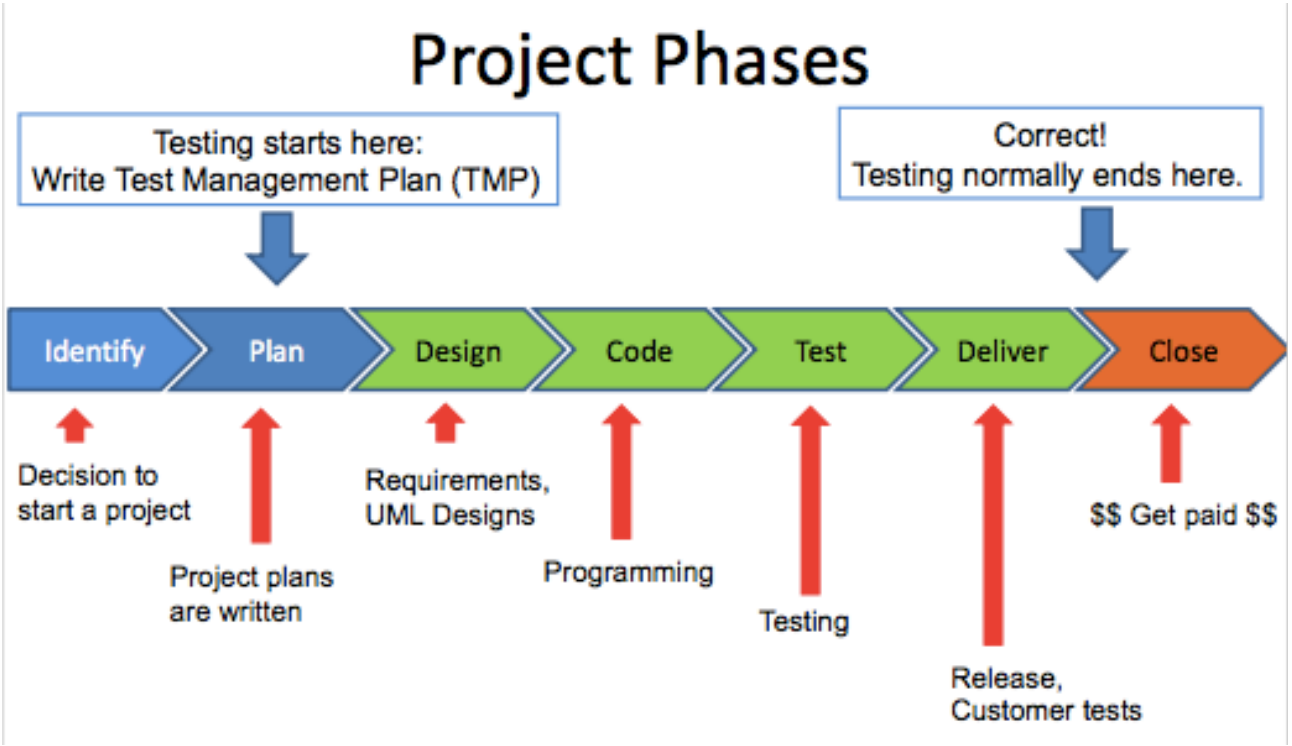
