第3章 静态技术

学习目标

编号	学习目标描述	级别
LO-3.1.1	了解可以通过不同的静态技术来检查软件工作产品的质量	K1
LO-3.1.2	描述在评估软件工作产品中运用静态技术的重要性和它的价值	K2
LO-3.1.3	结合测试对象、缺陷类型来说明静态测试技术与动态测试技术之间的不同，以及这些技术在软件生命周期中的作用	K2
LO-3.2.1	理解典型的正式评审过程中的阶段、角色和职责定义	K1
LO-3.2.2	解释不同类型评审的区别：非正式评审、技术评审、走查和审查	K2
LO-3.2.3	解释影响评审成功的主要因素	K2
LO-3.3.1	理解通过静态分析能够识别的典型缺陷和错误，并与评审和动态测试进行比较	K1
LO-3.3.2	举例描述静态分析的主要优点	K2
LO-3.3.3	列出通过静态分析工具识别的典型的代码缺陷和设计缺陷	K1

术语

术　语	含　义	解　释
Dynamic Testing	动态测试	通过运行软件的组件或系统来测试软件
Static Testing	静态测试	对组件/系统进行规格或实现级别的测试，而不是执行这个软件，例如代码评审或静态代码分析
Entry Criteria	入口准则	进入下个任务（如测试阶段）必须满足的条件。准入条件的目的是防止执行不能满足准入条件的活动而浪费资源
Formal Review	正式评审	对评审过程及需求文档化的一种特定的评审，例如审查
Informal Review	非正式评审	一种不基于正式（文档化）过程的评审。
Inspection	审查	一种同级评审，通过检查文档以检测缺陷，例如不符合开发标准、不符合更上层的文档等。这是最正式的评审技术，因此总是基于文档化的过程。参见 Peerreview
Metric	度量	测量所使用的方法或者度量标准
Moderator	主持人	负责检视或其他评审过程的负责人或主要人员
Peer Review	同行评审	由研发产品的同事对软件产品进行的评审，目的在于识别缺陷并改进产品，例如审查、技术评审和走查
Reviewer	评审人	参与评审的人员，辨识并描述被评审产品或项目中的异常。在评审过程中，可以选择评审人员从不同角度评审或担当不同角色
Scribe	记录员	在评审会议中将每个提及的缺陷和任何过程改进建议记录到日志表单上的人员，记录员要确保日志表单易于阅读和理解
Technical Review	技术评审	一种同行间的小组讨论活动，主要为了对所采用的技术实现方法达成共识。参见 Peer Review
Walkthrough	走查	由文档作者逐步陈述文档内容，以收集信息并对内容达成共识。参见 Peer Review
Complexity	复杂性	系统或组件的设计和/或内部结构难于理解、维护或验证的程度，参见 Cyclomatic Complexity
Cyclomatic Complexity	圈复杂度	程序中独立路径的数量。一种代码复杂度的衡量标准，用来衡量一个模块判定结构的复杂程度，数量上表现为独立现行路径条数，即合理的预防错误所需测试的最少路径条数。圈复杂度大说明程序代码可能质量低且难于测试和维护，根据经验，程序的可能错误和高的圈复杂度有着很大关系。圈复杂度 $=L-N+2P$，其中 L 表示为结构图（程序图）的边数；N 为结构图（程序图）的节点数目；P 为无链接部分的数目

续表

术 语	含 义	解 释
Control Flow	控制流	执行组件或系统中的一系列顺序发生的事件或路径
Data Flow	数据流	数据对象的顺序的和可能的状态变换的抽象表示，对象的状态可以是创建、使用和销毁
Static Analysis	静态分析	分析软件工件（如需求或代码），而不执行这些工作产品

3.1 静态技术和测试过程

与要求运行软件的动态测试技术不同，静态测试技术通过手工检查（评审）或自动化分析（静态分析）方式对代码或者其他的项目文档进行检查而不需要执行代码。静态测试的主要目的是从已有的说明（例如需求说明、设计说明等）、已定义的标准或项目计划和程序代码中发现缺陷和偏差。静态测试的基本思想是预防缺陷、尽可能早地在缺陷和偏差对将来的开发过程和测试过程产生影响之前识别并解决它们，避免将这些缺陷引入到下一个阶段，从而导致昂贵的返工。同时，静态测试的结果以及对结果的分析，可以有效用于软件质量改进以及开发过程和测试过程的改进。

静态测试包括人工检查（评审）和自动化检查（静态分析），是一个经常被低估或者被忽略的方法。和动态测试不同，静态测试不需要实际执行测试对象和输入测试数据，而是用阅读和分析替代具体的运行系统。也就是说，利用静态测试技术进行测试时，并没有真正地运行被测对象，而是通过人工或者自动化的方式对被测对象进行检查和分析。

静态测试可以是一个或多个人一起检查文档（评审），或使用特定的工具来完成文档或者代码的检查（静态分析）。软件开发项目中的所有文档都可以通过人工方式来检查。对遵循特定规则的文档可以通过工具进行静态分析。对于代码，既可以通过评审的方式人工进行检查（如代码走查），也可以通过工具进行自动化的静态分析。

