服务验证和测试

ITIL®4 实践指南

AXELOS.com

申明：

- 本文档由长河（微信achotsao）在机译的基础上经初步整理分解，精细化翻译工作正由ITIL先锋论坛组织的ITIL专家团队进行之中，预计到2020年年底之前全部完成。需要下载最终翻译版本请关注微信公众号：IT管理精英圈，或访问www.itil4hub.cn或www.itilxf.com。
- ITIL先锋论坛专家团队只是只是进行了这些著作的语种转换工作，我们并不拥有包括原著以及中文发行文件的任何版权，所有版权归Axoles持有，读者在使用这些文件（含中文翻译版本）时需完全遵守Axoles和TSO所申明的所有版权要求。

内容

1 关于本文件 3
2 一般信息 4
3 价值流和流程 21
4 组织和人员 26
5 信息和技术 29
6 合作伙伴和供应商 31
7 重要提醒 32
8 致谢 33

1 关于本文件

本文档为服务验证和测试实践提供了实用指南。它分为五个主要部分，内容包括：

- 有关实践的一般信息
- 实践的流程和活动以及它们在服务价值链中的作用
- 实践中涉及的组织和人员
- 支持实践的信息和技术
- 用于实践的用于合作伙伴和供应商的注意事项。

1.1 ITIL®4 鉴证方案

从本文档中选择的内容可作为以下课程的一部分进行检查：

- ITIL专家创建，交付和支持
- ITIL专家高速IT。

有关详细信息，请参阅相应的教学大纲文档。

2 一般信息

2.1 目的和描述

关键信息

服务验证和测试实践的目的是确保新的或更改的产品和服务符合定义的要求。服务价值的定义基于客户的输入，业务目标和法规要求，并作为设计和转换价值链实现价值的一部分记录在案。这些输入用于建立可测量的质量和性能或绩效指标，这些指标支持准则保证的定义和测试要求。

服务验证和测试实践涉及减少新的或更改的产品和服务引入运行环境的风险和不确定性。实践通过规划执行此操作并执行适当的测试。

系统越大越复杂，则需要进行的测试越多。但是，由于时间和成本的限制，即使是较小的简单系统也无法进行详尽的测试。因此，选择测试的内容很重要。定义范围和验证的级别以及进行测试时的关键注意事项是：

- 生产或服务必须满足的商定要求
- 影响以及偏离议定要求的可能性。

理解背景中的可能性和影响偏差的要求有助于对测试的重要领域有一个明智的了解。

该实践充满了对正在测试的服务的质量的信心。这与说它是完美的不同。通过测试可以赢得信心，以证明服务将按要求运行，符合要求并且没有重大缺陷。

2.1.1 服务验证

服务验证在生产和服务生命周期（概念和设计）的早期阶段执行。它着重于确认拟议的服务设计满足商定的服务要求，并着眼于为下一阶段（开发，部署和发布）建立客户或用户体验旅程。然后，将通过测试生产和服务组件，产品和服务来验证这些准则。

验证遵循服务要求的结构，通常涵盖功用，功效，体验，可管理性和合规性。其他要求也可能包括在内。

服务验证确保服务验收标准的定义，验证和文档，并告知范围和测试活动的重点。

2.1.2 测试中

基于通过服务验证识别的准则，开发并实施了测试策略和测试计划。

测试策略定义了一种总体测试方法。测试策略可以应用于环境，平台，服务集或单个产品或服务。测试涵盖的生产和服务生命周期阶段在组织内开发的产品和服务与从供应商获得的产品和服务之间可能有所不同。

架构管理，软件开发和管理，项目管理和基础设施和平台管理的更改对服务验证和测试实践产生了很大的影响。敏捷方法，IT基础设施的数字化，面向服务的架构以及软件开发和管理的自动化给服务验证和测试实践带来了新的挑战和机遇。为了满足当今的需求，服务验证和测试应该更快，更灵活并且不断发展。仅当实践与上述实践以及其他实践（包括发布管理，部署管理，事件和问题管理实践）紧密集成在一起时，才有可能。

有效的验证和测试基于测试人员，开发人员和操作团队之间紧密的协作，以及增强的工具和自动化方法。

另一个重要趋势是将验证和测试范围扩展到产品和服务的技术范围之外，包括用户体验和感知。

传统上，服务测试是根据基于需求规范定义的先验知识，通过检查有关软件应如何工作的期望来确认与明确要求相关的期望的行为。今天，测试还涉及探索和发现有关意外情况的信息，例如生产风险和有关以下方面的变量：

