前言

国际软件测试认证委员会（International Software Testing Qualifications Board， ISTQB）从 2002 年建立至 2009 年初已经有 40 多个国家分会组织，超过 11 万认证测试工程师。作为从 2006 年中国分会（CSTQB）建立以来，由于其在世界范围不断扩大的影响力，吸引了越来越多软件测试从业人员参加培训和考试。但由于考试大纲没有英文板，也缺乏足够的中文参考资料，给很多不习惯英文阅读者带来困难。

笔者结合自身参加基础级和高级考试的经历，多年软件测试经验，以及广泛吸取英文参考书的内容，编写此书，帮助更多有想法但缺乏参加培训条件的测试人员通过自学轻松通过考试拿到证书，为自己的职业生涯添加砝码。

如何阅读本书

本书面向两类读者，一类是或多或少有了一定的软件测试相关经验，有志于通过考试获取 ISTQB 认证；另一类是想通过自学对软件测试领域有一个系统的认识，了解和熟悉标准的测试概念，过程和技术。

本书共有两篇十六章。其中基础篇面向在准备 ISTQB 基础级认证的读者或踏入软件测试领域不久，希望较系统地学习软件测试基本理论的从业人员；进阶篇则更适合于参加高级认证考试和有比较丰富经验并希望通过学习 ISTQB 高级大纲完善理论提高解决具体问题能力的专业人员。全书严格按照 ISTQB 大纲（2010 年版）组织，斜线内容是直接翻译自大纲原文，关键词汇或定义附英文原文；带下滑线的内容是需要记忆的概念或定义。在每章后附带仿真例题，读者可以先自行尝试解决，检验学习的效果，然后再核对答案和分析。

由于语言文化上的差异和翻译水平的限制，直接翻译的文字（斜体字）较为生涩，读者不必为难以理解这部分内容而担心，本书会提供足够的内容和例子解释这些概念。

ISTQB 在中国范围内采用中文试卷。为了保持和中国分会（CSTQB）专业词汇翻译的一致性，全书尽量按照 CSTQB 提供的翻译术语。如该表中的某些翻译与行业通行的叫法有差异，书中会专门指出。书中为重要词语，概念和定义提供中英文对照，是出于两个考虑。第一是 ISTQB 的初衷之一是在世界范围内推行较为统一的术语，纯中文的材料和考试从一定程度上违背了此目标。所以让读者掌握统一术语的中英文也是考试学习的目标之一。第二是软件行业特别是测试还是相对新鲜的领域，很多语汇缺乏标准翻译，生硬的翻译晦涩难懂，不如英文原文来得形象，或不同概念对应相同的中文词语，造成歧异。另外虽然本书覆盖了大纲所有内容，有一定英文基础的读者仍应该将大纲原文作为最重要的参考材料和本书一起对照阅读，对于理解一些原理和概念大有好处。

水平所限难免有疏漏错误，衷心希望读者批评指正。

如有疑义请与作者联系：goal1860@hotmail.com

基础篇

1.1 测试基础

1.1 为什么测试如此重要（K2）

术语：缺陷（bug），缺陷（defect），错误（error），失败（failure），故障（fault），质量（quality），风险（risk）。

1.1.1 软件系统背景（K1）

软件系统在我们的生活中快速发展，从商用系统到日常消费都能看到。许多人都有这样的经历：一些软件的运行结果不完全符合我们的预期。这会引起很多后果，比如经济，时间和商业信誉上的损失，更严重的可能导致人员伤亡。

中心就是：软件系统总是存在着质量问题。虽然是 K1 的知识点，实际并没有什么真正需要记忆的。从考试的角度来说需要掌握的是：

- 了解什么是软件系统
- 常见的有那些软件系统
- 软件系统工作不正常的危害

对于软件测试人员来说这十分简单，就不多赘述了。

1.1.2 导致软件缺陷的原因（K2）

*任何人都可能**犯错**（**error**），在代码，软件系统或文档中产生**缺陷**（**defect, fault, bug**）。假如这个缺陷被执行到了，系统就没法完成应该做的是或者做不应该做的事，导致**失败**（**failure**）。有些软件，系统或文档中的缺陷能导致失败，但并不都会。*