评审是静态测试的重要组成部分。通过阅读分析可以检查和评估文档中的问题，具体是通过透彻阅读并尝试理解被检查的文档来完成评审。它是对软件工作产品（包括代码）进行测试的一种方式，评审可以在动态测试之前，也可以在任何其他阶段进行，例如可以对需求文档进行评审，也可对执行测试结束后得出的测试总结报告进行评审。修改在软件生命周期的早期发现的缺陷的成本要比修改在后期的动态测试中才发现的缺陷的成本低得多。例如由于需求理解错误引起的错误，在早期评审过程中发现和修改该缺陷只要花费较低的代价，如果等到完成设计工作并生成代码，再通过动态测试发现和修改该缺陷时需要花费更高的代价。如果该缺陷影响到系统的架构等核心部分，则导致的返工费用将会更加昂贵。

进行人工评审的主要活动是检查软件工作产品，并对它们提出修改意见。评审的主

要好处有尽早发现和修改缺陷、改善开发能力、缩短开发时间、缩减测试成本和时间、减少产品生命周期成本、减少软件缺陷以及改善和开发人员之间的沟通等。评审也可以在软件工作产品中发现一些遗漏或者冗余的内容，而这在动态测试中是很难发现的。

静态分析是静态测试的另一个重要组成部分。它一般是通过工具支持的方式来进行的。静态分析可以根据工作强度、形式、必需的资源（人员和时间）以及目的的不同，进行不同的分类。

静态测试（评审、静态分析）与动态测试有着共同的目标，即识别缺陷。但静态测试与动态测试也有所不同，静态测试直接发现文档或代码中的缺陷，而不是它们的外部表现，即失效。而动态测试通过运行被测对象，在运行过程中发现被测对象的缺陷的外部表现，即失效。它们之间是相辅相成的，不同的技术可以有效地发现不同的错误类型（缺陷和失效）。

与动态测试相比，静态测试更容易发现如下问题：与标准之间的偏差、需求的遗漏和错误、设计的缺陷、软件可维护性差和错误的接口说明等，而这些文档往往在动态测试过程中作为重要的测试依据。如果这些作为动态测试依据的文档的质量无法保障，则动态测试本身的质量也就值得怀疑，更无法有效保障被测对象的质量。

3.2 评审

评审类型是多样化的，既可以是不正式的评审，例如评审员没有文档化的指导性资料可参考，也可以是正式的评审，例如有团队参与、且有文档化的审查结果和管理审查的步骤。评审过程的正式程度和开发过程的成熟度、法律法规方面的要求或审核跟踪的需要相关。

采取什么样的评审类型由评审的目标决定，评审目标可以是发现缺陷、增加理解、培训测试人员和团队其他成员或对讨论和决定达成共识等。

3.2.1 正式评审过程

为了有效管理和监控评审，需要定义正式而系统的评审过程，即定义详细的阶段和活动。正式的评审过程由 6 个阶段组成：计划阶段、预备会阶段（Kick Off Meeting）、个人准备阶段、评审会议阶段、返工阶段和跟踪结果阶段（参考 IEEE Std 1028-1997）。

1. 计划阶段

为了有效开展评审活动，首先需要进行评审计划。评审计划阶段主要确定评审对象和评审目的，该阶段的主要活动包括测试经理或管理者选择主持人；与主持人一起选择评审员、分配角色和职责；为正式的评审类型（例如审查）定义入口准则和出口准则；选

择需要进行评审的文档或文档章节，以及针对更加正式的评审类型核对入口准则。

管理层必须在项目计划过程中，确定软件开发过程中的哪些文档需要进行评审，谁将参与评审以及采用什么评审类型。在项目计划中必须估算评审的工作量。在评审计划期间，主持人需要选择合适的评审员组成评审团队。通过和评审文档的作者一起，确保文档处于可评审的状态，即文档已经完备并且文档相关的工作已经完成。针对更正式的评审，还需要设定和检查入口准则以及设定出口准则。

通常情况下，从不同的角度对文档进行评审，或每个人只针对文档的某个方面进行评审，可以使评审更加容易成功。评审计划阶段需要确定评审的关注点。假如评审的目的是为了检查文档的基本质量，可以只对文档的高风险部分进行评审，或对评审对象进行抽样，并对抽样的内容进行评审。

假如需要针对评审召开预备会，必须选定预备会议的时间和地点。

2. 预备会阶段

假如评审计划中无法明确一些事情和注意事项，有时候在评审计划之后，还需要召开评审预备会。主持人召集评审员进行一个简短的会议，向评审员介绍评审的对象、评审的目的、评审的过程、入口准则和出口准则，给评审员分派评审任务和分发相关文档，以及介绍其他一些需要注意的事项。

3. 个人准备阶段

评审员明确了评审目的和各自的任务之后，接下来进入个人准备阶段。即在评审会议之前，每位评审员仔细阅读各自负责的评审内容，并根据提供的参考文档检查评审对象，标注评审对象中可能的不足、问题、意见和建议。评审员的充分准备是成功进行评审的一个重要前提条件。

4. 评审会议阶段

假如满足了评审的入口准则，就可以进入评审会议阶段。该阶段的主要活动包括通过文档化或会议纪要(针对更正式的评审类型)的方式讨论和记录评审过程和结果。评审员也可以简单地标识缺陷，提出建议来处理缺陷，或为如何处理缺陷做决定。

评审会议由主持人主持，主持人必须保证所有的评审员能够客观地表达他们的观点，确保评审对象关注在对产品的评估，而不是对作者的评估，并且预防或解决评审过程中可能出现的问题和冲突。主持人需要有良好的交际能力和技巧，以保证评审会议按照预期的目标进行，并激励参与人员为评审做出最大的贡献。

通常来说，评审会议的时间有限。评审的目的除了发现缺陷外，还包括判断评审对象是否满足需求、是否符合标准。评审的结果一般有接受、有条件接受和不接受。所有的评审员应该对这次评估的决定和综合结果达成一致。

