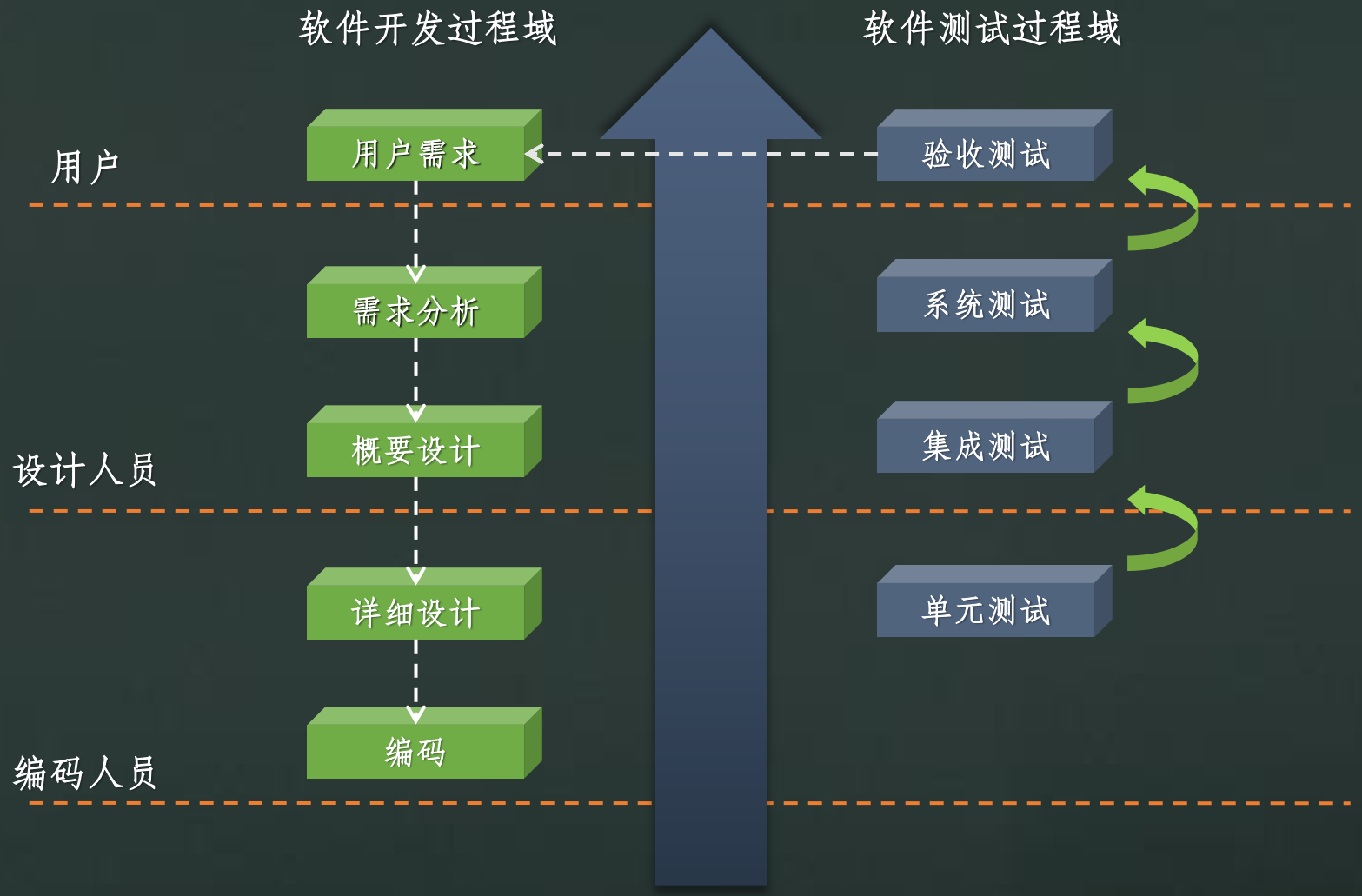
（6）避免既当球员又当裁判。

测试一般由独立的测试部门或第三方机构进行。

（7）尽量避免测试的随意性。

执行测试计划，保存测试用例、出错统计和最终分析报告，为维护工作提供充分的资料。

1.3软件测试/级别


软件开发过程域
软件测试过程域
用户
用户需求
验收测试
需求分析
系统测试
设计人员
概要设计
集成测试
详细设计
单元测试
编码
编码人员

1.3软件测试/级别/单元测试

(1) 基本含义

- 对象可以是模块、类、函数和对象等，由不同的软件语言来决定；
- 主要目的是验证单元是否满足了详细设计规格说明，发现需求和设计中的错误；
- 设计的主要输入是详细设计规格说明、软件设计和数据模型等；
- 主要采用白盒测试技术，黑盒测试技术作为单元测试的辅助；
- 应该覆盖功能需求和非功能需求；
- 使用测试驱动的方法（测试驱动开发）。

1.3软件测试/级别/集成测试

(1) 基本含义

- 集成测试，又叫组装测试、联合测试等；
- 集成测试是对组件之间的接口进行测试，以及和系统其他部分的相互作用；
- 最简单的形式是两个已经测试的单元组合成一个组件，来测试它们之间的接口和数据交换；
- 集成测试的主要工作：把单元测试通过的各个模块逐步集成在一起，来测试数据是否能够正确传递和调用，以及各个模块是否能正确的协同工作。

1.3软件测试/级别/集成测试

(2) 关注点

- 单元模块是否传输了错误的数据，或者没有传输数据；
- 接受数据的单元不能操作或者崩溃，比如单元功能缺陷、接口格式不兼容、协议不兼容等；
- 单元之间通讯正常，但是使用不同的方法来解析收到的数据，比如规格说明矛盾、理解错误等；
- 数据能正常传输，但是传输时间错误，比如时序问题，或者传输的时间间隔太短，比如吞吐量、负荷、容量等问题。

1.3软件测试/级别/系统测试

(1) 基本含义

- 系统测试是将已经集成好的软件系统，作为计算机系统的一部分，与计算机硬件、某些支持软件、数据和人员等系统元素结合起来，在实际运行环境下对计算机系统进行一系列严格有效的测试；

- 系统测试关注的是项目或产品范围中定义的整个系统或产品的行为；

- 在系统测试中，测试环境应该尽量和最终使用的目标或产品使用的环境相一致，从而减少和环境相关的失效。

1.3软件测试/级别/系统测试

(2) 测试目标

目标是确认整个系统是否满足了规格说明中的功能和非功能需求，以及满足的程度；

系统测试应该发现由于需求不正确、不完整或实现和需求不一致而引起的失效，并识别没有文档化或被忘记的需求；

常见的系统测试包括压力测试、容量测试、性能测试、安全测试、容错测试等。

1.3软件测试/级别/验收测试

（1）基本含义

验收测试通常是由使用系统的用户来进行，同时系统的其他利益相关者也可能参与其中；

验收测试目的是通过验收测试，对系统功能、系统特定部分或特定的系统非功能特征进行测试；

发现缺陷不是验收测试的主要目标，验收测试也可以用来评估系统是否可以在市场部署、用户使用系统的准备情况等。

1.3软件测试/级别/验收测试

（2）测试类型

用户验收测试

运行测试

合同验收测试

规范验收测试

Alpha和Beta测试

…

验收类型	执行者	测试目标	测试阶段
用户验收测试	关键用户	验证软件系统符合商业用户的商业需要	软件开发的任何阶段
运行测试	系统管理员	验证系统符合IT管理需要 • 系统备份/恢复测试 • 灾难恢复测试 • 用户管理测试 • 维护任务测试 • 安全漏洞阶段性检查	软件开发的任何阶段
合同和法规性验收测试	用户、系统管理员	验收软件符合合同中规定的验收准则	软件开发的任何阶段
α测试和β测试	潜在用户	在软件投入市场前，获取潜在用户关于软件的反馈信息	软件正式投入市场前