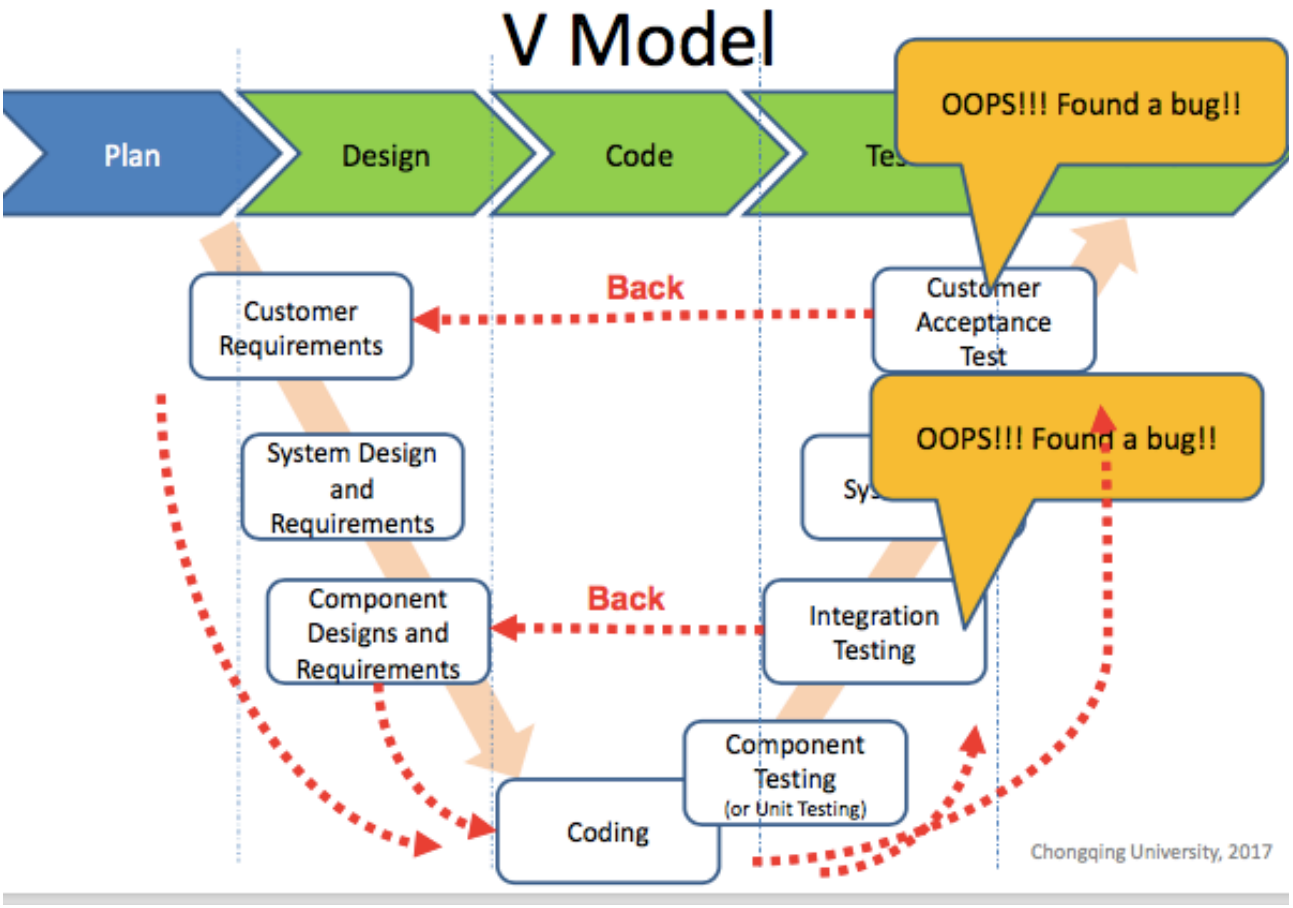
Software Testing Review