下面是评审会议阶段需要注意的一些通用准则。

(1) 评审会议的时间尽量控制在 2 个小时内。如果需要,可以在当天再发起另外一个评审会议。

(2) 如果一个或多个专家(评审员)没有出席,或者他们没有准备充分,主持人有权取消或中止会议。

(3) 评审的对象是文档,而不是作者,因此确保大家关注评审对象,而不是作者,这包括:

① 评审员必须要注意他们的言语以及表达的方式。

② 作者不应该成为被攻击的对象。作者应该把他人提出的意见和建议当作自己学习和提高的机会,并在必要时对自己的作品进行解释。

(4) 主持人不应该同时作为评审员。

(5) 修改方案的讨论不是评审的任务。

(6) 每个评审员必须有机会充分地表达他们的观点。

(7) 会议纪要必须充分表达评审员的意见。

(8) 评审会议中提交的问题不应该以命令的形式提交给作者(中肯的改进或修改建议有时候对质量改进是有帮助的,并且是明智的)。

(9) 针对不同的问题,划分不同的严重程度,缺陷按照严重程度分为:

① 严重缺陷(例如评审对象不能满足设计的目的,在批准评审对象之前必须修正相关缺陷);

② 重要缺陷(影响评审对象的可用性,批准评审对象之前应该修改相关缺陷);

③ 一般缺陷(小的偏差,基本不影响使用)。

(10) 评审团队应该对评审对象给出最后的意见,例如:

① 接受(无需修改);

② 有条件接受(需要修改,但不需要进一步评审);

③ 不接受(需要进一步评审或其他的检查措施)。

(11) 会议最后,所有参加人员需要签署最终的会议纪要。

会议纪要应包括会议中讨论的所有问题和异常现象。另外,评审总结报告应该收集评审过程中所有的信息和数据,包括评审对象、涉及的人员、人员的角色、重要问题的简短总结以及评审员建议的评审结果。当执行正式的评审时,还需要检查正式的出口准则。

5. 返工阶段

返工阶段的主要活动是修改发现的缺陷,通常由作者负责返工工作。作者同时需要更新相应的文档或者代码,以及更新之后的发布等。

6. 跟踪结果阶段

跟踪结果阶段的主要活动包括检查缺陷是否已修改、收集度量和检查出口准则(针

对更正式的评审类型)。

通常需要指定专门的人负责跟踪缺陷的修改,例如主持人本人、主持人指定的其他人。假如评审的结果为“不接受”,那么需要安排另外一次评审。重新进行评审的过程可以采用正式的评审过程,由于时间和成本等原因,通常情况下会采用更简单的方式。重新评审一般只对修改部分进行检查。

接下来需要对评审过程和评审结果进行评估,为评审的过程改进提供依据。例如根据评审结果,修改评审检查表,并保持它们是最新的。为了达到这个目的,必须收集和评估评审相关的度量数据。收集和分析连续发生或经常发生的缺陷类型,找出其根本原因,然后针对这些缺陷类型,计划和实施开发过程和测试过程改进,例如若发生该缺陷类型的根本原因是具体人员的知识和技能不足,可以通过培训进行弥补。同时,可以将这些缺陷类型加到评审检查表中,以提高评审的效率和有效性。

3.2.2 角色和职责

评审过程中需要有不同角色的人参与,他们在评审中的职责是不一样的,这一节将会对不同的角色和职责进行详细描述。

1. 经理或管理者

经理或管理者决定文档或者代码是否需要进行评审,若需要,则在项目计划中保留和分配足够的时间和资源。同时,在评审结束之后判断是否达到评审的目标。经理选择评审对象并确保基础文档和必需的资源可用,以及确定主持人和评审员的人选(也可由主持人来确定评审员)。

通常不建议管理者或者管理者代表参与评审会议,主要原因如下。

(1) 首先,文档作者或评审员担心经理或管理者通过评审对他们进行考核,从而导致参加的评审员无法进行自由讨论。

(2) 其次,除了项目管理人员参与的项目计划评审以及类似的评审之外,评审的对象文档更多的是关于技术方面的。作为经理或管理者,他们一般没有必要也不一定理解技术文档的具体内容。

2. 主持人

主持人负责文档或文档集的评审活动,其主要的职责包括制定评审计划、召开评审会议和跟踪评审结果,以及负责和评审有关的管理工作,确保评审有序进行从而达到期望的目标。主持人的另一个重要职责是收集评审数据和发布评审报告,用于软件开发和测试过程以及评审过程的改进。主持人还需要进行评审员不同观点之间的协调。

主持人对评审的成功至关重要,他们需要具备各种相关的技能。首先,主持人必须擅长评审会议的组织和协调,通过选择合适的策略引导会议有效地进行。其次,主持人

必须能够在不打击参与者积极性的情况下，中止不必要的讨论，以及在评审员之间存在观点冲突的时候进行调解，以中立的立场协调参与者之间的讨论。最后，他们必须保持中立，不对评审对象发表自己的看法。

3. 作者

作者是提交评审的文档的创建者。如果多个人参与文档的创建，该文档的主要负责人可以指定为作者，由他负责该角色相关的职责和任务。

作者的主要职责包括使评审对象满足它的评审入口准则，例如确保文档处于合理的完成状态；根据作者对文档内容的理解和相关知识，支持文档的评审；作者需要负责文档评审以后的任何返工，并且使得评审对象满足它的评审出口准则。

