TMMi® FOUNDATION

# DevOps 世界中的 TMMi®

## #TMMi 的成功

中文发布版 1.1

由 TMMI®基金会中国分会负责翻译

www.tmmi.org

TMMi®旨在独立于生命周期。本文档解释了如何在 DevOps 环境中有效地应用 TMMi®测试改进模型。将逐一讨论 TMMi®2 级和 3 级过程域及其目标。并说明这些过程域与 DevOps 的关系，以及实践在 DevOps 中通常是什么样子的。如果某项实践因为没有附加价值而不希望在 DevOps 环境中执行，也会明确说明。请注意，本文档并不是完整的 DevOps 测试大纲，而是“仅”展示如何在 DevOps 背景下解释 TMMi®目标，并为进一步研究提供思路和方向的文档。TMMi 在 DevOps 领域也并非旨在作为完整 TMMi 模型的替代描述。TMMi 在 DevOps 中的应用应始终与 TMMi 参考模型文档结合使用。

# 贡献者

Katalin Balla (Hungary)

Suresh Chandra Bose (USA)

Kari Kakkonen (Finland)

Mattijs Kemmink (The Netherlands)

Rik Marselis (The Netherlands)

Szilard Szell (Hungary)

Erik van Veenendaal (The Netherlands)

# 修订历史

本节总结了截至本文档发布的主要修订。

本节仅提供信息。

| 版本 | 修订说明 |
| --- | --- |
| V1.0 | “DevOps 世界中的测试”文档的初始版本 |
| EN1.0_CN1.0 | 本文档“DevOps 世界中的 TMMi®”中文版本地化完成 |
| EN1.1 | 在 DevOps 背景下增加了对 TMMi3 级过程域、目标和实践的解读，同时添加了 DevOps 专用术语的术语表。 |
| EN1.1_CN2.0 | 本文档“DevOps 世界中的 TMMi® v1.1”中文版本地化完成 |

## 参与中文本地化的 TMMi®专家

| 主要内容 | 翻译专家 | 评审专家 | QA 专家 |
|---|---|---|---|
| 英文封面&第 1-31 页 | 李颖丽 | 张楠婕 | 周震漪 |
| | 张楠婕 | 李颖丽 | |
| | 叶岚 | 刘晓玲 | |
| | 刘晓玲 | 叶岚 | |
| 英文第 32-40 页 | 李颖丽 | 刘晓玲 | 叶岚 |
| 英文第 41-49 页 | 刘晓玲 | 李颖丽 | |

# 目录


目录 ..... 5

1 DevOps 世界中的 TMMi® ..... 7

1.1 介绍 ..... 7

1.2 DevOps 和 DevSecOps ..... 7

1.3 测试成熟度模型集成（TMMi®） ..... 8

1.4 DevOps 环境下的测试 ..... 9

1.4.1 DevOps 环境下的测试挑战 ..... 9

1.4.2 TMMi®和 DevOps ..... 10

1.4.3 ISTQB 与 DevOps 大纲的关系 ..... 11

2 TMMi®2 级已管理 ..... 12

2.1 过程域 2.1 测试方针与策略 ..... 12

2.1.1 SG1 建立测试方针 ..... 12

2.1.2 SG2 建立测试策略 ..... 13

2.1.3 SG3 建立测试绩效指标 ..... 16

2.2 过程域 2.2 测试策划 ..... 17

2.2.1 SG1 执行风险评估 ..... 17

2.2.2 SG1 建立测试方法 ..... 18

2.2.3 SG3 建立测试估算 ..... 20

2.2.4 SG4 开发测试计划 ..... 21

2.2.5 SG5 获得测试计划的承诺 ..... 21

2.3 过程域 2.3 测试监督和控制 ..... 22

2.3.1 SG1 根据计划监督测试进度 ..... 23

2.3.2 SG2 根据计划和预期监督产品质量 ..... 24

2.3.3 SG3 管理纠正措施直至关闭 ..... 25

2.4 过程域：测试设计与执行 ..... 25

2.4.1 SG1 使用测试设计技术执行测试分析与设计 ..... 26

2.4.2 SG2 执行测试实施 ..... 27

2.4.3 SG3 进行测试执行 ..... 28

2.4.4 SG4 管理测试事件至关闭 ..... 29

2.5 过程域 2.5 测试环境 ..... 30

2.5.1 SG1 开发测试环境需求 ..... 30

2.5.2 SG2 执行测试环境的实施 ..... 30

2.5.3 SG3 管理和控制测试环境 ..... 31

3 TMMi®3 级已定义 ..... 32

3.1 过程域 3.1 测试组织 ..... 32

3.1.1 SG1 建立测试组织 ..... 32

3.1.2 SG2 为测试专家建立测试职能 ..... 32
3.1.3 SG3 建立测试职业路径 ..... 33
3.1.4 SG4 确定、计划和实施测试过程改进 ..... 33
3.1.5 SG5 部署组织的测试过程并合并经验教训 ..... 34
3.2 过程域 3.2 测试培训方案 ..... 35
3.2.1 SG1 建立组织测试培训能力 ..... 35
3.2.2 SG2 提供必要的测试培训 ..... 36
3.3 过程域 3.3 测试生命周期与集成 ..... 36
3.3.1 SG1 建立组织测试过程资产 ..... 36
3.3.2 SG2 集成测试生命周期和开发模型 ..... 38
3.3.3 SG3 建立主测试计划 ..... 38
3.4 过程域 3.4 非功能测试 ..... 39
3.4.1 SG1 执行非功能产品风险评估 ..... 39
3.4.2 SG2 建立非功能测试方法 ..... 40
3.4.3 SG3 制定非功能测试分析与设计 ..... 41
3.4.4 SG4 执行非功能测试实施 ..... 42
3.4.5 SG4 执行非功能测试 ..... 42
3.5 过程域 3.5 同行评审 ..... 43
3.5.1 SG1 建立同行评审方法 ..... 43
3.5.2 SG2 执行同行评审 ..... 45
**词汇表 DevOps 特定术语 ..... 47**
**参考文献 ..... 50**

# 1 DevOps 世界中的 TMMi®

## 1.1 介绍

本文档解释了如何将 DevOps 软件开发方法与 TMMi®测试过程改进模型相结合，并进一步阐述了这种结合是实现业务目标的可能途径。它还解释了如何在 DevOps 环境中有效地应用 TMMi®。为实施 TMMi®实践的传统测试方法提供了经过验证的基于 DevOps 的替代方案。本文档介绍了 TMMi®如何帮助 DevOps 组织，无论该组织在实施 DevOps 过程中有多成熟，通过提供关键测试实践的提示，而这些关键测试实践经常随着组织的发展和项目压力的增加而容易被忽略。本文档还将逐一讨论每个 TMMi®过程域及其特定目标，并说明它们与 DevOps 的关系以及相关实践通常是什么样子的。如果某项实践因为没有附加价值而不希望在 DevOps 环境中执行，也作了明确说明。

在当今快节奏的 DevOps 世界中，持续集成、持续交付和自动化驱动着软件开发，在此，确保每个阶段的质量是至关重要的。测试成熟度模型集成（TMMi®）提供了一个结构化的框架来评估和改进测试实践。随着 DevOps 团队在不牺牲质量的情况下努力实现快速发布，整合 TMMi®后提供了一个路线图来增强测试成熟度、与业务目标保持一致，并交付结果。TMMi®和 DevOps 之间的协同作用创造了一种平衡的方法来加速开发，同时保持高标准的质量和可靠性。

本文档有许多目标受众，主要群体是：

- 成熟的传统组织，它们希望在保持过程成熟度的同时向 DevOps 转型。
- 不太成熟的组织，它们希望开始应用 TMMi®来提高测试成熟度并向 DevOps 转型。
- 敏捷组织，它们已经拥有结构化敏捷测试过程，现在希望向 DevOps 转型。
- DevOps 组织，它们是成功且正在成长的 DevOps 组织。因此，他们需要一定的过程成熟度，同时希望保持敏捷的好处。

本文档不可脱离原始 TMMi®模型独立使用，而是旨在为 DevOps 环境中实施测试过程改进的人员提供关于 TMMi®各项目标及实践的解释说明的指导。本文档是对原始 TMMi®模型的补充，应作为补充文档使用。

## 1.2 DevOps 和 DevSecOps

DevOps 是一种哲学和一套实践，旨在打破软件开发与 IT 运维团队间的隔阂。通过培养协作和共同责任的文化，DevOps 致力于流水型软件的开发、部署和维护。其核心理念是打破开发和运维间的传统壁垒，实现更高效、更有效的软件交付。DevOps 的核心在于自动化和持续化过程。自动化在减少手动任务、减少错误和提高速度方面发挥着至关重要的作用。包括通过持续集成（CI）和持续部署（CD）实践，实现自动化代码变更、测试及部署过程。另一关键要素是 DevOps 文化转型：即自主性、领导力、协作、高频实验、缩短开发周期、提升部署频率以及构建更可靠的发布，确保与业务目标及客户高度契合。

DevOps 方法带来了许多好处，包括更快的上市时间、团队之间更好的协作、更高的软件质量和可靠性，以及更高的可扩展性和灵活性。通过更快地检测和解决问题，DevOps 还可以减少停机时间并增强系统弹性。最终，DevOps 不仅关乎技术和过程，也关乎文化变革。它需要转变思维方式，让开发和运维团队朝着共同的目标努力，接纳反馈，并不断努力改进。通过整合这些元素，DevOps 帮助组织更高效、更可靠、更安全地交付软件。

**DevSecOps**

DevSecOps 是开发、安全和运营的简称，是一种将安全实践集成到 DevOps 过程中的方法。该方法旨在确保从初始开发到部署及后续的整个 IT 生存周期中，安全成为一项共同责任。DevSecOps 致力于更快速、更高效地交付安全软件，从而降低安全漏洞风险。

## 1.3 测试成熟度模型集成（TMMi®）

TMMi®框架由 TMMi®基金会开发，作为测试过程改进的指南和参考框架，旨在解决对测试经理、测试工程师、开发人员和软件质量专业人员至关重要的问题。在 TMMi®中，"测试"被广泛定义为涵盖所有与软件产品质量相关的活动。TMMi®使用成熟度级别的概念进行过程评估和改进。并定义了过程域、目标与实践。应用 TMMi®成熟度标准可以优化测试过程，实践证明其对提升产品质量、测试工程效率及缩短周期时间具有显著成效。TMMi®的开发是为了支持组织评估和改进其测试过程。

TMMi®采用阶段型架构进行过程改进，它包含一个组织在测试过程从临时的和未管理的阶段或级别发展到已管理、已定义、已测量，直至优化阶段或级别。实现每个阶段确保了该阶段的所有目标都已实现，并为下一阶段奠定基础。TMMi®内部结构包含丰富的测试实践，可以通过系统化学习与应用来支持持续改进的高质量测试过程。TMMi®中有五个级别，规定了测试过程改进的成熟度等级和进化路径。每个级别都有一组过程域，组织必须实施这些过程域才能在该级别实现成熟度。TMMi®每个成熟度级别的过程域如图 1 所示。

TMMi®的核心原则是，它作为通用模型适用于各类生存周期模式与环境。TMMi®定义的大多数目标与实践已被证明适用于顺序和迭代生存周期模型，也包括敏捷。然而，在模型的最底层，很多子实践及示例会因采用的生存周期模型存在（显著）差异。注意，在 TMMi®中，只有目标是强制性的，实践不是。TMMi 可从 TMMi®基金会官网免费获取，也可以以出版书籍的形式获得。该模型已被翻译成多种语言，包括西班牙语、法语和中文。


图 1 TMMi®成熟度级别与过程域

## 1.4 DevOps 环境下的测试

### 1.4.1 DevOps 环境下的测试挑战

在 DevOps 环境中，由于持续集成与持续交付（CI/CD）流水线（pipelines）的特性、开发与运营团队的协作以及快速迭代的开发周期等给测试带来了独特挑战。一些关键的测试挑战包括：

- 持续测试：必须快速频繁地执行测试，以匹配 CI/CD 流水线（pipelines）的节奏，高度依赖自动化测试，这需要稳定可靠的测试脚本和框架。
- 环境管理：确保适当的测试环境管理和为持续测试提供实际的测试环境可能很困难。生产环境和测试环境往往是不一样的，这可能会导致未被发现的问题，另一个挑战是跨多个环境管理和维护一致的配置。环境通常应按需自动部署和配置，并在测试后删除。
- 测试数据管理：虽然测试数据应该是类似生产级的、具有一致性的、按需部署的，同时也要满足隐私合规要求。
- 集成与回归测试：集成多个具有复杂相互依赖关系的组件和服务，因此对其进行测试具有挑战性。还需通过全面回归测试套件确保新变更不会影响现有功能。
- 工具和基础设施：将各种测试工具与 CI/CD 流水线和其他 DevOps 工具集成，并确保测试基础设施能够扩展以处理增加的负载和复杂性。

- 文化和组织挑战：促进开发、测试和运营之间的合作，传统上，这些部门可能以相互矛盾的激励措施各自为政。需确保团队成员具备处理自动化测试、持续集成和部署实践所需的技能。

- 性能测试：在受控测试环境中模拟真实负载和条件，同时将性能测试集成到 CI/CD 流水线，以实现持续监测和优化性能。

- 信息安全性测试：随着流水线的快速发展，信息安全性测试往往会被边缘化。在不减缓开发速度的情况下集成信息安全性测试（DevSecOps）并确保及早发现漏洞，这需要专门的工具和策略，这对许多团队来说都是一个挑战。信息安全性测试需要在早期引入，并使用自动化工具融入整个开发生命周期，以执行静态和动态信息安全性测试。

- 反馈和报告：基于测试结果向团队提供快速、清晰和可操作的反馈。生成全面的报告，提供对构建质量的见解，同时保持过程的敏捷性。

- 左移实践：通过在生命周期的早期将任务向左移动来提高质量。测试左移意味着在过程的早期进行测试，例如通过（代码）评审、静态分析和全面的单元测试。

- 右移实践：由于尽可能频繁地将变更部署到生产中是关键，因此测试策略需平衡持续集成（CI）环节的测试活动与持续交付（CD）中对生产系统实际运行的观察。

- 监测和反馈循环：在 DevOps 中，测试不会随着部署而结束；持续监测和收集生产环境的反馈至关重要。建立有效的监测工具和解读生产数据以发现缺陷并优化性能往往被忽视或发展不力。

### 1.4.2 TMMi®和 DevOps

错误观点认为 TMMi®与 DevOps 方法互不相容。DevOps 方法和 TMMi®不仅可以共存，而且成功集成后能带来显著效益。还有一个挑战是以不同的方式看待测试，将其完全融入开发过程，并建立 CI/CD 流水线，以及理解这在"测试“改进计划的背景下意味着什么。请注意，TMMi®的“i”指的是测试应该是软件开发的一个组成部分，而不是被视为完全独立的环节。使用 TMMi®模型可以提醒团队关注那些在 DevOps 环境中经常被“遗忘”的关键测试实践，例如产品风险分析。本文档将通过实例展示 TMMi®和 DevOps 方法可以有效地协同工作。

在实施 TMMi®时，必须考虑到 TMMi®模型的意图不是将一套实践“强加”给组织，也不是将其作为必须“证明合规性”的标准来应用。如果使用得当，TMMi®可以帮助企业根据业务目标定位最具改进价值的测试领域。这与采用何种生命周期模型无关。务必牢记，TMMi®实践是期望组件，它可以通过相对于已定义的 TMMi®实践的所谓“替代”实践来实现。始终思考，此实践的意图是什么？理论依据何在？以及它如何为业务增加价值？通常在 DevOps 环境中，实践的意图往往已通过替代方式实现。通常，“任何”解决方案都是合规的，只要它是由业务需要驱动的！使用 TMMi®时切忌教条主义——这原本就违背其设计初衷。始终根据实际情况来解读 TMMi®的目标和实践。一般来说，先厘清具体业务场景下的过程要求，才能合理制定过程改进的优先级策略。

TMMi®和DevOps之间出现的大多数冲突要么是基于历史上TMMi®对“良好实践”实施形态的固有认知，要么是基于对DevOps实践的基本原理的误解，即它们应该如何支持DevOps的工作方式和价值观。TMMi®专家，包括（主任）评估师，需要重新审视并可能重新思考那些可能无意中传达的信息，即符合 TMMi® 的“良好实践”在实施时应该是什么样子。当DevOps方法与TMMi®过程恰当结合时，将促进测试实践的有效实施，而不应将其删除。请注意，除了 TMMi®中的特定（测试）实践外，还有通用实践。通用实践的目的是支持过程域的制度化，这实际上意味着确保组织具备相应的基础设施，以便在新员工入职或组织内部发生其他变化时支持该过程域。