- 软件
- 软件解决方案的想法
- 制品从想法中创建
- 用户体验和用户接口设计
- 模型和线框
- 架构和代码设计
- 码
- 工具
- 流程。

2.2 术语和概念

2.2.1 基于风险的测试

基于风险的测试是测试行业中的通用术语。通常，人们将基于风险的测试理解为平均测试（尤其是探索性测试），该测试由与所测试功能和生产组件相关的不同类型的生产风险构成和驱动。

专注于风险是有益的，因为它突出了服务可能如何失败。然后可以对此进行调查，以发现有关该软件及其质量的信息。

通常在软件测试中，人们关注测试的类型。测试类型的示例包括职能型，回归，性能或绩效，安全，可用性，跨浏览器，可访问性，端到端和集成测试。这些类型的测试关注于不同类型的风险。例如，职能型测试着重于职能型风险，而回归测试则着重于软件回归的风险。

尽管他们倾向于考虑10到15种测试类型，但许多团队在测试策略中仅包括5到8种测试类型。因此，由于存在影响服务的生产风险的类型很多，而很少与某种类型的测试相关联，因此将重点放在基于风险的测试上非常重要。

2.2.1.1 发现风险

识别生产风险的服务验证和测试实践活动与确认风险已得到有效解决的活动一样重要和有价值。

在生产的早期阶段进行的服务验证和测试活动生命周期输出有关生产风险，变量，未知数等的信息。相反，在生产生命周期的后期阶段进行的测试活动会发现问题以及有关服务实际情况的其他信息，然后组织可以响应这些信息。即使服务是运行的，组织也应继续发现有关风险，变量和未知数的信息。该反馈仍在继续，但会导致更长的反馈循环返回到想法，用户故事和设计。

例如，在软件开发中，敏捷用户故事制品和客户或用户体验旅程制品很少关注生产风险。这些制品中的文本通常与有关软件功能或互连性的一般期望有关。在定义客户或用户体验旅程时，识别与用户故事相关的风险非常重要。

识别后，应捕获风险。思维导图是用于此目的的常用工具，因为它们创建了一个风险映射，该映射易于访问，轻巧，易读，并可以在整个生产生命周期服务设计活动以及以后的阶段进行探索性测试中使用。

识别不同种类的生产风险可能很困难，但是有一些方法可以构造客户或用户体验旅程并测试理性分析KT法成功的机会，例如：

- 在整体级别上考虑测试的对象，然后仔细地进行测试，包括有形和无形的制品。积极考虑生产，服务或组件：
  - 潜在目的
  - 属性
  - 各种用户
  - 集成零件
  - 架构
  - 等等
- 探索每个方面的变量。
- 识别并讨论与变量有关的生产风险。可以确定的风险示例包括：
  - 可达性风险
  - 可用性的风险
  - 魅力/喜好风险
  - 兼容性风险
  - 环境集成风险
  - 职能型的风险
  - 界面风险
  - 本地化风险
  - 可维护性风险
  - 可观察性风险
  - 性能或绩效的风险
  - 携带风险

- 可靠性风险
- 响应风险
- 可扩展性风险
- 安全风险
- 稳定性风险
- 可测试性风险
- 可用性风险。

- 评估风险，并决定是否要花费更多的时间和精力来减轻或测试它。有关此主题的更多信息，请参考风险管理实践指南。
- 如果存在重大风险，请创建一个风险映射。风险映射是面向服务设计人员和开发人员的制品。它们还有助于阻止测试章程，该章程涉及通过测试特定领域中的特定风险来构造探索性测试。