这段话看似简单实则包含了一组相互联系的重要概念。加粗的三个术语是互相有逻辑关系但又不同的概念。错误可能转化为缺陷，也可能不会。缺陷可能导致系统失败或失效，也可能不会。举个例子：

有一行代码： if a>0 then do…，程序员犯了**错误**，写成了 if a>=0 then do…，但是由于某些外部限制，a=0 的情况不可能出现，所以这个错误也就不具备变成**缺陷**的条件。另有一行代码：if a=0.83975, then do…，程序员犯了**错误**，写成了 if a=0.93975, then do…，并且输入值完全有可能是 0.83975 或者 0.93975，所以就具备条件成了**缺陷**，但是由于出现该输入值的几率非常之小，以至于一直都未发生过，也就不能成为**失败或失效**。

这组概念常考，不难，但容易被忽视。

缺陷的产生是因为软件的制造者人类本身就是会犯错的，另外还有一些客观原因比如时间紧，代码复杂，基础件的复杂，技术更替或者一些系统原因。

说了一大堆，最终还是人本身的局限性，有谁能保证不犯错呢？

实效也可能源自环境条件：比如射线，电磁场，污染导致硬件条件的变化，进一步影响软件的执行。

随着硬件工艺的发展，这种情形在现实中出现得较少。但并不是不可能，而且一旦出现都可能是灾难性的。这就要求对工作场所特别是重要服务器，存储设备的存放地点做严格控制管理，无论温度湿度电磁场以及发生自然灾害的可能性都要考虑周全。

1.1.3 软件测试在软件开发，维护和运作中扮演的角色（K2）

严格的系统和文档测试有助于减少问题发生的风险，发布前发现缺陷可以提高软件系统的质量。

软件测试还可以确保软件遵守法规要求和行业标准。

1.1.4 测试和质量（K2）

有软件测试的帮助，就能用发现缺陷的多少来度量软件的质量，这对于软件的功能性和非功能性需求或特征（可靠性，可用性，效率，可维护性和可移植性）都同样有效。了解非功能性测试请阅读第二章，了解软件特征请阅读“软件工程-软件产品质量”（ISO9126）

这里提到了软件测试的第一个作用：量化软件的质量。有一个问题经常摆在软件工程师或项目经理面前：你们做的软件到底质量如何。定性的评价通常比较困难，就需要有具体的数字来衡量。什么度量数值可以客观的反映软件的质量？缺陷数量虽然不是唯一的指标，但却是最容易的。很容易理解，一个很容易找到 bug 的软件质量肯定有问题。这里提及了一些软件质量特征，会在后续章节详细介绍。

当软件测试只能找到很少或根本没有缺陷时，我们就能对该软件有足够的信心。设计合适的测试通过大大降低了该系统的风险。即便有缺陷发现，修复这些缺陷也能提高软件的质量。

挖空心思却找不到缺陷的软件当然让人放心。这是产品经理梦寐以求的目标。

要从以往项目中吸取教训。从对过往缺陷的分析，可以帮助我们不断改进开发过程，在未来的版本或产品中避免类似问题的出现，从而提升质量。这是质量保证的一个重要内容。测试应当和某项质量保证活动结合起来。

总结以上，软件测试有三点主要作用：提供质量度量（Measure Quality），提供软件产品信心（Provide Confidence），以及提供过程改进的依据（Improve Process）。理解和记住这三点就足够了。

1.1.5 多少测试才足够？（K2）

到底要进行多少测试取决于风险程度和项目约束条件。风险包括商业风险和技术风险。项目约束条件包括了诸如时间和经费等。

软件测试应当为相关方提供足够的信息来决定是否发布，是否进入下一个开发环节或递交给客户。

一句话，测试做到什么程度算完没有一个固定答案。只要满足两个显式条件和一个隐含条件就要一直进行。显式条件 1：项目有多大风险，人命关天的系统当然要测试测试再测试，自己写着玩的程序就没那么严格了。2：项目有多少钱让你折腾。时间=金钱。所以说到头还是钱。这也是通常为什么大公司做的软件产品可靠性要高一些。他们烧得起钱，做这个测试，那个测试。隐性条件：老板们从当前的测试结果已经获得了足够的信心，或者彻底摧毁了信心。只要他们还在犹豫咱就得继续干活。其实大家都是在按照这几条做事，只不过有的能在开始之前就计划好，有的则走一步看一步。