从基于传统的软件开发模式转向 DevOps，这也可以对当今所定义的过程进行裁剪，使之更精简和优化。通过这种方式，基于 TMMi® 的组织将从 DevOps 所蕴含的敏捷和精益思维方式中受益。人们倾向于对 TMMi®模型进行过度解读，从而创造出不必要的、无附加值的过程和工作产品。回归根本和改进目标，按照 TMMi® 模型的初衷来使用它，将有助于使真正有价值的过程与实际的过程要求和目标保持一致。以敏捷/精益的思维模式来精简和优化过程，将形成真正反映人们实际操作的过程，并确保只收集真正有用的数据。这种思维方式也会让人们把注意力放在尽可能简单的事情上，这通常并不容易，但会给那些实施 TMMi®的组织将带来收益。在 DevOps 中，通过DORA绩效指标，例如，变更交付时间、部署频率、变更失败率和失败部署恢复时间[DORA16] [DORA24] ，进行的改进通常会通过能够迅速采取行动的小型授权团队来实现，这也是 TMMi® 从应用 DevOps 中获益的另一种方式。尽管 TMMi® 是全面的，但组织无法应用其每一项内容。为了取得成功，组织必须明确自身需优先关注的关键测试实践与改进领域。

### 1.4.3 ISTQB 与 DevOps 大纲的关系

TMMi® 与 ISTQB 已结成联盟，共同进一步推动软件测试行业的发展。在本文档编制过程中，TMMi® 技术委员会与 ISTQB“DevOps 中的质量”教学大纲工作小组也进行了合作。TMMi®作为测试改进模型，提供了一个基于 TMMi® 模型结构的预定义改进方法及优先级。本文档的重点在于为在 DevOps 环境中工作的人员提供对 TMMi® 改进目标的解读。而非详细描述良好的DevOps工程实践。ISTQB“DevOps 中的质量”教学大纲是一份基于内容的文件，旨在详细描述良好的 DevOps 工程实践，例如，质量工程（包括测试）实践。因此，这两份文档在本质上是高度互补的：TMMi® 指出了应该改进什么（哪些过程），ISTQB DevOps 教学大纲描述了应当如何去做（工程最佳实践）。

# 2 TMMi®2 级已管理

## 2.1 过程域 2.1 测试方针与策略

测试方针与策略过程域的目的是开发并建立测试方针，以及组织或项目范围内的测试策略，其中明确定义了测试活动（如测试类型和测试级别）。为了衡量测试绩效、体现测试活动的价值并揭示改进领域，需引入测试绩效指标。

### 2.1.1 SG1 建立测试方针

任何组织在启动测试改进项目时，都应首先定义测试方针。测试方针定义了组织的整体测试目标、目的和有关测试和测试专业人员的战略观点。测试方针与组织整体业务和质量方针保持一致非常重要。测试改进应由明确的业务目标驱动，这些目标应记录在测试（改进）方针中。测试方针对于在组织内的所有利益相关方之间就测试及其目标达成共识来说是很有必要的。这种共识可以使整个组织的测试（过程改进）活动保持一致。请注意，测试目的本身永远不要成为目标，它们应源于建立工作软件和产品质量的更高级别的目标。

在一个拥抱 DevOps 文化和实践的 DevOps 软件开发组织中也是如此。事实上，在许多组织内部，对 DevOps 软件开发中测试角色的变化、测试的独立性、测试自动化以及专业测试人员等进行了大量的讨论。这些议题以及其他相关问题通常需要与管理层和其他利益相关方讨论，并在测试方针中予以记录。任何想要启动测试改进项目的组织，包括那些采用 DevOps 实践的组织，都需要明确并界定推动此类项目的业务驱动因素和要求。否则，启动改进项目的意义又何在？通过投入时间来准确捕捉真实的业务需求，可以为确定（测试）改进工作的优先级提供依据，例如，确定应重点关注哪些过程域。需要注意的是，测试方针通常是一份在组织层面的单页精简文档、网页或墙报/挂图，而非项目层面的文档。

TMMi®的特定目标建立测试方针，包括其特定实践，完全适用于采用 DevOps 软件开发方法的组织。然而，在 DevOps 中，开发、测试和运营通常包含在同一个过程中，或者至少它们是紧密协同的。因此，测试方针也有望成为整体 DevOps 方针的一个不可或缺的部分，更加侧重于质量工程而不仅仅是测试。DevOps 中的测试活是通过平衡“左移”和“右移”方法，采用协作实践和“测试先行”的思维方式来组织的。在 DevOps 中，测试活动应当支持 DevOps 中定义的三种方式：流程、反馈以及持续实验和学习。

如前所述，测试目标应侧重于平衡“左移”和“右移”；主动、设计和测试先行的思维方式，在部署前（在持续集成/持续交付（CI/CD）流水线内和周边）构建所需的质量和测试活动，以捕获缺陷，同时采取应对措施来减少生产环境中因故障而导致的服务停机时间，例如，采用自动回滚机制以减少停机时间、使用蓝绿部署[1]或金丝雀发布来降低影响。测试活动的目标是降低将故障（有缺陷的软件）部署到生产环境

[1] 蓝绿部署是一种部署策略，即创建两个独立但完全相同的环境。其中一个环境（蓝色）运行当前的应用程序版本，另一个环境（绿色）运行新的应用程序版本。

的概率，而部署策略则用于减少任何给定失效的影响，从而提升整体质量。为实现减少影响，应简化部署流水线中的测试活动，以改善流程和交付周期，同时总体风险保持低风险水平。

除了在整个软件开发过程中进行测试外，“右移”（即在生产环境中进行测试）也是 DevOps 中的常见实践，这同样会影响方针中定义的测试目标的设定，重点是根据遥测信息对生产环境中的问题做出快速响应，以减少停机时间、提高可用性，进而提升质量。站点可靠性工程（SRE）实践涵盖了监测系统、服务和应用程序的状况，并创建闭环自动化机制以应对问题/偏差状况[Beyer]。这也被称作自我修复/自愈。

由于 DevOps 中变更频繁，测试目标高度聚焦于功能和非功能层面的回归测试。一些非功能特性可以在部署后在生产环境中测量（右移），而其他特性则需要在投入生产之前进行测试，例如信息安全性方面。

*测试目标示例:*

- *测试旨在确保变更能够交付预期的价值（验收测试），并且不会破坏任何现有功能（回归测试）。*
- *测试应支持组织的发展目标，例如降低变更失效率，或提高部署频率。*

在 DevOps 背景下，独立性和根据测试方针定义的高级过程通常也会受到影响。关于独立性而言，测试是开发组织中的一个角色，很多时候被称为 DevTest，以表示其承担的开发与测试的双重角色。因此，其独立性通常较低。组织通常会依赖一个所谓的平台团队，该团队由具有测试能力的软件开发工程师（Software Development Engineers in Testing - SDET）组成，他们的主要职责是开发和维护产品和测试自动化解决方案的可测试性，供 DevTest 角色使用。此外，对于一些更复杂的领域，如非功能测试或复杂的端到端场景，可能会设立专门的测试团队。在这种情况下，虽然这些团队的独立性相对较高，但与开发团队的协作仍然非常紧密。关于高级测试过程定义，在 DevOps 中，测试现在是开发过程或 CI/CD 策略的一个组成部分，因此测试过程也会相应地成为这个整体过程的一部分。

确保测试方针（通常为轻量级/精益文档）分发给整个组织/价值流/敏捷发布列车/部落非常重要。由于测试是敏捷和 DevOps 团队工作的重要组成部分，而非独立组织，所有相关和受影响人员都应知晓该方针。

## 2.1.2 SG2 建立测试策略

测试策略遵循测试方针，并作为项目内测试活动的起点。测试策略通常是在整个组织或整个项目群范围内定义的。典型的测试策略基于高级别的产品风险评估，并将包括要执行的测试类型和测试级别的描述。仅仅声明例如在每个项目中都将进行集成测试和验收测试是不够的。我们需要定义单元测试和验收测试的含义；这些测试通常是如何进行的，以及它们的主要目的是什么。经验表明，当定义并遵循测试策略时，各种测试活动之间的重叠情况更少，从而

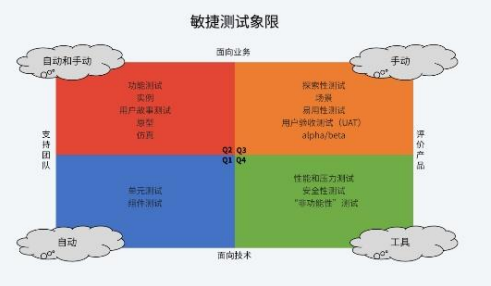


图 2 敏捷测试象限

能够使测试过程更加高效。此外，由于各种测试类型和测试级别的测试目的和测试方法保持相互协调一致，因此遗漏的测试点会更少，进而使测试过程更加有效，从而提高产品质量水平。

在 DevOps 环境中，测试策略是一份至关重要的文档。它从高级别上定义了团队主要在 CI/CD（持续集成/持续交付）策略中要进行的测试工作；明确了要执行的测试级别和测试类型，并从高级别上概述了它们的实施方法。对于每个测试级别和测试类型，都定义了其目的、职责、主要的持续测试任务，以及验收标准/质量门禁。测试策略通常遵循敏捷测试的四象限模型（参见图 2），但它是围绕 CI/CD 流水线构建的，以体现流水线内各项活动的流。这里的流水线不仅指由 CI/CD 工具执行的活动，还包括团队在处理待办事项（backlog item）时所做的所有工作。像评审和探索性测试这样的测试活动，虽然是在 CI/CD 工具之外进行的，但它们是流程/流水线/价值流中不可或缺的一部分。需要注意的是，流水线不仅可以被视为一种技术/工具，还可以被视为活动、过程和人员的可视化表示（参见图 3）。

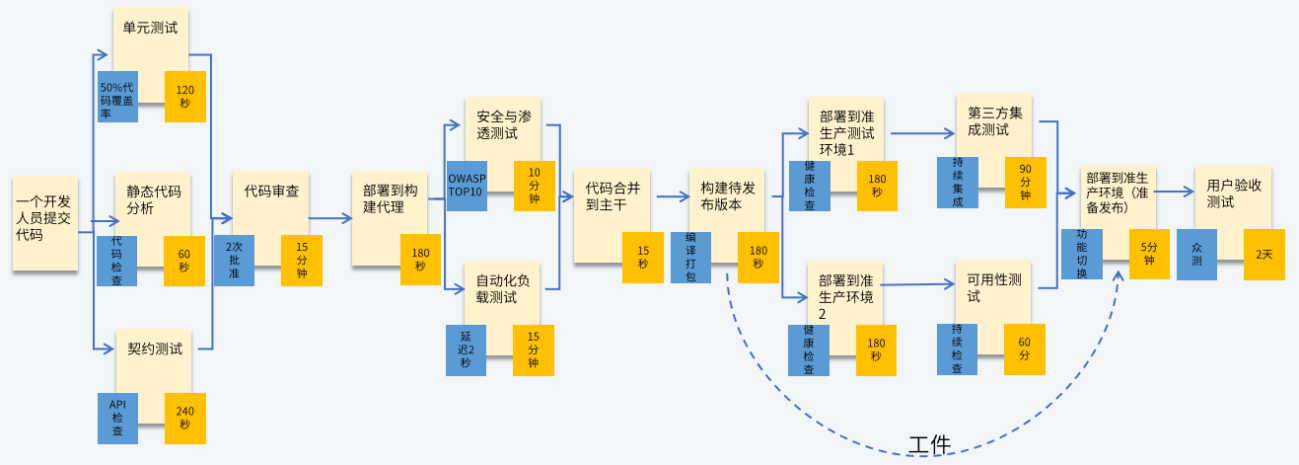


图 3：以高层级流水线形式呈现的测试策略

在采用 DevOps 工作方式时，测试策略同样是一份至关重要的文档。对于许多组织来说，DevOps 相对较新，因为这样的一份通用的组织级测试策略将指导所有相关人员，并说明在流水线中于何处以及如何进行测试。它还能避免每个项目/流水线重复“造轮子”。

由于测试是开发流水线中不可或缺的一部分，测试策略应与开发策略完全保持一致。在制定 DevOps 测试策略时，需要考虑以下一些（按主题分类的）要点：

*测试级别：*

- 测试级别主要体现为 CI/CD 流水线中的测试活动/任务。
- 测试策略通常围绕开发的用户故事的验收测试构建，采用行为驱动开发(BDD)实践，要求在流水线每个阶段实施持续测试。
- 测试策略通常会定义一个像契约测试这样的测试级别，以降低开放应用程序编程接口（API）使用风险，并为集成测试提供支持。

*底层测试：*

- 需考虑拉取请求、代码合并和代码评审等底层开发和测试实践。
- 静态测试等底层测试活动通常由 CI/CD 工具强制执行（例如作为拉取请求的自动化检查）
- 基于左移原则，重点实施单元测试、达成约定的代码覆盖率标准以及集成测试。

*测试自动化：*

- 持续测试无缝集成到软件交付流水线和 DevOps 工具链中
- 测试自动化策略是 DevOps 测试策略的核心组成部分。

*准入和准出标准：*

- 准入标准体现为 DevOps 团队的准备就绪定义（Definition-of-Ready - DoR）。DoR 应强制要求有高质量的测试依据作为测试输入，例如由团队共同定义的验收标准，通常采用 Given-When-Then 陈述的形式呈现。重要的是，在达到“准备就绪”状态之前，要明确定义并商定风险和测试目标，因此估算也涵盖测试要求。
- 准出标准分为 DevOps 团队的完成定义（Definition-of-Done-DoD）和 CI/CD 流水线中各测试任务的质量门禁。不同待办项层级（史诗/特征/用户故事）可定义不同的 DoD 标准，反映各阶段测试完成度和置信水平。除成功的验收测试外，DoD 通常还需满足作为所有待办项约束条件的非功能需求测试。在 DevOps 团队中，只有当待办项部署至生产环境并通过系统/服务监测验证运行正常，才视为“”完成″。

*测试类型：安全测试：*

- 明确对成功至关重要的非功能特性（如信息安全性、性能效率、可靠性或易用性）及其测试方法。
- 鉴于信息安全的重要性，静态应用程序的信息安全测试（SAST）、动态信息安全分析工具（DAST）、交互式应用程序信息安全测试（IAST）、软件成分分析（SCA）、渗透测试以及 API 信息安全测试等可作为安全测试方法的一部分。

*测试环境：*

- CI/CD 流水线中各测试阶段（如开发、预发布、准生产）对应不同的测试环境，每个环境聚焦不同测试目标，并与测试级别（单元测试、集成测试、系统测试和验收测试）对齐。DevOps 强调生产环境部署与配置的高度自动化（如″配置即代码″），该理念同样适用于测试环境设计与构建。基于云环境按需创建测试环境并在使用后销毁已成为标准实践，从而实现标准化、可控且稳定的测试环境。采用蓝绿部署等策略时，当前版本产品的测试环境在通过质量门禁和 DoD 后可直接转为生产环境，原生产环境则复用为下一测试环境（实现蓝绿环境用途切换）。

对于 DevOps 组织而言，精简的测试策略文档通常是一个良好的解决方案，可以避免制定过于详细（且针对具体项目）的测试计划。TMMi®中的特定目标“建立测试策略”，包括其特定实践，完全适用于采用

DevOps 软件开发方法的组织。与测试方针一样，重要的是要确保测试策略能够传达给整个组织、价值流、敏捷发布列车或部落。

### 2.1.3 SG3 建立测试绩效指标

测试方针中定义的测试改进业务目标需要转化为一组关键的测试绩效指标。这些指标与测试方针共同为组织提供明确方向，并作为沟通测试绩效期望值与实际达成值的工具。绩效指标必须向利益相关方展示测试及测试过程改进的价值。由于过程改进投资需要获得长期管理支持，定量衡量改进计划的收益对保持管理层积极性至关重要。需要注意的是，此TMMi®特定目标旨在定义少量（如2-3个）核心测试绩效指标，而非建立完整的度量体系。这些指标应能反映测试价值在不同交付环境中的变化。

与敏捷开发类似，DevOps 的核心理念也强调团队协作和系统思维。这可能会导致指标相应地扩大到开发和部署过程绩效指标，而不仅仅局限于测试本身的细节。DevOps 的核心绩效指标通常遵循 DevOps 研究与评估（DevOps Research and Assessment-DORA）指标体系。值得注意的是，在 DevOps 中，用在生产环境部署期间的失效率（即失败部署的百分比）替代了传统的(测试)覆盖率作为关键度量项。