2.2.2 在不同环境中进行测试

基于风险的方法对于测试环境以及确定测试的位置也很有帮助。

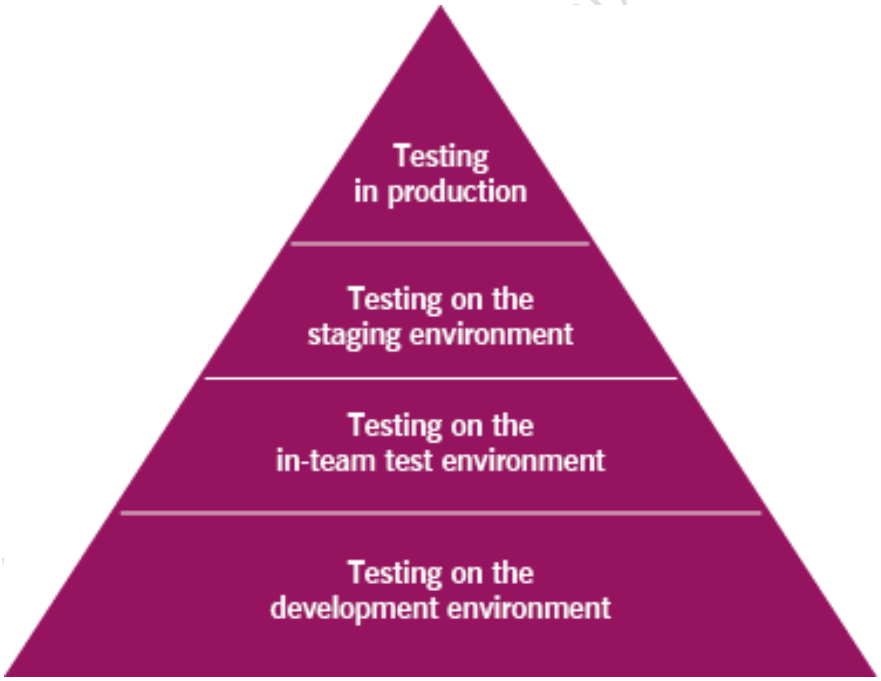
可以在开发环境中测试许多风险。开发在供应环境中的反馈周期非常快，因为通常可以快速编写代码并将其用于测试，以应对多种生产风险，然后在需要时重构代码。但是，某些风险无法在开发环境中进行测试。他们可能需要更严格地集成的环境，例如专用的测试环境。

使用专用的测试环境可能会比较慢，因为创建环境需要时间，并且如果发现任何问题，则反馈环的重构代码会更长，需要在开发环境中进行重新测试，然后将这些修订再次合并到测试环境中。无需重复在开发环境中完成的测试，但是可能需要对开发环境中无法测试的风险进行测试。

