> **Author: MQ_Adventure**
>
> **Duration: June, 2022**

软工复习笔记

Tools to generate JUnit tests automatically
JavaDoc
Cyclomatic Complexity
Martin's Coupling Metric
Reverse Engineering
Tools for Static Analysis
DevOps
Smoke Test
Misuse cases
Protection
git 命令

- Agile (extreme programming, XP)
- Theoretical (Waterfall)
- Formal (RUP, 统一软件开发过程)
- Distributed, open-source （Bazaar）

: Customer, Developer, Coach

： Write stories（用户故事）, planning game（规划游戏）, test-first（测试先行）, pair programming（团队开发）, continuous integration （持续综合）, refactoring（重构）

（产出）: User stories, tests, code

:

- Project initiation （项目初始化）
- Project monitoring and control（项目跟进与控制）
- Software quaity management（软件质量管理）

- Requirements （需求）
- Design（设计）
- Implementation（实现）

- Installation（安装）
- Operation and support（操作支持）
- Maintenance（维护）
- Retirement（软件退役）

- Verification and validation（核实检验）

- software configuration management（软件配置）
- Documentation development（文档）

The application of a systematic, disciplined, quantifiable approach to the
of software.

- Change control (tools: bugzilla, mantis, jira)
- Version control (tools: cvs, subversion, git)
- Building (make, ant, mvn)
- Releasing (Maven Central and Nexus)

跟踪变化以及支持版本回退

使用 SVN 创建本地仓库（ ）

```
svn checkout <address_to_remote> <name_of_local_dir>
```

使用 SVN / git 提交本地更改 ( )

```
svn commit -m "msg"
git commit -m "msg"
```

更新本地仓库 （ ）

```
svn up
git pull upstream master
```

tell svn about a new file to track ( )

```
svn add <name_of_new_file>
```

- svn st (显示当前 svn 目录下的文件状态)
- svn rm （删除被跟踪文件系列中的某一个文件）
- svn mv (移动文件)
- svn diff (比较两个版本的不同)

- Trunk
- Branches
- Tags

- working directory
- staging area
- git directory(repository)

- Modify files in your working directory (modified)
- Stage files, adding snapshots of them to your staging area.(staged)
- Commit (committed)

SVN 是中心化（centralized）管理，Git 是分布式(distributed)管理

- Identification
- Control
- Status accouting
- Audit and review

Baseline is a software configuration item （软件配置项, SCI） that has been , and that can be changed only through formal change control procedures.

Intermediate versions that haven't been reviewed are SCIs but not baselines.

Release is a software configuration item (SCI) that the developers give to other people.

It should be a baseline.

- Smoke Test: ensure that the system still runs after you make a change（确定新的程序代码不出故障，保障整个版本的稳定性）
- Unit Test: Ensure that a module is not broken after you make a change
- Regression test（退化测试）: Ensure that existing code doesn't get worse as you make other improvements.

- Requirements
- Design
- Implementation
- Integration or Testing (Verification)
- Maintenance

Main ideas:

- 不写过多文档
- 逐个实现功能
- 频繁发布代码
- 与甲方紧密联系
- 与队友紧密交流

- Iteration: two week cycle
- Plan each iteration in an iteration meering that is held at the start of the ieration
- Iteration is going to implement set of user stories
- Divide work into tasks small enough to finish in a day
- Each day, programmers work in pair to finish tasks.

A feature customers want in the software, written by customers and not by developers

表达一个 user story 的宏观大小

衡量一个团队进展的速度

- To recover simplicity, you must refactor the code
- To refactor safely, you should have a rigorous set of unit tests.

mistake < fault （程序错误的原因）< failure （程序表现出来的错误）

error 是推导过程，错误与原因分析

Error 是计算、观察或测量值或条件，与真实、规定或理论上正确的值或条件之间的差异，是能够导致系统出现 failure 的系统内部状态。

Failure 是当一个系统不能执行所要求的功能时，即为 Failure

- Assertions（测试想要的结果）
- Sharing common test data among tests（测试之间共享测试变量）
- Test suites for easily organizing and running tests（易于组织的测试套件）
- Test runners（测试程序）

