质量保证与软件测试复习

赵晓琼

基本概念

Basic concept

	Quality Assurance （QA）	Testing
Orientation	Process/Policy Oriented	Products/Deliverables Oriented
Defect Goals	Prevent Defects	Discover Defects
Means	Audit/ Governance	Kinds of Test Methods
Methodology	Objective	Skill and tech to meet the objective

分类

按照测试方法划分

- 白盒测试
  - 语句覆盖：
  - 条件覆盖
  - 路径覆盖
  - ……
- 黑盒测试
  - 等价类法
  - 边界值法
  - ……
- 灰盒测试

按照测试的粒度或测试的范围

- Unit Test
- Integration Test
- System Test
- Acceptance Test

- Smoke Test 与 接受

- Regression Test 与 退化

按照被测软件的评价标准

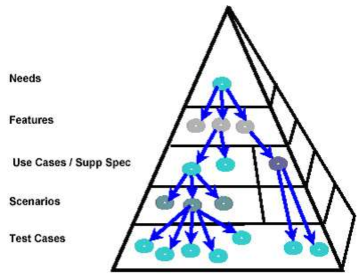
- 功能性测试
- 性能测试
- 安全性测试
- 高可用性测试

α测试和β测试

- α测试
  - 开发团队内部测试
  - 针对软件的各个方面进行测试（功能，性能，安全，可移植，可扩展，可用性）
  - 测试方法较多（黑盒，白盒）
  - 测试目标 [0.1，1.0)
- β测试
  - 最终用户测试
  - 主要针对功能测试
  - 使用黑盒测试
  - 测试目标为有限的production candidate

产出物的内在逻辑1：Traceability

- 文档化是附加产出
  - 也可能是最重要的产出
- 文档化的逻辑结构呈金字塔型
  - 其内在逻辑叫做Traceability

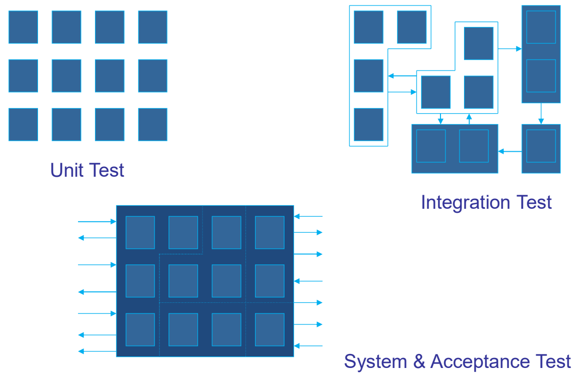


产出物的内在逻辑2 ： Coverage

	UC1				UC2		UC3		
TC	MF	AF1	AF2	AF3	MF	AF	MF	AF1	AF2
TC1.1	●				●		●		
TC1.2	●				●		●		
TC1.3		●			●		●		
TC1.4		●			●			●	
TC1.5		●			●				●
TC1.6			●		●				●
TC1.7			●		●				
TC2.1			●						
TC2.2				●					
TC2.3				●		●			
TC2.4				●		●			

SDP and V-model

Test Techniques – By Granularity



按照测试的粒度或测试的范围

- V-Model

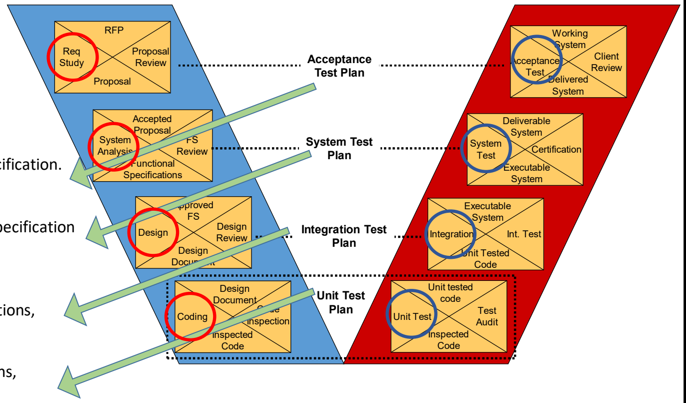
V-Model Documentation

✓ the User Requirements Specification.
✓ （URS）

✓ the System Requirements Specification
✓ （SS）

✓ the System Design Specifications,
✓ （SDS）

✓ Detailed Design Specifications,
✓ （DDS）



V-Model Documentation

- Each document produced is associated with pairs of phases in the model.
- These are the
  - (a) Detailed Design Specifications,
  - (b) the System Design Specifications,
  - (c) the System Requirements Specification,
  - (d) the User Requirements Specification.