有时，使用发布之前的环境（分段环境）是明智的。某些风险只能在共享的测试环境中进行测试，例如与数据流量有关的风险，平台风险或某些集成风险。

最后，某些风险只能在实际的生产环境中进行测试。

图片2.1代表了执行大多数测试的环境。



图片2.1测试三角形

可以在开发环境中尽早测试大多数风险。其余大多数可以在团队测试环境中进行测试。剩余的大多数组件都可以在环境中进行测试。其余的可以在生产环境中进行测试。

2.2.3 断言（脚本）和探究（调查）测试

测试为生产或服务提供了决策信息。信息是已知的还是未知的。

已知信息有两种状态：

- 明确的信息
- 隐性信息。

有两种状态的未知信息：

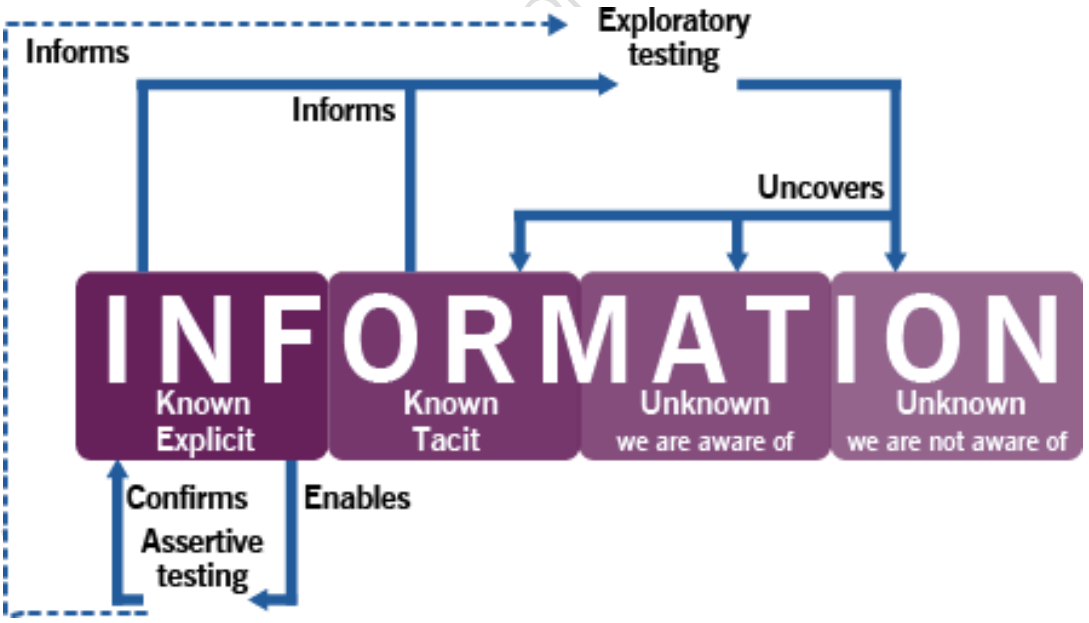
- 已知存在但尚未访问的信息
- 未知的信息。

有关信息类型及其相关实际含义的更多详细信息，请参阅知识管理实践指南。

根据信息的不同状态，关于软件测试有两种观点：

- 断言（或脚本）测试旨在验证组件，生产或服务是否满足基于议定要求的预定义准则。
- 探索性（或调查性）测试旨在发现有关服务，组件，生产，服务或环境的未知信息，以识别预定义准则尚未解决的风险。

两种方法应结合起来并保持平衡；单独遵守一个会减少质量有关产品和服务的信息，这可能会导致管理决策不理想。图片2.2说明了测试如何影响可用信息。



图片2.2测试有助于确认和发现信息

2.2.3.1 断言测试方法

断言测试确认是否满足对服务应该如何设计，开发和执行的明确期望。

这种类型的测试依赖于明确表达和记录的期望（通常以客户或用户体验旅程的形式）。它还需要创建经过测试的人工制品。

可以根据要测试的内容以及组织是否具有所需的工具来手动或自动执行断言测试。无论哪种方式，断言测试都基于已记录的测试脚本，这些脚本以人工或编程语言描述客户或用户体验旅程，测试操作以及通过/失败准则。自动化测试在软件和数字化基础结构中很常见，但是它可以应用于服务的其他方面，包括控件，通信，系统集成以及与用户的交互。

断言测试受到其性质的限制：它只能用于在有限的情况下减少已知和记录的风险。它也受要测试的生产或服务的测试策略的限制；为了提供足够而不是详尽的保证，可以省略一些已知的风险。

2.2.3.2 探索性测试方法

探索性测试基于调查生产，服务或环境中的未知数，目的是发现与服务的感知质量和价值相关的信息。它依赖于横向和批判性思维技能，并且通常基于对可能的生产漏洞和相关威胁的探索。

探索性测试通常被误认为是发起人或赞助人，并且是非结构化的。实际上，它是由称为测试宪章的小型测试任务构成的，这些任务将测试重点放在目标区域上，以调查特定的生产风险。

这种方法对于敏捷开发的背景以及不断增长的信息和组织系统的复杂性至关重要。它在整个生产和服务生命周期中实现了快速的学习和反馈循环，从而实现了产品和服务的持续改进。

2.2.4 连续验证和测试

服务验证和测试实践不仅用于测试可发布的运行的生产或服务。这些活动应该在整个服务生命周期中进行，如图片2.2中所示。

验证和测试活动创建了重要的反馈回路，这些反馈回路告知数字化产品生命周期的每个步骤，如表2.1所示。

表2.1 验证和整个数字化产品生命周期的测试

数字化产品生命周期相和相关的制品	验证	断言测试	探索性测试
主意	-	-	对这些想法及其相关性至客户和组织的需求的探索
史诗，用户故事，功能，促成因素等	史诗般的验证，用户故事以及功能/启用器，	-	探索史诗，用户故事以及可识别的功能

	客户或用户体验旅程的开发用于UX / UI设计以及架构和代码设计		不一致和错失的机会
UX / UI设计	设计的验证，以确认它们基于史诗，用户故事和功能，并且与定义的准则相匹配 架构和代码设计客户或用户体验旅程的定义或更新	测试UX / UI线框，以确认它们符合架构和设计策略和准则（如果适用）	探索UX / UI线框以识别不一致和错过的机会
架构和代码设计	设计的验证确认它们基于史诗，用户故事和功能并与定义的准则相匹配 客户或用户体验旅程的开发代码	对架构和代码设计制品进行测试，以确认它们满足架构和设计策略和准则（如果适用）	探索架构和代码设计制品以识别不一致和错过的机会
代码单位	该代码的验证，以确认它是根据商定的设计开发的，已经完成，并且遵守商定的架构标准 适用于运行的软件的客户或用户体验旅程更新	自动化的（有时是手动的）单元测试，以确认每个单元均按照商定的准则进行设计	同行评审，结对编程和其他探索性测试，以识别客户或用户体验旅程未涵盖的错误和机会
运行的软件	基于同意的准则的软件的验证，以确认它是否完整，并遵守同意的架构标准 开发以及部署和发布的客户或用户体验旅程更新	根据公认的准则，进行自动化且有时是手动的单元测试以确认软件是否按设计执行	评审和软件探索，以识别客户或用户体验旅程未涵盖的错误和机会
部署和发布流水线	部署和发布工具的验证，流程以及确认它们是否符合协议的方法 要求和遵循	对流水线工具和流程进行自动化（有时是手动）测试	探索流水线工具和流程，以发现未发现的错误和机会