一些测试的方法

- assertTrue(boolean)
- assertTrue(String, boolean)
- assertEquals(Object, Object)
- assertNull(Object)
- Fail(String)

```
public class Cal{
    public long add (int a, int b){
        return a + b;
    }
}

public class calcTest{
    private Cal cal;
    @Test public void testAdd(){
        calc = new Cal();
        assertEquals((long)5, calc.add(2, 3));
    }
}
```

1. Add a test
2. Run all tests and see the new one fail
3. make a little change to code
4. run all tests and see them all succeed
5. refactor to remove duplication

白盒测试

衡量测试有多彻底

Tool: Jacoco

Code coverage cannot ensure test quality

Tool: PIT

can help to generate JUnit tests automatically, it uses to generate and whole test suites towards satisfying a coverage criterion.

project management and comprehension tool

single configuration file that contains the majority of information required to build a project

- Number of people on project
- Time taken, money spent
- Bugs found/ reported
- Bugs fixed, features added

- Size of code
  - Number of files
  - Number of classes
  - Number of processes
- Complexity of code
  - Dependencies / Coupling / Cohesion
  - Depth of nesting
  - Cyclomatic complexity

Number of independent paths through the procedure

Give an upper bound on the number of tests necessary to execute every edge of control graph

= Number of branches(if, while, for) + 1

coupling: 耦合，模块之间的依赖程度

Cohesion: 内聚，模块内部的依赖程度

Afferent coupling: 传入耦合，依赖模块内部的外部模块的数量，体现了模块的职责

Efferent coupling: 传出耦合，依赖于外部模块的数量

Instability: $\frac{Ce}{Ca+Ce}$

- 过多的传出耦合会使得模块不稳定，因为模块容易随着外部依赖项变化而改变
- 过多的传入耦合会使得模块的改变变得非常难，因为可靠性太高

= number of abstract classes in module / number of classes in module

WMC for a class is the sum of the complexities of the methods in the class

Maximum length from a class to the root of the tree.

Number of subclasses

Number of other classes to which a class is coupled

Number of methods in a class or called by a class

The more methods that can be invoked from a class, the greater the complexity of the class

$$LOCM = Num_{don'tshared-instance} - Num_{shared-instance}$$

Forward: From requirements to design to code

Reverse engineering: From code to design, maybe to requirements, analyze an existing system

Reengineering: From old code to new code via some design

Static analysis involves no dynamic execution of the software under test and can detect possible defects in an early stage, before running the program.

Tools:

- Checkstyle (help programmers write Java code that adheres to a coding standard)

- PMD (find potential problems like possible bugs, dead code, suboptimal code, overcomplicated expression, duplicates code)
- FindBugs (concentrates on detecting potential bugs and performance issues)
    - Correctness
    - Bad practise(code that drops exceptions or fails to close file)
    - Performance
    - Multithreaded correctness
    - Dodgy (unused local variables or unchecked casts)

Making sure that is a form of defensive programming

- Failure detection
- Damage assessment
- Fault recovery
- Fault repair

- Record system state at specific events(checkpoints). After failure, recreate state at last checkpoint
- Backup of files
- Combine checkpoints with system log that allows transactions from last checkpoint to be repeated automatically.
- Test the restore software

Execute independent implementaion in parallel, compare results, accept the most probable

- Barriers, separate parts of a complex system
- Authentication（验证） & Autorization（授权）
- Encryption

```
/*
Below is overall description
*Return a synchronized map backed by the given map.
....
Below is block tags
@param map the map to synchronize, must not be null
@return a synchronized mpa backed by the given map
@throws IllegalArgumentException if the map is null
*/
```

Javadoc 从 /** 开始，以 */ 结束

Javadoc 应该描述如何使用类、方法以及构造器等等，但是不应该描述类或者方法内部的工作，也不应该告诉谁来使用方法

Rules for writing summaries: the first line should summarize the purpose of the element, for methods, omit the subject and write in the third-person marrative form. 例如：Finds the first blank in the string.