软件测试和开发模型二者有什么关联？

2.1开发模型和测试

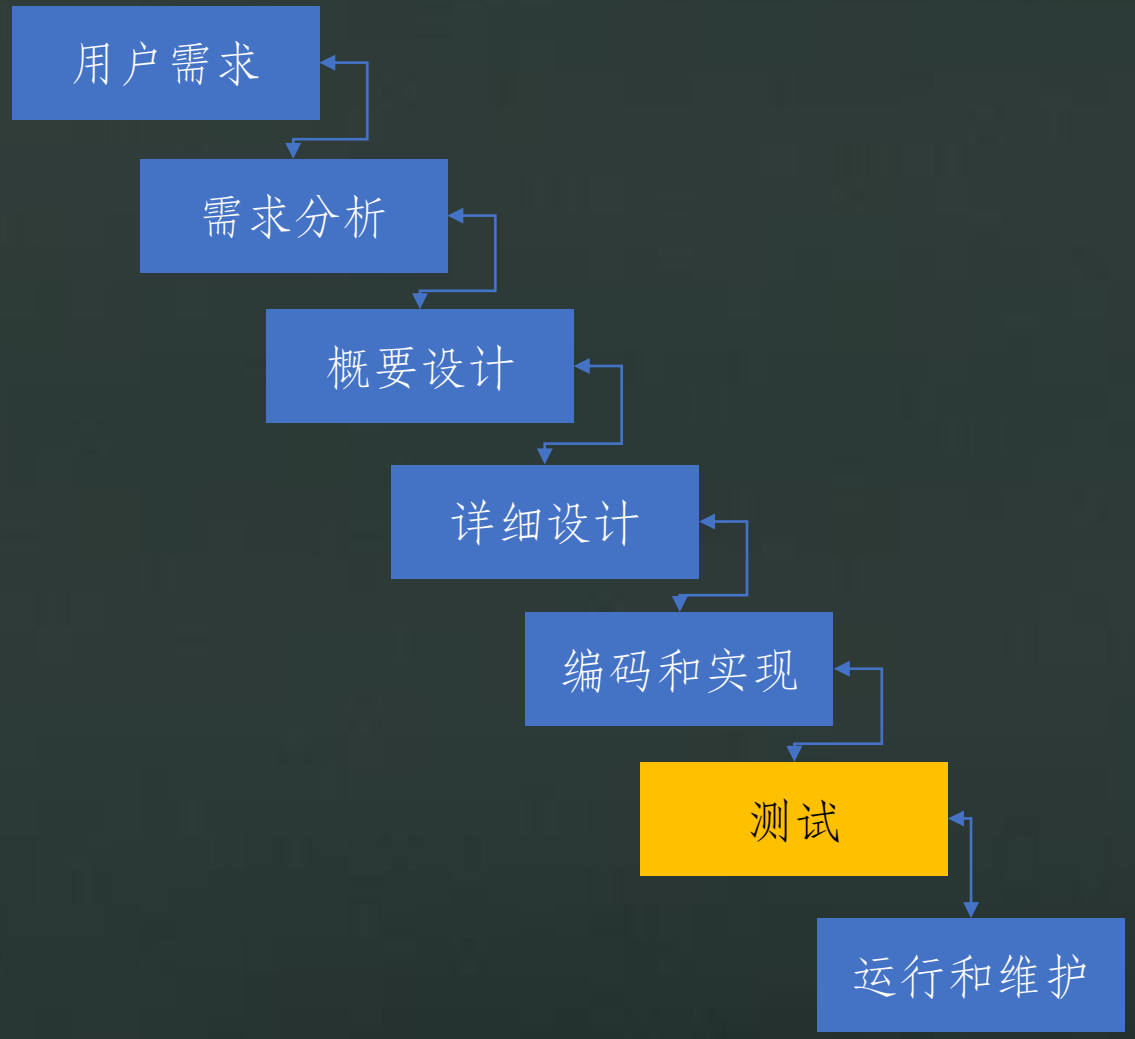
软件测试不是孤立存在的，测试活动与开发活动息息相关；

软件测试不仅仅包括测试执行，它应该贯穿于整个软件的生命周期之中；

不同的开发生命周期模型需要对应不同的测试阶段、测试活动和测试方法。

2.2开发模型/瀑布模型

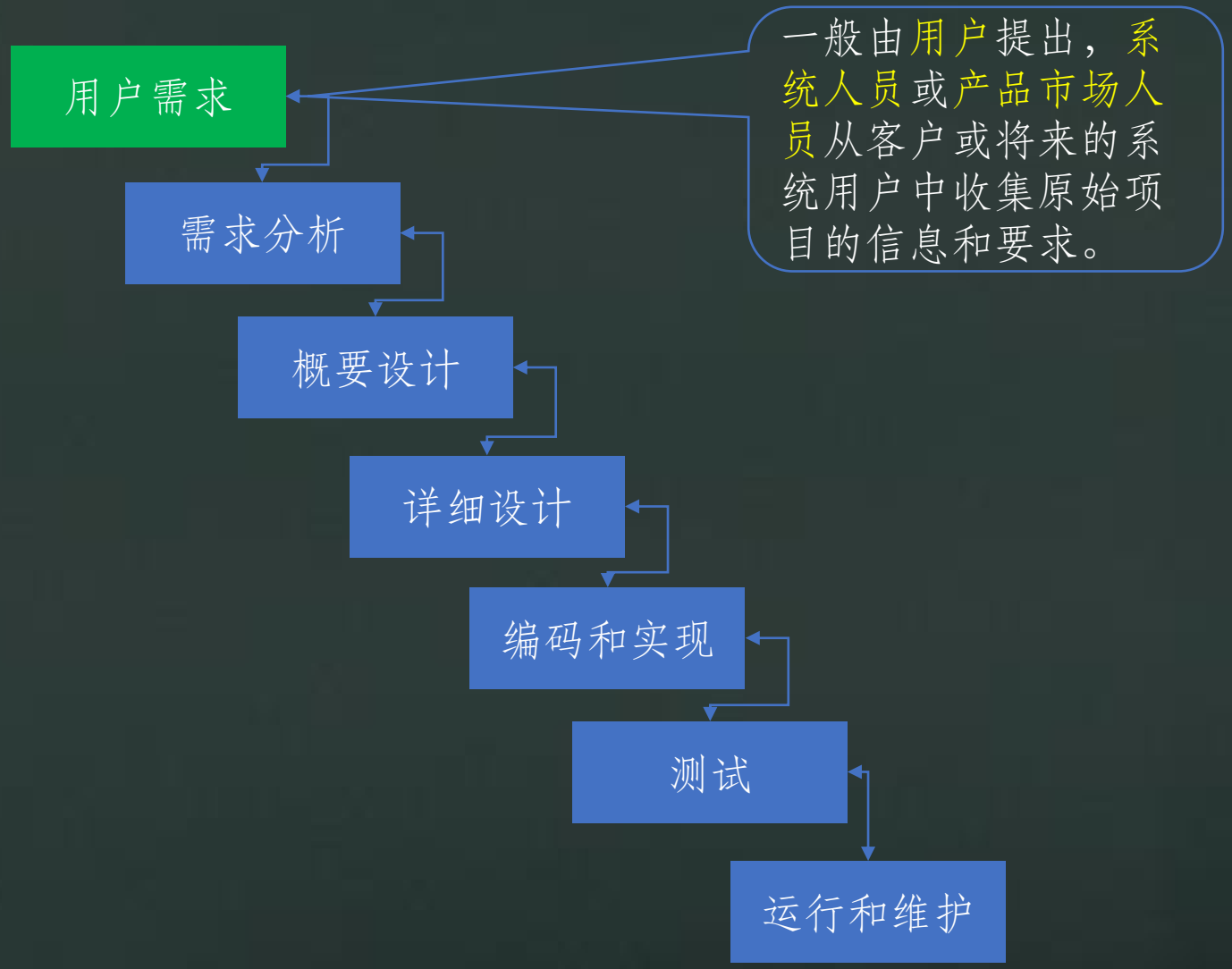
（1）瀑布模型的结构



软件测试作为整个软件生命周期中的一个阶段。

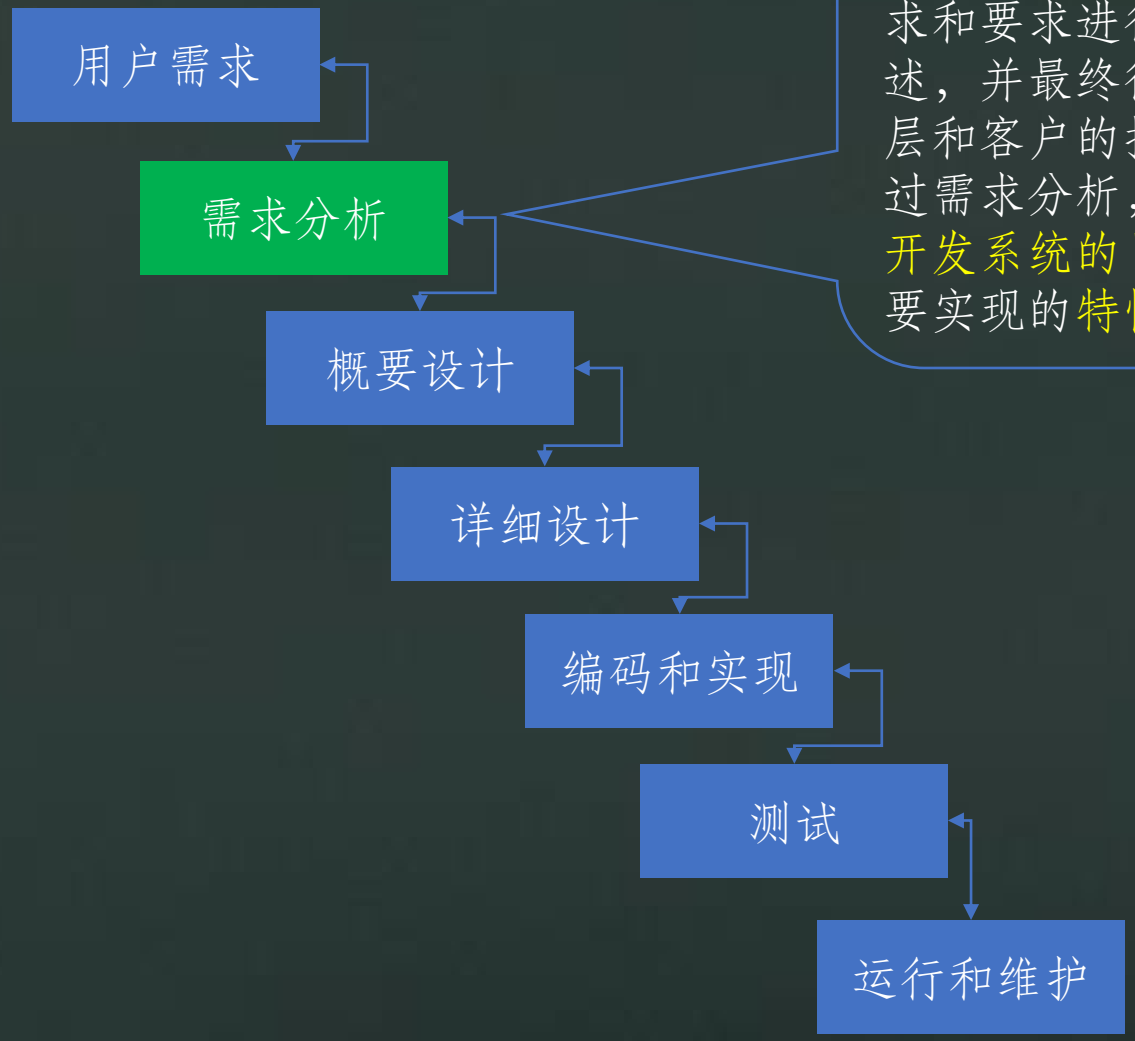
2.2开发模型/瀑布模型

（2）瀑布模型的阶段



2.2开发模型/瀑布模型

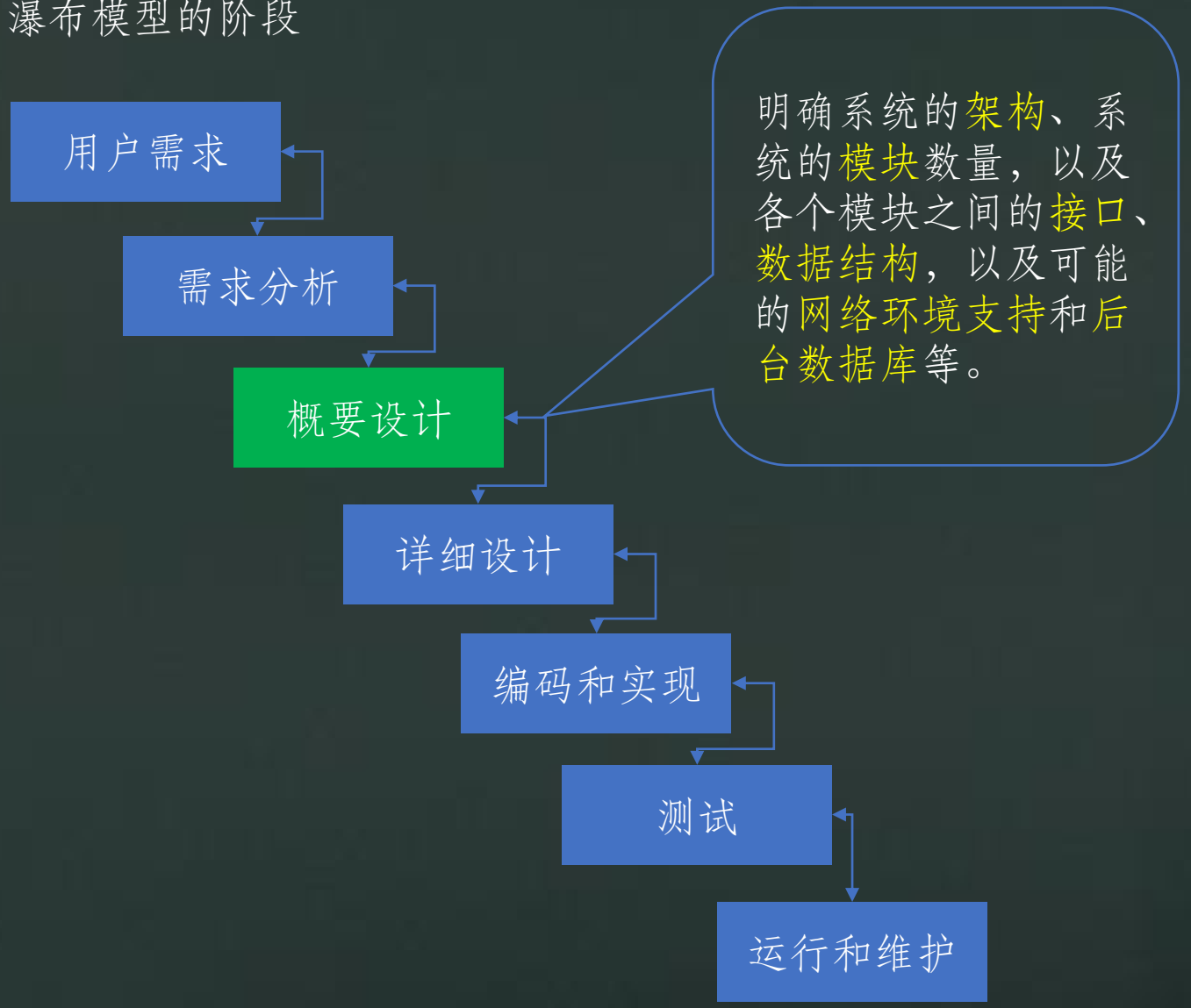