V-Model Documentation

- Requirements Gathering produces the User Requirements Specification (URS), which is both the input to Analysis, and the basis for Acceptance Testing.
- Analysis produces the System Specification (SS) – also know as the Software Requirements Specification (SRS) – which is both the input for Software Design, and the basis for System Testing.
- Design produces the System Design Specification (SDS), which is both the input for the detailed Specification phase, and the basis for Integration Testing.
- The Specification activity produces the Detailed Design Specifications (DDS), which are both used to write the code, and also are the basis for Unit Testing.

V-model Advantages

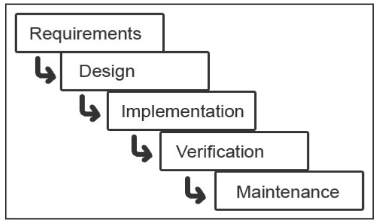
- It is simple and easy to manage due to the rigidity of the model,
- It encourages Verification and Validation at all phases:
- Each phase has specific deliverables and a review process.
- It gives equal weight to testing alongside development rather than treating it as an afterthought.

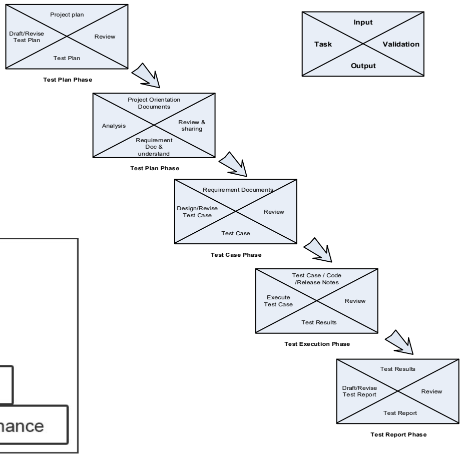
V-model Disadvantages

- Its disadvantages are that similarly to the Waterfall model there is no working software produced until late during the life cycle
- It is unsuitable where the requirements are at a moderate to high risk of changing.
- It has been suggested too that it is a poor model for long, complex and object-oriented projects