- Application System Reuse: reusing an entire application by incorporation of one application inside another. Development of application families.
- Component Reuse: components of one application reused in another application
- Function Reuse: reusing software components that implement a single well-defined function

- Quality
- Productivity
- Cost

CBSE is an approach to software development that relies on reuse.

integration conflicts removed by making `code-level` modification to the code

used when component library provides a `component extension language or API` that allows conflicts to be removed or masked.

requires the introduction of pre- and post-processing at the component interface to remove or mask.

- Applications generators for business data processing （商用数据处理应用生成器）
- Parser and lexical analyzers generators for language processing （语言处理字符解析生成器）
- Code generators in CASE tools.（CASE 工具中的代码生成器）
- User interface design tools （用户界面设计工具）

- Procedural abstraction
- Data abstraction
- Control abstraction

coupling: Measure of interconnection among modules. The degree to which one module depend o nothers

cohesion: Measure of interconnection within a module. The degree to which one part of a module depends on another

Frameworks are `sub-system` design containing a collection of abstract and concrete classes along with interfaces between each classes, frameworkk are reusable ent

Sub-system is implemented by adding components to fill in missing design elements and by instantiating the abstract classes.

Continuous integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integration per day. Each integration is verified by an automated build(including test) to detect integration errors as quickly as possible.

- Automate the build: Compile working executable on a daily basis
- Make your build self-testing (Automated tests: Tests that can be run from the command line)

developers -> (code commit) Source Repository -> (Trigger buiild) Continuous Integration Server -> (Deploys application) Web Server

Continuous Delivery = CI + automated test suite

Continuous Delivery 在完成测试之后是手动地部署到应用上面，Continuous Deployment 是自动地部署到应用上面

Continuous Deployment = CD + automatic deployment

Re-running test cases from existing test suites to ensure that software changes do not introduce new faults.

- Confidentiality （可信性）
- Integrity（完整性）
- Availability （可获得性）
    - Access to a system or its data that is normally available may not be possible

- Infrastructure security
    - is concerned with maintaining the security of all system and networks that provide an infrastructure and a set of shared services to the organization（维护基础设施）（实际基础设施的配置与部署）
    - Application security
    - is concerned with the security of individual application systems or related groups of systems. (单个应用层次上的安全)（安全系统的设计）
- Operational security
    - is concerned with the secure operation and use of the organization's system（操作层次上的安全）

Test which could fail or pass for the same code （即有时候成功有时候失败的，对同一份代码的测试）

原因：Concurrency, Environment / setup problems, Non-deterministic or undefined behaviors

- Interception threat: allow an attacker to gain access to an asset（侦听 ）
- interruption threat: allow an attacker to make part of the system unavailable（拦截）
- Modification threat: allow an attacker to tampler with a system asset（允许系统被篡改）
- Fabrication threat: allow an attacker to insert false information into a system（伪造）

- Vulnerability avoidance
- Attack detection and elimination
- Exposure limitation and recovery

- Protection
  - Platform - level protection (对于系统运行平台的控制)
  - Application - level protection （应用程序自身的保护机制）
  - Record-level protection（对于特定记录访问的保护）

  三点综合形成层次保护体系结构（layered protection architecture）
- Distribution

  means that attacks on one system do not necessarily lead to complete loss of system service

- Change control (tools: bugzilla, mantis, jira)
- Version control (tools: cvs, subversion, git)
- Building (tools: make, ant, mvn)
- Releasing (tools: Maven Central and Nexus)

requirement -> design -> implementation ->integration or testing -> maintenance

和 waterfall process 不一样的是，用 collaborative 和 iterative 取代了僵硬的waterfall process.