对作者而言，重要的是不把评审员针对文档提出的问题，看作是对他个人的批评。作者必须明白评审的目的是帮助改进产品的质量。

4. 评审员

评审员一般是指具有专门技术或业务背景的人员，也称为检验员(Checker)、审查员(Inspector)，他们在必要的准备后，标识和描述被评审对象存在的问题(缺陷)。他们是涉及评审对象内容相关技术方面的专家。评审员应该在评审过程中代表不同的观点。

评审员应该识别评审对象中存在的问题，并对它们进行适当的描述。他们可以代表不同利益相关者的观点(例如项目发起人、需求分析人员、设计人员、编码人员、安全相关人员、测试人员等)，但是他们表达的观点必须和评审对象相关。

为了保证评审的有效覆盖率，有时候需要给一些评审员分配特定的评审主题。例如有的评审员可以关注特定标准的一致性，有的关注语法，有的关注整体的一致性。主持人应该在计划评审的时候分配这些角色。

评审员应该为评审会议做充分的准备。在评审对象描述不充分的地方和可能错误的地方做上相应的标记，并以作者能够修正的方式文档化。

5. 记录员

记录员记录所有在评审会议中提出的不足、问题(包括采取的措施、决定和建议等)，以及在会议过程中标识的未解决的问题。记录员必须能够以简短和准确的方式记录评审中发现的问题，抓住评审过程中讨论的中心思想，清晰地表达问题。

有时，作者担当记录员这个角色是比较合适的，因为作者能够较容易地理解评审员提供的信息，同时清楚以什么样的准确度和详细程度对评审员提供的意见和建议进行记录。

3.2.3 评审类型

评审的类型是多样化的，不同的文档或者代码有时候需要经历不同的评审类型，因

此，对同一个评审对象需要在不同的时间采用不同的评审类型。例如，技术评审之前首先进行非正式评审，在走查之前可能要进行需求说明审查。评审的对象既可以是项目的工作产品，也可以是项目当前的状态，评审员在技术、成本、进度等方面对项目的状态进行评估。

评审的类型主要包括非正式评审(Informal Review)、走查(Walkthrough)、技术评审(Technical Review)、审查(Inspection)。

1. 非正式评审

非正式评审是评审的精简版，它以一种简单的方式遵循评审的通用过程。通常情况下，文档作者发起非正式评审。评审计划局限在选择评审员和要求他们在规定时间内提交他们的意见和建议。非正式评审通常不召开评审会议，也不在评审员之间交换各自发现的问题。在这种情况下，评审只是作者和评审员之间的交互。评审的结果不需要明确的文档化，有时一个评审清单或修订文档就足够了。

非正式评审是一种由一个或多个同行完成的交叉阅读，结对编程、结对测试、代码交换以及类似的工作形式都可以认为是非正式评审的一种。非正式评审非常普遍，并且由于工作量小和简单方便而被广泛接收。

非正式评审的主要特点如下。

(1) 没有正式的过程。

(2) 可以由程序员的同行们或技术负责人对设计和代码进行评审。

(3) 评审结果可以文档化。

(4) 评审者不同，评审作用可能会不同。

(5) 其主要目的是以较低的成本获得收益。

2. 走查

在测试实践中，走查既可能是非常正式的评审活动，也可能是非常不正式的评审活动。走查的主要目的是发现文档中的错误、改进产品质量、考虑替换的实现方案以及评估文档内容和标准规格之间的符合程度，也可以是相互学习、增加理解的一个过程。

走查的关注点是召开评审会议(没有时间限制)。相对于其他类型的评审，走查的准备时间是最少的，有时甚至可以省略。在走查会议上，作者向评审员介绍或演示文档内容或者产品，例如根据软件处理事件的顺序，以检查典型的用例，有时候也称为场景，也可以模拟单个的用户使用场景。评审员通过自发的提问来发现可能的缺陷。

走查的过程适合 5～10 人的小型开发和测试团队，因为准备工作和后续工作不需要占用很多资源，因此成本较低。走查可以用来检查那些重要性较低的文档。

由于作者主导评审会议，因此他对走查过程有最大的影响力，例如可以决定会议讨论的重点。同时作者负责走查后续的跟踪，对于走查的后续活动，没有要求更进一步的检查。

以下方法也可用于走查：会议前评审员提前准备，评审结果写入会议纪要，列出所有的发现而不是让作者标记它们。实际上，从非正式到正式走查有很多变种。

走查的主要特点如下。

(1) 由作者召集开会。

(2) 以情景、演示的形式开展走查活动，并且以同行参加的方式进行。

(3) 开放式模式。

(4) 评审会议之前的准备、评审报告、发现的问题和记录员(不是作者本人)都不是必需的。

(5) 在实际情况中可以是非常正式的，也可能是非正式的。

(6) 其主要目的是学习、增加理解、发现缺陷。

3. 技术评审

技术评审既可以是正式的，也可以是非正式的。技术评审的对象可以是正式的说明等技术文档，其关注的焦点是技术文档与其他参考文档之间的一致性，例如需求说明、标准等。参与技术评审的评审员必须是有资质的技术专家。为了避免项目盲点，参与评审的一些评审员可以不是项目的参与者，管理层也不需要参与。评审员写出他们发现的缺陷和建议，并在评审会议前提交给主持人。主持人(理想状态是经过培训的人员)根据他们认为的重要性为所有意见和建议设置优先级。评审会议上，只对主持人选择出的重要意见和建议做相关的讨论。