1.2 什么是测试？（K2）

术语：调试（debugging），需求（requirement），评审（review），测试用例（test case），测试（testing），测试目标（test objective）

背景：对于测试（testing）的一般理解是运行测试（tests），比如执行一个软件。这只是测试的一部分，但决不是测试活动的一部分。

注意第一个“测试”是动名词，指的是测试的过程，第二个测试是名词，可以是测试用例，也可以是一个单次测试。这在中文里很容易混淆。

测试活动存在于测试执行前与执行后：比如测试计划和控制，选择测试条件，设计测试用例，和检查测试结果，评估出口准则，测试报告和结束活动。测试也包含了文档（包括代码）评审和静态分析。

这里虽没有明确指出，但显然“检查测试结果”之前的内容都是在执行前发生的，而包括“检查测试结果”的其它活动都是在执行之后。这里出现了一系列关于测试过程的概念，要在后续章节中详细介绍，但读者不难从字面上做出初步理解。最后加下划线的一句请牢记于心，评审也是测试！静态分析也是测试！

动态测试和静态测试作为不同的方法来达成类似的目标，并为被测系统本身和开发流程提供持续改进所需的信息。

再次强调了静态测试和动态测试同等重要。

软件测试有以下目标：

- *发现缺陷*
- *获取信心和提供信息*
- *防止缺陷*

以上目标和 1.1.4 中软件测试的作用一一对应。

在开发周期较早阶段（通过测试设计来验证测试基准）开始考虑测试设计有助于防止缺陷进入代码。

测试基准（test basis）是能提供测试依据，系统需求的文档，也是整个开发流程的源头。需求中的缺陷一般指的是歧义，矛盾，缺乏可测试性等问题，也可能是对真正需求的误解。后者具有很大隐蔽性，一旦进入了编码阶段一般要到后期，甚至验收测试阶段才发现，危害不言而喻。由测试组和领域专家共同在早期对这些文档进行评审固然能发现一些错误，但是也很容易忽视细节。而测试设计是一个很严谨的过程，很容易发现细节上的问题。比如有如下系统需求：

“此系统用户密码加密存储且在数据库中相同密码显示的密文不能相同。”此需求从安全的角度来看很合理，描述也很清晰。但当你设计测试用例的时候就会发现问题了：对应于一个密码来说，期待的加密结果是什么呢？如何测试来保证相同密码加密后不会相同呢？

*不同角度的测试目标也是不同的，比如开发阶段测试（组件，集成和系统测试）的目标是尽可能多找到缺陷，以便尽快修复。而验收测试则是证明开发的系统符合预期，对于系统符合需求增添信心。有时候测试的目的只是评估软件的质量，并无意于修复缺陷，作用仅在于为相关方提供评估发布时间的信息。维护测试的目标是测试代码改动没有引入新的缺陷，而**运行测试**的目标则可能是评估系统的某些特征，比如可靠性和可用性。*

这里说了那么多，只是为了说明并不是所有测试都是相同目的的。目的不同就会有心态不同。出现了一些新概念，比如**运行测试**，指的是在实际环境中运行操作。这些概念在后续章节或高级大纲中会涉及，这里大致了解即可。

*调试和测试是完全不同的。测试是为了展现因缺陷（defect）而产生的系统失效（failure），而调试是一种开发活动，是为了发现缺陷原因，修复代码以及确认缺陷已经被修复。接着测试人员进行**确认测试**，保证系统失效确实被修复了。两种活动的职责是截然不同的：测试人员测试而开发人员调试。*

这段话貌似废话，但其实是有一定针对性的。我们常鼓励测试人员熟悉系统结构和代码，尽可能精确地定位缺陷，调试能力似乎也是白盒测试的一个标志，但有一定根本性的东西不能忘，测试人员的天职是发现系统失效和缺陷。在实际项目中测试人员的配备往往不足，他们的时间捉襟见肘，应当首先用在刀刃上，尽快发现和报告缺陷。