*DORA 指示示例:*

- 软件交付绩效指标：部署频率、变更前置时间（从代码提交到生产环境部署）、服务恢复时间（MTTR）、变更失效率（生产环境部署失败比例）
- 运营绩效：可靠性指标，例如平均修复时间（Mean Time To Repair-MTTR）和平均无故障时间（Mean Time To Failure-MTTF）

绩效指标应当成为 CI/CD（持续集成/持续交付）活动不可或缺的一部分，并且相关数据应当几乎实时地提供给组织内的所有人员。除了 DORA 指标外，流程指标也被用于支持价值流管理与优化。测试活动可以按照上述指标进行衡量，以确保其贡献是清晰可理解的。

*与价值流管理相关的测试流程指标示例:*

- 测试执行交付周期：指从代码变更触发测试流程开始，到测试完成所需的时间。每个代码变更在 CI/CD 流程中的测试活动成功率或失效率：衡量测试活动在 CI/CD 流程中针对每个代码变更的成功或失败比例。
- 流程指标：测试执行交付周期如何影响整体交付周期，即分析测试执行交付周期在整个软件交付周期（从代码提交到生产部署）中所占的比例，以及它对整体交付速度的影响。

面临的挑战在于如何定义一套恰当的指标组合，这些指标既要与 DevOps 方法和系统思维相关联，又要能够准确反映基于 TMMi®（测试成熟度模型集成）的测试改进计划所取得的成绩。尽管所选和应用的绩效指标可能比仅与测试相关的范围更广，且不仅仅局限于测试领域，但 TMMi®中的特定目标“建立测试绩效指标”（包括其具体实践）仍然完全适用。当然，指标范围的扩大确实会使绩效指标的分析和解读更具挑

战性。事实上，这些指标甚至可能不以"测试绩效指标"命名。只要包含测试相关要素并用于评估测试改进进展，这在 TMMi®框架下仍是可行的。

## 2.2 过程域 2.2 测试策划

测试策划的目的是基于已识别的风险和已定义的测试策略来定义一套测试方法，并为执行和管理测试活动建立和维护有充分依据的计划。

请注意，成功的测试策划的关键在于前期思考（“活动”），而不是定义相关的测试计划（“文档”）。

对于 DevOps 生命周期，通常有两种规划：发布规划和迭代规划。TMMi®2 级的测试策划过程域关注发布规划和迭代规划中的测试相关活动。发布规划在项目开始时着眼于产品的发布。发布规划需要明确的产品待开发列表，并可能涉及将较大的用户故事细化为一系列较小的故事。然而，请记住，DevOps 和 Agile 通常采用固定的时间线和灵活的内容。务必确保仅发布那些已按照测试方法和 DoD 进行测试的待开发列表。发布规划为涵盖所有迭代的测试方法和测试计划奠定了基础。发布计划是概要的，侧重于整体策略或方法，而不是具体的任务。完成发布规划后，将开始第一次迭代的迭代规划。迭代规划着眼于单次迭代的结束，并关注迭代待开发列表。

### 2.2.1 SG1 执行风险评估

穷尽测试是不可能的，需要始终做出选择并设定优先级。因此，这个 TMMi®目标也适用于 DevOps 项目。对于 DevOps 项目，应在发布规划阶段基于产品愿景文档或一组概要产品目标进行概要产品风险评估。在每次迭代中对于每次迭代，应根据该迭代的用户故事或迭代的其他需求，在迭代规划会议中开展更详细的产品风险评估会议。有时，这已经在细化会议中完成，在会议上会讨论用户故事并分析风险细化会议上已经完成用户故事的理解并分析风险。与采用遵循顺序生命周期模型的传统项目相比，DevOps 项目中的产品风险评估过程将采用更轻量级的模式。轻量级产品风险评估技术的示例包括风险扑克 [Veenendaal12] 和 ROAM（已解决、已拥有、已接受、 已缓解）[Baah]。在发布规划或程序增量规划（PI 规划）中，由了解发布中特征的业务代表提供待开发功能的概述，其中将包括非功能性需求作为约束，整个团队，包括测试人员，将协助进行风险识别和评估包括测试人员在内的整个团队将协助进行风险识别和评估。

在迭代规划期间，DevOps 团队将会根据即将在下一迭代中实施的用户故事，识别并分析产品风险。最好是敏捷团队的所有成员以及其他一些利益相关方者都参与产品风险会议。会议的结果是最终会形成一份按优先级排列的产品风险项列表，其中明确确定了需要测试的关键域。这进而将有助于确定应分配的适当的测试工作量，以便用足够的测试，覆盖每种风险，并以优化测试工作的有效性和效率的方式安排这些测试的顺序对这些测试进行排序，从而优化待完成测试工作的有效性和效率。可以根据相关的产品风险等级

对估算的预估任务进行优先级排序。与较高风险相关的任务应尽早开始，并投入更多的测试工作量。与较低风险相关的任务应稍后开始，并投入较少的测试工作量。

鉴于安全性在许多组织中的重要性，威胁建模除了上述用途外，还可用于识别安全需求、精准定位安全威胁和潜在漏洞、量化威胁和漏洞的严重程度，以及确定缓解措施的优先级以进行风险缓解并确定缓解措施的优先级。通过使用量化每个威胁的可能性和影响的指南，对安全威胁进行优先级排序，以确定其严重程度安全威胁的优先级排序基于量化每种威胁发生可能性和影响的准则，最终确定其严重程度。

### 2.2.2 SG1 建立测试方法

为了缓解已识别并确定优先级的产品风险，需要制定测试方法测试方法的定义是为了缓解已识别和确定优先级的产品风险。对于特定的迭代，待测试项和功能在迭代规划中确定在迭代计划期间确定要测试的项和功能。此活动也是基于产品风险会议的结果开展。待测试项的优先级列表通常与本迭代需要待测试的用户故事相关。这些功能通常涉及多种需要测试的软件质量特性等内容本次迭代中测试的功能。这些功能通常与待测试的各种软件质量特性相关。在迭代过程中，可能会出现新的产品风险，需要进行额外测试新产品风险可能在需要额外测试的迭代期间变得明显。像需要额外测试的新产品风险这类问题，通常会在每日站会中进行讨论通常会在每天的站立会议上讨论需要额外测试的新产品风险等问题。

在 DevOps 中，测试方法是基于通过 CI/CD 流水线进行的持续测试来进行定义的。持续测试意味着尽早并频繁地使用自动化工具进行测试，并提供针对交付流水线每个阶段提供可操作操作性的反馈。持续集成 (CI) 和持续部署 (CD) 工具通常在代码提交时自动调用，并协调构建、集成、测试、安全、合规性和部署活动。CI/CD 流水线中的测试活动通常基于使用以敏捷测试象限确定的测试级别为基础，并由团队持续优化。

在迭代级别或待办开发列表级别,为缓解风险而制定的测试方法定义用于降低风险的测试方法，例如，对用户故事和验收标准进行额外评审；根据风险等级分配相应比例的测试工作量；依据风险等级和类型选择合适的测试技术等。可以涵盖例如对用户故事和验收标准的额外审查、与风险级别成比例的测试工作、根据风险级别和风险类型选择合适的测试技术。该测试方案它还可以建议仅在部署后执行的测试活动，例如在生产中进行众包测试以收集用户体验反馈，或使用暗发布启动测试真实用户流量的性能。在 DevOps 的背景下，行为驱动开发 (BDD) 和验收测试驱动开发 (ATDD) 是驱动测试用例识别和设计的常用流行技术。发布级别的测试方法将属于需要更高的级别，并且应基于项目软件或组织级别已上定义的测试策略。测试方法通常会保存或者展示在团队/项目的维基盘（wiki）上进行保存或展示。

回归风险是迭代开发中存在的一个重要风险。测试方法需要定义如何管理回归风险。通常，这可以将通过创建一个特定的回归测试集来完成，该测试集最好是自动化的。在这种情况下，测试自动化金字塔会有所非常有帮助用[Cohn]。它展示了如何最大化回归测试自动化的价值，从金字塔最底层的单元测试开始，逐步过渡到接下来是服务级测试。用户界面测试位于最顶层。单元测试快速而可靠。服务层允许在 API 级或服务级别测试业务逻辑，它不这样就不会受到用户界面 (UI) 的束缚。测试级别层级越高，测试速度越

慢，也越脆弱。由于自动化是 DevOps 的重要组成部分，因此测试自动化是 DevOps 测试方法的重点，而不仅仅是支持回归测试。应对回归风险的另一种方法是依靠系统的可观测性另一种方法来应对回归风险是依据系统的可观察性，持续监测其状态和使用情况，并建立构建自动化机制来规避风险，例如回滚到之前的稳定版本。

入口准则（也称为“就绪定义”- DoR）通常是已定义测试方法（特定实践 2.3）的一部分，但在 DevOps 开发中，其处理方式可能有所不同。在 DevOps 软件开发中，测试是团队过程和持续活动不可或缺的一部分。以往使用的入口准则清单被拆分为两部分：一部分体现在 “就绪定义” 中，例如将验收标准定义为采用 “Given-When-Then” 语句的测试场景；另一部分则整合到 CI/CD 机制内的检查中，例如拉取请求中的检查、在开始大规模测试执行活动前在新部署的测试环境上运行冒烟测试，或是检查所需的测试数据是否已部署以前使用的入口准则清单分为“就绪定义”和“持续集成/持续交付”的检查，前者包括“就绪定义”，例如使用“给定-何时-然后”语句将验收标准定义为测试场景；后者包括拉取请求中的检查、在启动更大规模的测试执行活动之前，在新部署的测试环境上运行冒烟测试，或者检查所需的测试数据是否也已部署。因此具体的应该在 CI/CD 工作流中设置专门的实施用于确定测试是否可以启动的检查表或检查阀，用于确定测试是否可以启动，并像其他缺陷一样，在不符合条件合规的情况下拒绝构建。这同样适用于 DevOps 团队中，组件的测试从一个测试级别（例如，单元测试）进入到另一个测试级别（例如，验收测试）的情况。

测试出口准则（特定实践 2.4）是所谓的完成定义 (DoD) 的一部分，完成定义它严重在很大程度上依赖于在 CI/CD 流水线中完成的质量阀。重要的是，DoD 必须具有与特定的测试相关的特定标准，例如，测试覆盖率和产品质量。迭代应实现约定商定的一组用户故事，并满足 完成的定义 DoD 中规定定义的（测试）出口准则。在 DevOps 中，“DONE”通常意味着与待办开发列表事项相关的代码更改已部署到生产中，并通过了 CI/CD 流水线内的所有质量阀。如果采用按需发布模式发布是按需进行的，并且使用功能切换开关切换等技术将发布与部署分离，那么在发布级别也会有一个 “完成定义”，该定义可能涵盖多个迭代则在发布级别也有一个可以跨越多个迭代的 DoD，但 DevOps 的目标是尽可能频繁地发布小型版本。发布级别的完成的定义通常也会 DoD 包含与覆盖范围和产品质量相关的标准通常会再次具有覆盖范围和产品质量相关的准则，这些标准通常侧重于基于生产环境中的测试活动（如众包测试或金丝雀测试），验证是否已向用户交付预期价值通常侧重于根据生产中发生的测试活动（如众包测试或金丝雀测试）来验证是否已向用户交付期望价值。

特定实践 2.5 定义暂停和恢复准则很可能与 DevOps 生命周期不相关。由于测试是 DevOps 软件开发过程中不可或缺的一部分，因此它不会被视为与其他迭代活动分离且独立的活动。当出现可能被视为对测试进度构成潜在或实际威胁的阻塞性问题时，这些问题会在每日站会中进行讨论如果存在阻碍性问题，可能会被视为潜在或实际对测试进展的威胁，这些问题将在 每日站立会议中讨论。在这个讨论中，团队将决定需要采取什么行动来解决问题。因此，正式的暂停和恢复准则标准不是必要的，也不需要被定义。相关问题将作为常规 DevOps 日常工作惯例的一部分进行处理。因此，DevOps 日常工作惯例可作为此这种特定实践的一种替代。

## 2.2.3 SG3 建立测试估算

对于 DevOps 团队，在迭代规划迭代计划期间将对测试进行详细估算。概要的高级别阶（测试）估算将在发布规划期间进行，也可能在待开发列表待办事项细化会议中进行。所有估算当然均以团队估算的形式进行，其中包含交付每个故事所需的全部工作量。确保在估算会议中切实考虑到测试活动至关重要务必确保在估算会议期间确实考虑到了测试活动。要做到这一点，可以将测试活动确定为独立任务，这可以通过将测试活动确定为单独的任务，然后分别估算每个测试任务来实现，或者通过估算每个用户故事来完成，从而明确及考虑到需要进行的测试活动。由于在 DevOps 中，几乎所有工作都要求实现自动化正如 DevOps 期望几乎所有内容都实现自动化一样，因此与测试相关活动的自动化也应进行估算测试相关活动的自动化也应如此，。估计包括测试自动化、测试环境部署自动化、测试数据部署自动化，以及根据确定的迭代测试方法对流水线的改进。在估算过程中期间，作为 DevOps 团队的一部分，测试人员应参与估算会议。计划扑克（卡片）、T 恤尺寸、泳道尺寸估算（桶式估算桶尺寸）和点投票法都是 DevOps 软件开发中使用的典型估算技术。

下一次迭代要完成的工作通常由用户故事定义。用户故事的规模要尽可能地小些，以便可以估算。可估算是 INVEST 定义的用户故事标准之一，适用于迭代的用户故事可适用于迭代中的一部分。在迭代规划期间，敏捷团队通常会识别和定义与测试相关的任务。测试任务将与其他开发任务一起记录在任务板上。在回顾会议上，团队应该回顾他们的估算准确性（如果被确定为问题），并根据迭代中完成的各个任务的规模来改进。

在版本发布规划中，通常会对用户故事或故事分集史诗进行更高层次的定义，但并不会将其详细描述为特定的任务。这当然会使估算更加困难且不准确。如上所述，DevOps 项目不会建立工作分解结构作为估算的基础，但在发布阶段，可以从关于“待办事项应开发什么以及如何开发” 的信息中获益开发待开发列表的内容和方式的信息中受益，这些信息足以满足所需的估算准确性精度，足以使待开发列表待办事项的优先级排序成为可能。由于 DevOps 的目标是优化价值流程，因此估算需要为支持下一步开发什么提供构建的决策，而不是创建一个详尽完整的项目时间表。

虽然 DevOps 敏捷团队会以相对非正式的方式进行估算，但估算的理念应该清晰明确是很清楚的（即需要考虑哪些因素）。基于这些依据的展开讨论，可以提高估算的准确性。通常，在 DevOps 项目中，估算侧重于规模（使用故事点）或工作量（使用理想人天理想化工日作为估算单位）。成本通常不会作为 DevOps 项目的估算过程的一部分。需要注意的是，在 DevOps 环境中，基础设施成本通常很高，其成本估算通常是在特定的估算环节中确定的，而不是在团队估算环节中。需要密切监测和控制基础设施成本，例如公共云的使用，并且可以嵌入到 CI/CD 流水线中，及具有自动报告功能。（例如，执行的每个 GitHub Action 作业都以使用的 vCPU 分钟数显示）。

因此，除特定实践“ 3.2 定义测试生命周期”这一特殊实践外，TMMi® 特定目标及其特定实践完全适用。迭代和 DevOps 软件开发的基础之一是以小批量块来开展工作。因此，已经确定的任务通常足够详细，可以作为（测试）估算的基础。因此，在 DevOps 环境中，也需要为测试活动定义一个生命周期，作为估算的额外依据作为估算的估算基础。

## 2.2.4 SG4 开发测试计划

需要重复说明的是再次强调，测试计划是关于前期思考（即“活动”），而不是定义相关的测试计划（即“文档”）。在 DevOps 中，大多数测试规划活动（正如 TMMi®特定目标所定义）将在版本发布和迭代规划期间执行，以及待开发列表细化会议中执行。然而，这些活动的结果通常不会记录在测试计划中，尤其是在迭代规划中，它们可以反映在任务板上。由于测试是发布和迭代规划不可或缺的一部分，因此最终的“进度表”也将包含测试活动。与按顺序生命周期制定的详细进度表不同，DevOps 项目中的进度表更像是用户故事（待开发列表待办事项）和任务的排序，反映了发布和迭代的优先级，例如基于期望的业务价值交付。请记住，测试执行主要在 CI/CD 流水线中进行，由代码提交触发。额外的手动测试或在生产环境中进行的活动可以在迭代板上进行规划。思维导图通常用作辅助工具技巧。任务板将反映迭代优先级。因此不需要制定明确的（测试）时间表。期望为用户故事定义明确的发布和迭代优先级，分别要执行的任务，包括测试任务。