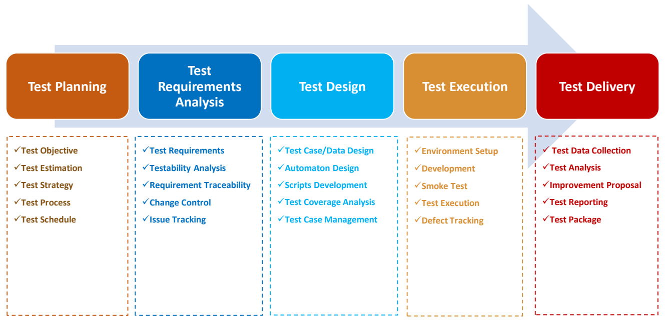
Test process

Waterfall

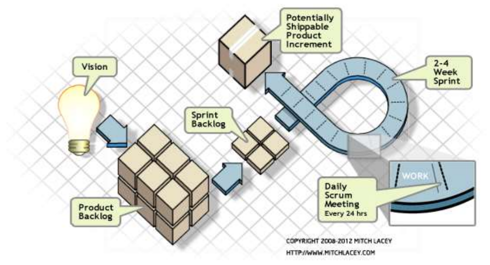




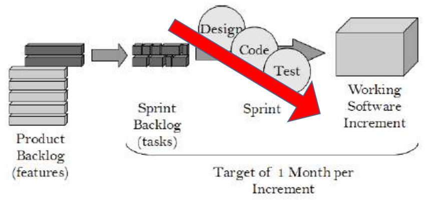
Test process



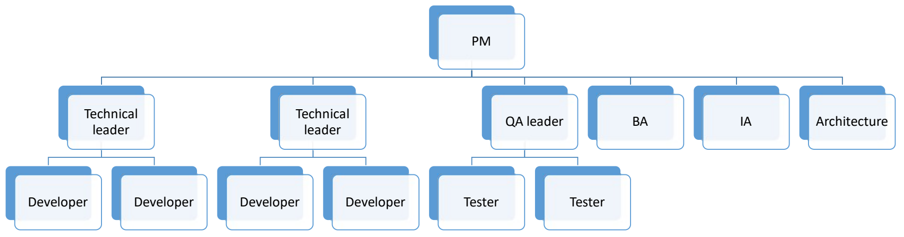
Scrum


Vision
Potentially Shippable Product Increment
2-4 Week Sprint
Sprint Backlog
Product Backlog
Daily Scrum Meeting Every 24 hrs
WORK
COPYRIGHT 2008-2012 MITCH LACEY
HTTP://WWW.MITCHLACEY.COM

- Backlog
- Sprint
- Product increment


Design
Code
Test
Product Backlog (features)
Sprint Backlog (tasks)
Sprint
Working Software Increment
Target of 1 Month per Increment

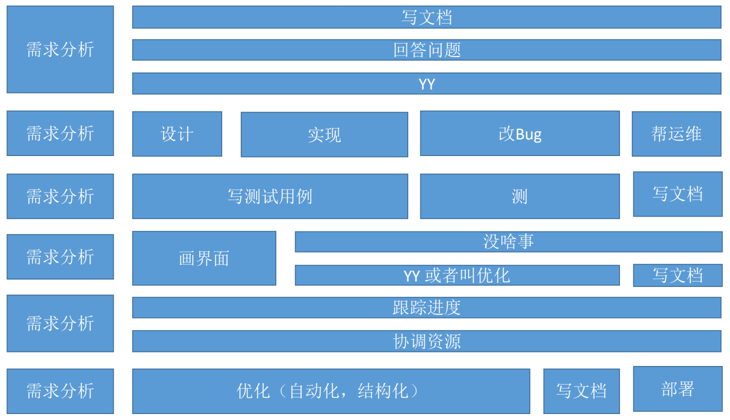
全功能提交团队



22

常态化中的一个迭代

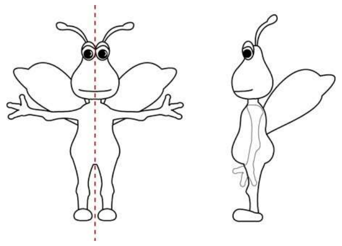
- BA
- DEV
- QA
- IA
- PM
- OP



Bug lifecycle

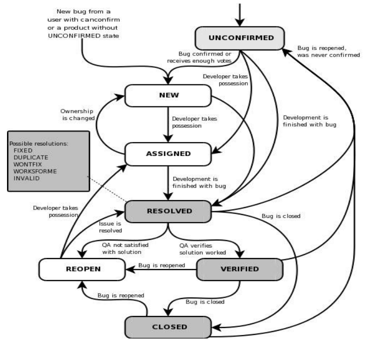
Defect Attributes

- Summary
- Reported By
- Reported Date
- Assigned To
- Description
- Severity
- Component
- Environment
- Release/Build
- Status
- Priority
- Type
- Detection Stage
- Fix Time
- Comment
- ……



Bug Life Cycle

Defect life cycle, also known as Bug Life cycle is the journey of a defect cycle, which a defect goes through during its lifetime. It varies from organization to organization and also from project to project as it is governed by the software testing process and also depends upon the tools used.