1.3 七大测试原则

术语：穷尽测试(Exhaustive testing)

测试原则

前人在差不多 40 年的时间内总结出了一些测试原则，对于各种测试都同样有效。

经典啊！都要烂熟于心才行。

原则 1 – 测试只是展示缺陷

测试只能表明缺陷存在，却不能证明没有缺陷。测试能降低未发现缺陷留存的概率，却不能证明软件是绝对正确的。

正如某些数学命题，你可以穷举 1-n，证明其正确，却依然无法证明对于 n+1 仍然正确。

原则 2 – 穷尽测试是不可能的

测试所有的输入和条件组合是不可能的，除非是极其简单的情况。可以取而代之的是基于风险和优先级的测试。

当不懂装懂的老板要求你彻底测试一个软件的时候，这是你反驳的最好支持，当然要说的委婉一点。

原则 3 – 早期测试

要较早发现缺陷，就要在软件周期尽可能早的时候开始测试，而且要专注于已定义的测试目标。

尽早开始测试！这句话估计早就把大家的耳朵磨起茧了。为什么要早？因为越早发现问题，解决的代价就越小。

原则 4 – 缺陷簇生

要对缺陷发现率高的模块投入更多的测试。少量的模块往往隐藏了大部分的缺陷。

这不仅仅是所谓的物以类聚。缺陷发现率高的模块往往于需求不清，设计不当，编码复杂度高等内在原因关联，所以从风险的角度来看必然较高，多花些时间绝对值得。

原则 5 – 杀虫剂悖论

相同的测试再重复多次后就无法再找到缺陷了。要克服“杀虫剂悖论”，测试用例要不断评审修改，不断添加新的和不同的测试，就有可能找到更多缺陷。

随着对系统的加深理解，必然会有更多的测试用例产生。另外缺陷本身也是新用例的很好来源。

原理 6 – 测试是上下文相关的

*测试在不同上下文环境中的执行是不同的。比方说**安全关键系统**（safety critical system）和电子商务网站的测试方法就有很大不同。*

这个原理相对难理解。这里其实强调的是不能用相同的态度和手段来测试不同类型的系统。安全关键系统的概念要到高级大纲中才出现，指的是对系统安全要求苛刻的系统，较之一般的电子商务系统的测试要求更为严苛。

原理 7 – 无错谬论

假如建立的系统不稳定或不能满足用户需要和期望，那么发现和修复缺陷就毫无帮助了。

缺陷数量往往用来评估某软件的质量，但要是系统本身背离了用户要求，那就算缺陷再少也没用，因为没有人会去用它。所以测试时要注意验证（verification）和确认（validation）的区别。需求规格说明和其他文档只是需求的不完全载体。文字说明必然有遗漏和偏差，而各人的理解更有可能出错。要不断通过各种途径保证所生产的的确就是用户需要的。常用的方式就是邀请领域专家或用户尽可能多地参与到开发活动来，特别是需求评审和演示（Demo）。

1.4 基本测试过程

术语：确认测试（confirmation testing），再测试（re-testing），出口准则（exit criteria），事件（incident），回归测试（regression testing），测试依据（test basis），测试条件（test condition），测试覆盖（test coverage），测试数据（test data），测试执行（test execution），测试日志（test log），测试计划（test plan），测试归程（test procedure），测试方针（test policy），测试套件（test suite），测试总结报告（test summary report），测试件（testware）

*背景：对于测试来说最显而易见的部分是测试执行，但是为了让测试工作更有效和高效，**测试计划（test plan）**也应该包括用于**测试计划（planning tests）**，用例设计，测试准备和评估结果所需要的时间。*

第一个测试计划是计划文档，而第二个测试计划是过程。

基本测试过程由以下测试过程构成：

计划与控制

分析与设计

实施与执行

评估出口准则和报告

测试结束活动

基本测试过程的五个主要活动是基础级考试的必考内容，需要记忆并理解每个活动包含的具体内容。接下来会展开介绍，所以请先记住这五项活动。

以上活动尽管在逻辑上是顺序发生的，但是也会有重叠和并行的情况。经常需要根据项目和系统具体情况进行裁剪。

请注意两个问题。第一是虽然说是五项活动，其实其中四项是两个活动的合并，所以要明白在基本测试过程中其实包括了九项活动。第二点必须注意的是虽然这里的知识点都是 K1，即识记，但死记硬背定义没有意义，要联系自己的项目经验来判断具体的活动应该归到哪一类中。另外顺序很重要，图 1 可以帮助读者理解整个过程中相互间的顺序和关系。