在项目层面，测试人员是构建团队组建的一部分；对测试资源或多技能资源的需求需预先确定预先确定了对测试或多技能人力资源 的需求。随着项目的变化或规模的增长，人们可能很容易忘记为特定团队/项目初步选 择个人的理由，例如具体的技能需求或与团队/项目相关的特定经验。因此，将人员的技能需求记录下来，并附上他们被选入该团队 / 项目的原因作为佐证信息，是很有必要的这为写下人们的 技能需求提供了一个很好的理由，为团队/项目选择原因提供相关的备份信息。一旦 DevOps 团队确定，测试人员或多或少就已经基本固定。在迭代规划期间，可以根据需要讨论在迭代中执行测试所需的（额外）资源和技能，例如非功能测试，以确保团队拥有足够的测试资源、知识和技能来执行所需的测试。

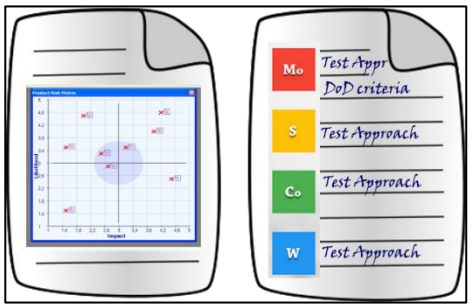


图 4：一页的测试计划

最初始项目风险会议应成为发布和迭代规划的一部分。迭代期间进一步识别项目的风险（和管理）是每日站立会议和常规 scrum-of-scrum 会议的一部分，通常通过障碍日志记录。测试问题也应记录在障碍日志中，这一点很重要。障碍应在每日站立会议上讨论，直至解决；如果团队没有级别或权限解决这些障碍，则应在 scrum-of-scrum 会议上讨论。

虽然通常不会制定和记录具体详细的测试计划文档，但特殊实践 4.5 建立测试计划在 DevOps 环境中仍然适用。然而，在测试规划中进行的讨论结果可能会以轻量级的形式记录下来，可能是思维导图或一页纸的文档（参见图 4）、团队级别的敏捷测试象限以及测试活动在 CI/CD 流水线的内测试活动。

## 2.2.5 SG5 获得测试计划的承诺

在 DevOps 中，制定开发和建立测试方法和测试计划的过程是一项基于团队的工作，可能由测试专业人员（团队成员之一）领导。产品质量是团队的共同责任。因此，只要团队遵循正确的过程，并将商定的测试活动嵌入到 CI/CD 流水线中，对（测试）方法和（测试）计划的承诺就已是发布和迭代规划的必然结

果，因为这是团队共同努力的结果因为它是一项团队努力。当然，这与传统环境中的工作方式有着巨大的区别，在传统环境中，通常由测试人员准备测试计划，然后需要获得明确的承诺。CI/CD 流水线以部署的测试环境和执行的自动化测试的形式， 实施测试计划所需的基础设施资源，需要考虑并由利益相关方者提供。

在 DevOps 环境中，团队（包括产品负责人）必须理解并同意按优先级排序的产品风险列表的优先级列表以及需要执行的测试缓解措施。例如，在发布规划或迭代规划期间，先进行一次简短的演示可以通过简短的演示，然后在发布或迭代计划期间进行讨论，向团队解释产品风险、测试方法及其原理，然后展开讨论，从而达成理解和承诺。

在估算环节（参见 SG 3 建立测试估算）会估算工作量。团队的（测试）资源由 DevOps 团队的设置决定。估算并选择开发的用户故事，会将可用资源作为出发点（约束条件）以可用资源为起点（约束）。因此，再次强调，协调工作量和资源量并不是一项有意义的活动，但持续关注技术资源（例如云计算/存储/网络可用性）至关重要，且需要为团队提供这些资源需要将其提供给团队。因此，特定实践 35.2 协调工作和资源级别 协调工作量和资源量通常与 DevOps 环境无关。每日站立会议和 Scrum-of-Scrum 会议将用于解决迭代期间的任何资源问题，并立即（重新）分配适当的资源，或者从迭代中移除某个可交付成果，以便在未来的迭代或发布规划中进行协调或者减少可交付物并协调到未来的迭代 中，或调整版本发布计划。

## 2.3 过程域 2.3 测试监督和控制

测试监督和控制的目的是了解测试进展和产品质量，以便当测试进度明显偏离计划并且产品质量明显偏离预期时，可以采取适当的纠正措施。

与传统项目相比，DevOps 项目中的一些监督和控制有着根本性的区别差异。虽然监控监督和控制是 DevOps 中必不可少的核心元素，但这并不意味着其目标是必须坚持严格遵循的计划是目标。从测试监督和控制的角度来看，这意味着我们并非不是由计划驱动的，而是不断审视评审我们的测试进度和测试结果，并在适当情况下调整根据需要适当调整我们的计划和方法，例如新产品的风险变得明显时显现时。

测试计划通常分散在涵盖经过微调的验收标准、待开发列表待办事项（包括其风险级别）和任务中并可以进行微调，它是一个有生命动态变化的实体，需要随着新信息的出现或反馈的给出，不断进行评审和更新需要随着新的依据信息的出现或得到反馈而不断进行审查和更新。测试活动的目的由给定待开发列表待办事项的验收标准来定义、团队针对给定给定待开发列表待办事项更新测试的方法以及在待开发列表待办事项级别定义的具体的“完成的定义的”标准。DevOps 项目还需要着眼于牢记“全局”，不仅需要在并在 CI/CD 级别，也需要在和迭代级别进行监督和控制。

需要值得注意的是，由于测试是一个过程并完全融入 DevOps 团队整体过程，因此测试监督和控制也是 DevOps 团队整体监督和控制机制不可或缺的一部分。因此，测试人员不像传统项目那样向测试经理报告，而是向团队报告。

### 2.3.1 SG1 根据计划监督测试进度

DevOps 团队使用各种方法和工具来监督和记录测试进度，例如在敏捷任务板上记录测试任务和故事的进展，以及燃尽图。然后，这些信息可以通过维基仪表板、仪表板式电子邮件等媒介传达给团队其他成员，也可以在站会通过口头方式进行沟通。团队可以使用 wiki 仪表板和仪表板式电子邮件等媒介，以及在站立会议中口头传达这些信息给团队的其他成员。团队可以使用燃尽图来跟踪整个发布版本和每个迭代中的进度。燃尽图通常会显示进度与预期速度以及待实现功能的积压情况（因此，它显示了团队的绩效）。此外，DevOps 团队会使用在线仪表板进行持续的监督，以自动化方式在整个 CI/CD 流水线中监督基础设施、应用程序和网络。测试环境资源会根据迭代规划期间商定的资源以及 CI/CD 流水线的使用情况进行持续监督。另一种常见做法是通过任务板跟踪测试环境问题，例如，在故事卡上使用标记为“测试环境受阻环境原因测试被阻断”的标签，或者在任务板上创建一个单独的列，将所有被环境阻塞的故事放入其中，直到阻塞解除。这一切都是为了让大家清楚看到测试环境对进度造成的阻碍影响这一切都是为了使阻断进程的测试环境影响变得可见，并促使让团队在进行任何处理其他任务之前先解决此问题。

为了即时、详细直观地及时提供团队和 CI/CD 当前状态（包括测试状态）的详细的可视化呈现，团队通常可以使用任务板。在任务板上记录故事卡、开发任务、测试任务以及在迭代规划期间创建的其他任务。在迭代过程中，通过将这些任务在任务板上的不同列之间移动，如 “待办”“进行中” 和 “已完成”，来管理进度通过将这些任务在任务板上的移动划分为“待办”、“进行中”和“已完成”等列来管理进度。为了保持流程流畅，通常会在看板上不同状态中对“不同状态进行中在制品”（Work in Process - WIP）限制，以规范控制团队行为以完成正在（进行中的）任务，而不是着手开始处理新任务。DevOps 团队通常会借助使用工具在任务板上维护故事卡，这些工具可以自动生成化的仪表板和状态概览。不过然而，有些团队不会为各项个活动创建特定的任务，而可能仅只是使用故事卡并在卡片上注释关于测试的评论测试、相关参考工具或记录测试内容的维基（wiki）页面可能记录测试的 wiki。任务板上的测试任务通常与为为用户故事定义的接受验收标准以及作为作为每个待开发列表待办事项约束条件的非功能需求相关。当测试任务的测试自动化脚本和探索性测试达到通过状态时，测试任务会移至任务板的“已完成”列。

已完成的定义（DoD）是衡量进度的出口标准。完成定义还应该与测试活动相关联，并展示了一个在用户故事的开发和测试完成之前，需要满足的所有标准。请注意，与测试任务相关的测试标准只是团队所达成一致同意的完成的一部分，即“已完成定义”的一部分。完成定义准则通常应用于多个级别，例如故事级别和史诗级别。整个团队会定期（通常在每日站立会议期间）评审任务板的状态，以确保任务板以可接受的速度推进。如果任何任务（包括测试任务）没有移动或移动速度过慢，则会触发基于团队团队层面的讨论，分析可能阻碍这些任务进展的问题。

每日站立会议涉及 DevOps 团队的所有成员，包括测试人员。在这个会议上，他们会向团队其他成员汇报当前状态和实际进度。任何可能阻断开发或测试进度的障碍都会在每日站立会议中沟通传达，以便整个团队了解情况并采取相应措施。通过这种方式，项目风险管理也被融入整合到这些日常会议工作中。团队可以定期进行 ROAMing 练习，以支持对项目风险状态的监督 [ROAM]。任何项目风险，包括与测试相关的风险，例如测试环境不缺乏可用性，都可以在每日站立会议上进行沟通和处理解决。（请注意，监督产品

风险是监控产品质量的一部分，因此将在下文中围绕特定目标 SP2 监控产品质量是否符合计划和预期根据计划和期望监督产品质量这一具体目标进行讨论。）用于日常任务管理的每日站立会议，以及与任务方向修正相关的 DevOps 团队实践，都是经过验证的验证的行之有效的技术方法良好技术，能够满足符合 TMMi 测试监督和控制过程域中特定实践的意图。

在迭代结束时，会根据商定的冲刺目标举行冲刺评审。团队会展示已取得的成果，例如已完成的故事或史诗。通过 CI/CD 流水线进行集成当然是持续进行的。测试的完成情况，例如对照完成“完成定义”的检查，将成为迭代评审的一部分。团队会组织与利益干系人组织演示活动，邀请利益相关方参加，共同讨论交付产品的商业价值和质量。产品经理负责人代表利益相关方者参与迭代规划、迭代评审（DEMO 演示活动）和回顾会议。产品经理会参与关于产品待办事项的讨论，并在整个迭代过程中为用户故事的细化和测试设计提供反馈与输入产品负责人通过讨论产品待开发列表列表参与其中，并在整个迭代过程中为用户故事的阐述和测试设计提供反馈和意见。其他利益相关方者在每次迭代结束时参与迭代评审。由于产品经理负责人代表利益相关方者已经融入到已融入敏捷工作方式中，因此不需要专门监督对利益相关方干系人的参与情况进行具体监控。

### 2.3.2 SG2 根据计划和预期监督产品质量

产品质量监督的机制与进度监督大致相同（参见上文 SG1）。使用在线仪表板监督产品质量，团队不仅仅在开发和测试过程中会这样做使用在线仪表盘进行产品质量监控不仅在团队的开发和测试阶段进行，这也是它也是一个持续的过程——，在生产环境中持续监督软件产品的质量。运维人员通常使用可观察观测性来持续监督产品质量，通常定义为服务级别指标（ 群 Service Level Indicator - SLI）。SLI 的退步回归或偏差会触发纠正措施，以保证质量和用户体验。在 DevOps 中，产品风险监督的重点是在在于定期会议中评审产品风险清单（包括安全风险/威胁），而非评审不是任何详细的风险文档记录。新识别的产品风险或发送变化的产品风险的变化，（例如由于探索性测试发现的的结果风险）将进行讨论并就所需的测试行动达成一致商定所需的测试执行。如有需要，可能会使用额外的 ROAM 练习 [ROAM]。各种产品风险的状态通常通过仪表板上的图表来显示。

DevOps 团队会使用类似的基于缺陷的指标来监督和改进产品质量，这些指标与传统开发方法中所收集的指标类似（例如测试通过/失败率、缺陷发现率、发现并修复的缺陷数量）来监控和改进产品质量。在每日站会例会期间，需要对以下内容进行监督：应该监控迭代期间过程中发现和解决的缺陷数量、持续集成活动中发现的缺陷数量，以及可能成为下一迭代待开发列表待办事项的一部分的未解决缺陷的数量。为了监督和改进整体产品质量，许多 DevOps 团队还会使用客户满意度调查来收集获得有关于产品是否满足满足客户期望的反馈。

测试出口准则（例如测试覆盖率和产品质量）是″“完成定义”（DoD）的一部分。一般通过任务板来监督对达成一致的出口标准的遵守情况，即只有符合 DoD 标准的故事才能被标记为“已完成”。DoD 标准通常包括将给定的待开发列表待办事项部署到生产环境中（部署后才算“完成”，反之则不成立）。通常，

会为持续测试定义了具体的出口准则。在 CI/CD 流水线中持续监督缺陷，作为是持续测试的一部分，在 CI/CD 流水线中持续监控缺陷。通过可观察观测性监督生产环境中产品预期质量特性的达成满足情况。自动化质量门（基于已定义的阈值）通常在 DevOps 流水线的不同阶段实施，以便在产品质量未达到预期时阻止部署。这些门控门禁可以评估多种指标，如测试通过率、性能基准、代码覆盖率和静态代码分析结果等等指标。

性能和可靠性通常是重要的非功能质量特性，因此是产品质量监督的一部分。性能测试（例如负载和压力测试）可以集成到流水线流程中，以对照根据预期基准监督系统性能。当性能阈值被突破时，会触发警报，通知团队立即解决问题。此外，（日志）监督工具也可以用于跟踪生产环境中的实时错误率、内存泄漏和系统崩溃。及早发现这些问题有助于确保产品达到满足可靠性的预期。

每日站立会议是一种几乎可以持续开展进行产品质量评审的机制。团队组织面向向利益相关方干系的人进行演示活动 DEMO 演示，进行验证，并一起组织讨论交付产 品的商业价值和质量。产品质量根据定义的完成准则进行验证和确认。

继“测试规划”过程域之后按照过程域测试计划一致，监督入口，暂停和恢复标准的特殊具体实践可能与此无关。更多信息，请参阅测试规划过程域的特定具体目标 2 建立测试方法，其中解释了为什么这些标准在 DevOps 环境中通常不适用。

### 2.3.3 SG3 管理纠正措施直至关闭

DevOps 团队能够快速发现问题，比如注意到燃尽图中出现与的预期不符的偏差，以及/或者任务板上（测试）任务和故事没有缺乏进展等问题。最重要的是，如果生产环境中的软件质量下降，能立即被察觉团队会立即注意到，并通常会触发纠正措施，例如回滚到已知的稳定版本。这些问题以及其他问题（例如阻碍测试进度的问题）会在每日站立会议上进行沟通，以便整个团队都能了解情况。随后，团队会开展集体讨论这随后会触发基于团队的讨论，来分析对这些问题进行分析。团队将与产品负责人共同决定采取哪些纠正措施。在基于 DevOps 的项目中，管理纠正措施主要是自组织团队的基本责任。团队可以定义并实施适当的纠正措施，或将任何问题作为“障碍”上报给 Scrum Master。针对产品质量的纠正措施有时是会自动化的，并监控对根本原因进行监督以避免再次发生。这是 DevOps 团队持续改进文化的一部分。Scrum Master 通常负责支持团队处置管理问题直至结束关闭。讨论和管理纠正措施的典型场合事件是每日站立会议和回顾会议。已商定达成一致的纠正措施可以作为“任务”或“待办事项”，通过产品待开发列表待办列表或（在迭代内）通过任务板作为“任务”或“待开发列表”来进行管理，直至完成结束。