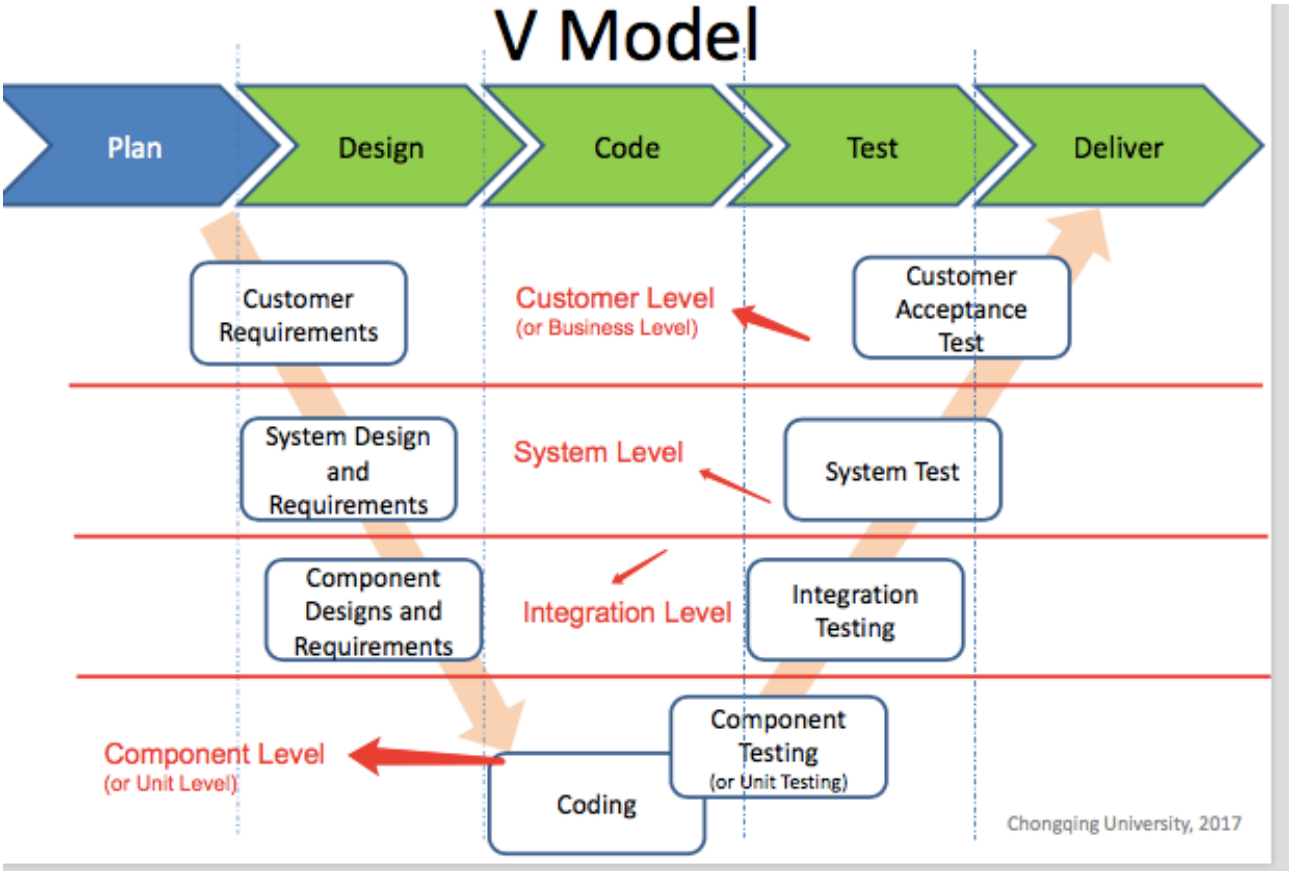
Course Overview/Objectives

Lectures Contents:

- Introduction to Software Testing
- Short Introduction to V-Model SDLC
  - Test Levels (Unit, Integration, System, Acceptance Testing)
- Planning of Testing
  - Writing a Test Plan
  - Test Prioritization Strategies
- Test Execution
  - Test Reporting
  - Static Testing
    - Reviewing Requirements
    - Extracting Test Scenarios from designs
    - Static Code Analysis
  - Dynamic Testing
    - Black, White and Grey Box Test Techniques
  - Test Automation
    - Tools, Simulators, Stubs
  - Security Testing
  - Non-Functional Testing




V Model
Plan
Design
Code
Test
OOPS!!! Found a bug!!
Customer Requirements
Back
Customer Acceptance Test
System Design and Requirements
OOPS!!! Found a bug!!
Component Designs and Requirements
Back
Integration Testing
Component Testing
(or Unit Testing)
Coding
Chongqing University, 2017


V Model
Plan
Design
Code
Test
Deliver
Customer Requirements
Customer Level
(or Business Level)
Customer Acceptance Test
System Design and Requirements
System Level
System Test
Component Designs and Requirements
Integration Level
Integration Testing
Component Level
(or Unit Level)
Component Testing
(or Unit Testing)
Coding
Chongqing University, 2017

Test Management is only one of the four processes of a project's quality system

Test Management Plan
Contains:
1.purpose
2.scope
3.test object
4.test resource
5.test level strategy
6.test prioritization strategy
7.test reporting
Test Level Strategy
Unit Testing ,Integration Testing,System Testing,Customer Acceptance Testing.
Test Prioritization Strategy
1.Customer Requirements-Based Prioritization 基于客户需求的优先级
    1.Customer-Assigned priority 客户分配优先权
    2.Implementation/Requirements Complexity 实施/需求复杂性
    3.Requirements Volatility需求波动
2.Coverage-Based Prioritization
  Types of Coverages :
    1.Requirements Coverage
        1.Initial Requirements Coverage初始要求覆盖
        2.Total Requirement Coverage(requirements from all levels)总要求覆盖
        3.Additional Requirements Coverage(added/changed)附加要求覆盖
    2.Statement Coverage(testing the paths by calling only main methods)声明覆盖
    3.Code Coverage代码范围
3.Cost Effective-Based Prioritization 基于成本效益的优先级
    this strategy if often applied in combination with other strategy.
4.History-Based Prioritization
5.Risk-Based Prioritization

1.Basic Steps of Structured Testing?