技术评审的大部分工作集中在准备工作阶段。在评审会议上，记录员记录所有的问题并准备评审结果的最终文档。评审结果必须获得所有参与人员的一致通过并签名。不同的意见应该记录在会议纪要中。对评审的结果做出决定不是评审员的工作，而是管理层的职责。如果是非常正式的技术评审，需要定义评审的入口准则和出口准则。

技术评审的主要特点如下。

(1) 对发现的缺陷需要进行文档化。

(2) 需要同行和技术专家的参与。

(3) 没有管理者参与的同行评审。

(4) 理想情况下由专门接受过培训的主持人(不是作者本人)来主持。

(5) 会议之前需要进行准备。

(6) 可以使用检查表、评审报告、发现的问题列表等。

(7) 在实际情况中可以是非常正式的，也可能是非正式的。

(8) 其主要目的是讨论、评估、发现缺陷，解决技术问题，检查与规格及标准的符合程度。

4. 审查

审查是最正式的评审，它遵循正式严谨的评审过程。通常每个评审员都是从作者的

直接同事或同行中选出的，并且具有固定的角色。按照一定的规则要求定义评审过程，针对审查的不同对象，使用包括审查标准（正式的入口准则和出口准则）在内的检查表。

审查的重点是发现文档的不清晰要点和可能的缺陷、度量文档质量、改进产品质量和开发以及测试过程。审查计划阶段，需要确定审查的目的，并且只对文档的特定部分进行检查。审查开始之前，根据正式入口准则检查审查对象，以确定是否可以开始审查活动。审查员（在审查过程中的评审员也可称作审查员）采用过程、标准和检查表等手段来准备审查内容。

审查会议遵循下面的议程。

主持人主持会议。主持人首先介绍参加人员和他们的角色，同时简单介绍需要检查的对象。主持人询问每个参与者是否准备充分，可以询问评审员用了多少时间以及发现了多少问题，来检查每个人是否为这次会议做了充分的准备。然后讨论文档格式的问题，并写入会议纪要。

其中的一个评审员用简单的、合理的方式来介绍审查对象的内容，也可以大声地朗读文档。在介绍文档的过程中，评审员开始提问，并对选择的审查内容进行仔细地检查。作者回答相关的问题，但是这个过程通常是被动的。如果作者和评审员对某个有疑问的意见不一致，可以在会议的最后继续进行讨论。

主持人必须在讨论失控的时候进行干预，还要保证评审会议覆盖了所有需要评审的内容以及整个文档。同时主持人还要保证记录员记录了所有的问题和疑问，并且进行跟踪。

在会议的最后，需要检查评审过程中发现的所有问题，以保证记录的完整性。对有争论的问题需要重新进行讨论，以决定它们是否是缺陷。如果依旧没有解决，需要将争论的观点写入会议纪要。

最后，根据评审过程中发现的问题和建议，确定审查对象是评审通过，还是需要返工。在审查完成后，需要管理后续的跟踪工作和重新审查工作。

审查过程中还有一个重要的工作，即收集数据对开发和测试过程和审查过程进行质量评估。审查工作除了可以评估被审查文档的质量外，还适用于开发过程和测试过程的改进。通过分析收集到的数据，可以发现开发过程和测试过程中存在的弱点。过程改进后，通过把以前的数据和当前的数据做比较以检查过程改进的有效性。

审查的主要特点如下。

（1）由专门接受过培训的主持人（不是作者本人）来主持。

（2）通常是同行检查。

（3）定义了不同的角色。

（4）引入了度量。

（5）根据入口准则、出口准则和检查表定义正式的评审过程。

（6）会议之前需要进行准备。

（7）需要审查报告和发现问题列表。

(8) 有正式的跟踪过程。

(9) 可以进行过程改进。

(10) 其主要目的是发现缺陷。

3.2.4 评审成功的因素

评审类型的选择很大程度上取决于对评审以及评审对象质量的要求,以及需要花费的工作量,同时也取决于项目环境。评审类型的选择,会受到各种因素的影响,例如:

(1) 评审结果的形式。例如,是否需要具体的评审结果文档,或者只需要非正式的评审结果。

(2) 是否能找到 5～7 个技术专家都合适的评审时间?

(3) 是否需要有不同领域的技术知识?

(4) 需要多少有资质的评审员参与?

(5) 评审的收益(期望结果)和投入评审的工作量是否相一致,或者说是否值得?

(6) 评审对象是否具有正式的记录格式?是否可以通过工具支持来开展分析活动?

(7) 管理层是否支持评审活动?项目面临进度压力的时候,管理层是否会缩减评审时间和工作量?

根据组织和项目的特殊要求,可以对评审类型进行相应的裁剪,以提高评审的效率。另外,参与项目的不同人员之间的协同合作有利于项目质量的提高,因此项目团队相互检查各个阶段输出的工作产品,就可以在早期发现一些缺陷和含糊不清的地方,例如极限编程中建议的结对编程,可以认为是两个人进行评审的一种固定模式。假如项目是分布式开发的,组织评审就会比较困难,此时需要采用其他的一些手段支持评审,例如网络评审、视频评审、电话会议等。

成功开展评审,会受到各种因素的影响,例如:

(1) 为每次的评审预先确定评审目标。评审的目的可以是发现评审对象中的缺陷,提高评审对象的质量,也可以是相互学习和交流、决策和评估候选方案等。

(2) 针对评审目标,有合适的评审人员的参与。选择合适的评审员是评审成功的关键,要选择具有一定资质的人员参与,避免让领导或管理人员在不必要的情况下参与评审。

(3) 对发现的缺陷持欢迎态度,并客观地描述缺陷。对作者来说,应该把评审当作一次学习交流的机会,对意见和建议持欢迎的态度。对评审员来说,应该对事不对人,并能客观、中性地发表意见和建议。