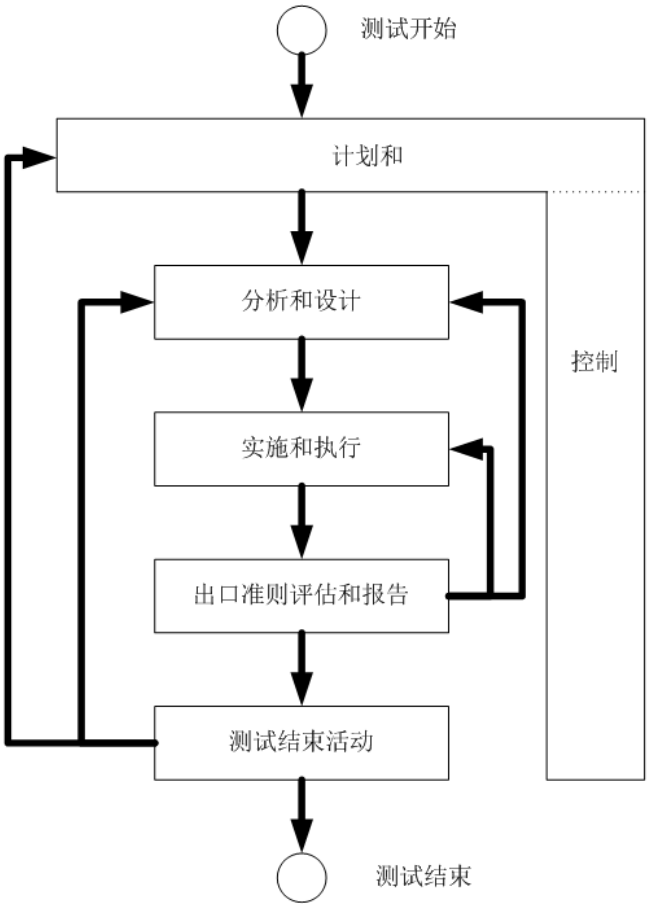

图 1 基本测试过程

基本测试过程是各种测试过程的简化模型，其重要性不言而喻。基本测试过程在 ISTQB 高级大纲部分会被再次提及，并且更加深入讨论。

1.4.1. 测试计划和控制（K1）

测试计划是定义测试目标及测试活动规格说明以满足特定目标和使命的过程。

没有必要去记忆如此复杂的定义，其实计划就是计划，我们首先要知道这是一个过程，而不是完成一份计划文档，需要所有相关人员的参与并达成共识，否则提交的文档没有任何价值。有一些快速开发的小型项目没有翔实的计划文档，但所有项目成员在一起进行了细致的讨论，确定了项目开展的细节，那这样的计划活动效果要远好过测试经理闷头熬夜赶出来的计划文档。

要计划的内容有测试方法，人员，策略，时间表，度量等等。简言之，计划应当包含所有与测试执行相关的信息，但是切忌大而空，内容要有用，提供的评估和度量要具有可行性。有人把计划总结为：什么人，在什么时间内，根据什么，做什么。个人认为总结还是颇为精辟的。最后提交的测试计划文档包含计划的结果。IEEE829 测试计划规格说明书提供了指导性建议。ISTQB 基础级考试并不要求太深入了解，有兴趣可以在互联网查阅相关文件。

测试控制是持续比较实际进展和计划的活动，并报告状态，比如实际和计划的偏离。这也包括采取一定的措施来符合项目使命和目标。为了能有效进行测试控制，我们需要在整个项目周期内对测试活动进行一定监控。测试计划也应当纳入测试监控的反馈。