	适用的政策和准则 部署和发布的客户或用户体验旅程更新	确认他们按照约定工作	由客户或用户体验旅程覆盖
软件部署	验证的完整性和正确性 发布客户或用户体验旅程的更新和回归测试准则	对已部署的制品进行自动化测试，以确认它们是否符合约定的客户或用户体验旅程	探索已部署的制品和环境，以识别客户或用户体验旅程未涵盖的错误和机会
服务发布	已发布的服务的验证，以确认它是否完整并且符合商定的设计规范 实时服务验证准则的更新	服务运营的自动和手动测试，包括用户销售活动测试	探索发布的服务，以发现客户或用户体验旅程未涵盖的错误和机会
运维中的服务	服务质量的验证（功用，功效和体验级别）基于公认的准则和服务级别管理信息。	回归测试以确认以前的测试结果仍然有效	混沌工程和正式的服务质量控件未涵盖的混沌工程至探索服务漏洞以及其他错误和机会

2.3 范围

服务验证和测试实践的范围包括：

- 将产品或服务的需求转换为部署和发布管理客户或用户体验旅程
- 建立测试方法并为新的或更改的产品和服务定义测试计划
- 通过测试消除了风险以及新产品或服务变更的不确定性
- 通过测试发现有关新的或更改的产品和服务的新信息
- 不断审查测试的方法和方法以测试改进和效率

尽管活动和责任范围仍与服务，验证和测试密切相关，但许多活动和职责范围并未包含在服务验证和测试实践中。表2.2中列出了这些内容，以及对可以找到它们的实践的引用。重要的是要记住，ITIL实践只是价值流的背景中使用的工具的集合；根据情况，应将它们组合在一起。

表2.2与其他实践指南中描述的服务验证和测试实践相关的活动

实现价值	实践指南
建立新的或更改的生产或服务的功用和功效的详细要求 分析现有功用和功效选件之外的服务新要求	业务分析
维护财务控制过测试定义测试预算	财务管理
开发和管理软件	软件开发和管理
开发和管理基础架构	基础设施和平台管理
运行的与用户的交流和反馈的收集	服务台
部署服务和组件	部署管理
发行服务	发布管理
正在进行的管理和改进的实施	持续改进

2.4 实践成功因素

实践成功因素

实践的复杂职能型组件，是实践实现其目的所必需的。

实践的成功因素（PSF）不仅仅是一项任务或实现价值，因为它包括所有服务管理四维模型的组件。活动的性质和实践中PSF的资源可能有所不同，但它们共同确保实践有效。

服务验证和测试实践包含以下PSF：

- 定义并同意验证的方法以及组织的产品，服务和组件的测试，以符合组织的速度要求和服务的质量更改
- 确保新的和更改的组件，产品和服务符合商定的准则

2.4.1 定义并同意验证的方法以及组织的产品，服务和组件的测试，以符合组织的速度要求和服务的质量更改

服务验证应该建立一种方法来捕获任何生产，服务和组件的所有功用和功效需求。此方法应涉及不同的利益相关者以及利益相关者的信息源，例如客户和用户要求和反馈，业务要求，内部和外部合规性和法规要求，风险和安全以及其他要求源。这种方法还应该提出将需求转换为服务的客户或用户体验旅程的方法。

测试策略定义了考虑项目的目标应如何执行测试。测试规划应该基于测试策略。测试策略还定义了测试管理方法，包括如何组织和控制测试。

测试策略定义了范围中的测试阶段（或级别）和类型。测试阶段包括：

- **开**发人员用来验证他们开发的内容符合要求的单位。一个单元通常是整个系统中经过隔离测试的组件。
- **集**成当开发足够完整以开始集成不同的系统时进行，与系统之间集成的测试有关。
- **系**统在经过验证可以集成系统的组件后，系统测试考虑了系统的端到端功能。
- **销**售活动用户销售活动测试（UAT）是正式的测试阶段，最终用户在此阶段验证并确认要交付的产品是否满足其要求。

