#### 软件工程

###### 系统测试（评审）

* 尽早、不断的进行测试。
* 程序员避免测试自己设计的成勋。
* 你要选择有效、合理的数据，也要选择无效不合理的数据。
* 修改后应进行回归测试。
* 尚未发现的错误与该成勋发现的错误数成正比。

###### 测试类型

动态测试：在计算机上实际运行程序进行软件测试，包括黑盒测试、白盒测试。

* 黑盒测试-功能测试-数据驱动。
* 白盒测试-结构测试-逻辑驱动。
* 灰盒测试-介于两者之间。

静态测试：程序不在机器上运行，采用人工检测代码质量，保罗桌前检查、代码走查、代码审查。

* 桌前检查-程序员自查。
* 代码走查-程序员与架构师共同检查。
* 代码审查-正规审查/轻量级审查（非正式）/结对编程。

###### 测试用例

驱动模块--前--被测模块--后--桩模块。

黑盒测试用例：

* 等价类划分：把输入字段内容划分不同类型，每个类型用一个代表性的输入字段即可。两个测试用例的设计原则：①设计一个新的测试用例，使其更多的覆盖有效等价类。②设计一个新的测试用例，使其覆盖一个尚未被覆盖的无效等价类。
* 边界值分析：把参数范围内的参数取边界值，测试极端参数的情况。
* 错误推测：凭经验、直觉测试。
* 因果图：由结果反推原因。

白盒测试用例（基本路径测试、循环覆盖测试、逻辑覆盖测试）：

* 语句覆盖SC：逻辑代码里面所有语句都执行一遍，覆盖层级最低。
* 判定覆盖DC：判断语句条件真假分支都覆盖一次。
* 条件覆盖CC：每个判断条件每个独立条件都要执行一遍真和假（一个判断里面可能有多个条件）。
* 条件判定组合覆盖CDC：同时满足判定覆盖和条件覆盖。
* 路径覆盖：把所有可行路径都覆盖，覆盖层级最高。

###### 测试阶段

* 单元测试：针对一个小的功能模块测试。测试依据是软件详细设计说明书。
* 集成测试：检查模块之间的问题，测试依据是软件概要设计文档。
* 系统测试：目的是验证软件是否和系统有联系，偏向于性能测试。
* 确认测试：验证软件系统的各个功能是否有问题，测试依据是需求文档。
* 内部确认测试：由软件开发内部人员测试，按SRS（需求规格说明书）进行测试。
* Alpha测试：用户在开发环境下测试。
* Beta测试：用户在实际使用环境下测试。
* 验收测试：交付前以用户为主进行的测试。
* 回归测试：针对改正的bug进行测试，测试bug是否改掉，并且测试bug是否导致了其他的bug。

###### 非功能性测试（负载测试、强度测试、容量测试）

* 恢复测试。
* 安全性测试。
* 压力测试。
* 性能测试。
* 可靠性测试。
* 可用性测试。
* 可维护性测试。
* 安装测试。

###### 测试策略

* 自底向上：从底层模块开始测试，最后完成整个系统的测试。
* 自顶向下：先测试整个系统，再往下测试单个模块。
* 混合式：既有第一种，也有第二种。缺点是测试工作量大。

###### 面向对象的测试

* 算法层(单元测试)：包括等价类划分测试、组合功能测试（基于判定表的测试）、递归函数测试、多态消息测试。
* 类层（模块测试）：包括不变式边界测试、模态类测试。
* 模板层/类数层（集成测试）：包括多态服务测试和展平测试。
* 系统层（系统测试）。

###### 测试自动化优点

效率上：

* 提高测试执行的速度。
* 提高运行效率。
* 保证测试结果的准确性。
* 连续运行测试脚本。
* 模拟现实环境下受约束的情况。

成本上：

* 所有测试活动都可以自动完成。
* 减少人力成本。
* 可以免费获得降低测试工作量。

###### 软件调试

调试是找出错误的代码和原因

软件调试方法：

* 蛮力法：主要思想是”通过计算机找错”，低效，耗时。
* 回溯法：从出错处人工沿控制流程往回追踪，直至发现出错的根源。复杂程 序由于回溯路径多，难以实施。
* 原因排除法：主要思想是演绎和归纳，用二分法实现。

软件调试与测试的区别：

* 测试的目的是找出存在的错误，而调试的目的是定位错误并修改以修正错误。
* 调试是测试之后的活动,测试和调试在目标、方法和思路上都有所不同。
* 测试从一个已知的条件开始,使用预先定义的过程，有预知的结果；调试从一个未 知的条件开始,结束的过程不可预计。
* 测试过程可以事先设计,进度可以事先确定;调试不能描述过程或持续时间。

###### 验证与确认

* 验证是指在软件开发周期中的一个给定阶段的产品是否达到在上一阶段确立的需求过程。
* 确认是指在软件开发过程结束时对软件进行评价以确定它是否和软件需求相一致的过程。
* 测试是指通过执行程序来有意识的发现程序中的设计错误和编码错误的过程，测试是验证和确认的手段之一。