（2）瀑布模型的阶段



对用户需求进行可行性分析，并对用户需求和要求进行详细描述，并最终得到管理层和客户的批准，通过需求分析，定义了开发系统的目的和需要实现的特性和功能。

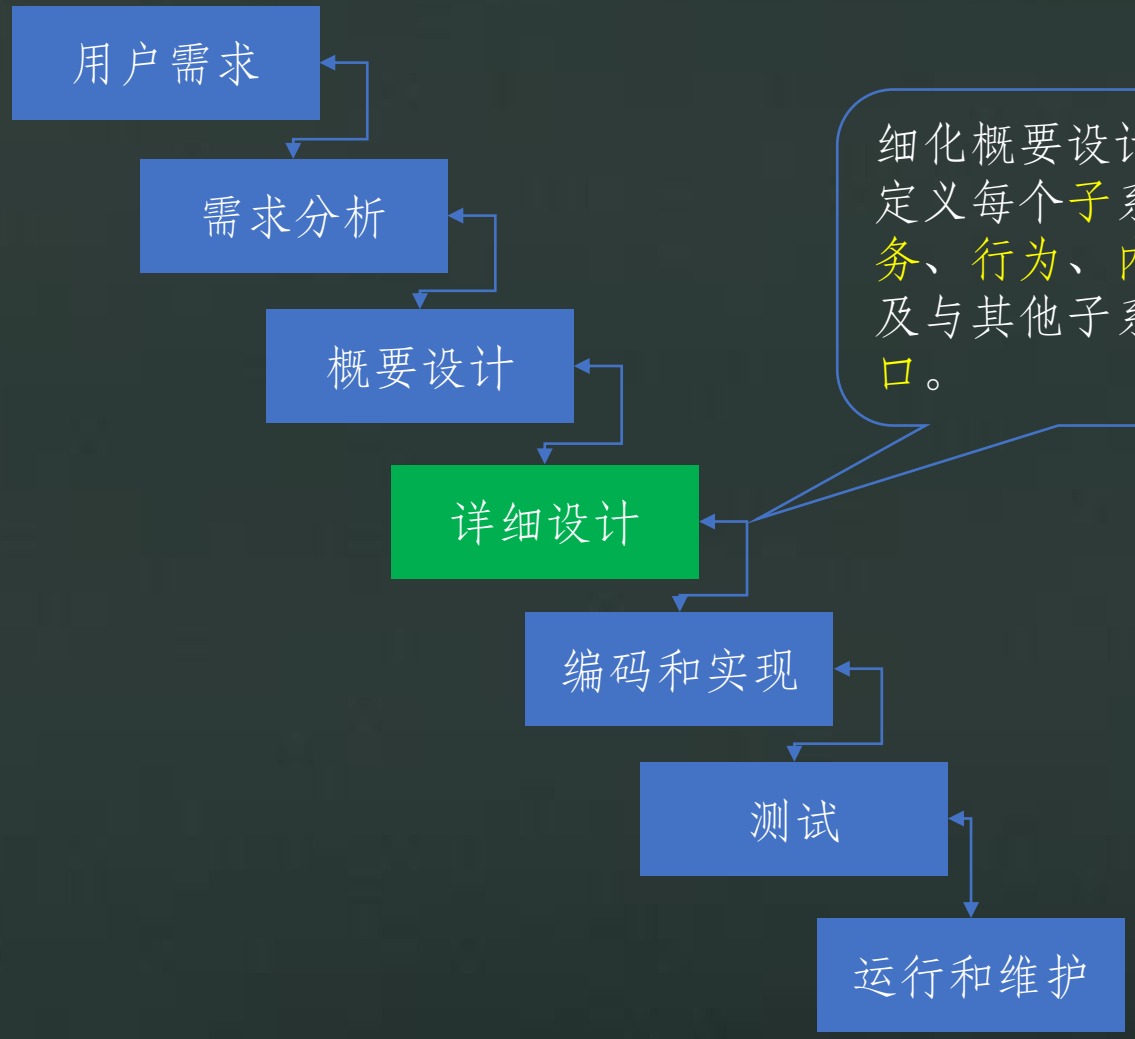
2.2开发模型/瀑布模型

（2）瀑布模型的阶段



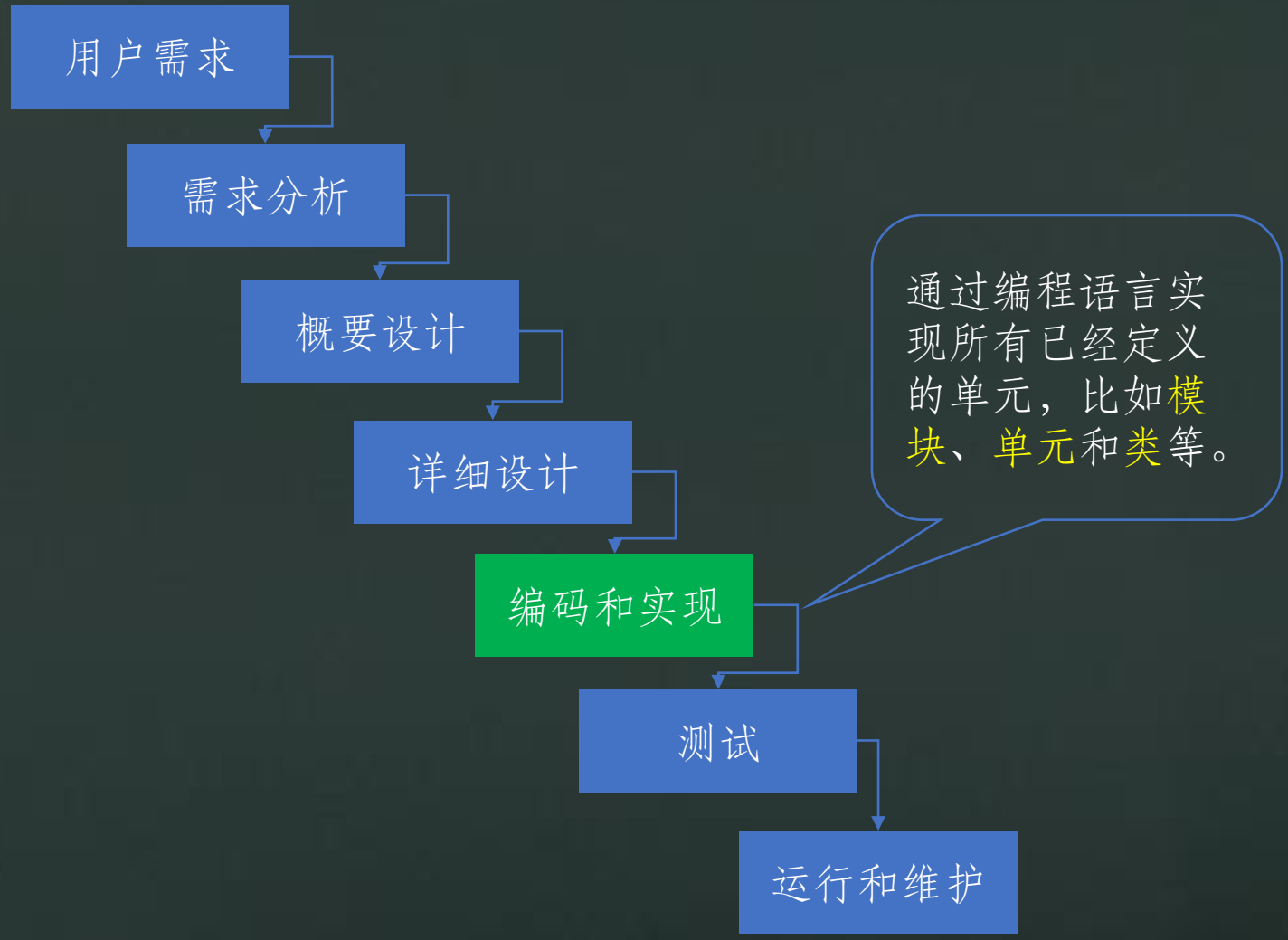
2.2开发模型/瀑布模型

（2）瀑布模型的阶段



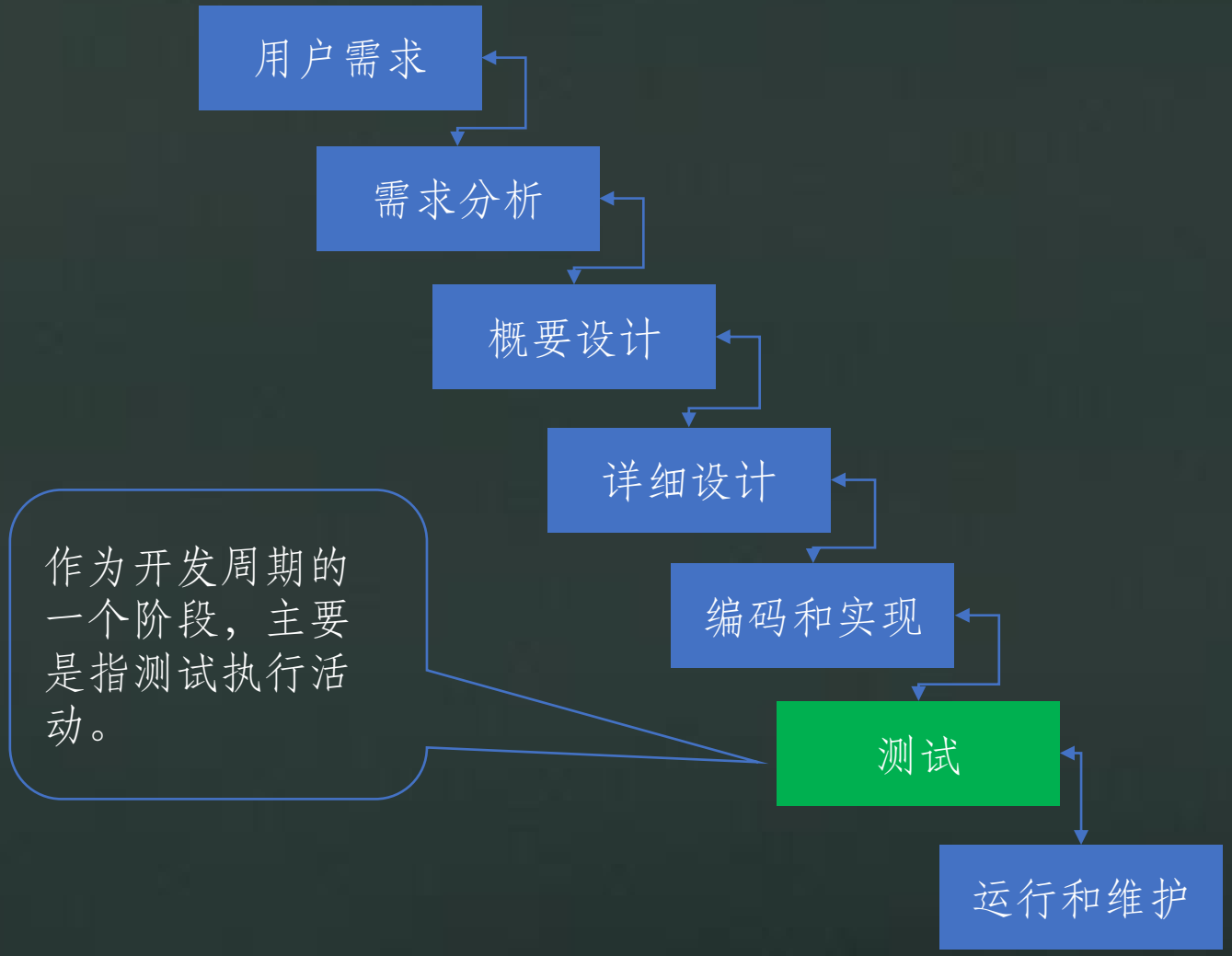
2.2开发模型/瀑布模型

（2）瀑布模型的阶段



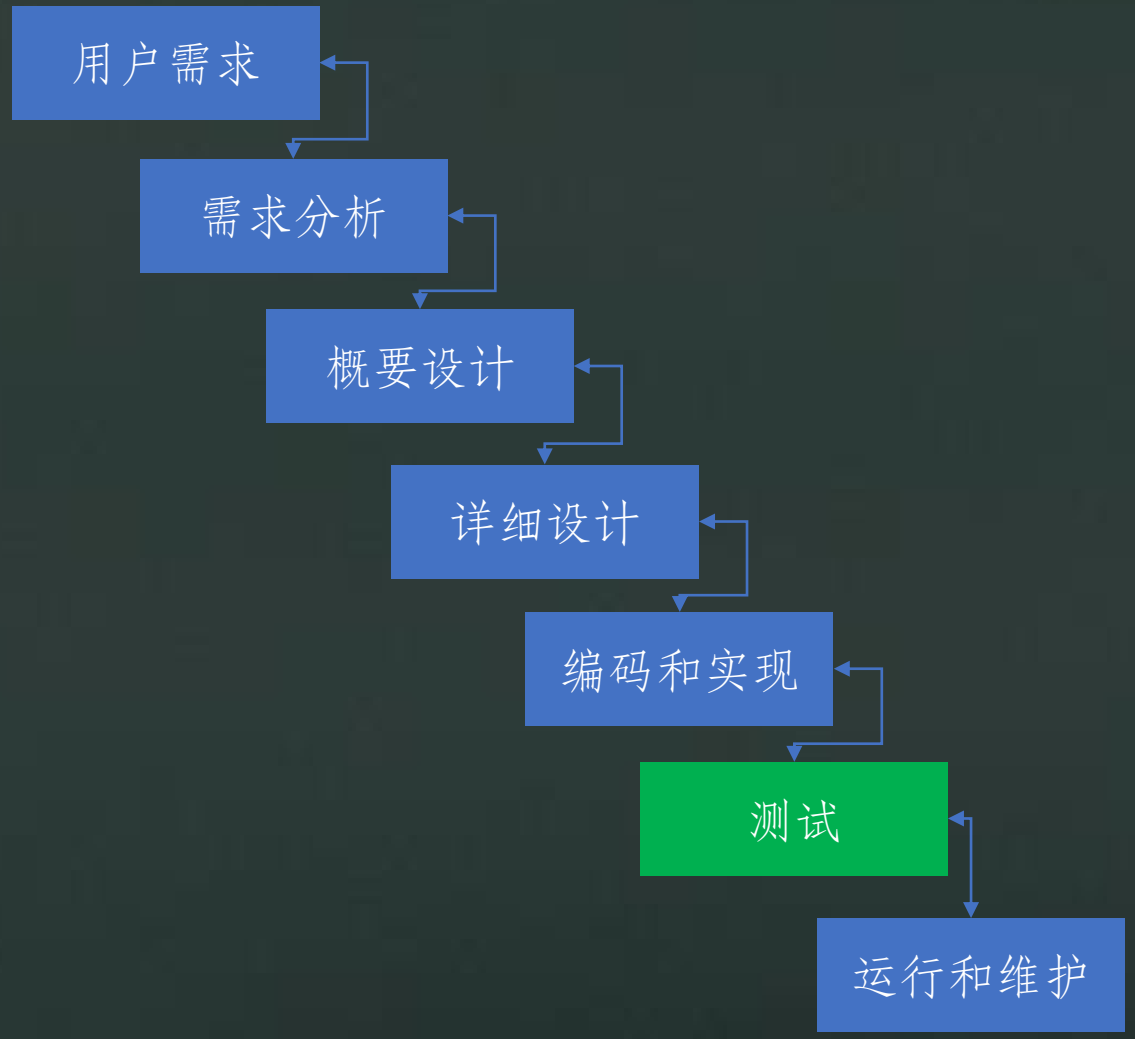
2.2开发模型/瀑布模型

（2）瀑布模型的阶段