在每个测试阶段中，测试策略都必须考虑哪种测试类型合适。测试类型包括：

- **职**能型测试即将交付的系统会做什么。
- **非**职能型测试系统与其职能型要求没有直接关系的方面。非职能型的常见方面是：
  - **性**能或绩效行为在正常条件下。
  - **用**增加的负载加载行为。
  - **接**近运行的上限时，应给行为施加压力。
  - **安**全授权和身份验证系统控件。
  - **可**用性系统的用户如何将契动与系统配合使用。
- **回**归新的发展（进展）和错误修复（调试）可能会引入意外的系统行为。回归测试旨在验证变更之后，系统是否仍按要求运行。

测试计划定义了每个测试阶段的详细活动，估计和时间表。因此，测试策略定义了整个范围和方法，而测试计划则详细说明了每个测试阶段。表2.3对此进行了概述。

表2.3 测试策略

	测试策略			
类型/级别	单元	集成	系统	燕麦
职能型	单元测试计划	集成测试计划	系统测试计划	UAT test plan
非职能型				
回归				

2.4.2 确保新的和更改的组件，产品和服务符合约定的准则

没有两个项目是相同的，并且测试策略必须适合于相关的项目和组织结构。每个测试策略都应致力于：

- 通过平衡效果和效率在可利用的时间内实现最佳的测试覆盖范围
- 务实并适合方案的需求，可用资源和可用技能
- 与开发方法论，所采用的技术以及正在开发的系统的性质保持一致
- 尽早建立对软件交付的高度信心
- 确认提供的软件的准确性（职能型属性）
- 减轻与实施新软件有关的业务风险的水平
- 随着项目的展开，继续使用改进和优化，测试，流程
- 识别与测试相关的风险，问题和脆弱区域，并提供适当的建议。

为此，测试策略需求解决：

- 测试组织
- 测试规划和控制
- 测试分析和设计
- 测试的准备和实施
- 测试进度和报告
- 事件管理
- 测试关闭并退出准则。

测试策略必须考虑采用的开发方法。瀑布开发模型通常允许对捕获的用户要求进行早期静态测试和验证。在编码开始之前，一种更具迭代性的方法可能无法提供完整的用户要求。适当的测试策略需求。

测试策略还必须考虑所涉及的系统/ 服务的类型。例如，在年底测试财务系统所需的方法与测试电子商务网站的方法截然不同。考虑测试环境的基础很重要：验证质量所需的流程，系统，资源和管理。

测试不仅限于软件制品；数据迁入，培训，运行的准备情况，发布管理和报告是需要特别关注测试的其他领域。

2.4.2.1 测试组织

那些测试和系统应该与开发系统的那些独立。测试人员和开发人员的心态不同。开发人员通常旨在证明自己拥有的东西

开发的产品符合要求：测试人员旨在证明已满足要求，并且未引入其他问题。

测试组织应鼓励将测试的效果与改进分开。应该明确定义参与测试的人员的角色和职责，包括测试，管理和测试分析人员角色，以及涉及事件管理，配置管理，变更控制，部署和发布的支持角色。

2.4.2.2 测试规划和控制

如果软件开发生命周期遵循基于冲刺的方法，则冲刺规划应该包含测试。即使要求使用存根和驱动程序，每个冲刺都应交付可测试的制品，然后应将它们放在冲刺范围中。

可用于测试的发行版包括进度有效载荷（新的事物）和回归影响（需要测试以确认它们可以继续按照要求继续使用职能的事物）。

在渐进和回归方面，对任何发布的威胁通常包括：

- 引入了满足需求（渐进和回归威胁）的新功能。该威胁源自现有的项目。
- 错误修复了新功能（渐进和回归威胁）。该威胁源自现有的项目。
- 产品服务的修补程序（回归威胁）。该威胁源自生产服务提供者。
- 维护发布用于生产服务（回归威胁）。该威胁源自生产服务提供者。

2.4.2.3 测试分析和设计

仅就总覆盖率的百分比报告测试进度不支持知情的风险评估。为了使测试上的报告取得有意义的进展，应使测试符合方案的可交付成果和要求。

每个要测试的发布都包含一个有效负载。有效负载可以分为有效负载元素（PE）。每个PE都有一个离散的测试包，据报告。

例如，在基于Web的订单条目系统中，方案计划将冲刺定义为：

- PE 1：提供客户短代码查找工具（带有回归影响的渐进交付物）
- PE 2：允许通过信用卡付款（进度交付物和回归影响）
- PE 3：将总订单价值作为自动更新的字段交付到订单输入网页上，从而代替了用户手动使用“计算订单总数”职能（渐进式交付物和回归影响）的需求
- PE 4：此冲刺中提供了多达十个开发小时的错误修复，以解决先前sprint（渐进和回归交付物）中的未解决错误。