(4) 能够正确处理人员之间的问题以及心理方面的问题(例如对作者而言,能让他觉得有积极正面的体验)。

(5) 采用的评审技术适合于软件工作产品的类型和级别以及参与的评审人员。

(6) 选用合适的检查表或定义合适的角色,可以提高评审过程中发现问题的效率。

(7) 提供评审技术方面的培训,特别是针对正式的评审技术,例如审查。

(8) 管理层对良好评审过程的支持(例如在项目计划中安排足够的时间来进行评审活动)。

(9) 强调学习和过程的改进。不断从评审过程本身来学习经验教训,例如评审过程的持续改进等。

3.3 静态分析与工具支持

静态分析指的是不需要运行程序代码,借助工具对测试对象进行检查的技术。而动态测试需要真正运行软件的代码。静态分析可以发现在动态测试中很难发现的问题。与评审一样,静态分析通常发现的是软件的缺陷而不是软件运行的失效。静态分析工具能够分析程序代码(例如控制流和数据流),同时产生 HTML 和 XML 等格式的信息输出。

和评审一样,静态分析的目的是发现文档或者代码中的缺陷或者可能存在的缺陷隐患。不同的是,静态分析一般需要工具的支持。例如,拼写检查工具可以认为是静态分析的一种形式,它可以发现文档中的拼写错误,从而有利于文档质量的提高。静态分析的另外一个目的是得到度量数据,从而对测试对象的质量和复杂度进行度量和验证。

在使用工具对文档进行分析和检查时,被分析的文档必须以特定的结构和标准来组织。静态分析的对象可以是各种类型的正式文档,例如代码、技术文档、需求文档、软件架构或者软件设计、UML 中的类型图模型等。HTML 和 XML 格式产生的输出也可以通过工具支持来进行分析。也可以对设计阶段开发的正式模型进行分析,找到其中的不一致。目前,程序代码是软件开发过程中常见的可以进行静态分析的文档。

在组件测试或集成测试过程中,开发人员通常会使用静态分析工具,检查被测对象是否满足编程指南或编程规范。集成测试过程中,需要分析测试对象是否满足接口说明。

静态分析和评审是紧密联系的。若在评审之前进行了静态分析,可以发现很多的缺陷或异常,则可以显著提高评审效率。由于静态分析通常是工具支持的,因此其工作量会比评审少得多。

(1) 假如对象是正规的文档,可以使用工具支持的静态分析,只要花费很少的成本就可以发现一些错误和不一致处,从而可以缩短评审的时间。

(2) 通常,即使没有计划评审活动,静态分析也是应该经常使用的。通过静态分析发现和移除缺陷或异常,可以提高文档的质量。

并不是所有的缺陷都可以通过静态分析来发现,有些缺陷只有在程序运行的时候才能显现出来,即这些缺陷只有通过运行时才会转变成失效并被发现。例如,除法中的分母是一个变量,给这个变量的赋值为 0 时,运行程序就会出现问题。通过静态分析,发现

这种类型的缺陷并不容易，除非常量 0 赋给了这个变量。另外一种方式是分析所有可能的路径来解决这个问题，但这种情况可能会出现潜在的风险：项目的延期。

另外，动态测试也不容易发现程序中的一些不一致和可能存在问题的区域。例如，测试对象和编程标准的差异、禁止使用具有错误倾向的程序结构等类型的缺陷，它们更容易通过静态分析（或评审）发现。

所有的编译器都会对程序代码进行静态分析，用来确认程序代码是否使用了编程语言的正确的语法。除了编译器外，还有称为分析器的工具，用它们对单个软件组件或集成系统进行分析。

用静态分析工具可以发现程序中的如下问题。

（1）引用一个没有定义值的变量。

（2）模块和组件之间的接口不一致。

（3）从未使用的变量。

（4）不可达代码或死代码。

（5）逻辑上的遗漏与错误（潜在的无限循环）。

（6）过于复杂的结构。

（7）违背编程规则。

（8）安全漏洞。

（9）代码和软件模型的语法错误。

静态分析可以用来发现安全性问题。很多安全性漏洞是由于使用了错误倾向的程序结构，并且没有进行必要的检查而发生的。例如缺少缓冲区溢出保护，或者没有检查输入数据越界等问题。静态分析工具可以发现这种类型的缺陷，因为它们有标准的格式来查找和发现这种缺陷。

静态分析的优点如下。

（1）在测试执行之前尽早发现缺陷和问题。

（2）通过度量的计算（例如高复杂性测量），早期警惕可能存有问题的代码和设计（高风险区域）。

（3）可以发现在动态测试过程中不容易发现的一些缺陷和异常。

（4）可以发现软件组件之间的相互关联的不一致性。

（5）可以改进代码和设计，增强可维护性。

开发人员通常在组件测试和集成测试之前或期间使用静态分析工具（例如检查预先定义的规则或编程规范），而设计人员在软件建模期间也会使用静态分析工具。

静态分析工具通常会产生大量的警告和注释信息。为了更加有效地使用工具，产生的大量信息必须进行适当的处理，例如通过设置工具参数，按照一定的顺序或规则控制产生的信息，否则使用这种工具的效率无法体现。编译器也可以为静态分析提供一些帮助，包括度量的计算等。