用更加易于理解的话来说，测试控制就是持续追踪测试活动的进度，及时发现与计划的偏差，并采取措施消除偏差或修改计划。

第五章将详细定义测试计划和控制相关的具体任务。

1.4.2. 测试分析和设计（K1）

测试分析和设计是将抽象测试目标转化为具体有形测试条件和测试用例的过程。

测试分析和设计有以下主要任务：

- *评审测试依据（test basis）（例如需求文档，软件完整性级别（风险级别），风险分析报告，构架设计，设计，界面规格说明书）。*
- *评估测试依据和目标的可测试性（testability）*
- *基于测试项，规格说明，行为和软件结构进行分析并确定测试条件（test conditions）以及优先级*
- *设计高级测试用例（high level test case）并确定优先级*
- *确定必要的测试数据（test data）来支持测试条件和测试用例*
- *设计测试环境（test environment）并确定所需的基础设施（infrastructure）和工具*
- *建立测试依据和测试用例间的双向可追溯性（traceability）*

象这类知识点的学习目标是给出某项具体的任务，能够确切地对应到五项主要活动之一。比如一说“编写测试用例”，就是属于分析设计阶段的任务。接下来对以上任务进行简单解释。要注意这些任务间并非是绝对相互独立的，很多是互相依赖包含的。

测试依据在很多时候指的就是需求文档，毕竟需求是一个开发项目的“根”。设计文档能帮助测试者更好地了解系统细节，但必须注意设计本身未必是正确的。

可测试性分析在测试分析阶段很重要，但很容易被忽视。有些看上去很清楚的需求说明从测试角度分析后就能看出很多问题。1.2 中提到的例子就很说明问题。

接下来就是分析测试条件了。测试条件离测试用例已经不远了。事实上在很多快速开发的项目中测试条件代替了测试用例的作用。

高级测试用例不含具体步骤和测试数据，只是测试条件的细化而已。

最后说一下什么是双向可追溯性。测试用例来源于需求，本身有个对应关系。我们必须要维护这种对应关系，使得任何测试都有据可依。一旦需求变化了测试用例也要跟着变化，并且要很容易地定位到具体的需求描述。为了达到这个目标，我们就要对需求项和测试用例进行编号管理，并指明相互间的映射关系。有时我们想知道某个测试用例所对应的需求描述，有时我们又关心某个需求项是否有足够的测试覆盖到了，这就是一种双向关系了。

1.4.3. 测试实施和执行（K1）

测试实施和执行是结合测试用例确定**测试规程**（test procedure），**脚本**（test script）以及执行顺序，建立测试环境并运行测试的过程。

可以把该定义分成两部分来理解：测试实施是搭环境，选用例，而测试执行自然就是所说的“跑测试”了。这里要注意的是测试规程，脚本和用例的区别。测试用例可以是高级（抽象）的，也可以是具体可以遵照执行的。而测试用例中可以遵照执行的动作序列就是测试规程。测试脚本在不同语境下有不同含义，有时泛指自动化和手工脚本。在这个定义里显然是要区别于手工测试脚本，因而是特指自动化脚本。

测试实施和执行包含以下主要任务

- *确定，实施测试用例（包括确定测试数据），以及制定优先级*
- *开发测试规程以及制定优先级，创建测试数据，和准备**测试用具**（test harness）及自动化脚本（非必须）*
- *根据测试规程创建测试套件以达到高效测试的目的*
- *验证测试环境搭建正确*
- *验证和更新测试依据和测试用例间的双向可追溯性*
- *根据预定的执行次序使用手工方式或工具执行测试规程*
- *记录测试执行的输出，被测试软件的版本或标示，测试工具和**测试件***
- *比较真实结果和预期结果*

- *将（真实结果和预期结果）比较的差异作为**事件**来报告，并且分析其原因（比如代码缺陷，测试数据，测试文档或测试的错误执行方法等）*
- *重复执行引起执行差异的测试活动，比如**重测试**上次失败的测试来确认缺陷是否被修复了（**确认测试**），执行改正后的或原有的测试以确定修复缺陷的代码没有引入或揭示新的缺陷（**回归测试**）*