测试已经审查了方案时间表，获得了要求和职能型设计文档的副本，并估计要覆盖所有冲刺的进度交付品，需要45个测试案例。另外，在考虑冲刺的回归风险时，测试确定了另外25个测试案例，因为冲刺的发展影响了系统的核心功能。

如果计划对一个职能型测试阶段进行两个三天的周期测试，那么测试的总包装尺寸为70个测试盒。

在第一个测试周期内进行报告可能会指出，第二天结束之前80%的测试案例已经运行并通过。但是，如果不确认已运行哪些测试以及仍要运行哪些测试，就不能认为这是一个好的结果。运行并成功通过所有回归测试将表明冲刺没有引入回归问题。

测试功能并记录成功的结果，但不执行回归测试会导致人们对新开发的产品充满信心，但并不表示系统并未受到损害。重要的是要考虑解决PE并预先报告它们。

在上面的示例中，知道时间表中还有一天需要测试，有十项尚未完成的回归测试和五项尚未完成的进度测试，这将成为测试其余工作重点的重点。

为了进一步细分回归测试，从业人员可以考虑测试下系统的核心功能并定义重点领域。订单输入可能是一个重点领域，客户计费可能是另一个重点领域。通过对重点领域的回归测试进行分类，可以制造出更好的评估，从而获得出色的测试风险。

2.4.2.4 阶段和周期

定义了所需的测试范围之后，从业人员可以按PE和重点领域考虑测试的时间表。部署和发布活动至测试环境完成后，应立即开始测试。重要的是要考虑对PE和重点领域进行测试的顺序。默认情况下，应尽快将测试最复杂或最新的发展（通常是最高的风险）发展为测试。

确定了测试范围估计至测试的执行持续时间是必需的，并且考虑到缺陷影响测试的执行情况，缺陷率需求也将被估算并计入测试执行计划中。

规划很难测试新的系统。对于计划而言，基于结果的估计非常重要，并且由于系统已通过迭代测试，因此可以完善这些估计。

根据测试案例来表达影响缺陷可能很有用。

例：

测试团队已分析了最新销售订单处理系统中的下一个发布。对实体和重点领域的分析已经完成。已经确定了172个测试案例可以涵盖PE，但是已经对150个测试案例的标准回归包以及由于PE的性质而进行的任何其他60个回归测试进行了范围划分，得出的总回归包大小为210 测试案例。在此示例中，回归测试是手动运行的。

在测试规划，假定所有PE的20%（34）和10%的回归测试（21）将导致缺陷。

并非所有缺陷都是平等的。有些很容易分类和修复，而有些则微不足道。缺陷被分类为复杂，标准或琐碎的问题，并分配了应解决的时间。

表2.4概述了估计总共有55个缺陷的信息。

表2.4缺陷信息示例

分类	固定利率	假设百分比	假定缺陷	加权因素	调整后总计
复杂	3天	50	27（55的50%）	2	54 (27 * 2)
标准	2天	30	17（55的30%）	1.5	26 (17 * 1.5)
不重要的	1天	20	11（55%的20%）	1	11 (11 * 1)
总					91

加权因素可用于调整某些缺陷的复杂性。调整后的总数可以视为要执行的其他测试案例。将这些添加到测试的情况下，范围会为缺陷修复留出余地。

对于测试执行规划，需要关于测试员运行测试需要多长时间的假设。

在“每天每个测试人员测试”（TPTPD）的基础上进行工作可能会很有用。与缺陷估计一样，并非所有测试都是相等的：某些测试比其他测试花费更长的时间。

表2.5根据382个测试案例（172个PE和210个关注区域）的示例测试范围概述了测试案例信息。

表2.5示例382 测试案例的结果

TPTPD	假设百分比	测试案例数	测试一位测试人员的持续时间
5	40%	153	31天
3	35%	134	45天
1	25%	96	96天

表2.5显示了估计的全覆盖测试持续时间的余量。包括更多测试仪或范围的减少将减少测试的持续时间。

在以上示例中，PE和重点领域已得到同等对待。通过分别估计这些值，可以实现更高的精度。

测试时间表应按阶段组织，并分成一个或多个周期。每个周期都明确定义了范围中用于测试的PE和重点领域，通常首先测试最高的风险。

较大的测试相通常采用3周期方法。表2.6概述了3周期方法。

表2.6 3周期测试阶段

周期1	周期2	周期3最终测试周期- FTC)
高风险/ 优先级PE	较低的优先级PE	最终修复
高优先级重点领域	优先级的下焦点区域	优先关注区域清洁运行
	周期1的修复	优先PE 待办项
	周期1的待办项	