1.Write Test Management Plan
2.Select what has to be tested
3.Decide when, how and to what extent the testing needs to be done.
4.Develop test scenarios/cases
5.Execute the tests
6.Write Test Reports

2.**SMART　Reviewing Requirement复查需求**

S-Specific是否是明确指定的

M-Measurable是否是可测量的，比如数据的准确性，而不是best, optimal(最佳的，最理想的), fastest, around, about

A-Achievable是否是可实现的

R-Relevant是否是需求相关的

T-Traceable是否是可追踪的，是否有标志符序号

Static Testing Method:

1.Review Requirement
2.Static Code Analysis

Dynamic Testing has 3 types of Tests methods

1.Black Box Testing methods
    1. No programming knowledge required
    2.Based On requirements and specifications
2.White Box Testing methods
    1.Programming knowledge is required
    2.Based on detailed designs/requirements
3.Gray Box Testing methods
    1.Methods can be used for Black Box testing
    2.Methods can be used for White Box testing

2. 白盒测试

白盒测试也称结构测试或逻辑驱动测试，它是知道产品内部工作过程，可通过测试来检测产品内部动作是否按照规格说明书的规定正常进行，按照程序内部的结构测试程序，检验程序中的每条通路是否都有能按预定要求正确工作，而不顾它的功能，白盒测试的主要方法有逻辑驱动、基路测试等，主要用于软件验证。

"白盒"法全面了解程序内部逻辑结构、对所有逻辑路径进行测试。"白盒"法是穷举路径测试。在使用这一方案时，测试者必须检查程序的内部结构，从检查程序的逻辑着手，得出测试数据。贯穿程序的独立路径数是天文数字。但即使每条路径都测试了仍然可能有错误。第一，穷举路径测试决不能查出程序违反了设计规范，即程序本身是个错误的程序。第二，穷举路径测试不可能查出程序中因遗漏路径而出错。第三，穷举路径测试可能发现不了一些与数据相关的错误。

3. 灰盒测试

灰盒测试，确实是介于二者之间的，可以这样理解，灰盒测试关注输出对于输入的正确性，同时也关注内部表现，但这种关注不象白盒那样详细、完整，只是通过一些表征性的现象、事件、标志来判断内部的运行状态，有时候输出是正确的，但内部其实已经错误了，这种情况非常多，如果每次都通过白盒测试来操作，效率会很低，因此需要采取这样的一种灰盒的方法。

灰盒测试结合了白盒测试盒黑盒测试的要素.它考虑了用户端、特定的系统知识和操作环境。它在系统组件的协同性环境中评价应用软件的设计。

灰盒测试由方法和工具组成，这些方法和工具取材于应用程序的内部知识盒与之交互的环境，能够用于黑盒测试以增强测试效率、错误发现和错误分析的效率。

灰盒测试涉及输入和输出，但使用关于代码和程序操作等通常在测试人员视野之外的信息设计测试。

Black Box Method:
1.Control-Flow Testing
2.Path-Testing
3.Execution of the Scenarios
Gray Box Method
Boundary Testing
White Box Test Techniques:
1.Unit Test Code 单元测试
2.Loop Testing 循环测试
3.Control Structure Testing控制结构测试
Grey Box Test Techniques:
Boundary Value Analysis 边界值分析
2.Equivalent Partitioning 等效分区
Black Box Test Techniques:
1.Path Testing 路径测试
2.Cause-Effect Graphing 因果图形
3.Comparison Testing 比较测试
4.Fuzz Testing 模糊测试
5.Monkey Testing 随机测试
Error Categories:
1.Syntax Errors
2.Compilation Error (or Compile-Time Error)
3.Semantic Error(or Logical Error)
4.Run-Time Errors

Error Categories Overview

	Caused In	Caused by	Discovered During	Code Compiles?	Application Runs?	Discover Difficulty	Note
Syntax Errors	Code	Programmer	Compiling (at latest)	No	No	Easy	
Compilation Errors	Code Compiler	Programmer, Compiler Software	Compiling	No	No	Easy up to Difficult	
Semantic Errors	Requirements, Designs, Code	Requirements eng., Designer/Architect, Programmer	After application startup	Yes	Yes	Easy up to Difficult	Results in wrong system behavior
RunTime Errors	Requirements, Designs, Code	Requirements eng., Designer/Architect, Programmer	After application startup	Yes	Yes	Easy up to Difficult	Results in system (functions) failure (e.g. crash)