2.2开发模型/瀑布模型

（3）瀑布模型的特点



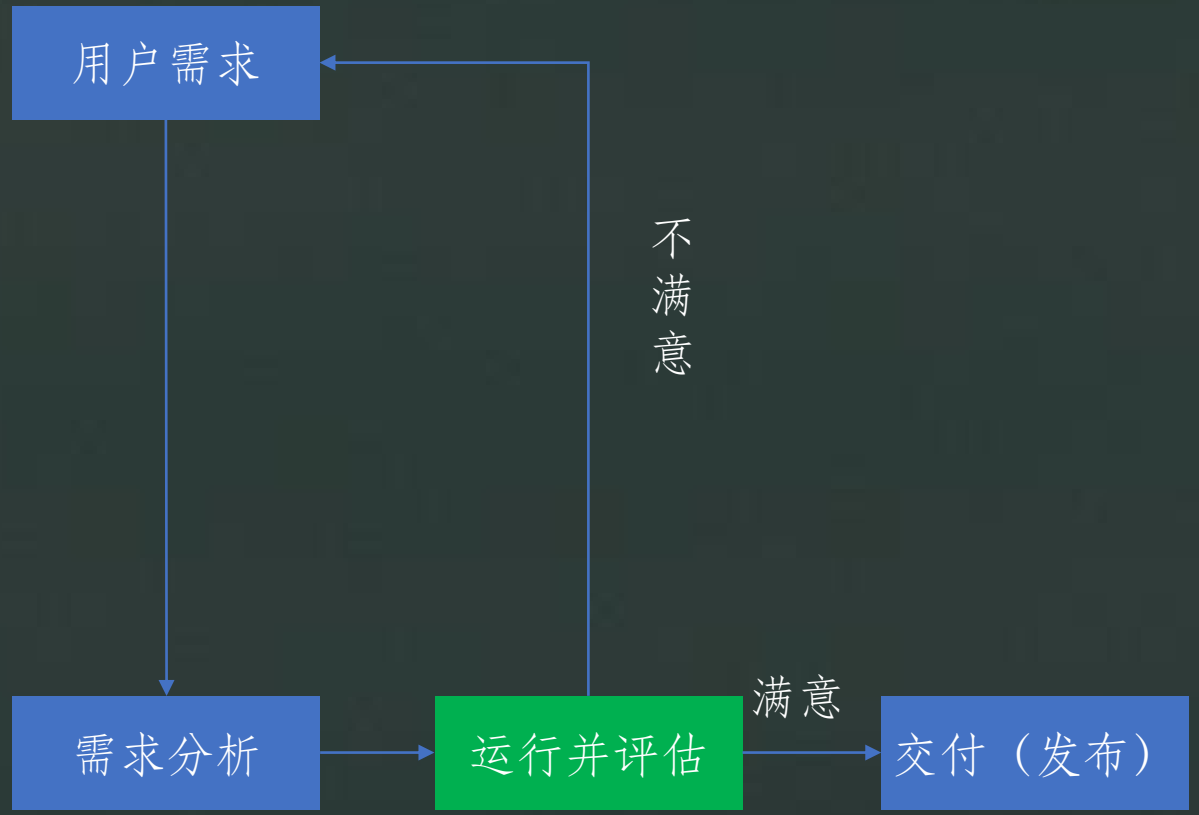
1. 软件测试是开发中的一个阶段，对产品质量进行的最后检查；

2. 在客户需求明确，以及开发过程中没有频繁需求变更，比较适合瀑布开发模型；

3. 假如需求不明确，或者需求经常发生变更，采用瀑布模型是非常不适合的。

2.2开发模型/边做边改模型

（1）边做边改模型的结构



1. 在没有需求规格说明书和系统设计的条件下开发软件，反复对产品进行编码以得到正确稳定的产品；
2. 适用于需求非常简单、容易明白，软件期望的功能行为容易定义，实现的成功或失败容易检验的工程；
3. 缺少计划和设计环节，软件结构容易越修改越糟；
4. 忽略了需求环节，风险极大；
5. 没有考虑测试和程序维护，也没有任何文档，后期维护困难。