这里列出的大多数任务都是很显而易见的，但是有些名词和概念需要着重解释一下。

测试用具（test harness）并不是一个很正式的术语，但是在英语国家里被广泛使用。它一般是针对自动化测试的，指的是一个执行环境或框架，开发出的测试用例可以独立地在其之上运行。而测试用具的设计本身也是测试框架设计和测试构架设计的过程。

维护测试追溯性往往会被忽视掉，在测试最后阶段来做这个事情只能是事倍功半。很多项目之所以测试管理混乱就源自于此。选择合适的工具或平台可以极大降低这项工作的复杂度。

最后一项任务的描述其实是解释了三个概念之间的关系：**重测试**（re-testing）等同于**确认测试**（confirmation testing），是执行之前失败的相同测试；**回归测试**（regression testing）要进行更多的测试以避免新代码带来的新缺陷。这几个概念无论考到的概率还是在工作面试中出现的概率都非常高。

1.4.4. 分析出口准则和报告（K1）

评估出口准则是指将测试执行与先前定义目标做比较评估的活动。这对于每个测试级别都是有效的（参见 2.2）。

分析出口准则包括以下主要任务：

- *将测试日志和测试计划中规定的出口准则做比较*
- *确定是否需要更多的测试或者修改规定的出口准则*
- *为相关方提供测试总结报告*

ISTQB 中英文术语表将 Exit Criteria 翻译成出口准则总觉得有些别扭，另一个更通行的翻译是测试退出条件。这可能更容易理解，即当规定的条件满足是测试结束。

测试出口准则不是一成不变的，根据实际情况要不断修正。比如原来规定的是所有 bug 都要修复，但在项目进行当中发现时间不够，可以通过某些正式的流程将出口准则修改为修复“普通”以上级缺陷。

1.4.5. 测试结束活动（K1）

测试结束活动指的是从已完成的测试活动中整理经验，测试件和事实数据等。测试结束活动发生于某些项目里程碑，比如当软件系统发布，软件项目完成（或取消），一个里程碑达成，或一个维护版本发布。

测试结束活动包括以下主要任务：

- *检查计划交付物都已经交付*
- *关闭时间报告或提交更改记录*

- *编写系统验收文档*
- *完成和打包测试件，测试环境和测试基础设施，以便未来重用*
- *将测试件交付给维护组*
- *分析经验教训以确定将来项目所需要的改进*
- *用收集的信息来增进测试成熟度*

这部分内容不难理解，总之这个阶段的任务就是总结和整理，积累智力资产和为今后的项目做一些准备。

1.5 测试心理学

术语：错误推测（error guessing），独立性（independence）

背景：测试评审过程中和软件开发过程中人的心态是不同的。尽管拥有正确心态的开发人员也可以测试自己写的代码，但是分离出独立的测试人员来完成测试不但可以更加专注，还可以获得一些额外的好处，比如受过训练的专业测试人员可以有独立的视角。独立的测试对各个测试级别都有效。

一定程度的独立性常使测试人员能更有效地发现缺陷和失效。但是独立性也不能取代对系统的熟悉。开发人员可以更有效地在自己代码中找到缺陷。这里从低到高定义了独立性级别：

- *被测试软件开发者来设计测试（最低独立性）*
- *其他人（比如其他开发人员）来设计测试*
- *同组织内其他组的人员（比如独立测试组）或测试专业人员（比如可用性或性能测试员）来设计测试*
- *不同组织或公司的人员（外包或外部授权体）来设计测试*

人员和项目是被目标驱动的。每个人总是倾向于根据管理者和相关利益者的目标来制定计划，比如测试到底是为了找缺陷还是为了确认软件符合需要。所以清晰表述测试目标非常重要。

在测试中寻找系统失效有可能被理解为对产品和其作者的批评，因而尽管测试对于产品风险管理具有建设性意义，却经常仍然被看作是非建设性的活动。寻找系统实效需要的是好奇心，职业悲观主义精神，挑剔的眼睛，对细节的关注，与开发同伴之间的良好沟通和错误推测的经验。

假如错误，缺陷和失效能得到良好的沟通，那么测试人员和分析，设计，开发人员之间的不良感觉是完全可以避免的。无论在开发过程中还是在评审过程中都是这样。

测试人员和测试组长需要有良好的人际能力，以围绕缺陷，进度和风险事实信息为依据进行建设性的沟通。对于软件或文档的作者来说，缺陷信息可以帮助他们提高技术能力。测试过程中的缺陷发现和修复可以节约后面的时间和金钱，并降低风险。