3.3.1 编译器分析工具

通过编译器分析工具可以发现编程语言语法错误，并且以错误或警告的方式进行报告。很多的编译器也会产生其他的信息，并且执行其他的检查。例如：

(1) 产生不同程序元素的交叉引用列表(例如变量、函数)。

(2) 检查编程语言中数据和变量的类型是否一致。

(3) 检查没有定义的变量。

(4) 检查不可达代码。

(5) 检查域边界的上限或下限(静态选择)。

(6) 检查接口的一致性。

(7) 检查所有作为跳转开始或跳转结束标签的使用。

这些信息通常提供在一个列表中。工具中报告的“疑似”结果通常是可能的缺陷隐患，因此，需要更进一步的分析。

3.3.2 规范标准一致性

通过静态分析工具还可以检查测试对象是否与规范、标准相一致，例如是否遵循了编程规范和标准。这种检查方式几乎不需要花费多少时间和人力成本。另外，通过静态分析工具检查还有一个优点：假如编程人员知道代码需要和编程规范进行一致性检查，他们会更乐于按照编程规范来工作。

3.3.3 数据流分析

数据流分析通过检查程序代码中的数据流来发现数据流异常。需要注意的是数据流分析通常发现的是数据流异常，不一定是缺陷。异常指的是由于不一致而可能导致的失效，也可能是代码的冗余。异常是一种风险，可能会触发系统的失效。

数据流异常有各种表现形式，例如变量没有赋值之前，就对该变量进行读取操作；或者变量赋值之后，但在程序中根本没有使用或读取该变量。在数据流分析过程中，检查每个变量的使用情况，并且根据每个变量的使用情况定义了三种不同的变量状态。

(1) 已定义(D-Defined)：变量已经赋值，表示此变量有确定的值。

(2) 被引用(R-Referenced)：读取或使用变量的值，表示此变量的值被使用了。

(3) 未定义(U-Undefined)：命名了变量，但还没有对变量赋值，或已经释放了变量(模块或函数结束时)，此时的变量没有确定的值。

跟踪每一个变量，通过分析变量的状态变化情况发现问题或异常，这里存在三种数据流异常的情况，即如果某个变量的先后状态连续出现 UR、RU 或 DD。

(1) UR-异常：程序路径上的某个变量的状态从U(未定义)转化到R(被引用)，表示读取了没有赋值的变量或已经释放的变量。

(2) DU-异常：某个变量的状态从D(已定义)转化到U(未定义)，表示变量已经赋值，但该变量未曾使用就已经是无效了，因为此变量又被重新命名了，原先的值已不复存在。

(3) DD-异常：某个变量的状态从D(已定义)转化到D(已定义)，表示变量被连续赋值了两次，在前一次赋值还没被使用的情况下又赋了第二个值，同时第一个赋值不复存在。

下面的例子中解释了不同类型的异常情况(以C++语言为例)。函数的目的是假如变量Min的值大于变量Max的值，则通过变量Help的帮助，来交换参数Max和Min的整型值。

```
void exchange (int& Min, int& Max) {
int Help;
        if (Min > Max) {
        Max = Help;
             Max = Min;
             Help = Min;
        }
    }
```

通过如表3-1所示的数据流分析表可以对例子中的数据流进行分析。

表3-1 数据流分析表

行号\变量	Min	Max	Help
1 (in)	D	D	
2			U
3	R	R	
4		D	R
5	R	D	
6	R		D
7			
8 (out)	U	U	U

通过分析变量的使用情况，可以发现以下异常。

(1) 变量Help的UR异常：变量的范围局限在函数内。变量的第一次使用是在赋值语句的右侧部分(语句2)。这个时候，变量还没有赋值，而在语句4进行了直接的引用。变量声明的时候没有初始化(假如高级别的警告打开，这种异常也可以通过常用的编译器来发现)。

(2) 变量Max的DD异常：在赋值语句4和赋值语句5的左边，Max变量连续使用了两次，因此Max赋值了两次。第一次的赋值就被忽略，第一次赋的值在被使用前已被

第二次赋的值覆盖了。

(3) 变量 Help 的 DU 异常：函数的最后一个赋值(语句 6)，变量 Help 被赋了一个任何地方都不能用的数值，因为变量 Help 只有在函数内是有效的。

这个例子中，异常情况是明显的。但是在实际情况中问题可能会更复杂，函数的规模也可能会很大。这时候异常就不会这么明显，通过手动检查，例如评审，容易遗漏这些问题。而分析数据流的工具可以发现这些异常。

不是每个异常都会导致程序或者软件的不正确行为，成为一个失效。例如，DU-异常就不会有直接的影响，程序可以继续运行。所以我们应该去分析这些异常：为什么这个异常的赋值会出现在程序的这个特殊位置(语句 6)？通过分析找出引起这些异常的根源。通常，存在异常和问题的程序部分，更加需要进行检查以便进一步地发现不一致。

3.3.4 控制流分析

假如将程序的结构以控制流图的方式表示，控制流图中的节点代表代码中的可执行语句。可以将没有分支的顺序系列语句表示为一个节点，因为这一系列语句在程序执行时其控制流并不会发生变化。假如执行这系列语句的第一句，系列内的其他语句也一定会被执行到。

程序执行中顺序可能会发生变化的地方表示为分支，分支可以是判定语句，也可能是循环语句。例如判定语句 IF ，若判定计算的值为 TRUE，则程序继续执行以 THEN 开头的部分。若判定计算的值为 FALSE，则程序执行 ELSE 部分。假如是循环语句，循环会回到前面的语句，所以会重复执行控制流中的部分语句。