## 2.4 过程域：测试设计与执行

*测试设计与执行的目的是通过建立测试设计规格说明、使用测试设计技术、实施结构化测试执行过程，以及管理测试事件直至关闭来提高在测试设计与执行期间测试过程的能力。*

虽然在 DevOps 项目和传统顺序方式的项目的基本目标（“消除风险并测试软件”）是相同的，但测试方法通常存在显著的差异。在 DevOps 方面，灵活性和能够对变化做出响应是重要的起点。此外，测试分析与测试设计、测试实施和测试执行不是顺序的测试阶段，而是在快速反馈循环的支持下，并行执行、重叠和迭代地执行。测试文档建立的详细程度是另一个关键区别，主要依赖于测试自动化代码作为测试文档的一部分。对于手工测试，在 DevOps 项目中通常使用更多的是基于经验和基于缺陷的技术来补充测试自动化。此外，手工测试也可用于众包测试，在将软件部署到生产环境后，将其发布给一个预选人群组，他们遵循一些指令或测试章程，并就特定的质量特性（通常是易用性或易访问性）向开发提供额外反馈。基于规格说明的测试技术可能仍然适合并可以使用，特别是作为协作式的需求分析一部分，列出可能的示例场景，以增强用户故事的验收准则。随着对组件测试和集成测试的重视度的加强，诸如语句测试和判定测试的白盒技术，以及对应用程序编程接口进行的契约测试也更广泛地被应用。最后一个主要区别是，由于高变更率导致的回归风险的水平，这要求在各个测试级别进行更多的回归测试。理想情况下，回归测试是高度自动化的，在 CI/CD 流水线内的多个测试阶段中自动触发并执行。尽管存在诸多差异，但最终在“测试设计与执行”过程域背景下，始终是关于创建测试、缓解产品风险、运行测试和发现缺陷。

### 2.4.1 SG1 使用测试设计技术执行测试分析与设计

在 DevOps 中，测试分析与设计以及测试执行是相互支持的活动，通常持续地嵌入整个产品团队的敏捷工作方式中。在 DevOps 项目中，测试人员是团队的一部分，该团队协作创建和优化用户故事。频繁的非正式评审是在开发需求的同时进行，包括形成每个用户故事的验收准则。这些准则是业务代表、开发人员和测试人员之间协作定义的。通常，测试人员基于独特的视角，通过识别缺失的细节、可替代场景保证可测试性，使得用户故事更为完善。因此，测试分析并非一个明确的独立活动，而是测试人员在协作用户故事开发中作为其角色的一部分执行的一个隐含活动。

基于对用户故事的分析，识别出测试条件。从测试的角度来看，通过分析测试依据来确定可以测试的内容——这些就是测试条件[Veenendaal24]。只要规定的验收准则足够详细和清晰，就可以很好地承担传统测试条件的作用。在 DevOps 背景下，通常使用 Gherkin 语法为驱动开发（BDD）和验收测试驱动开发（ATDD）等技术定义场景并识别要测试的条件，将其记录为给定用户故事的验收准则。同样，非功能需求特定的测试条件也通常以验收准则的格式定义。测试条件随后被转化为测试，这些测试需要覆盖已定义并达成一致的验收准则。有意义的测试条件也可以记录在测试章程中，用于探索性测试。当然，以上不仅仅适用于用户故事级别，也适用于史诗级别和特征/功能级别的测试。史诗和特征/功能级别的测试条件通常比用户故事级别的更抽象，并跨越多个用户故事，或代表非功能需求。基于规格说明的测试技术通常有助于从用户故事和验收准则推导出测试条件。但是，在 DevOps 背景下，这些测试技术通常不那么明显，因为测试人员根据他们的经验掌握了这些技术，并能够在上下文中灵活的使用它们。

随着测试在先（test-first）在 DevOps 中应用，覆盖测试条件集的测试将在代码开发之前，或至少与代码开发同步进行（并可能实现自动化）。这种方法有时在待办事项梳理和细化时就已经开始。对于自动

化的单元测试，可以考虑测试驱动开发（TDD）等方法。对于更高级别的测试，如前所述，行为驱动开发（BDD）和验收测试驱动开发（ATDD）是目前流行的方法，它们也与测试自动化高度相关。

对于大多数手工测试，随着测试团队的测试推进，在测试执行时测试还将被识别/完善。测试通常不像传统项目那样有详细的文档化记录，而是以探索性测试并以测试想法的形式记录。对于复杂和关键领域，采用更传统的测试设计和测试用例开发方法（使用正式的测试设计技术）可能是覆盖风险的最佳方式。然而，即使采用这种方法，与传统的顺序生存周期环境中的测试相比，文档量也将是有限的。测试的优先级遵循其所覆盖用户故事的优先级。用户故事的优先级取决于商业价值；最高优先级对应最高价值。重要的是要建立测试以覆盖功能和非功能风险是很重要的，但在 DevOps 中，特别重要的是覆盖回归风险。测试优先级通过将测试添加到 CI/CD 流水线作业中来实施。

确定了支持测试条件和执行测试所需的特定测试数据。然而，在 DevOps 中，与传统环境不同，所需的测试数据通常不会首先被指定为测试规格说明文档的一部分。一旦必要的工具和/或功能可用，就马上提供测试数据。可在手工测试开始前立即创建测试数据，以允许立即开始执行手工测试。但是，对于自动化测试，通常需要预先指定数据。测试数据通常以与测试环境相同的方式进行管理，并且可以按需部署到这些环境中。测试数据准备工具可以根据隐私和 GDPR 法规的要求，对生产数据进行匿名化、随机化和脱敏处理。

需要建立和维护需求、测试条件和测试之间的可追溯性。团队需要明确表达在他们的测试部分已经覆盖了各种用户故事和验收准则。因此，团队需要工具支持为每个用户故事分配标识符来组织和管理其需求（用户故事），以便这些标识符可用于确保测试根据约定的准则完成。通常，工具中链接的测试和用户故事提供实时覆盖度测量的可能性。通过自动生成的 CI/CD 流水线测试作业状态和日志，可以看到测试到用户故事的可追溯性。

### 2.4.2 SG2 执行测试实施

测试实施是将所有需要启动测试执行的工作事项到位。通常，支持测试执行的测试文档（例如测试规程）的开发被最小化。但是，自动化（回归）测试脚本应是优先考虑并被开发。回归测试准备也应尽快启动，并与其他测试活动并行进行。

在 DevOps 环境中，用于手工测试的详细测试规程并不常见。在一个知识完备的团队中工作时，测试很可能会在更高的抽象层上被记录下来，例如，用于探索性测试的测试章程。这对于执行测试的人员来说已经足够了，因为已经假设他们具备执行测试所需的领域知识和技术知识的水平。那些执行测试的人员在团队内工作，因此他们就能更好地理解代码的内容和方式，以及功能如何使用于各种使用方式。由于迭代内和迭代间的变化程度通常很高，开发详细的测试规程也会产生大量的维护工作。通常，许多手工测试是通过探索性测试来执行的。在探索性测试中，不制定详细的测试规程，而是使用高级别的单页测试章程，描述测试思路和指导测试人员提开展工作。当然，大多数 DevOps 团队使用自动化测试。当前典型的方法是测试驱动开发（TDD）、行为驱动开发（BDD）和验收测试驱动开发（ATDD）。虽然行为驱动开发本身不是测

试自动化，但它可以作为自动化验收测试的基础。用行为驱动开发（如 Gherkin 语法）编写的场景通常关联到测试自动化框架。使用一种或多种这些方法，将创建自动化测试脚本，例如，详细说明 Gherkin 场景，作为测试实施活动的一部分。测试脚本也作为测试文档，有足够的注释以便理解，并采用良好的编码实践。通常，会创建测试套件来对已开发的测试脚本进行分组。这些套件随后链接到 CI/CD 流水线中的测试作业。

在测试分析和设计活动中指定的特定测试数据，最好以可重新部署的格式创建。测试数据有时被设置为代码，即测试数据以模式文件和规范的格式定义。在 CI/CD 流水线内实施准入测试。该测试有时称为信心或冒烟测试，用于在流水线中每次测试执行作业开始时，决定测试对象是否准备好进行详细和进一步的测试。冒烟测试旨在当测试失败时向开发团队和利益相关方提供即时反馈。此外，通常为 CI/CD 流水线内的每个测试环境/阶段定义自动化的部署健康检查清单。在迭代规划中，在团队任务板上对与测试数据准备、测试环境变更或测试自动化实施相关的任务进行估计和规划。

关于特定实践 2.4 开发测试执行进度表，自动化测试执行由 CI/CD 流水线内任何代码的变更触发。识别触发场景并编码（Web 钩子）以在变更时启动测试执行。手工测试执行根据迭代规划进行安排，通过任务板作为测试任务进行管理，并在每日站会中讨论。

### 2.4.3 SG3 进行测试执行

根据每次更改时指定的 CI/CD 流水线作业，以及团队根据迭代计划开发的新用户故事执行测试。为提供快速反馈定义测试作业的先后顺序。如果有足够的资源，测试作业可以并行执行以最大化的反馈。云成本和可持续性是优化 CI/CD 流水线的关键。报告缺陷/事件，并生成测试日志。在 DevOps 项目中，软件尽可能频繁地交付到生产环境，并由代码变更触发。测试是该过程的组成部分，通过协作交付高质量的产品。在执行更大的测试集之前，执行冒烟测试作为 CI/CD 流水线中定义的测试作业的一部分。如果失败，流水线的执行不会继续到下一个作业，并将反馈提供给开发。如果失效的原因被确定为缺陷，则报告一个缺陷。

测试执行按照迭代规划中定义的优先级进行。部分测试可以使用文档化的测试规程执行，但通常情况下，更多的测试将使用探索性测试和基于会话的测试作为其框架执行。在探索性测试会话中，以预先准备的测试章程为指导，并且有时间限制。测试也可以在软件发布给所有用户之前在生产环境中执行。在众包测试的情况下，选定一组 Beta 测试人员在产品大规模发布前就产品的某些质量方面提供反馈。要求特征/功能能够根据预定义的前提条件（如用户或地理位置）开启和关闭（特征/功能的切换）。

随着 DevOps 开发，对组织和结构化回归测试的需求在越来越大。特别是在单元和集成测试级别的回归测试是持续集成过程的一部分。它基本上是一个自动化的“构建和测试”过程，发生在每次代码变更时，以便尽早快速地检测缺陷。持续集成允许定期运行自动化回归测试，并向团队快速反馈代码质量和实现代码的覆盖度。测试执行也根据由代码变更而触发的指定 CI/CD 流水线作业进行。这些自动化测试脚本将提供测试结果和测试日志。通常可以按史诗、用户故事、测试用例、测试套件、测试作业以及整个流水线的汇总和视图呈现测试日志。

实际结果与期望结果之间的差异被报告为缺陷。缺陷管理通常是开发系统的一个组成部分，并提供执行和代码变更的完整可追溯性。此外，所有系统都会自动提供测试日志，以按时间顺序记录测试执行的相关细节。

随着团队开发和测试新的用户故事，在敏捷项目中通常会有关于是否应记录所有发现的缺陷的讨论。特别是在团队同地办公的情况下，测试人员需要与开发人员进行沟通，那些可以立即修复并纳入在下一个构建中的缺陷，可能不需要记录，只需要修复即可。有些团队只记录逃逸出迭代的缺陷，有些团队记录那些当天无法修复的缺陷，有些团队只记录高优先级缺陷。如果并非所有发现的缺陷都被记录，则必须有准则来确定哪些缺陷应该记录，哪些不应该记录。

即使在 DevOps 团队中，在测试执行期间记录数据以确定被测试项是否满足其定义的验收准则并确实可以标记为“完成”，这也被认为是一种良好实践。测试数据记录应由使用的自动化提供支持。在应用基于经验的技术（如探索性测试）时也是如此。可能需要记录的有用信息示例包括：测试覆盖率（覆盖了多少，还有多少需要测试）、测试期间的观察结果（例如，被测系统和用户故事是否稳定）、风险列表（哪些风险已被覆盖，哪些仍然是重要的风险）、发现的缺陷和其他问题，以及可能的未决问题。记录的信息应以一种方式捕获和/或汇总成某种形式的状态管理工具（例如，测试管理工具、任务管理工具、任务板），以便团队和利益相关方能够更容易了解所有进行的测试的当前状态。当然，团队需要的不仅仅是单独测试任务的状态，而且需要知道用户故事的整体状态。当一个待办事项满足完成的定义时，它就完成了。当 DevOps 团队使用持续部署作为实践时，完成的定义通常也包括成功部署作为标准。

### 2.4.4 SG4 管理测试事件至关闭

在 DevOps 中，“事件”一词通常用于表示影响用户的真实生产事件。因此，在本节余下部分，我们将使用“缺陷”一词来表示 TMMi®上下文中的测试事件。由于 DevOps 团队也负责运维，他们必须快速做出反应。DevOps 内的事件管理是一个 IT 服务管理（ITIL）过程：“通过尽快恢复正常服务运营来最大限度地减少事件的负面影响是一种实践。”

由于在 DevOps 中几乎所有内容都是代码化的，所有发现的缺陷实际上都是某些软件中的缺陷。如果是环境问题，那是环境的基础设施即代码（Infra as Code）中的缺陷；如果是测试数据的错误，那是测试自动化代码缺陷；如果是其他执行不可靠性，则可能是流水线代码缺陷等。如上一段所述，在 DevOps 环境中，并非所有发现的缺陷都会被记录。此特定目标仅适用于那些被记录并因此需要管理直至关闭的缺陷。原则上，在 DevOps 中管理事件很简单。如果事件被记录在任务板上，它会可视化为一个阻塞某个用户故事及其任务完成的缺陷。标准的 DevOps 工作方式将与阻碍进展的任何其它障碍或任务一样应用。它将在站会期间讨论并分配给团队解决。

如果决定将发现的缺陷推迟到另一个迭代，它们就变成了产品的另一个期望选项（或变更）。将它们添加到待办事项列表并相应确定优先级。当优先级被设置得足够高时，团队将会在下一个迭代中处理它们。

在每日站会和回顾会议中，监督迭代过程中发现和解决的缺陷数量以及未解决的缺陷数量，以及可能成为下一次迭代的待办事项中的一部分的未解决缺陷数量，这是一个良好的做法。

## 2.5 过程域 2.5 测试环境

*测试环境的目的是建立和维护一个适当的环境，包括测试数据，使我们以可管理和可重复的方式执行测试成为可能。*

通过测试环境过程域，建立并维护一个充足的测试环境，其中包括测试数据，在该环境中可以以可管理和可重复的方式执行测试。当然，在 DevOps 软件开发项目中，足够的测试环境也是不可或缺的。为每个 CI/CD 流水线阶段使用云技术，可以在受控状态下按需设置环境。由于迭代周期短，测试环境需要非常稳定并保持可用。测试环境中的问题，例如在 CI/CD 流水线中，将总是立即影响迭代的进度和结果。因此，正确管理测试环境和测试数据的配置和变更也至关重要。

### 2.5.1 SG1 开发测试环境需求

在项目早期进行测试环境需求的规格说明，此后在基于持续学习的基础上不断进行。理解产品价值流、生产环境、技术架构和正在使用的开发工具链，对于确定环境要求并随后制定驱动环境实施的需求非常重要。评审需求规格说明以确保其正确性、适用性、可行性以及对“真实”操作环境的准确表示。早期进行需求规格说明的优势在于提供更多时间来获取和/或开发所需的测试环境，这些环境基于虚拟化、容器化和其他在 DevOps 背景下常用的云原生方面。构建流水线（流水线的 CI 部分）中的环境通常以小型和简单的规模部署，并随着性能和可靠性期望的增长而持续开发。然而，在发布流水线（流水线的 CD 部分）中，可能需要进行端到端业务过程测试，因此需要相应复杂得多的测试环境。

在梳理和改进会话期间，并根据 CI/CD 剧本或平台团队（platform team）定义的基础设施即代码和配置即代码实践和标准，确定并定义测试环境的要求。测试环境和数据需求的优先级是基于相应的待办事项优先级。有时会指定一个带有验收准则的单独技术探针来定义更复杂的环境需求。需求通常使用符合基础设施/配置即代码的语言编写，遵循组织的 CI/CD 实践和术语。需求被映射到价值流或 CI/CD 流水线阶段。开发工具链与运营和部署技术保持一致，以确保其必要性和完整性。

### 2.5.2 SG2 执行测试环境的实施

实施部署流水线以及生产和测试环境应尽早开始，以便在第一次迭代开始时拥有环境的初始版本。测试环境是类生产环境，包括通用测试数据，用于在给定的 CI/CD 测试阶段执行相关的测试级别。高效的 DevOps 团队也会重用生产自动化来部署测试环境，并按需部署它们。所有与环境实施相关的任务都应像任何其他待办事项一样在待办事项列表中处理。

在 DevOps 背景下，基础设施即代码通常用于定义和实施环境组件，配置即代码用于定义和实施环境变量和配置，两者都通过自动化脚本和流水线实现。通常重要的是拥有按需提供环境的能力，在 CI/CD 流水线中使用自助服务，也用于探索性测试会话。编码环境的通用测试数据使用自动化测试数据生成和管理工具创建。通常在部署环境实例后、在环境用于其他活动之前，在流水线中运行冒烟测试。