假如测试人员只是扮演缺陷信息传递者的角色，沟通问题就会显得尤为突出。但是也有一些方法来改进测试人员和其他角色之间的沟通和相互关系：

- *进行合作而不是争斗 – 提醒所有人质量更好的系统才是共同目标*
- *围绕事实基础进行中立客观的沟通，而不是批评相关人，比方说我们可以写下目标和实际事件报告评审测试发现的问题*
- *尽力理解其他人的感受和反应*
- *确认他们理解你所说的，你也理解他们所说的*

本节阐述了与测试相关的人员一般心理和测试人员沟通方面的软能力要求。正因为是软能力，本节内容较难体现在考试卷上，其更大价值还是体现在工作面试和日常工作中。概括成三个要点如下：

每个角色在项目过程中处于不同的心里状态。当有矛盾出现时要确认共同目标，尽量求同存异，解决问题

对事不对人

沟通很重要，沟通技巧是测试人员不可缺少的能力

另外本节开头的测试独立性是一个很重要的概念，考试中通常出现排序题，理解之后难度并不大。

1.6 道德规范（K2）

参与测试的个人会接触到保密和需要权限才能了解的信息。一套职业道德规范可以保证这些敏感信息不会被滥用。ISTQB 承认并采用一部分 ACM 和 IEEE 的工程师职业道德规范：

公共 *– 认证软件测试员的行为应当与公共利益一致*

客户和雇主 *- 认证软件测试员的行为应当符合客户和雇主的最大利益，并与公共利益一致*

产品 *- 认证软件测试员应当保证他们的提交物（测试的产品和系统）符合可能的最高职业标准*

判断 *- 认证软件测试员的职业判断应当具有正直的品格和独立性*

管理 *- 认证软件测试经理应当在软件测试管理中遵循和推广符合职业道德的工作方式*

职业 *- 认证软件测试员应当不断推进本身的职业正直品格和声誉，并符合公共利益*

同事 *- 认证软件测试员应当公平对待并支持他们的同事，并增进与软件开发人员的合作*

自身 *- 认证软件测试员应当参与与之工作实践相关的终身学习，并提高工作时间的职业道德工作方式*

目前中国在软件测试职业道德方面的规范还很欠缺，而且职业道德规范本身大多不具有强制性，并且与所处的文化环境相关联。而真正和大多数测试人员关系比较密切的是不能违反相关的保密隐私信息法律法规。有不少公司为图方便把真实用户信息用于测试，这会损害用户利益。测试人员有责任通过一定途径提出劝告。本节内容做了解即可。

习题

例题1. 下列哪个是测试计划阶段的主要任务之一?

A. 为测试分析和设计安排时间
B. 初始化更正活动（如修改缺陷）
C. 监控测试进度和覆盖度
D. 度量和分析测试结果

例题2. 调试活动通常由谁完成?

A. 开发人员.
B. 分析人员.
C. 测试人员.
D. 事件管理人员.

例题3. 测试退出准则是在哪个测试过程中确定的?

A. 测试计划.
B. 测试准则评估和报告.
C. 测试结束活动.
D. 测试控制.

例题4. 以下哪个是测试实施和执行阶段的主要任务之一?

A. 度量和分析结果
B. 作为事件报告测试差异
C. 发现测试条件或测试需求
D. 评估是否需要添加更多测试

例题5. 效率测试（比如性能测试）套件在基本测试过程的哪个阶段创建?

A. 实施和执行
B. 计划和控制
C. 分析和设计
D. 测试结束活动

例题6. 关于确认测试和回归测试以下哪个表述是正确的?

A. 确认测试是测试一组缺陷的修复而回归测试是测试代码变更是否引入新的缺陷
B. 回归测试测试是测试一组缺陷的修复而确认测试是测试代码变更是否引入新的缺陷
C. 确认测试和回归测试都是测试代码变更是否引入新的缺陷
D. 确认测试和回归测试都是测试一组缺陷的修复