2.2开发模型/快速原型模型

（1）快速原型的结构

优点：

（1）克服瀑布模型的缺点，减少由于软件需求不明确带来的开发风险；

（2）适合预先不能确切定义需求的软件系统的开发；

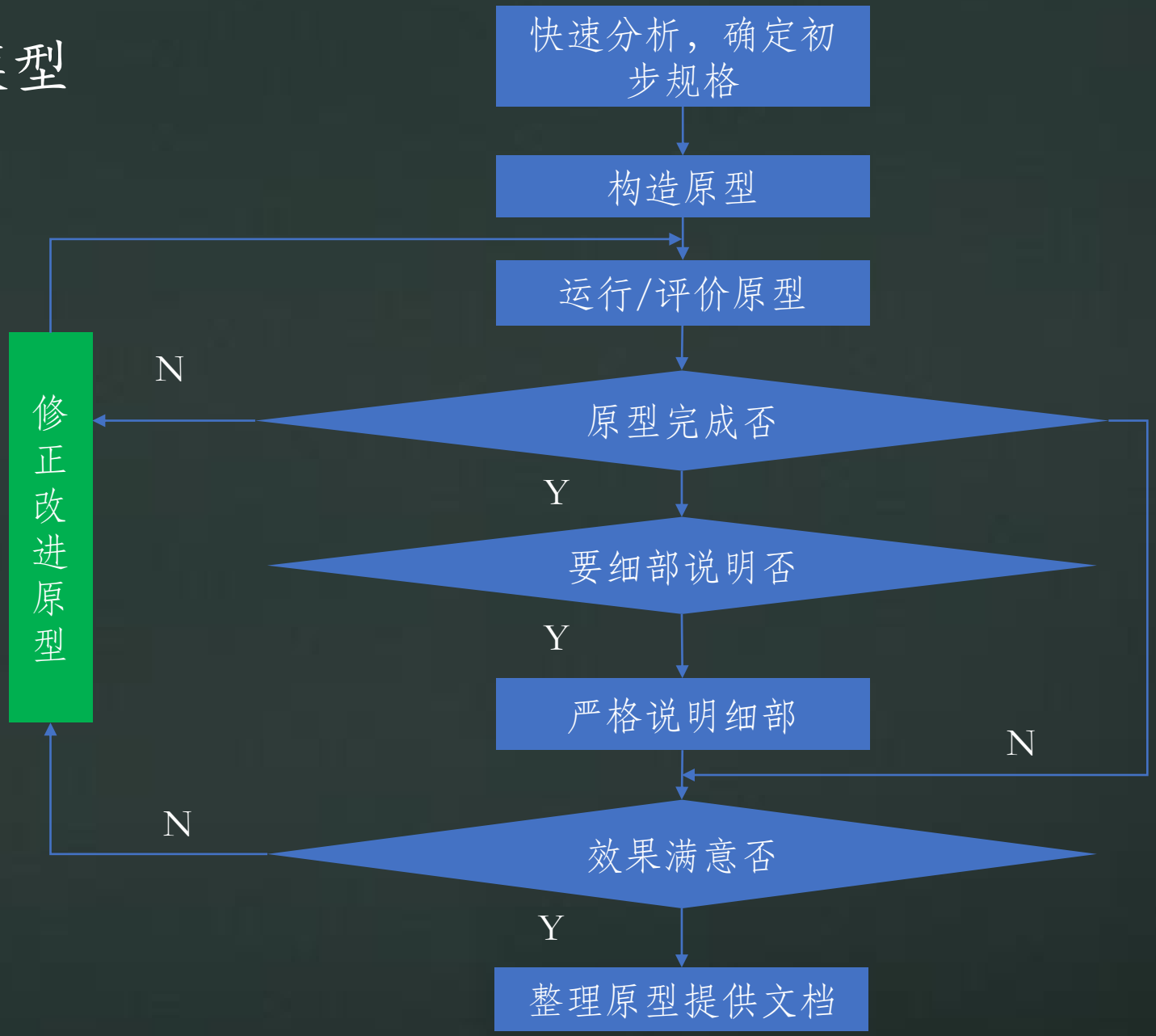
（3）能快速吸引用户，从而抢占市场先机。

缺点：

（1）没有考虑软件整体质量和长期维护；

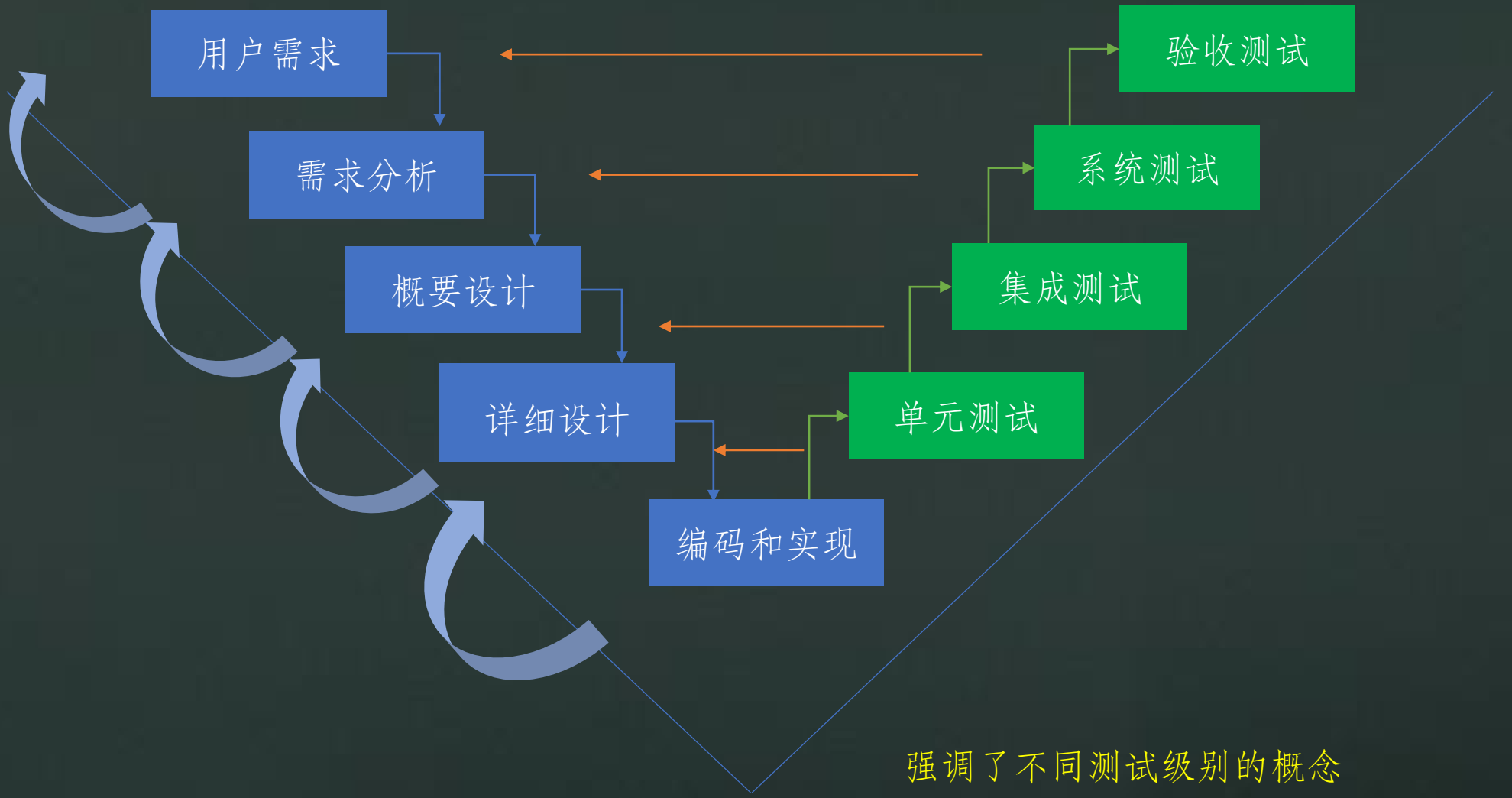
（2）大部分开发都不适合，往往只用于演示功能；

（3）若达不到质量要求，就会被抛弃，并重新设计。



2.2开发模型/V模型

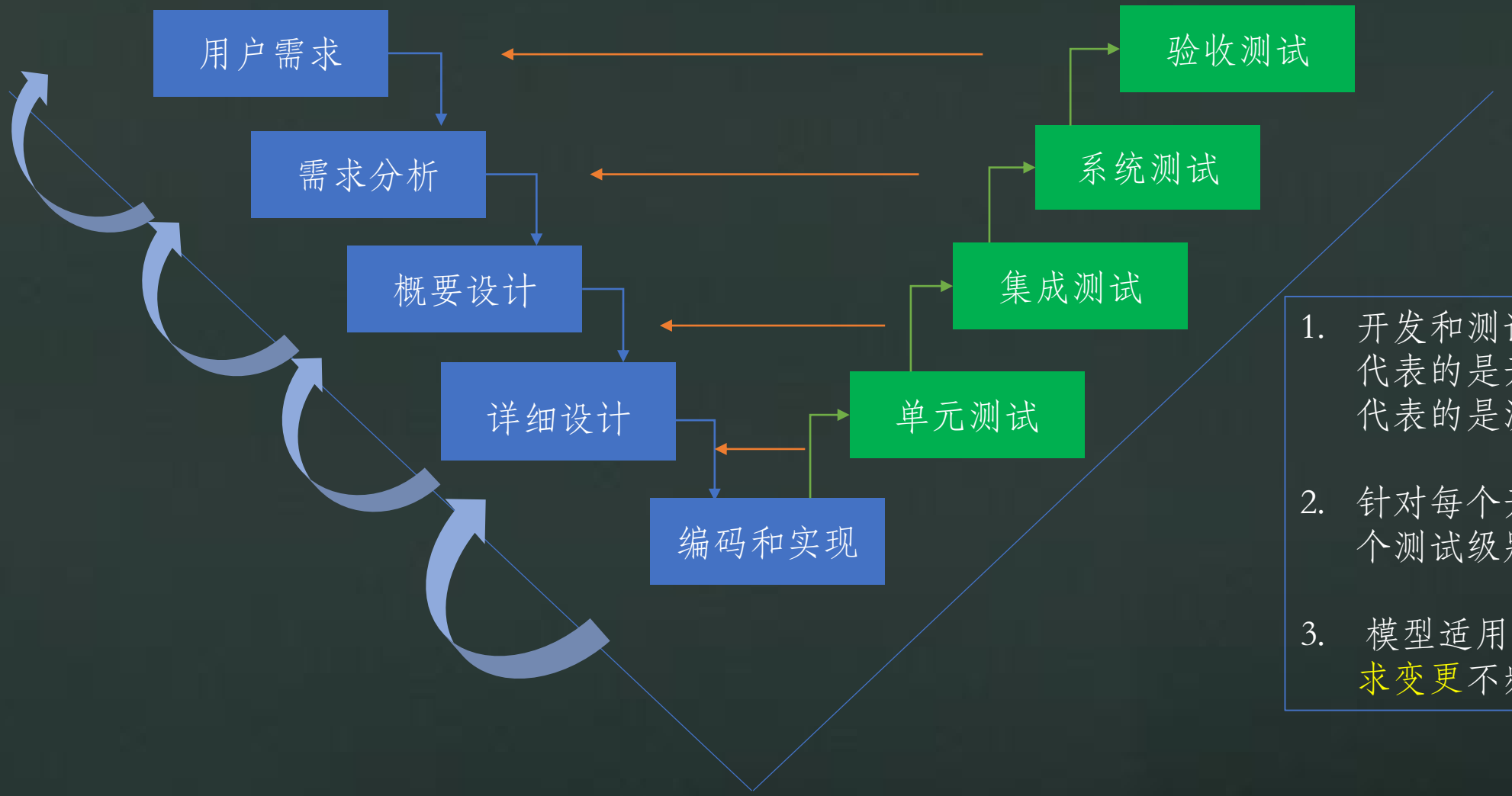
（1）V模型的结构



强调了不同测试级别的概念

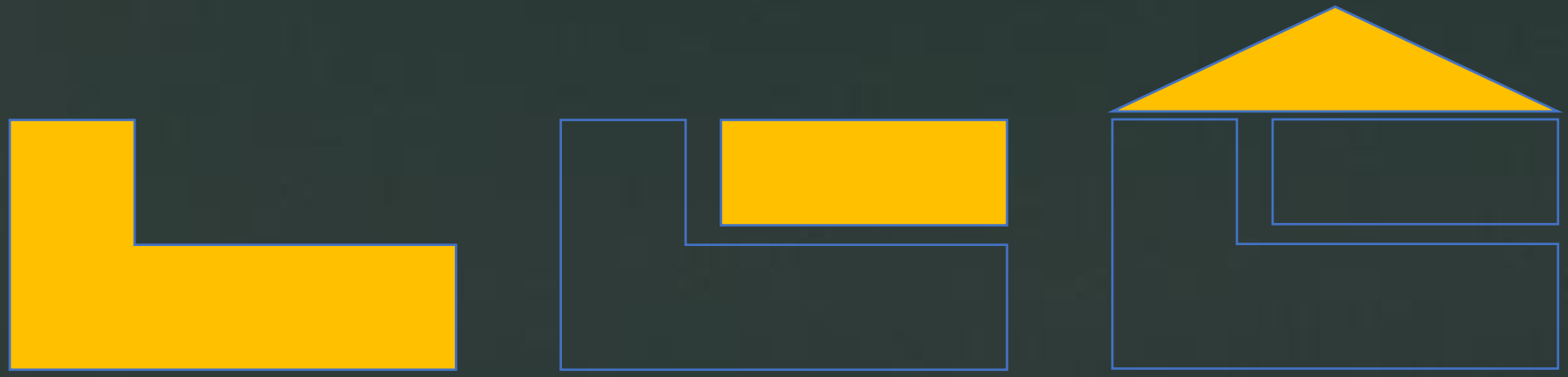
2.2开发模型/V模型

（2）V模型的特点



1. 开发和测试同等重要，左侧代表的是开发活动，而右侧代表的是测试活动；
2. 针对每个开发阶段，都有一个测试级别与之相对应；
3. 模型适用于需求明确和需求变更不频繁的情形。