Most common types of Non Functional Tests:
1.Usability Testing可用性测试:测试系统对特定用户组（禁用用户）的操作和可用性（ISO 9241，ISO 9126）
可用性测试（Usability testing），是一项通过用户的使用来评估产品的技术，由于它反应了用户的真实使用经验，所以可以视为一种不可或缺的可用性检验过程。 也就是说，可用性测试是指让用户使用产品（服务）的设计原型或者成品，通过观察，记录和分析用户的行为和感受，以改善产品（服务）可用性的一系列方法。

2.Recovery Testing恢复测试:测试系统恢复和/或已实施的数据恢复机制。

恢复测试主要检查系统的容错能力。当系统出错时，能否在指定时间间隔内修正错误并重新启动系统。恢复测试首先要采用各种办法强迫系统失败，然后验证系统是否能尽快恢复。对于自动恢复需验证重新初始化（reinitialization）、检查点(checkpointing mechanisms)、数据恢复(data recovery)和重新启动 (restart)等机制的正确性；对于人工干预的恢复系统，还需估测平均修复时间，确定其是否在可接受的范围内。

3.Load Testing负载测试:衡量系统行为增加系统负载（示例：同时工作的用户数量，交易数量）

4.Performance Testing性能测试:测量特定用例的处理速度（CPU）和响应时间，通常取决于增加的负载

5.Stress Testing压力测试:系统行为过载时的观察。

压力测试（Stress Test），也称为强度测试、负载测试。压力测试是模拟实际应用的软硬件环境及用户使用过程的系统负荷，长时间或超大负荷地运行测试软件，来测试被测系统的性能、可靠性、稳定性等。

6.Volume Testing容量测试:观察系统行为取决于数据量（例如：处理非常大的文件）

容量测试是一种非功能性能测试，软件受到大量数据的影响。它也被称为洪水测试。通过增加数据库中的数据量，进行容量测试来分析系统性能。在容量测试的帮助下，当暴露于大量数据时，可以研究对响应时间和系统行为的影响。例如，当有数百万用户下载歌曲时，测试音乐网站的行为。

7.ReliabilityTesting可靠性测试：运行时操作测试来测量故障或故障率之间的平均时间。（系统运行多长时间没有错误？）

8.Robustness Test鲁棒性测试：测量系统对操作错误，编程错误，硬件故障等的响应以及检查异常处理和恢复。

鲁棒性亦称健壮性、稳健性、强健性，是系统的健壮性，它是在异常和危险情况下系统生存的关键，是指系统在一定(结构、大小)的参数摄动下，维持某些性能的特性。例如，计算机软件在输入错误、磁盘故障、网络过载或有意攻击情况下，能否不死机、不崩溃，就是该软件的鲁棒性。

9.Compatibility and Conversion Testing兼容性和转换测试：考虑给定系统的兼容性，数据的导入/导出等

10.Back-to-Back Testing背对背测试：测试系统的不同配置（例如：不同版本的操作系统，用户界面语言，硬件平台等）。

Software Testing Environment Tools

1.Emulators 模拟器

仿真器是在另一个计算机系统（主机）中复制（或模仿）一台计算机系统（客户机）的功能的硬件和/或软件。 全面实施所有功能。仿真器尝试复制设备/系统的内部工作。

2.Simulators 模拟软件

仿真软件是试图（重新）在计算机上创建（现实生活或假设）情况的软件，以便可以研究以了解系统的工作原理。 在大多数情况下：实施所需的功能。

Simulation Types:

Live Simulation:现场模拟：仿真在实时系统中执行

Virtual Simulations:虚拟模拟：模拟在受控时序下执行，时间可以减缓或加速。

Constructive Simulations:建构性模拟：基于事件序列，而不是时序来执行仿真。 不需要可视化。

3.Stubs/Test Stubs测试桩

一个存根是一段代码，用于支持一些其他编程功能。Stub是（一）指定行为（部分模拟）的实现。

For testing, but they do not replace System Testing.

Software Tests might involve testing the software for:

1.Confidentiality 机密性
2.Integrity 完整
3.Authentication认证
4.Authorization 授权
5.Availability 可获得性
6.Non-Repudiation 不可抗拒性

Techniques：

1.Illegal Access 非法访问
2.Data Integrity 数据完整
3.Password Guessing密码猜测
4.Password Cracking密码破解
5.Traffic Interception:交通拦截
6.Patch Management补丁管理
7.Security Awareness安全意识