### 2.5.3 SG3 管理和控制测试环境

测试和生产环境的管理和控制将贯穿整个 DevOps 项目。管理和控制环境以允许使用可用工具不间断地进行测试执行，这些工具支持创建、配置、分配和使用环境以及数据。在测试环境中执行系统管理，以支持供应、配置、分配、使用和退役，包括以下实践：

- 通过代码、自动化和监督来管理基础设施。
- 通过提供登录详细信息来管理对测试环境的访问。
- 在测试执行期间对影响进度的问题提供技术支持。
- 提供监督和遥测，可用于分析测试结果和缺陷原因。

此外，尽可能管理和自动化测试数据的供应、分配、修改和使用，以支持测试过程。

创建和分配测试环境是自动化的、可协调的和可测量的，以实现最大的可用性及效率。测试环境部署实现自助服务或按需环境供应（启动/关闭）以及环境实例的监督，特别是对于持久性的环境。在供应或使用测试环境时发生的问题由 DevOps 团队记录并管理直至关闭。当观察到问题时记录测试环境事件，无论是作为失败测试的一部分自动记录还是由环境用户记录。

作为测试环境过程域的主要结论，特定目标和实践在本质上仍然适用且没有改变。

# 3 TMMi®3 级已定义

## 3.1 过程域 3.1 测试组织

*测试组织过程域的目的是识别并组建一支具备高专业技能的团队，由其负责测试工作。除执行测试外，测试团队还会在充分理解组织当前测试过程与测试过程资产优劣势的基础上，对组织的测试过程及测试过程资产进行改进管理。*

### 3.1.1 SG1 建立测试组织

测试组织是一个经常被误解的过程域。许多人将其理解为 TMMi 需要一个独立的测试团队，甚至独立的部门来开展独立测试。由于 DevOps 团队本身是跨职能设计，旨在优化价值流动并强化反馈，设立独立的测试组织会违背这一理念。因此，此处所指的高技能测试专业人员，通常分散在各个跨职能团队与不同角色中。一般情况下，测试能力中心或测试社区侧重推动组织级测试过程与专业能力的改进，包括技能培养与职业发展路径，而长期存在的 Scrum 团队则聚焦于优化价值交付。二者共同构成组织改进的机制。

所谓测试能力中心的典型任务与职责包括：建立、管理并改进组织层面通用的标准测试过程，以及向各跨职能 DevOps 团队分配具备测试经验、知识与技能的人员。测试能力中心通常负责测试方法与测试流程的制定。。需要注意的是，测试人员在日常工作中一般属于 DevOps 团队，并因此参与持续集成持续（CI/CD）交付流水线中的持续测试，但除此之外，测试人员通常还隶属于一个专注于测试专业领域的测试组织，如测试能力中心或测试协会。

测试组织需要在与测试及质量相关的领域发挥领导作用。然而，测试必须被视为价值流中不可或缺的一部分，且与测试相关的活动应扩张到 DevOps 组织的各个角色中。为了在所需角色中建立测试技能，测试专家会提供质量支持。测试能力中心模式与 DevOps 非常契合，它能确保测试作为一门专业学科得以保留，并受到应有的重视。在 DevOps 背景下，测试组织以测试协会的形式存在。测试协会通常是一个非正式的、自我管理的测试人员群体，其成员来自不同的 DevOps 团队。测试协会的重要活动包括知识共享、小组学习、趋势观察等。协会的成员资格通常为自愿加入。测试协会也可以承担更正式的任务，例如制定测试职业发展路径、组织培训，以及确定、规划并实施测试过程改进。测试社区取得成功的一个重要约束条件是：需为其成员分配时间，用于他们参与到各项测试协会活动。

### 3.1.2 SG2 为测试专家建立测试职能

测试被视为一项有价值的职业与专业学科。DevOps 团队需要具备测试知识与技能，测试专家会在执行测试活动的同时，提供这些知识技能并指导团队其他成员。典型的 DevOps 测试人员所需的知识与技能与传统组织中的测试人员有所不同。当然，测试人员仍需具备“传统”测试知识与技能，但除此之外，通常还需要掌握测试自动化相关的知识与技能，并且能够执行测试之外的任务，例如脚本编写或需求工程。在

DevOps 环境中，行为驱动开发（BDD）和验收测试驱动开发（ATDD）是常用方法，因此这些也需成为 DevOps 测试人员技能组合的一部分。此外，持续集成持续交付（CI/CD）流水线设计与配置、部署自动化、配置管理以及部署策略，也应是测试人员的能力的一部分。由于 DevOps 测试人员在团队中工作，软技能与不断提升的技术技能同等重要。DevOps 测试人员属于所谓的 T 型测试人员，拥有多个扎实的知识领域，这一点应体现在测试职能的描述中。部分测试人员可能更擅长技术方面，而另一些则可更专注于设计思维与客户理解。一些基于 DevOps 的组织已设立了三种不同类型的测试职能：

- 软件测试开发工程师（SDET） - 负责开发测试和部署自动化解决方案，以及产品的可测试性功能，为开发测试（DevTest）成员提供支持。

- 开发测试人员（DevTest） - 同时负责产品测试的开发人员，能够分析需求、设计测试、实施并执行测试（通常利用软件测试开发工程师提供的可用测试自动化工具和可测试性功能）。

- 测试人员（或测试架构师） - 负责监督各 DevOps 团队之间的测试活动，识别并管理风险，必要时，也运行专注于产品的专项测试，但主要是了解整体测试工作情况。

无论采用何种形式，测试组织都应具备在 DevOps 环境中成为优秀测试人员所需的技能与积极性的成员组成，并被分配到特定的测试职能岗位（见上述示例）。尤其是在 DevOps 环境中，由于整个团队都会参与测试活动，因此至关重要的一点是，在其他非测试相关职能的描述中，明确其在测试活动中的期望、角色、职责以及所需的测试知识与技能。

### 3.1.3 SG3 建立测试职业路径

测试职业发展路径的建立是为测试专业人员提供提升知识、技能、地位和回报，从而使他们能够提升自己的职业生涯，例如，从 DevOps 测试工程师晋升为 DevOps 测试架构师、测试教练，或凭借客户理解能力成为产品负责人，或培养人际交往能力成为 Scrum Master。测试职业发展路径无需局限于测试领域，从长远来看，这将有助于在整个组织中传播质量理念。基于已定义的测试职业发展路径，需为测试组织的每位成员制定并维护个人测试职业发展计划。这一特定目标及其配套的特定实践，在 DevOps 环境中同样完全适用。DevOps 团队需要的测试人员，应了解他们自身工作的挑战性、能够指导团队其他成员，并且富有激情，具备工作积极性。其中许多问题都可以通过测试职业发展路径来解决。

然而，与遵循顺序生命周期的组织相比，DevOps 组织中测试职业发展路径的实际定义通常会有所不同。在传统组织中，测试职业发展路径通常关注纵向发展（从测试人员到测试经理），而在 DevOps 组织中，测试职业发展路径倾向于关注横向发展（从初级测试人员到高级测试人员，再到测试架构师，或许还能成为测试教练），或晋升为 Scrum Master 或产品负责人角色。

### 3.1.4 SG4 确定、计划和实施测试过程改进

测试改进是 DevOps 组织持续改进理念和价值流管理框架的基本组成部分，其理念聚焦于精益原则和系统思维。主要思想围绕强化流动增强价值流、消除浪费、增强强化反馈机制，并基于从 DevOps 流水线收集的事实数据（例如流动指标或四项 DORA 性能指标）来驱动改进。与其他改进措施事项一样，测试改进在组织的待办事项列表中会被进行记录、优先级排序和管理。

TMMi 特定目标及特定实践所描述的改进过程，主要通过在很大程度上由 DevOps 团队开展的回顾会议来实现，并依赖于流水线生成的数据和事实进行决策。回顾会议为 DevOps 团队提供了自我审视的机会，并制定具体的改进措施。回顾会议通常在迭代评审之后、下一轮迭代计划之前举行。作为流水线的一部分，团队会识别设计指标并建立数据收集机制，以便在价值流图上进行可视化，从而识别哪里出现了延迟、限制或产品质量问题。这些指标有助于识别（测试）改进的机会。DevOps 团队专注于测量与其工作密切相关的活动，例如测试执行时间、回归测试失败率、测试环境部署与配置时间；而大型组织则关注流动指标和四项 DORA 四项性能指标。

在回顾会议中，团队（部分基于度量）讨论本次迭代中哪些方面做得好、哪些方面有待改进，以及在下一迭代中承诺改进的事项。解决问题的技能也至关重要，例如可使用运用石川图或 5 Why 法（5 个为什么）等方法进行根本原因的分析。持续学习的文化鼓励团队成员不断获取知识，并积极分享 DevOps 相关的经验与想法。在每次回顾会议中，DevOps 团队都会识别并规划提升产品质量的方法，例如通过优化改进工作过程，或在团队层面或价值流层面调整“就绪定义”（Definition of Ready）和“完成定义”（Definition of Done）的标准。回顾会议可以产生与测试相关的改进决策，重点关注测试的有效性、测试效率以及测试质量。这些决策还可以涉及应用程序、用户故事、功能或系统接口的可测试性。所有团队成员，包括测试人员和非测试人员， 都可以对测试和非测试活动提出意见。回顾会议结束时，DevOps 团队应已明确将在下一迭代中实施的（测试）改进项。在下一迭代中实施这些改进是 DevOps 团队自身的责任，因此会以这些改进作为待办事项条目进行管理。测试人员通常负责测试改进的落实，以确保其在团队内得到执行。回顾会议是推动团队在项目生命周期中持续演变和改进的重要机制。

由于范围回顾会议的通常局限于单个迭代，因此所做的多为小规模但频繁的改进，主要聚焦于解决特定的团队问题。这些改进的重点往往不是跨价值流的学习以及改进成果的制度化。从（测试）过程改进的组织与管理方式来看，组织层面可能较少关注（测试）过程小组组织层面可能较少设立专门的（测试）流程小组，而更加强调团队的自我管理。虽然这本身并无不妥，但同样重要的是，必须建立一种机制，确保在局部取得成功的测试改进能够被组织内其他团队共享，同时也能解决那些超出单个 DevOps 团队能力范围的改进事项。一种可行的解决方案是，由测试组织（如测试能力中心或测试实践社群）指派代表参加各团队的回顾会议，以便在适当时机推动改进成果在整个组织范围内的共享与制度化。

大型组织通常还会在价值流层面专门组织回顾活动，以识别潜在的系统性问题。测试过程改进负责人也可从中识别提炼出影响整个组织测试工作的根本性问题或举措，并将其添加到价值流的待办事项列表中。

### 3.1.5 SG5 部署组织的测试过程并合并经验教训

此外，对于 DevOps 项目而言，根据需要尽可能地访问并复用组织标准的测试过程和测试过程资产，并从中产生附加价值，这一点非常重要。这些过程和资产当然需要与 DevOps 的工作方式保持一致，重点关注各级待办事项的质量保证和测试工作。测试组织将确保这些过程和资产在所有团队中得到推广和应用。（有关组织标准测试过程、测试过程资产及其与 DevOps 项目关联关系的更多信息，请参见“测试生命周期与集成”的过程域。）在 DevOps 项目中，自组织团队可对组织标准测试过程在 DevOps 项目中的实施以及测试过程资产的使用自行监督，可由自组织团队自行监督，例如在回顾会议中进行解决。一个推动前进的方向式是将过程和模板嵌入工具中，并通过流水线强制执行相关实践，例如，若代码覆盖率未达到目标，或违反了约定的编码规范，则自动拒绝任何拉取请求。若测试组织的代表（例如测试过程改进人员）能参与这些会议，便可跟踪实施情况，并协调解决跨项目问题。该代表还可将 DevOps 团队的经验教训和测试改进建议反馈给测试组织。这些经验教训和改进点随后可作为被整理为测试过程改进建议提交，并进行相应的管理。

## 3.2 过程域 3.2 测试培训方案

*测试培训方案过程域的目的是制定一项培训计划，以促进相关人员知识与技能的发展，从而能够高效、有效地执行测试任务和履行测试职责。*

测试培训方案的主要目标是为测试人员以及团队内外参与测试工作的其他相关人员提供必要的（测试）培训。一个有效的培训方案可确保所有参与测试的人员持续提升其测试知识与技能，并获得测试所需必要的测试的领域知识及测试相关的知识与技能。鉴于由于质量和测试是团队的共同责任，因此测试相关培训不仅应面向测试专业人员，还应提供整个 DevOps 团队。

### 3.2.1 SG1 建立组织测试培训能力

该过程始于识别组织在测试培训方面的战略需求。战略性的测试培训需求聚焦于为团队配备必要的技能与知识，以便在有效实施持续集成和持续交付流水线中有效实施测试实践，同时确保安全性问题得到解决。除了针对测试人员传统的及敏捷环境下的培训需求（如测试设计技术、探索性测试和覆盖率度量）外，当前还需获取与 DevOps 测试特性相关的知识与技能。具体示例包括：测试自动化工具与框架、安全测试（例如静态应用程序安全测试 SAST、动态安全分析测试 DAST 及渗透测试）、基础设施即代码、左移测试左移实践、测试驱动开发（TDD）、行为驱动开发（BDD）以及验收测试驱动开发（ATDD）。识别战略性测试培训需求不仅限于测试人员，同样至关重要的是，还需明确 DevOps 团队中其他成员所应具备的测试相关培训需求。

就像任何其他组织一样，测试培训需求将被转化为（轻量级）培训方案，可能也会同时解决一些特定的项目测试培训需求。培训方案需要进行评审，并获得 相应的承诺。请注意，尤其是在 DevOps 方面，通

过非正式沟通方式（例如在岗培训，午餐 培训课程，教练辅导和指导）可以有效地传授一些技能，而其他技能则需要正式培训。此外，DevOps 实践如嵌入知识和持续试验，将推动培训方案开发与实施的方法。

### 3.2.2 SG2 提供必要的测试培训

总体而言，在传统与 DevOps 组织中，SG2 “提供必要的测试培训”的特定实践是相同的。根据组织级培训计划，识别出需要接受培训的团队成员，以确保其能够有效履行测试职责。随后安排培训事宜，包括所需资源，并进行培训。培训活动，明确所需资源并组织实施。

如上所述，DevOps 组织通常会使用非正式的培训方式。例如，除了正式培训外，不断鼓励和使用在职指导和辅导。事实上，辅导被视为是组织各级人员的共同每个人的责任，并且在组织的 各个层面上都是需要的。结对工作也是团队常用的技能提升和知识传递方式。

所开展的测试培训所有已实施的测试培训（无论正式或非正式）均需建立培训记录，并对测试培训的有效性进行评估。参加过的（测试）培训也可在回顾会议中进行评价， 团队会讨论参与培训人员的知识与技能是否已得到提升，足以更好地完成他们的测试任务。

## 3.3 过程域 3.3 测试生命周期与集成

*测试生命周期与集成的目的是：建立并维护一套易用的组织级测试过程资产和工作环境标准，并将测试生命周期与开发生命周期进行集成和同步。这种集成的生命周期将确保测试在整个条价值流中得以尽早介入和全程参与。测试生命周期与集成的目的还包括：基于已识别的风险和所定义的测试策略上，定义覆盖多个测试级别（在 DevOps 语境下体现为 CI/CD 流水线中的阶段）的协同一致性测试方法；并依据所定义的测试生命周期，制定总体整体测试计划。*

### 3.3.1 SG1 建立组织测试过程资产

测试组织的一项重要职责是根据组织的测试方针与目标，定义、记录并维护一套标准测试过程。该过程通常包含对需执行的各类测试级别和测试类型的描述。由于 TMMi 并不强制要求详尽的书面化测试过程，因此采用带有示例说明的模板往往是建立统一工作方式的有效途径 [Galen]。例如，“敏捷测试象限”提供了一种简洁高效的方法，团队可借此就项目中所采用的测试级别与测试类型达成共识；而强制性的测试级别则可作为标准纳入“完成定义”中。此外，对 CI/CD 流水线中各测试阶段的概览，也是一种很好的通用文档形式。通过模板其结构明确了所需开展的活动，以及应收集的信息和需记录的结果。过程不必表现为严格的活动序列。若存在依赖关系，可在模板中以注释形式予以说明。模板的实际价值在于清晰传达过程的核心意图，同时避免“冗长”过程文档中常见的歧义并可有效避免传统冗长文字型流程文档中常见的歧义问题。在 DevOps 组织中，若已有行之有效的过程，可能仅需以轻量化的方式进行记录，或补充少量内