2.2开发模型/增量模型



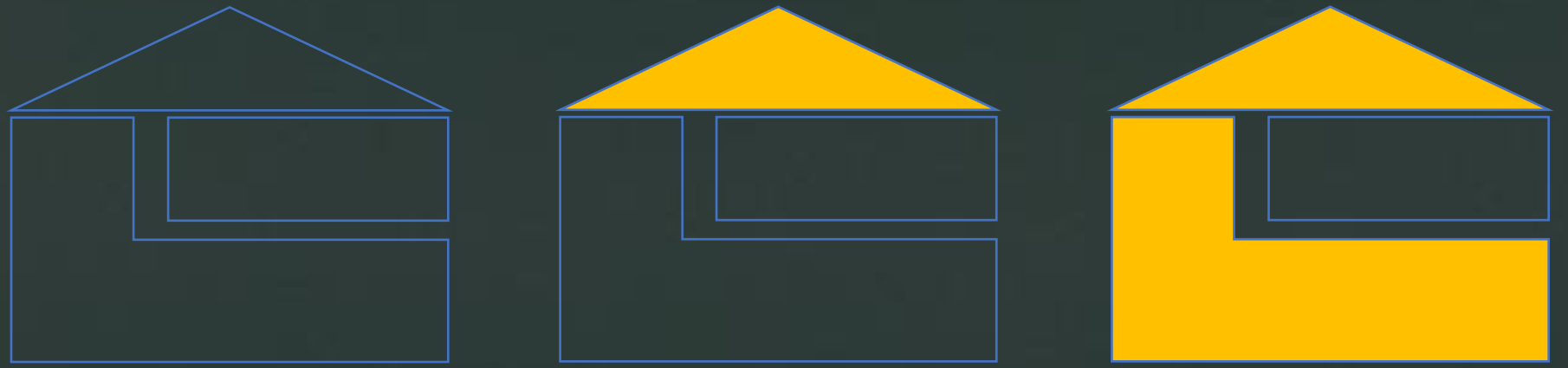
核心功能1　　核心功能1\2　　核心功能1\2\3

1. 每个阶段交付满足客户需求的一个子集的可运行产品，因此可以较好的适应变化，以及控制风险；

2. 增量的一个缺点是后面并入的构件不能破坏已构造好的系统部分，这需要软件具备开放式的体系结构；

3. 需求的变化是不可避免的，增量模型的灵活性可以使其适应这种变化的能力大大优于线性模型，但也很容易退化为边做边改模型，从而是软件过程的控制失去整体性。

2.2开发模型/迭代模型

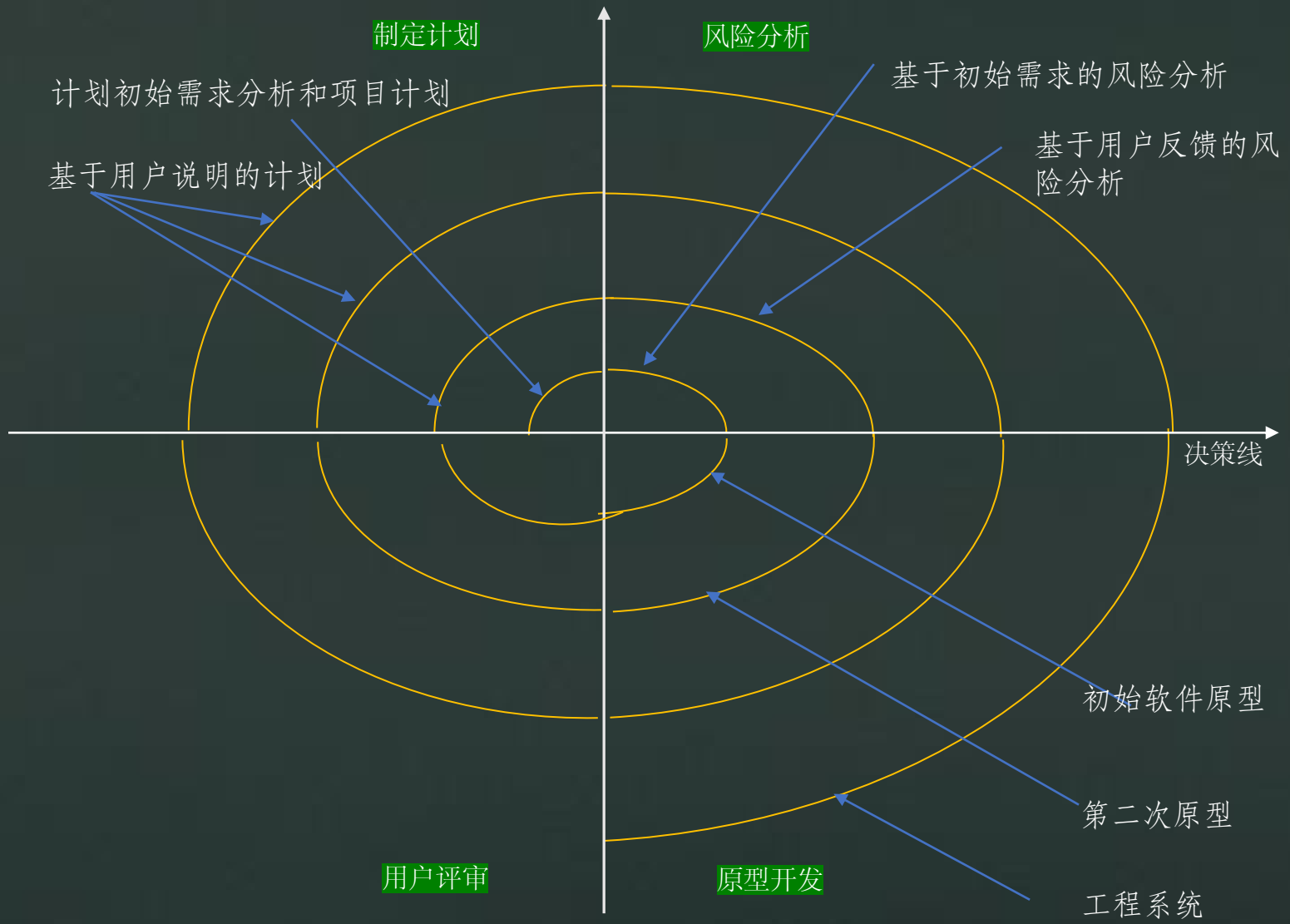


1. 迭代模型包括了一系列的迭代，每一个迭代都包括了一些或者很多的开发活动（需求、分析、设计、实现等等）；

2. 每个后续的迭代都建立在前一个迭代的基础上以使系统得到发展和细化，直到最终产品被完成；

3. 迭代模型中集成不是在项目的尾声进行的“大动作”，每一次迭代都以集成构建系统各部分结束，这样不断的积累将使日后的返工最小化。

2.2开发模型/螺旋模型



1. 螺旋模型是瀑布模型与边写边改模型相结合，并加入风险评估所建立的软件开发模式；

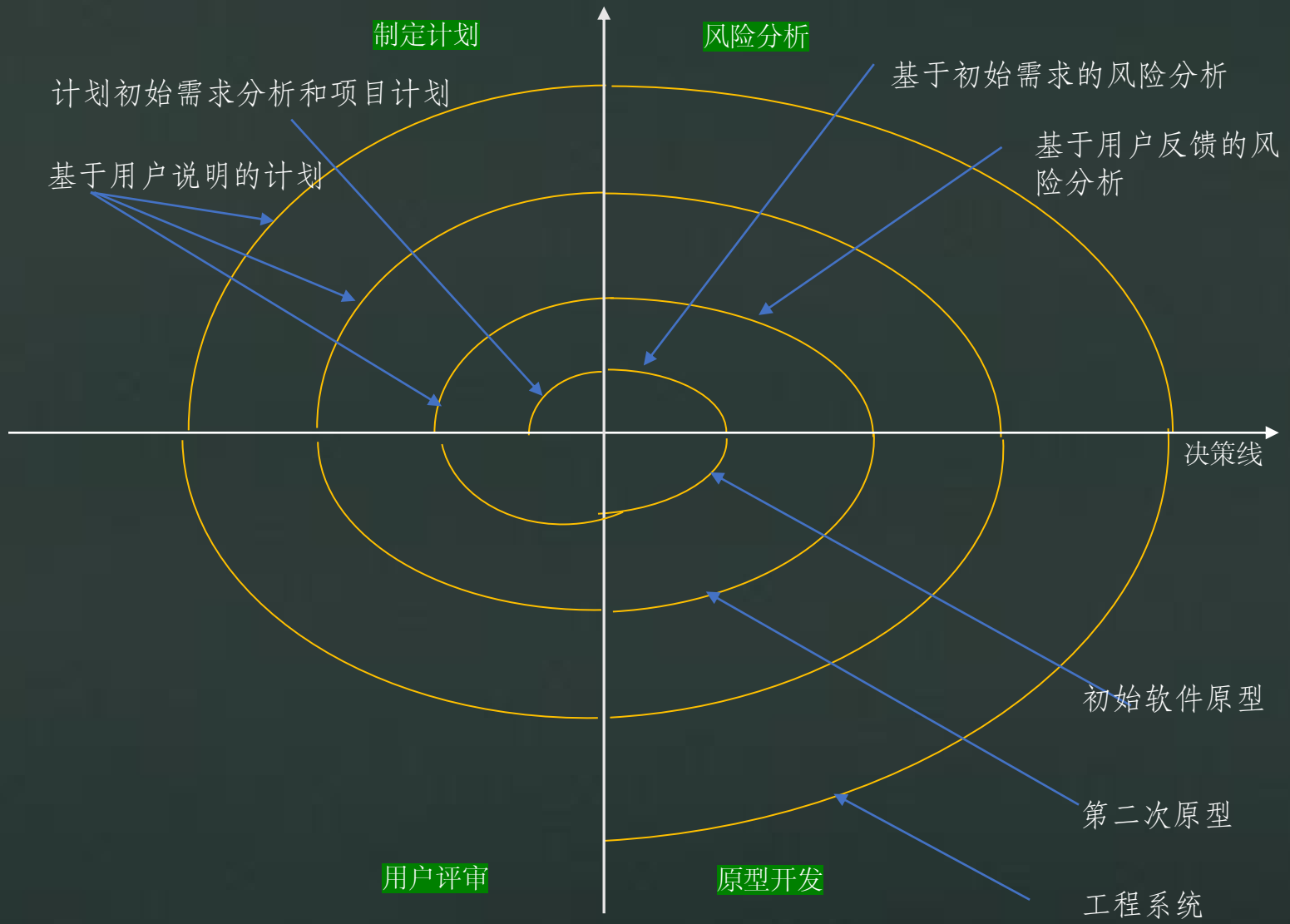
制定计划：确定软件目标、需求和选定实施方案，弄清项目开发的限制条件；

风险分析：评估所选方案，考虑如何识别和消除风险；

原型开发：实施软件开发、编码、测试等；

用户评审：评价开发工作，提出修正建议，规划下期任务。

2.2开发模型/螺旋模型



适合的项目

- 风险是主要的制约因素；
- 项目中的不确定因素和风险限制了进度；
- 用户对自己的需求不是很明确；
- 需求可能会发生重大的变更；
- 项目规模很大；
- 采用了新技术或新概念，需要验证。