- 不需要写过多的注释，主要写代码以及测试
- 逐个实现功能
- 频繁发布代码
- 与甲方紧密交流
- 与队友紧密交流

- Planning game for requirements
- Test-driven development for design and testing
- Refactoring for design
- Pair Programming for development
- Continuous integration for integration

mistake < fault （程序错误的原因）< failure （程序表现出来的错误）

error 是推导过程，错误与原因分析

Error 是计算、观察或测量值或条件，与真实、规定或理论上正确的值或条件之间的差异，是能够导致系统出现 failure 的系统内部状态。

Failure 是当一个系统不能执行所要求的功能时，即为 Failure

equal 有 s

1. Quickly add a test
2. Run all tests and see the new one fail
3. Make a little change to code
4. Run all tests and see them all succeed
5. Refactor to remove duplication

Class Coverage > Method Coverage > Branch Coverage > Statements Coverage > Instructions Coverage

Randoop

Evosuite

```
int foo(itn a, int b){
    if(a > 17 && b < 42 && a + > 55){
        return 1;
    }
    return 2;
}
```

2 by Eclipse Metrics Plugin

4 by GMetrixs

5 by SonarQube

Ca: Afferent coupling: measure incoming dependencies

Ce: Efferent coupling: measure outgoing dependencies

Instability: Ce / (Ca + Ce)

Discovering design of an artifact from lower level to higher level

Checkstyle

PMD

FindBugs

Development and IT Operation

A quick set of tests run on the daily build

cover most important functionalities of the software but NOT exhaustive

- Interception threats （Attacker gains access to an asset）
- Interruption threats （Attacker makes part of a system unavailable）
- Modification threats (A system asset is tampered with)
- Fabrication threats（False information is added to a system）

- Platform-level protection （System authentication、System authorization、File integrity management）
- Application-level protection（Database login、Database authorization、Transaction management、Database recovery）
- Record-level protection（Record authorization、Record encryption、Record integrity management）

```
//// 本地版本管理 ////
git commit 提交一个新的版本
git branch <branch_name> 创建一个名字叫做 branch_name 的分支
git checkout <branch_name> 切换到 branch_name 上面
git merge <branch_name> 将 branch_name 分支合并到 main 里
git rebase main 将当前指向的分支以线性的方式合并到 main 当中

git checkout C4 将分离 head 指针指向 C4（一个提交记录哈希值）对应的提交记录
main^^ 代表 main 节点的第二层父节点，为 git checkout 提供了一种相对引用的方式，也可以使用
HEAD^ 一直往上移动
也可以使用 HEAD~4 来说明 HEAD 往上走 4 层。
git branch -f main C6 将 main 分支移动到 哈希值为 C6 的提交记录上

git reset C1 将主分支的更改撤销，回到 C1 提交记录
git revert C1 撤销 C1 提交记录的更改，并将此次撤销作为一次新的提交记录

git cherry-pick C2 C4 将C2 以及 C4 提交记录抓过来放在主干分支下面
但是 cherry-pick 只能针对已知提交记录的哈希值的情况
如果不知道，就可以直接使用 rebase -i 来图形化操作

--------------------------------------------------------------------------------
--------------
// 远程版本管理 ///
git clone 复制远程仓库
git chekout o/main && git commit 来到远程仓库进行提交

git fetch 完成了两步操作：1. 从远程仓库下载本地仓库缺失的提交记录 2. 更新远程分支指针

git pull 就是 git fetch 和 git merge o/main 的缩写！！相当于先下载了远程仓库，再进行了一
次合并
git push 把本地仓库的工作结果推送到远程仓库

如何在远程仓库被修改的情况下推送自己的工作
git fetch && git rebase o/main && git push

git pull --rebase C2 在提交记录 C2 下面

git push 被限制，需要pull request?
通过：
git checkout -b feature 创建新分支
git push 推送到远程仓库
git reset 在本地回到上一层分支

git push origin main 将本地的main 推送到远程的 main当中
```

如果不希望一一对应，则
git push origin local_foo:main 将本地的 local_foo 推送到远程的 main 当中。数据流动方向： local_foo -> main
同理
git fetch origin main:local_foo 将远程仓库的 main 用于本地 local_foo 分支的更新。数据流动方向: main -> local_foo

git pull origin main:local_foo 将远程仓库的 main 用于本地 local_foo 分支的更新，之后将当前的主干分支 merge