2.4.2.5 测试的准备和执行

执行测试的规划可能是一项艰巨的任务。需要预见许多因素，以便测试的执行可以继续进行。需要计划诸如环境（s）设置，数据创建，用户帐户和角色配置之类的因素。

计划计划活动的活动，并定义测试准备阶段的角色和职责。每日站起来监视进度可能很有用。测试准备本身应作为项目进行处理，需要通常的项目管理技术来确保成功。

在开始执行测试之前，需要成功运行同意的环境和系统验证测试。此外，应严密保护准备好的测试数据。生成测试数据通常很耗时。仅当测试下的系统可以支持测试范围时，才应使用已创建的数据。

开始执行测试时，重要的是要确保动力保持不变。通常，测试的初始阶段会确定阻止程序；这应该是可以预期的。适当的解析器组应位于备用上，并具有增强的支持，以快速解决早期问题。通常，解析程序组可以保持忙碌状态，例如在备用上支持系统的下一个发布时，支持正在进行的开发工作。

通常，测试的最大威胁是事件。为确保始终专注于测试执行，缺陷经理的责任是确保快速解决缺陷并确定优先级以支持测试执行进度。

作为缺陷管理流程的一部分，需要明确定义严重性和优先级。严重性，以衡量影响的缺陷对发布和系统的能力。优先级，以支持测试计划。但是，重要的是要记住，关键严重性缺陷不一定是最高的优先级。

如果开发使用冲刺方法，则每个冲刺的小时数可以分配给测试缺陷调试（可以分配解析器组资源来支持测试）。测试缺陷经理应确保调试，修复和部署缺陷以进行重新测试，以支持测试执行计划。

测试缺陷经理应该监视KPI，例如发布中确定的缺陷数量，以便确定可能需要额外培训和支持的区域。KPI专注于那些有可能最大化改进点的领域。

2.4.2.6 评估出口准则和报告

至关重要的是，从业人员应该知道何时停止测试。

定义为测试分析和设计阶段的一部分的出口准则用于标识所测试的系统何时足够好。测试报告将引用出口准则和项目

成果，例如测试是否将按时完成。如果报告指示问题，则需要更正性能或绩效。从PE和关注区域的角度考虑测试的执行很有用。通过这种方式，报告使您可以更直观地了解所携带的风险。

如果在正常发布的回归测试中重复关注相同的特性区域，则标准化其测试的执行时间表，并将发布中测试的执行进度与最新的X版本进行比较，可以清楚地表明测试的轨迹。

至少，每天和每周测试状况报告都需要详细说明测试的执行范围以及PE和关注区域（回归）的通过/失败率。这些报告将提供有关测试能力的性能或绩效的度量标准，例如实际观察到的TPTPD，按严重性列出的缺陷率，调试率和首次修复率，并评估测试执行的轨迹。

2.4.2.7 测试关闭

一旦完成了最终测试（已满足退出准则），就可以启动测试关闭。根据系统的性质，这可能在测试结束时触发。在其他情况下，这可能会跟随成功的前期支持（ELS）或部署和发布之后的超级护理阶段。通常，关键的测试资源会在预定义的ELS阶段保留，而系统会稳定在生产环境中。

这将涉及：

- 从测试活动正式释放资源并过渡到日常业务（BAU）操作
- 归档和索引测试资产，包括测试策略，计划，报告和脚本
- 过渡到BAU可能需要参考测试资产，尤其是在考虑BAU支持运维所需维护的回归测试包时。

应该记下从测试中学到的经验教训，包括做得好的和不好的。为了支持持续改进实践，重要的是要确保将汲取的教训纳入测试策略。详细说明正在/未重复/未解决的课程。默认情况下，那些未解决和/或重复的课程应过渡到下一堂课程。

2.5关键指标

应该在每个实践所贡献的价值流的背景内评估ITIL惯例的效果和性能或绩效。与任何工具的性能或绩效一样，只能在应用程序的背景内评估实践的性能或绩效。但是，设计和质量的工具可能会有很大差异，这些差异定义了工具的潜力，或根据用途使用能力才有效。有关度量标准，关键性能或绩效指标（KPI）的其他指南以及可以帮助您解决此问题的其他技术，请参见度量和报告实践指南。

服务验证和测试实践的关键指标已映射到其PSF。它们可以用作价值流的背景中的KPI，以评估实践对这些价值流的效果和效率的贡献。表2.7给出了一些关键指标的示例。