模型名称	优点	缺点	适合项目
瀑布模型	➢ 各阶段划分清晰； ➢ 强调计划与需求分析； ➢ 适合需求稳定的产品开发。	➢ 单一流程，不可逆； ➢ 风险显露的晚，纠正机会少； ➢ 测试只是其中一个阶段，缺乏全过程测试思想。	➢ 项目的需求在项目开始前很明确； ➢ 解决方案在项目开始前也很明确。
边做边改模型	➢ 前期出成效快。	➢ 缺少计划和设计环节，软件结构容易越修改越糟 ➢ 忽略了需求环节，风险极大； ➢ 没有考虑测试和程序维护，也没任何文档，后期维护困难；	➢ 适用于需求非常简单，软件期望的功能行为容易定义，实现的成功或失败容易检验的工程；
快速原型模型	➢ 克服瀑布模型的缺点，减少由于软件需求不明确带来的开发风险； ➢ 适合预先不能确切定义需求的软件系统的开发； ➢ 能快速吸引用户，从而抢占市场先机。	➢ 没有考虑软件整体质量和长期维护； ➢ 大部分开发都不适合，往往只用于演示功能； ➢ 若达不到质量要求，就会被抛弃，并重新设计；	➢ 项目的需求在项目开始前不明确； ➢ 类似的项目如：开发新产品，验证技术可行性。
V模型	➢ 文档驱动的开发模型； ➢ 注重反馈和测试。	➢ 严格限定了开发的各阶段，缺乏迭代性； ➢ 集成测试是发现缺陷最多的时候，在模型中滞后严重； ➢ 不支持探测性测试； ➢ 不适合新领域，缺乏对变化的支持； ➢ 不支持频繁集成。	➢ 项目需求明确； ➢ 长期专职开发人员的小型项目开发。
增量模型	➢ 有效缩短开发时间，有效规避并降低开发风险； ➢ 开发人员与用户可通过原型充分地交流； ➢ 有利于用户培训、销售和开发的同步； ➢ 模型的灵活性可使其适应需求的变化。	➢ 软件必须是开放式的体系架构； ➢ 若缺乏严格的过程管理，容易退化为“边做边改”； ➢ 对产品需求分析要求高，若需求不全面，会影响产品设计的完整性。	➢ 项目开始，明确了需求的大部分，但是需求可能会发生变化； ➢ 对于市场和用户把握不是很准，需要逐步了解； ➢ 具有复杂功能的大型系统进行功能改进。
迭代模型	➢ 降低了产品无法按照既定进度进入市场的风险； ➢ 加快整个开发工作的进度； ➢ 使适应需求的变化更容易些。	➢ 不适用于较小的项目； ➢ 风险管理成本较高； ➢ 进度管理方面，对项目组成员的要求也非常高。	➢ 分析设计人员对应用领域很熟悉； ➢ 需求不甚明确、高风险项目； ➢ 用户可不同程度地参与整个项目地开发流程。
螺旋模型	➢ 设计上灵活，各阶段都可变更； ➢ 开发过程划分详细，成本计算更简单； ➢ 客户参与各阶段开发，保证项目可控； ➢ 强调风险分析，规避开发风险； ➢ 适合庞大、复杂并具高风险的项目。	➢ 需要相当丰富的风险评估知识与经验； ➢ 过长的开发周期，导致产品交付时，技术可能落后； ➢ 过多的迭代增加开发成本，延迟交付时间。	➢ 风险是最主要的制约因素； ➢ 项目中的不确定因素和风险限制了进度； ➢ 用户对自己的需求不是很明确； ➢ 需求可能会发生重大的变更； ➢ 项目规模很大； ➢ 采用了新技术或新概念，需要验证。

2.3 练习题

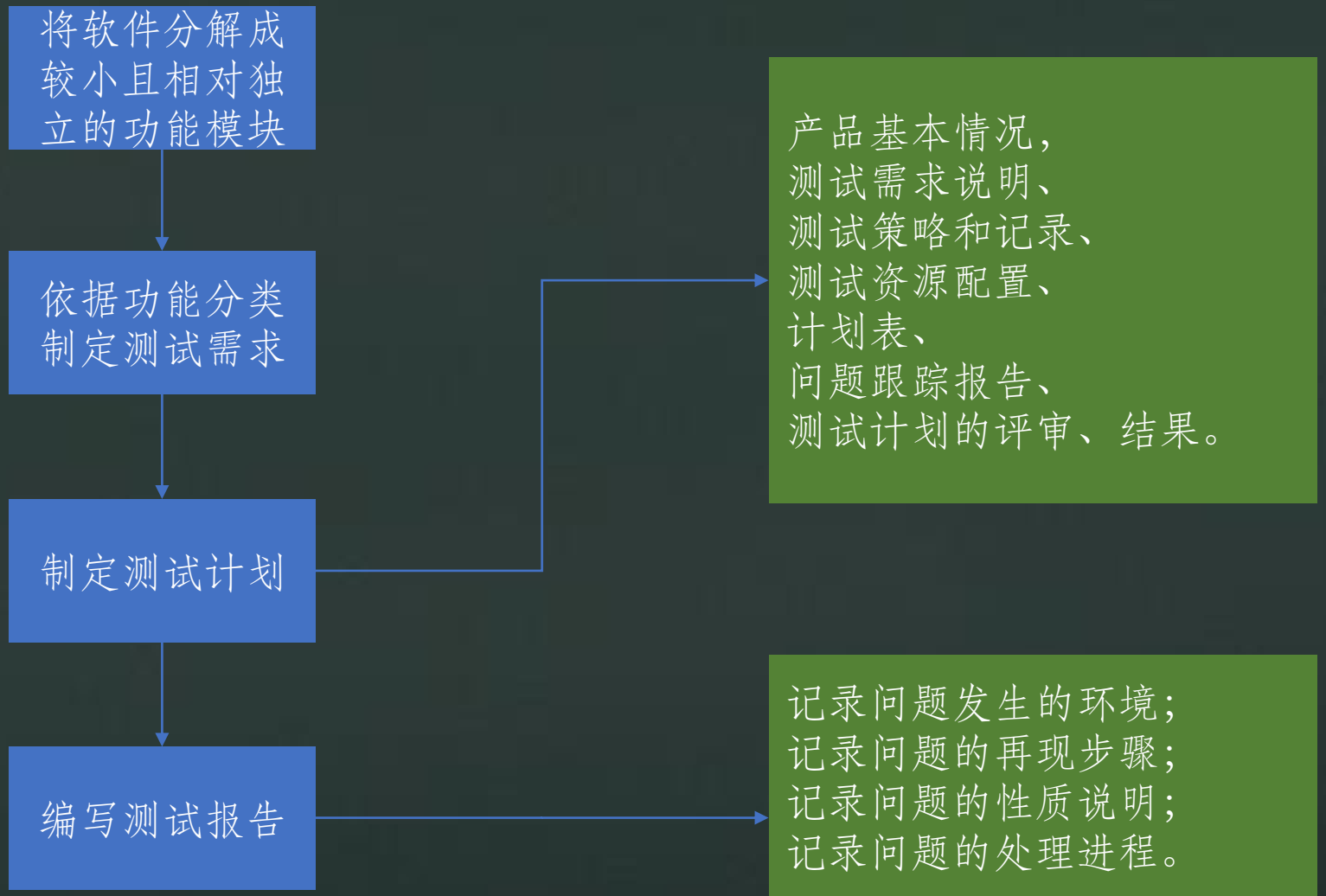
场景	可采用的测试模型
1. 软件需求较稳定，软件开发团队计划用三个月的时间进行需求分析、四个月的时间进行系统设计、四个月的时间进行编码，所有活动将采用阶段式完成方式进行。	?
2. 软件需求信息不足，项目组计划采取先设计部分原型系统，以便与客户进行交流来确认下一步的开发需求与系统改进。	?
3. 由于项目规模较大，开发过程中，项目组希望开发组进行多次发布，并进行已完成模块的集成，从而验证系统进展情况，以便做出适当的调整。	?