容即可。总体而言，DevOps 组织高度依赖自动化，其过程与规则通常被由工具捕获和强制执行。因此，尽管测试过程未必总是以独立文档的形式存在，但它们仍然是开发与部署过程中至关重要的组成部分。当然，即便采用此种工作方式，确保所有相关人员对应用的过程达成共同理解依然至关重要。DevOps 组织在建立过程时通常遵循以下一些关键指导原则：

- 将过程中中“必须做的事情“与准则建议性指南明确分开。
- 一个过程文档通常不超过两页。
- 过程中不包含“如何操作”信息或具体工具使用的说明，除非组织已决定跨所有项目（无论其大小和规模）统一强制执行的操作。
- 单独的指导方针包含剪裁/计划选项以及“如何操作”信息。请注意，通过“如 何操作”信息，人们也可以决定 参考现有的书籍或论文。

大型组织可采用平台工程实践，强制执行规定标准的待办项工作流或 CI/CD 流水线工作流。这意味着所定义的过程将通过特定工具予以实现，并要求各团队使用这些工具。若允许进行过程裁剪，则生命周期阶段是可选的则相应生命周期阶段为可选项。

在许多大型组织中，通常可见四层结构的过程资产，包括：方针、过程/实践、工作指导/规程以及支持工具/模板。这种分层结构并非 TMMi 模型的要求，而是源于许多大型组织在实施其过程资产时所采用的方式。尽管过程资产的具体形式由各组织自行决定，但当前许多 DevOps 组织发现，仅需两层结构的过程资产即可满足需求。这一目标通过以下方式实现：将方针声明与对应的过程描述进行整合，形成第一层级，用于明确执行过程时“必须完成的工作”。第二层级则提供“如何操作”的指导，包括过程的具体实施方法和裁剪方式。该层级可视为对过程裁剪的支持，赋予团队在既定框架内自主决定具体工作方式的权限灵活性，通常包含相应的支持性模板。在大多数 DevOps 组织中，一步一步的过程已被工具使用指南以及培训/辅导所取代。综上所述，过程是模板（如测试章程模板、测试会话表模板或测试计划模板）的使用，其中已隐含过程中必要的活动。这正是开发有效敏捷过程描述时常用的一组技术方法如测试备忘录模板、测试会话记录表模板或测试计划模板等模板，其结构本身即隐含了必要的过程活动，因而可直接作为过程的载体。这种将模板作为过程描述载体的方法，已成为构建高效敏捷流程描述的常见实践 [McMahon]。

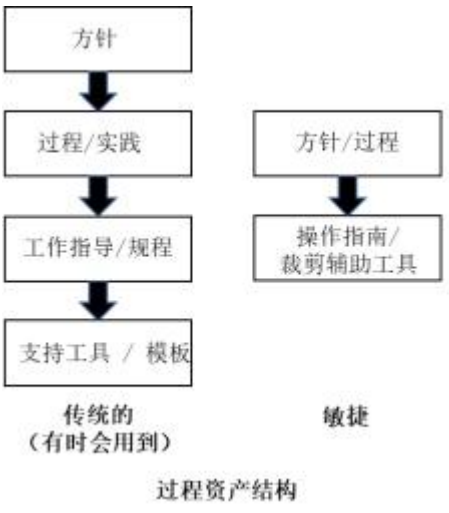


*图5：过程资产的层级*

组织的标准测试过程集可由项目或团队进行裁剪，以创建其特定的已定义过程。在特定实践 1.3“建立裁剪标准准则与和指南”的背景下，许多组织采用一种称为“向下裁剪”的方法。该方法要求相关人员花费付出精力去解释为何某项内容不适用。然而，当进行向下裁剪时，裁剪的幅度应有多大？是否有明确的最低限度是否存在一个明确定义的最低限度？如果以一个最小集合为起点，其中该最小集合是所有团队都必须执行的内容，则可以消除将“必须执行项”裁剪掉的风险。“向上裁剪”的方法更契合与敏捷和

DevOps 理念更为一致，且完全符合 TMMi 的要求。一组简单的准则有助于在“向上裁剪”时提出正确的问题。若缺乏相应的支持性准则，一些团队则容易出现过度向上裁剪的情况。

为支持标准测试过程的部署实施，通常会将建立并维护一个测试过程资产库，通常以维基（wiki）形式呈现，测试人员可在其中获取模板、最佳实践和检查清单等资源，以辅助其日常测试工作。这些模板通常作为工具实现的一部分，例如作为 CI/CD 流水线的蓝图、待办项管理工具中的必填字段或标准模板。在 DevOps 环境中，建立测试过程资产库便是有意义的，前提是只要团队在测试活动中使用这些所提供的资产能够获得附加价值。，建立测试过程资产库便是有意义的。类似的逻辑也适用于对于测试过程数据库而言，其逻辑基本相同，该数据库用于收集并共享测试数据（例如估算值、覆盖率、质量度量），并供供各团队使用。需确保所收集和提供的任何数据均能促进跨团队的学习与经验分享，从而为团队带来增加价值。

当然，在 DevOps 环境中，工作环境标准同样会用于指导项目工作环境的创建。

### 3.3.2 SG2 集成测试生命周期和开发模型

标准测试生命周期模型定义了各测试级别的主要活动和交付成果物。该模型与开发生命周期模型保持一致，以确保测试活动在阶段划分、里程碑、交付成果和具体活动方面与开发过程相集成。生命周期的集成应确保针对每个待办项的，测试活动能够尽早地参与到项目中。当然，在 DevOps 模式下，开发与测试应在整个生命周期中实现全面融合，测试应尽早可能提前开展。由于测试人员作为团队成员参与迭代计划，因此风险得以提前识别，测试活动得以事先协商确定，这会进而影响工作量估算和团队承诺，从而确保测试有效融入生命周期。

因此，本特定目标同样适用于采用开展 DevOps 软件开发模式的组织。在 DevOps 背景下，开发与测试在团队层面理应在团队层面已实现全面整合，并且团队需对此达成共识对集成方式应有共同的理解。在此类情况下，无需额外采取额外措施。而在其他情况下，若整合与早期介入尚未完全落实，则本特殊特定目标将是一个至关重要非常重要的改进领域，并将应推动促使在迭代内中实现更加协调一致的的开发与测试活动更协调一致的推进。

### 3.3.3 SG3 建立主测试计划

在 TMMi 3 级成熟度中，测试涉及主测试计划的制定，旨在协调各测试级别的测试任务、职责分工和测试方法。这有助于避免不同测试级别之间出现不必要的测试重复或遗漏，从而可显著提升整体测试过程的效率与质量。主测试计划描述了针对特定项目的测试策略的具体应用实施方案，包括将要执行的测试级别以及这些级别之间的相互关系。在 TMMi 2 级成熟度，（测试）计划工作已在“测试计划”过程域中作为组成部分，分别在发布层级和迭代层级予以实施同步开展。一个 DevOps 团队通常会执行多个测试级别，这些级别测试活动均应在发布计划和迭代计划中得到恰当的考虑和覆盖。

然而，一些遵循 DevOps 生命周期模型的项目采用了某种混合模式，即在 DevOps 团队内部开展测试的同时，部分测试活动在团队外部另行组织。在此类情况下，主测试计划将用于定义和管理 DevOps 团队与团队范围之外执行的测试级别或类型之间的关系，例如外包的非功能性测试（如安全性、性能测试）或与外部方协作开展的系统集成测试。此类外部测试活动通常在任务看板上作为一个额外的阶段予以体现。所有与该特定目标相关的特定实践在此类场景下通常也同样适用。

如果所有测试活动均在 DevOps 团队内部完成，则制定专门的主测试计划并无额外价值，因此在此类情况下，本 TMMi 目标可能不具适用性。此时，发布层级和迭代层级的计划应涵盖所有必要的测试内容。然而，有必要检查团队是否已建立一种通用的测试方法，并达成共识且持续改进。通过审视“完成的定义”中提及的具体测试级别、CI/CD 流水线/工作流，或一份简单的敏捷测试象限文档，这些内容也可被视为一种轻量化的主测试计划。

## 3.4 过程域 3.4 非功能测试

*非功能测试过程域的目的是提升测试过程能力，以便在测试计划、测试设计和执行阶段全面纳入非功能测试。*

非功能性测试是否重要，与所采用的生命周期无关。具体需要测试哪些非功能性方面，将取决于正在开发的产品。当然，这一点在 DevOps 开发模式中同样适用。因此，非功能性测试也完全适用于 DevOps 软件开发过程。

然而，根据非功能性方面的性质以及所采用的测试方法，某些非功能性特性——例如性能和可靠性——无法通过传统方式在短迭代中进行测试。因此，通常需要采取不同的方法来测试这些非功能性属性。这正是 DevOps 带来的一个根本性改进：质量属性得以尽早测试，而不再被拖延到项目后期。尽管并非所有组件或功能在迭代之初就能完全具备以开展全面的非功能性测试，但这些测试结果仍可作为早期识别质量问题的重要依据。

用户故事应同时涵盖功能性和非功能性要素，以确保为用户和客户开发出正确的产品。ISO 25010 质量特性可帮助梳理需求，测试人员也应充分考虑这些非功能性要素[ISO25010]。在 DevOps 实践中，除了标准的 ISO 25101 质量特性外，还特别引入了可观测性（由遥测技术提供支持）这一全新质量特性。

### 3.4.1 SG1 执行非功能产品风险评估

在测试策划过程域中，作为具体目标 SG1 执行风险评估的一部分，DevOps 项目所涉及的产品风险会议及相应技术，如风险扑克、ROAM 方法和威胁建模，现已明确扩展至涵盖非功能性方面及其相关风险。在 DevOps 实践中，团队通常特别关注系统的可观测性、安全性与合规性。随着企业日益向数字化和云端转型，IT 系统面临的潜在风险和漏洞也显著增加。威胁建模是一项在功能生命周期早期阶段开展的过程——即分析与细化阶段，此时会进行风险评估，以识别威胁，并借助结构化模型制定应对策略，从而有效降低风险。

其最终成果往往以非功能性验收标准的形式体现在功能层面，甚至可能推动更多用户故事的创建，或直接指导特定的架构决策。而在迭代层面，风险评估则会在用户故事细化过程中展开，重点围绕非功能性需求进行输入，并最终更新故事的验收标准。理想情况下，所有团队成员，包括产品负责人以及部分其他利益相关方，都应参与产品风险会议。对于某些非功能性领域，可能还需邀请相关专家提供支持。通过这些产品风险会议，团队通常会形成一份经过优先级排序的非功能性产品风险清单，进而更新功能与用户故事的相关文档。这份产品风险清单不仅为 CI/CD 流水线的设计及非功能性测试方法的选择提供了重要依据，更会主动影响产品的整体架构设计，以及开发过程中所选用的技术方案，例如通过容器化技术提升系统的可扩展性和可靠性。正如测试计划过程域中所强调的，在 DevOps 项目中，无论是所采用的过程，还是最终生成的文档，都将比传统项目采用顺序式生命周期模型时更为轻量化。

可观测性是指通过分析组件或系统的外部输出，来理解其内部状态的能力，从而实现监控、故障排除和优化。可观测性利用日志记录、链路追踪和指标，提供全面的信息，帮助用户清晰地掌握软件的运行状况。这些信息由遥测技术自动收集——遥测技术能够针对功能性和非功能性质量特性进行量化测量。例如，实时站点遥测用于监测软件的实际运行情况，如内存使用率或 CPU 负载；而用户遥测则侧重于衡量用户互动行为，比如点击跟踪以了解功能使用情况，或表单完成率等数据。

### 3.4.2 SG2 建立非功能测试方法

为降低已识别并优先排序的非功能性产品风险，我们制定了针对相关非功能性质量特性的测试方案。CI/CD 流水线的设计应涵盖非功能性测试环节。在具体迭代过程中，额外需测试的非功能性方面会在迭代规划阶段明确；同时，CI/CD 流水线中的非功能性测试将伴随每次变更自动执行。此外，非功能性质量特性还会在部署至生产环境后通过依赖可观测性的遥测技术进行持续监控与评估。在 DevOps 实践中，服务级别指标（SLI）和服务级别目标（SLO）是常见的管理方式。通常，待测试的非功能性方面优先级清单会与该迭代中开发并测试的用户故事紧密相关。然而，在迭代过程中，可能会暴露出新的非功能性产品风险，从而需要开展额外的测试工作。诸如需额外测试的新非功能性产品风险等问题，一般会在每日站会中进行讨论。

在迭代层面制定的非功能测试方法，旨在降低非功能风险，通常包括根据非功能风险的级别和类型，识别合适的非功能测试方法与测试技术。此外，它还通常涉及辅助工具的使用，以及针对非功能测试的自动化测试策略。而在发布层面，非功能测试的实施方法则需达到更高层次，并应基于已定义的 CI/CD 流水线展开。无论是发布层面还是迭代层面的非功能测试方法，均属于整体测试方法的一部分，将统一存放在或展示于团队/项目的 Wiki 页面上。同时，非功能测试的结果还需呈现在产品仪表盘中。

非功能性退出标准是“完成定义”（DoD）的一部分。确保 DoD 包含与非功能性测试相关的具体标准至关重要，例如 API 响应时间，或安全方面的预期，如“前端网页已通过 OWASP 十大风险列表的测试”。此外，在 CI/CD 流水线中，部署后所测量的服务水平指标（SLI）和服务级别目标（SLO）也可作为退出标准，通常要求避免 SLI 或 SLO 的性能退化。每次迭代都应落实经双方同意的一系列非功能性验收标准，甚至可

能包括用户故事，并满足 DoD 中规定的非功能性（测试）退出标准。不仅在迭代层面会为非功能性需求制定 DoD，通常在功能或发布层面也会有跨多个迭代的 DoD。另外，在 DevOps 背景下，人们更加注重“右移”策略，即通过持续监控，在生产环境中实时衡量非功能性特性，并利用部署后的 SLI 和 SLO 数据进行评估。

### 3.4.3 SG3 制定非功能测试分析与设计

这个具体目标主要沿用了相同的实践方法，但如今从非功能性的视角出发，例如针对测试设计与执行过程域中的 SG1：使用测试设计技术开展测试分析与设计。在测试分析与设计阶段，非功能性测试的实施思路会被转化为具体的测试条件和测试用例。在 DevOps 实践中，测试分析与设计以及测试执行是相辅相成的活动，通常会在整个迭代过程中并行开展。这一点同样适用于大多数非功能性测试。因此，非功能性测试分析并非一项独立的明确活动，而更多地是测试人员（DevOps 质量工程师）在协作式用户故事开发与优化过程中，作为其职责的一部分所自然开展的一项隐性活动。

基于非功能性验收标准和用户故事的分析，确定了非功能性测试条件。这些测试条件本质上是对需要测试或覆盖的“事项”的明确识别。只要所定义的验收标准足够详细且清晰，它们通常就能取代传统的测试条件角色。随后，这些验收标准会被进一步转化为非功能性测试用例。在进行非功能性测试时，往往建议从更高层次开展测试分析，而不仅仅局限于用户故事层面。例如，通过对某个功能、史诗或一组故事进行分析，可以识别出比用户故事层级更抽象、且跨越多个用户故事的非功能性测试条件。此外，在 DevOps 实践中贯彻“测试先行”原则，覆盖所有非功能性测试条件的非功能性测试用例将被提前识别出来（甚至可能实现自动化），这一过程可与代码开发同步进行，或至少与之并行展开。

对于大多数手动的非功能测试，测试用例会随着团队在非功能测试执行过程中的推进而逐步确定或完善。通常，这些测试不会像传统项目那样详细记录，而是以探索性测试中“测试思路”的形式呈现。然而，针对复杂且关键的非功能领域，采用更传统的测试设计方法，并结合正式的非功能测试设计技术来开发测试用例，可能是有效应对风险的最佳方式。不过，即便采用这种方法，文档的详尽程度仍会比传统顺序生命周期环境下的非功能测试更为精简。此外，非功能测试的优先级通常遵循其所覆盖用户故事的优先级排序；但有时，测试的优先级也可能由准备和执行特定非功能测试所需的时间决定。

识别可观察性和遥测需求及用例，也可能是非功能性分析与设计的成果之一，以帮助在生产环境中衡量非功能性指标。

为支持非功能测试的执行，已确定所需的特定测试数据。在 DevOps 实践中，通常不会首先将这些测试所需的数据完整地写入测试规范文档中，而是常常作为用户故事的一部分记录下来，例如以用户角色的形式呈现。不过，只要具备必要的工具和/或功能，用于支持非功能测试的测试数据通常会即时生成，以便快速启动非功能手动测试。相比之下，对于自动化非功能测试，数据往往需要事先明确指定。此外，由于部分非功能需求是在生产环境中进行测试的，因此测试数据也可以直接采用生产环境中的实际数据。当然，在这种情况下，隐私保护与 GDPR 合规性尤为重要。