通过控制流图的清楚描述，可以很容易理解程序结构，同时可以发现一些异常，例如是否会异常跳出循环体，或者有几个出口的程序结构等。这些异常并不一定会导致失效，但它们不符合结构化编程的原则。在实际应用过程中，通过手工方式生成控制流图是很困难的，因此需要有相应工具的支持。

假如控制流图的部分或者整体非常复杂，它们之间的相互关系和事件顺序难以理解，这时候就需要修正程序的架构和内容，因为复杂的程序结构常常意味着潜在的错误风险。而圈复杂度静态分析工具可以分析代码的复杂程度。

3.3.5 圈复杂度

静态分析工具除了上面提到的静态分析能力外，还可以提供度量值。软件系统的质量特性可以通过度量值进行度量。通过检查度量值，确认它们是否满足特殊的需求。

定义软件特殊属性度量的目的是获得抽象软件的定量测量。因此，度量只能对经过检查的软件部分提供判断，并且计算得到的测量值，需要和经过检查的其他程序或程序模块之间比较，这样才有意义。

圈数可以用来测量程序代码结构的复杂度，所以也称作圈复杂度，如图 3-1 所示。

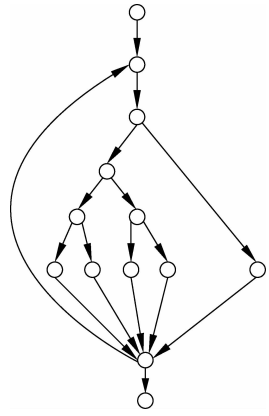

图 3-1　圈数计算的控制流图

对于程序或程序模块的控制流图 G，它的圈复杂度（圈数）可以通过下面的公式计算得到：

$$v(G) = e - n + 2$$

其中：

$v(G)$为控制流图 G 的圈数；

e 为控制流图中的边数；

n 为控制流图的节点数。

图 3-1 表示了程序模块的控制流，它是可以被调用的函数。圈数的计算式如下。

$$v(G) = e - n + 2 = 17 - 13 + 2 = 6$$

其中：

e 为控制流图的边数＝ 17

n 为控制流图的节点数＝ 13

计算得到的圈数为 6，根据 McCable 原则，这个数值属于可以接受的中间范围。McCable 原则一般认为圈数大于 10 是不可接受的，若圈数大于 10 则建议需要对程序代码进行重新设计。通过圈数可以获得不同组件/模块的相对复杂度，有效地安排资源，并确定测试的广度和深度。例如假如某个组件的圈数较大，说明此组件相对比较复杂、风险也较大，可能会考虑对此组件进行更深入的测试、花费更多的资源。

圈数也代表了程序模块的控制流图中独立路径的数目，这组独立路径代表了控制流图中的所有特征，而其他的非独立路径都可以通过这组独立路径运算得出。因此圈数可以用来估算程序代码的可测试性和可维护性。为了有效地测试控制流图中的所有特征，则至少需要覆盖所有的独立路径，因此圈数提供了关于测试用例数目的重要信息。维护代码最基本的要求是了解程序。程序圈数越大，理解程序模块的难度越大。

圈数的一个缺点是没有考虑用来选择控制流的条件复杂性。不管条件是由多个原子条件通过逻辑运算符连接而成的复合条件，还是单个条件，都不会影响圈数的计算。

更多关于圈的知识请参见相关资料。

3.4 习题

1.（K1）关于静态分析的描述，下列选项正确的是（　　）。

A. 开发人员通常在软件验收期间使用静态分析工具

B. 静态分析不需要运行被测软件，且能发现软件的失效

C. 通过静态分析能够发现模块和组件之间的接口不一致

D. 通过静态分析能够发现软件内的所有缺陷

2.（K1）下面不属于软件评审的好处的是（　　）。

A. 增加测试的时间

B. 尽早发现和修改缺陷

C. 改善开发能力、缩短开发时间

D. 缩减测试成本

3.（K1）“向评审员解释评审的目标”属于下列哪个阶段的主要活动？（　　）

A. 计划阶段

B. 预备会阶段

C. 个人准备阶段

D. 评审会议阶段

4.（K2）在评审过程中，主持人的主要职责是（　　）。

A. 决定是否需要进行评审

B. 主持文档或文档集的评审活动

C. 标识和描述被评审产品存在的问题（如缺陷）

D. 记录所有的事件、问题

5.（K1）参与正式评审的所有角色包括哪些？（　　）

A. 作者、评审员、记录员

B. 经理、作者、主持人、评审员、记录员

C. 经理、主持人、作者、记录员

D. 主持人、作者、评审员

6.（K1）下面哪个活动是属于主持人的主要职责？（　　）

A. 决定哪些文档需要进行评审

B. 主持文档或者文档集的评审活动，包括制订评审计划、召开会议和会议后对结果的跟踪

C. 负责文档的修改

D. 记录评审会议中的各种事件和问题

7.（K1）下面哪个缺陷是静态分析工具容易发现的问题？（　　）

A. 代码实现和设计要求不吻合

B. 软件的可维护性差

C. 引用了某个没有定义的变量

D. 内存泄漏

8.（K2）下面关于评审的作用，说法正确的是（　　）。

a. 评审能够发现缺陷、缩短开发时间

b. 评审应该尽早开展，可以减少动态测试的成本

c. 静态测试和动态测试是互补的，有些问题要到动态测试的时候才能发现

d. 评审员需要清晰地标注评审问题和结果，以帮助作者改进文档质量

A. a，b

B. a，c

C. a，b，c

D. a，b，c，d