软件测试的方法有哪些？

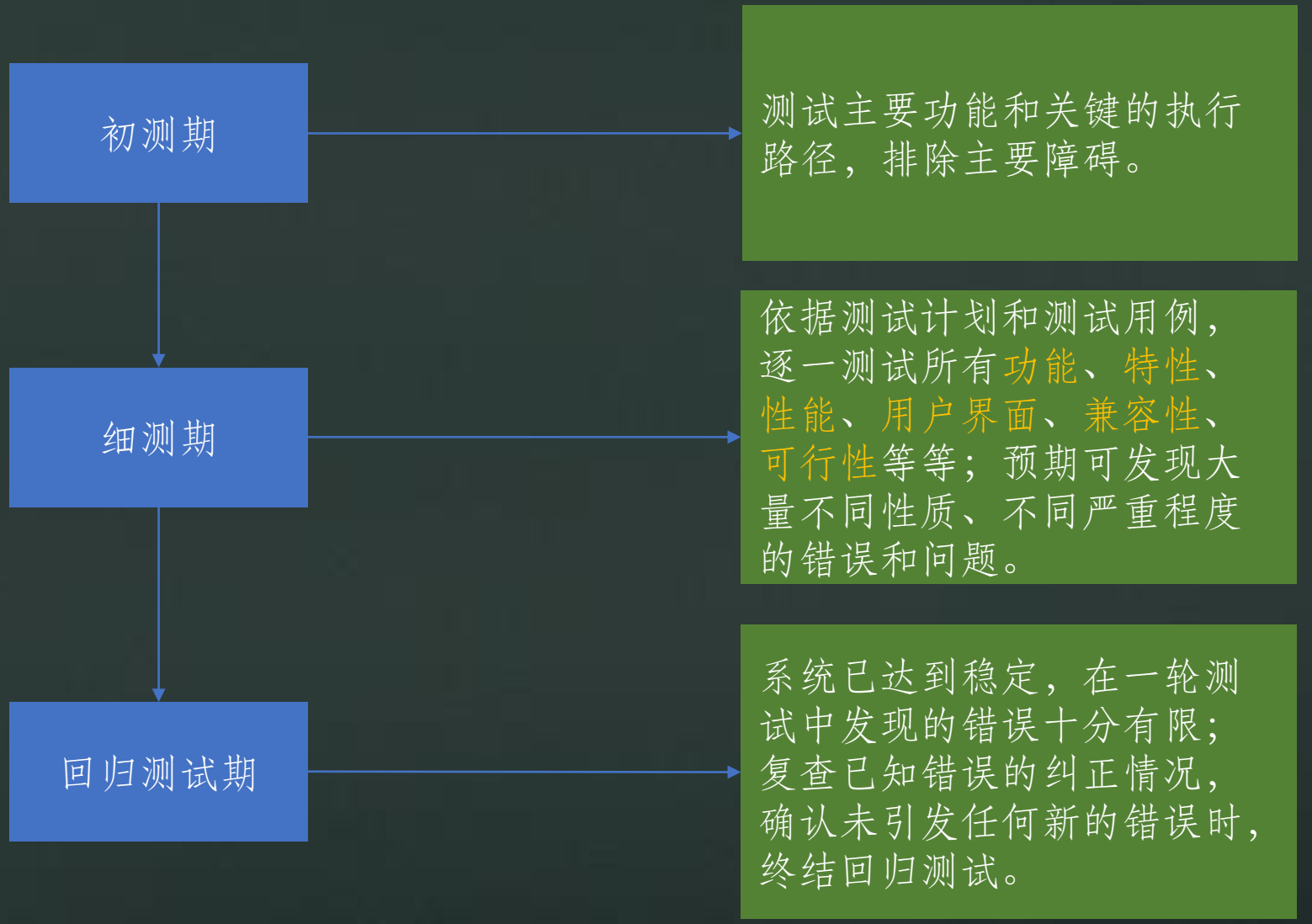
3.1软件测试的方法/常见方法

测试方法	解释
静态测试（分析）	不运行被测程序本身，评审文件和阅读代码等方式测试软件；代码风格和规则审核；程序设计和结构的审核； 业务逻辑的审核； 走查、审查与技术复审。
动态测试	运行被测程序，检查运行结果与预期结果的差异。
黑盒测试	功能测试、数据驱动测试或基于规格说明的测试，是一种从用户角度出发的测试。 测试只知道该程序输入和输出之间的关系或程序的功能，依据需求规格书考虑确定测试用例和推断结果的正确性。 用于证明软件功能的正确性和可操作性。
白盒测试	结构测试、逻辑驱动测试或基于程序的测试。依赖于对程序细节的严密检查，针对特定条件和循环集设计测试用例，对软件的逻辑路径进行测试。要求对某些程序的结构特性做到一定程度的覆盖，或者说是“基于覆盖的测试”。
回归测试	修改了旧代码后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。
α测试（非正式验收测试）	模拟各类用户对即将面市软件产品进行测试，尽可能逼真地模拟实际运行环境和用户对软件产品的操作并尽最大努力涵盖所有可能的用户操作方式。不能由程序员或测试员完成，手册文稿等应该在该测试前准备好。
β测试	发放一部分给用户进行测试，并要求用户报告异常情况、提出批评意见，然后软件开发公司再对软件进行改错和完善。

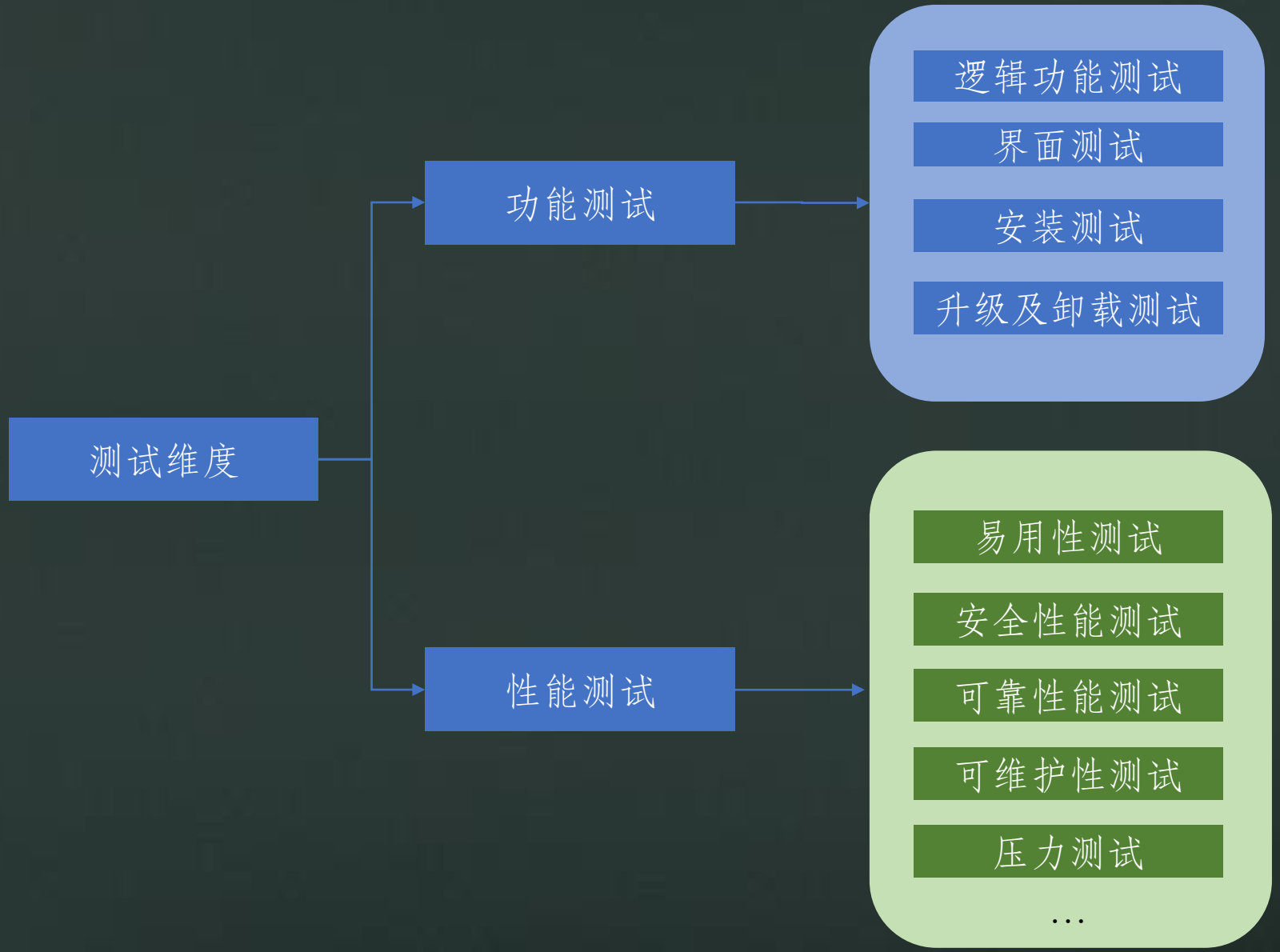
3.2软件测试的方法/流程



3.3软件测试的方法/执行过程



3.4软件测试的方法/测试维度



3.4.1软件测试的方法/测试维度/系统功能

子维度	测试内容
文档	评审需求规格书、技术要求等。
配置测试	确保软件在不同的硬件和软件配置上按预期运行的测试。
安装测试	确保在不同的条件下（磁盘空间不足或电源中断）按预期安装的测试。
基准测试	与已知的参考和系统的性能进行对比。
边界测试	测试产品特定的限制，如最大值，最小值，确定软件在这些特定的限制下能否正常工作。
正/逆向测试	确定正确使用软件时，软件某一特性产生的结果是否与需求一致。 确定软件无效输入或非法操作的处理是否合理。
数据准确性	数据结果是否正确，精度是否达标。

3.4.2软件测试的方法/测试维度/系统性能

- 易用性测试

子维度	测试内容
页面风格一致性	页面结构、导航、菜单、链接、搜索、翻页、字体、列表、日期和扫描控件、数据精度的风格是否一致。
易浏览性	具有必要的信息指导用户使用程序。
	输入、输出设计规矩，输出结果应简洁、直观、美观、方便阅读、易懂和使用。
	人机界面简洁、美观、实用，风格相对一致，符合办公习惯。
	在界面、人机交互、输出中的用语应与业务用语一致。
易操作性	具有严重后果的功能执行可逆，或者给出明显警告，执行前要求确认。
	软件操作简便，系统支持标准的鼠标、键盘操作，支持鼠标的单击、双击和右键操作，支持快捷键操作。
	提供辅助输入手段（如选择输入、默认值等），数据检索方便、灵活。

3.4.3软件测试的方法/测试维度/系统性能

- 安全性测试

子维度	测试内容
权限验证	权限进行测试，确保不同的用户能看到正确的菜单和操作。
信息所有权验证	验证具有同一权限的不同用户的信息，是否只能由该用户操作，而其他用户不能代为操作，从而保证每条信息的安全。

3.4.4软件测试的方法/测试维度/系统性能

- 可靠性测试

子维度	测试内容
成熟性	使用的容量达到规定的极限时，系统不崩溃、不异常退出也不丢失数据。
	产品描述中列出的其他程序或用户造成的错误输入时，系统不崩溃也不丢失数据。
	输入用户文档中明确规定的非法指令时，系统不崩溃也不丢失数据。
容错性	能屏蔽用户的误操作。
	对错误有正确提示。
	输入数据错误时，系统不崩溃、不异常退出也不丢失数据。
易恢复性	系统运行失效后，应能较快重建系统。
数据校验机制	应对数据项之间的逻辑关系进行校验，保证数据的有效性。
	应保证数据的完整性和一致性，不会因删除或反复的更新而被破坏或留下垃圾数据（系统更新或删除功能不影响系统数据）。
	对不符合要求的输入数据，系统应使用中文给出简洁、准确的提示信息，必要时应给出帮助。

3.4.5软件测试的方法/测试维度/系统性能

- 可维护性测试

子维度	测试内容
日志维护	业务操作记录都能自动记录到日志。
	日志内容中包含的所有信息都记录正确。
其他	代码可读性、程序文档可理解性等。 易分析性、易修改性、模块化、可重用性。

3.4.1软件测试的方法/测试维度/系统性能

- 其他

子维度	测试内容
文档	评审需求规格书、技术要求、概要设计文档、详细设计文档等。
并发性能测试	采用一定数量的用户并发使用系统进行测试，查看系统运行情况。
响应速度测试	改变系统与控制台之间的交互频次。 （可通过优化数据库处理语句定义和优化索引等方式提高数据库处理速度）。
容量测试	测试对象对于大量数据的处理能力的测试。
强度测试	确保系统可在遇到异常条件时按预期运行。 包括过大的工作量、不充足的内存、不可用的服务/硬件或过低的共享资源。

4复盘

- 什么是软件测试？

软件测试的定义、目的和意义、对象和特点、原则、级别。

- 软件测试和开发模型二者有什么关联？

瀑布模型、边做边改模型、快速原型、V模型、增量模型、迭代模型、螺旋模型。

- 软件测试的方法有哪些？

常见方法、流程、执行过程、测试维度等。

软件测试体系学习

ISTQB®
International
Software Testing
Qualifications Board

Chinese Software Testing Qualifications Board
CSTQB

中国软件测试认证委员会
理事单位


欢迎批评指正
谢谢！