###### 软件度量

软件的两种属性：

* 外部属性：面向用户的属性，一般是性能指标。可直接测量。
* 内部属性：软件本身的属性，如可靠性等。只能间接测量。

McCabe度量法：又称为环路复杂度，假设有向图中有向边数为m,节点数为n,计算有向图的环路复杂度为m-n+2。

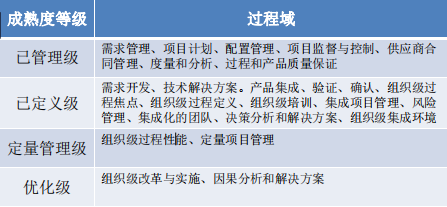
###### 软件质量

质量保证与质量控制（QC/QC）(关注过程/关注结果)（事中控制/事后检查）

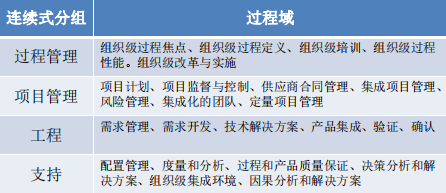
* 质量保证一般是每隔一定时间（例如每个阶段末）进行的，主要通过系统的质量审计和过程分析来保证项目的质量。
* 质量控制是实时监控项目的具体结果，以判断他们是否符合相关质量标准，制定有效方案，以消除产生质量问题的原因。
* 一定时间内质量控制的结果也是质量保证的是质量审计对象，质量保证的成果又可以指导下一阶段的质量工作，包括质量控制和质量改进。

###### 软件过程改进（CMMI）

* 阶段式--组织能力成熟度。



* 连续式--软件过程能力。



###### 配置管理

* IEEE对配置下的定义为硬件、软件或二者兼有的集合，为配置管理指定的，在配置管理过程中作为一个单独的实体对待，可作为配置项管理的有：外部交付的软件产品和数据、指定的内部软件工作产品和数据、指定的用于创建或支持软件产品的支持工具、供方/供应商提供的软件和客户提供的设备软件。
* 典型配置项，包括项目计划书、需求文档、设计文档、源代码、可执行代码、测试用例、运行软件需要的各种数据，他们经评审和检查通过后进入软件配置管理。
* 开发库（动态库，程序员库，工作库；动态系统，开发者系统，开发系统，工作空间）。
* 受控库（主库、系统库；主系统、受控系统）。
* 产品库（备份库、静态库、软件仓库；静态系统）。

###### 软件开发环境与工具

软件开发支持环境、软件生存期支持环境。

环境机制：

* 环境信息库。
* 过程控制和消息服务。
* 用户界面规范。

工具集：

* 系统规划工具。
* 建模工具。
* 分析设计工具。
* 编程工具。
* 测试工具。
* 项目管理工具。

###### 系统转换计划

遗留系统：计算修改和演化也不能满足新的业务需求的信息系统。

特点：业务不能满足需求，技术落后，大型软件系统维护工作困难，很难管、开发、理解。

新旧系统的转换策略：

* 直接转换策略-成本低，风险高-小系统。
* 并行转换策略-成本高，风险低-一般系统。
* 分段转换策略-区域：试点-推广；大型系统：子系统-逐步转换。

数据转换与迁移：旧数据库-抽取-转换-装载-新数据库（ETL）三种方式：

* 系统切换前通过工具迁移。
* 系统切换前采用手工录入。
* 系统切换后通过新系统生成。

###### 系统维护

软件维护是生命周期的个完整部分，可以将软件维护定义为需要提供软件支持的全部活动，这些活动包括在交付前完成的活动，以及交付后完成的活动。交付前完成的活动包括交付后运行的计划和维护计划等;交付后的活动包括软件更改,培训、帮助 资料等。

系统运行和维护包括：硬件维护、软件维护、数据维护。

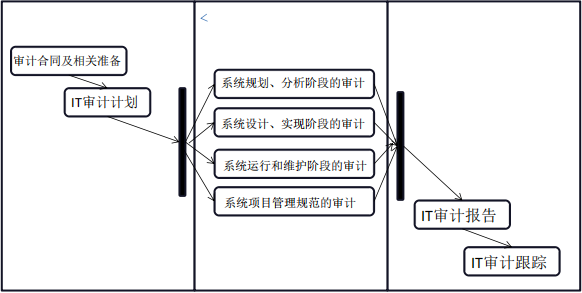
系统的可维护性：为维护人员理解，改正，改动和改进这个软件的难易程度。

系统评价指标：易分析性、易改变行、易测试性、稳定性。

软件维护类型：

* 改正性维护（25%）：正确性，发现bug而修改。
* 适应性维护（20%）：由于外部环境，被动惊醒软件的修改和升级。
* 完善性维护（50%）：基于用户主动对软件提出更多需求。
* 预防性维护（5%）预防未来可能会发生的bug。

###### 系统审计



###### 系统评价

系统性能评价、系统效益评价、系统建设评价。

系统评价流程：

* 确定评价对象，下达评价通知书，组织成立评价工作组合专家咨询组。
* 拟定评价工作方案，收集基础资料。
* 评价工作组实施评价，征求专家意见和反馈给企业，撰写评价报告。
* 评价工作组将评价报告报送专家咨询组复核，向委托人送达评价报告和选 择公布评价结果建立评价项目档案。

系统评价分类：

* 立项评价：系统开发前的评价，分析是否立项开发，做可行性分析。
* 中期评价：项目开发中的阶段评审。
* 结项评价：系统投入使用后，了解系统是否达到预期要求。

#### 软件架构设计