New bug from a user with canconfirm or a product without UNCONFIRMED state
UNCONFIRMED
Bug confirmed or receives enough votes
Bug is reopened, was never confirmed
Developer takes possession
NEW
Ownership is changed
Developer takes possession
Development is finished with bug
Possible resolutions: FIXED DUPLICATE WONTFIX WORKSFORME INVALID
ASSIGNED
Development is finished with bug
Developer takes possession
RESOLVED
Issue is resolved
Bug is closed
QA not satisfied with solution
QA verifies solution worked
REOPEN
Bug is reopened
VERIFIED
Bug is reopened
Bug is closed
CLOSED

- •**New -** Potential defect that is raised and yet to be validated.
- •**Assigned -** Assigned against a development team to address it but not yet resolved.
- •**Active -** The Defect is being addressed by the developer and investigation is under progress. At this stage there are two possible outcomes; viz - Deferred or Rejected.
- •**Test -** The Defect is fixed and ready for testing.
- •**Verified -** The Defect that is retested and the test has been verified by QA.
- •**Closed -** The final state of the defect that can be closed after the QA retesting or can be closed if the defect is duplicate or considered as NOT a defect.
- •**Reopened -** When the defect is NOT fixed, QA reopens/reactivates the defect.
- •**Deferred -** When a defect cannot be addressed in that particular cycle it is deferred to future release.
- •**Rejected -** A defect can be rejected for any of the 3 reasons; viz - duplicate defect, NOT a Defect, Non Reproducible.

White box testing and code coverage pattern

Testing and Debugging

- Defect testing and debugging are distinct processes
- Defect testing is concerned with confirming the presence of errors
- Debugging is concerned with locating and repairing these errors
- Debugging involves formulating a hypothesis about program behavior then testing these hypotheses to find the system error

White box testing

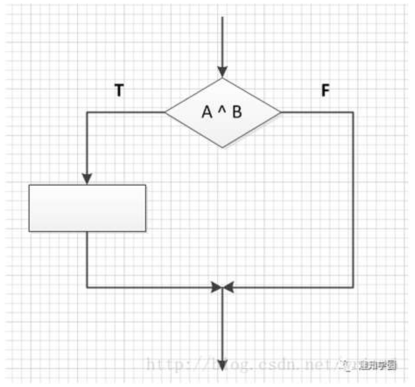
- White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of software testing that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the expected outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).
- White-box testing can be applied at the unit, integration and system levels of the software testing process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for integration and system testing more frequently today. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

White box testing coverage pattern

- In computer science, test coverage is a measure used to describe the degree to which the source code of a program is executed when a particular test suite runs. A program with high test coverage, measured as a percentage, has had more of its source code executed during testing, which suggests it has a lower chance of containing undetected software bugs compared to a program with low test coverage. Many different metrics can be used to calculate test coverage; some of the most basic are the percentage of program subroutines and the percentage of program statements called during execution of the test suite.

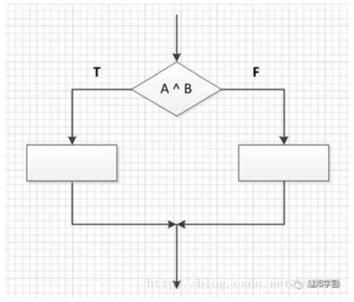
- Statement coverage – has each statement in the program been executed
- Branch coverage – has each branch (also called DD-path) of each control structure (such as in if and case statements) been executed
- Condition coverage (or predicate coverage) – has each Boolean sub-expression evaluated both to true and false

Statement coverage

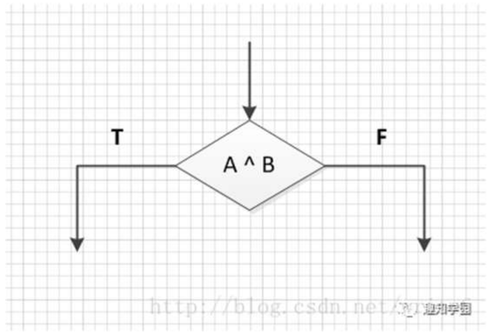


Determine coverage

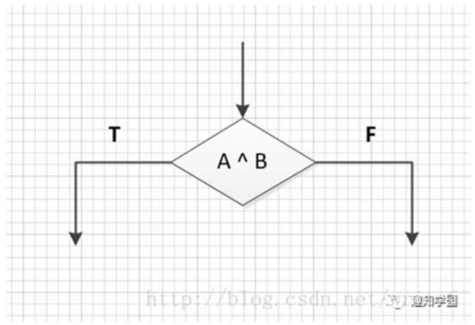
- Branch coverage



Condition coverage



Branch conditional coverage



谢谢