需要建立并维护非功能性需求、测试条件与测试之间的可追溯性。团队需明确表明，他们已在测试过程中覆盖了各类非功能性用户故事及其验收标准。在大多数 DevOps 组织中，用户故事是制定一系列相关验收标准并进一步设计测试的基础。采用这种方法，可实现从需求到测试的横向可追溯性。

### 3.4.4 SG4 执行非功能测试实施

测试实施是指将启动测试执行所需的一切准备工作就绪。通常，用于支持测试执行的测试文档开发（如测试过程）会被尽量精简，取而代之的是优先开发并自动化的（回归）测试脚本。在 DevOps 模式下，凡是能够自动化的部分，都应尽可能实现自动化。

非功能测试的实施将遵循许多已在特定目标 SG2“执行测试实施”中描述的做法，该目标属于测试设计与执行过程域。具体需要完成哪些工作，以及如何开展非功能测试，很大程度上取决于所采用的方法、使用的技术，以及需测试的具体非功能性特性及其测试级别。例如，探索性测试虽然准备工作较少，适合用于可用性测试，但通常不太适用于大规模的可靠性与性能测试。

对于一些非功能性质量特性，测试数据的可用性至关重要，需要在测试实施过程中专门创建。这一点与传统项目基本相同，尽管传统项目中的相关文档可能较为简略，但仍便于后续用于回归测试的重复利用。如前所述，由于部分非功能性需求已在生产环境中得到验证，因此这些测试数据也可以直接采用生产环境的数据。

当然，测试实施与准备工作将尽快与其他（测试）活动并行展开。这并非一个独立的阶段，而是一系列需要完成的活动（已列在任务看板上），以确保测试能够高效、有效地执行。

### 3.4.5 SG4 执行非功能测试

与本过程域中的先前目标一样，我们将主要参考本文前面讨论的测试设计与执行过程域中相关的具体目标 SG 3 执行测试。在 DevOps 环境中开展非功能性测试、报告测试缺陷以及编写测试日志的实践，与传统 DevOps 环境中的功能测试基本相同。通常，这类工作所需的文档强度会远低于传统项目，甚至常常无需编写详细的测试过程和测试日志。需要注意的是，非功能性测试与指标还将在部署后继续进行，并通过服务级别指标（SLI）和服务级别目标（SLO）来有效管理这一过程。

非功能测试的执行是 CIO/CD 流水线的重要组成部分，其开展方式与迭代计划阶段所确定的优先级保持一致。部分测试可能采用已文档化的测试过程来执行，但通常情况下，许多非功能测试仍会以探索式测试和基于会话的测试作为主要框架。随着迭代开发的推进，对回归测试的组织与结构化需求也日益增加。为此，我们将借助自动化回归测试及配套工具来实现。当然，回归测试同样适用于系统中已被确认为关键测试的非功能性方面。

在测试过程中发现的非功能性缺陷可能会被团队记录并上报。通常，在 DevOps 项目中，各方会讨论是否应将所有发现的缺陷都记录下来。有些团队只记录那些逃过迭代阶段的缺陷；有的则会在当前无法修复

时才记录；还有的仅记录高优先级的缺陷。如果并非所有发现的缺陷都会被记录，那么必须制定明确的标准，以决定哪些缺陷该记、哪些无需记录。有时，缺陷会被直接记录在任务看板上——可以是作为任务标签贴附，也可以单独创建一个任务，以便直观地显示该缺陷正阻碍故事及其相关任务的完成。此外，也有团队选择使用专门的工具（如缺陷管理或缺陷跟踪工具）来记录和归档发现的缺陷。不过，这类工具应尽量保持轻量级，避免强制提交那些对非功能性测试缺陷报告毫无价值或价值甚微的内容。若非功能性需求已在生产环境中进行监控，当系统性能指标（例如 SLI 未达 SLO 要求）触发降级时，系统将自动回滚至先前版本，从而确保整体系统性能不受影响。

在 DevOps 团队中，记录非功能测试执行过程中的数据也被视为一种良好实践，以便判断被测项目是否满足其设定的验收标准，从而真正能够被标记为“已完成”。所记录的信息应被收集并汇总至某种形式的状态管理工具中（如测试管理工具、任务管理工具或任务看板），以确保团队和利益相关方能够轻松了解当前所有已执行测试的进展情况。

## 3.5 过程域 3.5 同行评审

同行评审过程域的目的是验证工作产品是否满足其特定要求，并及早、高效地消除选定工作产品中的缺陷。一个重要的附带效果是增进对工作产品以及可能预防缺陷的更好的理解。

### 3.5.1 SG1 建立同行评审方法

DevOps 团队通常不进行正式的同行评审，即他们通常没有固定的时间点让人们聚在一起对产品提供反馈。然而，他们通过在开发过程中持续进行非正式的同行评审来实现同行评审的目的。这在许多 DevOps 组织中是一种常见实践。由于 DevOps 更侧重于流动，并在软件开发中实施精益实践，任何评审实践都应支持这种快节奏的开发模式，而不是成为流程中的瓶颈或造成延迟。评审较小的工作项是典型做法，例如，频繁地对几行代码进行代码评审，以及通过协作定义待办事项的验收标准。评审实践非常重要，甚至被纳入流程中的强制环节，例如，强制要求每次拉取请求都必须进行评审，或强制要求符合"就绪定义"。TMMi 的这一目标涵盖了在项目内建立同行评审方法的实践。评审方法定义了评审活动得开展方式、场景、时机，以及这些活动是正式的还是非正式的。建立同行评审方法也适用于 DevOps 项目。尽可能将评审方法的部分内容通过工具落地，使其成为不可规避的环节，例如，在管理工作流的任务看板各阶段之间、在拉取请求中、或作为 CI/CD 流水线中的质量阀。

在 DevOps 项目中通常执行的同行评审示例：

- 在迭代过程中定期与团队和业务干系人举行用户故事的待办事项梳理会议。
- 与其他团队成员每日开会，讨论正在开发的工作产品（例如，代码或测试）并提供反馈。
- 尽早并频繁地向客户演示产品，至少在迭代结束时的迭代评审环节进行演示。

- 评审工作产品，如用户故事、通过拉取请求进行的代码更改、作为基础设施即代码一部分的脚本和配置文件。
- 在迭代期间结对完成各类活动。

规格说明不完善往往是项目失败的主要原因。规范问题可能源于用户对其真实需求缺乏洞察、缺乏系统的全局视野、功能冗余或矛盾，以及其他沟通不畅的情况。在 DevOps 开发模式中，编写用户故事是为了从业务代表、开发人员和测试人员的角度梳理需求。在顺序开发模式中，对特性的共同愿景是在需求编写后通过正式评审完成的；而在 DevOps 开发中，这种共同愿景是在需求建立过程中通过频繁的非正式评审完成的。这些非正式评审会议通常被称为待办事项梳理会议。在梳理会议期间，团队和业务代表使用评审技术来建立共识，确定实现所需的详细程度，并澄清待解决的问题。

团队对正在开发的工作产品进行几乎不间断的评审是整个团队方法的一部分。进行这些评审的目的是及早识别缺陷，并找出改进的机会。在 DevOps 中应用的方法和技术，如 XP 或结对编程，也将同行评审作为核心实践，为团队创建反馈循环。当然，在遇到高复杂度或高风险的情况下，团队可以选择应用半正式或正式的评审技术，例如，审查。在这些情况下，有明确的理由投入精力开展更正式的评审，采用更规范的工作方式。作为评审方法的一部分，可以定义标准，说明何时应用更正式的评审技术。

虽然需求工程强调通过非正式评审、走查、审查或基于视角的阅读等方法来验证需求，但敏捷和 DevOps 方法更倾向于通过频繁和早期的反馈来验证需求，以快速实现有价值的产品增量。通过以原型或部署到生产环境的集成产品增量的形式快速展示成果，减少了对早期正式验证的需求。如果增量没有完全满足干系人的要求，其差异部分会以新需求的形式重新纳入产品待办事项列表，并与所有其他待办事项一起确定优先级。在迭代评审期间，对迭代周期内实际交付成果进行演示，是一种非常有效的方式，能够围绕具体产出物激发基于验证的对话。没有什么比能够实际看到功能运行状，以及了解已部署特性对最终用户行为的影响更能聚焦对话了。演示是在迭代评审期间执行的一项活动。团队会向相关方展示并讨论已集成到系统中的功能，必要时对产品待办事项列表或发布计划进行调整，以反映讨论中获得的新认知。

同行评审标准可以通过自动化方式定义，也可以作为检查清单嵌入工作流程中。对于代码评审，定义了入口准则，如强制性的单元测试执行、代码复杂度度量、代码风格指南符合性。这确保了只有符合入口准则的代码才会被评审。作为出口标准，代码需要通过多名团队成员的评审。如果未通过，则需要在更新代码后重新开展评审。这通常意味着代码无法进行合并操作。

对于待办事项，入口准则通常仅适用于正式评审，因此在 DevOps 项目的待办事项优化会议中，准入标准往往不具备适用性。。然而，出口准则的使用是适用的并且具有附加价值。INVEST[Wake] 就是一套在 DevOps 项目中被广泛参考的准出标准，常用于用户故事的评审与更新工作。INVEST 是一个首字母缩略词，包含以下构成良好用户故事的要求：

图 6：INVEST 用户故事标准

- 独立的（Independent）
- 可协商的（Negotiate）
- 有价值的（Valuable）
- 可估算的（Estimate）
- 小的（Small）
- 可测试的（Testable）

当然，也存在其他与质量相关的出口准则，并且也可以有益地使用。

评审出口准则在不同的工作产品的"完成定义"中定义。不满足标准的项将被移回，不能进入下一阶段。

### 3.5.2 SG2 执行同行评审

当然，任何工作方法都不应只停留在既定共识的层面，更需要得到严格遵循与落地执行。团队需要在迭代期间花费大量时间在待办事项梳理会议上，将非正式（以及可能的正式）评审作为日常工作的一部分，并定期进行干系人演示。期望至少在每个迭代结束时进行干系人/客户演示。

测试人员作为 DevOps 团队的成员，参与各类评审会议至关重要。尤其是在讨论、评审那些将作为迭代测试依据的工作成果时 —— 例如待办事项优化会议和协同设计会议，测试人员的参与更是不可或缺。通常，测试人员凭借其独特的视角将能够通过识别需求中缺失的细节或非功能性需求来完善用户故事。测试人员尤其能够支持特定用户故事的验收标准的识别与定义。可以使用 BDD（行为驱动开发）和 ATTD（验收测试驱动开发）等技术以验收测试的形式定义验收标准，从而确定用户故事是否满足可测试性标准。因此，测试人员通过向业务代表提出关于用户故事的开放式"“假设”"问题、提出测试用户故事的方法以及确认验收标准来做出贡献。

特定实践 2.3 分析同行评审数据与正式评审尤其相关。正式评审需投入大量精力，为了确保这些投入兼具效率与效益，需要收集同行评审数据并传达给团队，以便学习和调整评审过程。如前所述，正式评审在 DevOps 项目的应用频率相对较低。虽然数据收集是正式评审的重要组成部分，但在非正式评审中却并不常见。在决定是否开展同行评审数据收集工作时，可通过以下两个问题进行判断：

- 如果收集这些数据，其使用主体是谁？
- 这些数据与我们的业务目标有何关系？

如果没有人会有效地使用这些数据，就不必浪费宝贵资源开展此项工作。然而，在 DevOps 中，通常使用工具来收集评审结果和数据，并将其作为信息显示在仪表板上，从而大大提高效率。此外，在迭代回顾会议中，当团队随后分析评审数据和发现以理解根本原因并识别改进机会时，这些数据就变得非常有意义了。

# 词汇表 DevOps 特定术语

| 术语 | 中文翻译 | 说明 |
|---|---|---|
| Blue-green deployment | 蓝绿部署 | 一种部署策略，维护两个相同的生产环境一个处于活跃状态，一个处于空闲状态，以便在空闲环境中测试软件的新版本，一旦验证通过，将所有流量从活跃环境切换到空闲环境。 |
| Canary testing | 金丝雀测试 | 一种测试方法，即将变更发布给一小部分用户进行验证，然后再逐步推广。 |
| Change fail percentage | 变更失败百分比 | 已部署变更中出现失败的比例。 |
| Change lead time | 变更前置时间 | 一次提交进入生产环境所需的时间。 |
| CI/CD pipeline | CI/CD 流水线 | 交付新软件版本所需的一系列步骤。 |
| Containerization | 容器化 | 一种轻量级的虚拟化形式，将应用程序及其依赖项打包成一个称为容器的可执行单元。 |
| Continuous delivery (CD) | 持续交付 (CD) | 一种自动化软件开发流程，其中代码变更会被自动构建、测试，并部署到生产环境。 |
| Continuous deployment (CD) | 持续部署 (CD) | 一种自动化软件发布流程，其中通过所有测试的代码变更会自动部署到生产环境，无需人工干预。 |
| Continuous monitoring | 持续监控 | 自动收集性能指标，以实时了解用户和系统行为。 |
| Cross-functional DevOps team | 跨职能 DevOps 团队 | 一群拥有不同但重叠的知识、技能和能力的人，为共同的目标而协作。 |
| Dark launch | 暗发布 | 一种部署策略，在不公开发布的情况下发布新功能或产品，允许团队在生产环境中测试和监控性能。 |

| 术语 | 中文翻译 | 说明 |
| --- | --- | --- |
| Deployment frequency | 部署频率 | 软件发布到生产环境的频率。 |
| DevOps | DevOps | 一种思维模式、文化和技术实践，支持有效开发和运营解决方案所需的集成、自动化和协作。 |
| Failed deployment recovery time | 失败部署恢复时间 | 失败部署后恢复正常运营所需的时间。 |
| Feature toggle | 功能开关 | 一种无需更改代码即可修改系统行为的机制。 |
| Feedback | 反馈 | 为改进产品、服务和流程而持续进行的信息交换。 |
| Flow | 流 | 工作产品在相关价值流中从一个步骤到下一个步骤的平滑、线性且快速的流动。 |
| Infrastructure as Code (IaC) | 基础设施即代码 (IaC) | 使用机器可读的配置文件来管理和配置 IT 基础设施，并对其应用软件开发实践。 |
| Observability | 可观测性 | 通过分析组件或系统的外部输出来理解其状态的能力，以实现监控、故障排除和优化。 |
| Pull Request (PR) | 拉取请求 (PR) | 一种机制，允许通知团队成员代码库中所做的变更，提议对这些变更进行评审、讨论，最终合并到主代码仓库中。 |
| Service Level Indicator (SLI) | 服务等级指标 (SLI) | 服务可靠性的可量化指标。 |
| Service Level Objective (SLO) | 服务等级目标 (SLO) | 服务等级指标的可靠性目标。 |

| 术语 | 中文翻译 | 说明 |
| --- | --- | --- |
| Site Reliability Engineering (SRE) | 站点可靠性工程 (SRE) | 一种将软件开发实践和原则应用于基础设施和运维的方法，同时使用自动化反馈循环来平衡服务可靠性与新功能开发的速度。 |
| Value stream | 价值流 | 向客户交付产品或服务所需的端到端活动序列，包括信息和物料的流动。 |
| Virtualization | 虚拟化 | 模拟其他软件运行所依赖的软件和/或硬件。 |

# 参考文献

[Baah] A. Baah (2017), Agile quality assurance, Bookbaby

[Beyer] B. Beyer, C. Jones, J. Petoff and N.R. Murphy (2016). Site Reliability Engineering: How Google Runs Production Systems, 1st. O'Reilly Media, Inc.

[Cohn] M. Cohn (2009), Succeeding with Agile: Software Development using Scrum, Addison-Wesley

[DORA16] A. Brown, N. Forsgren, J. Humble, N. Kersten and G. Kim (2016), 2016 State of DevOps Report, Puppet Labs, DORA (DevOps Research & Assessment]

[DORA24] DORA [2024], DORA's software delivery metrics: the four keys, https://dora.dev/guides/dora-metrics-four-keys/, DORA (DevOps Research & Assessment]

[Veenendaal12] E. van Veenendaal (2012), PRISMA: Product Risk Assessment for Agile Projects, in: Testing Experience, Issue 04/12, December 2012

[Veenendaal24] Erik van Veenendaal, Rex Black, and Dorothy Graham (2024), Foundations of Software Testing - ISTQB Certification (5th edition), Cengage

[Wake] B. Wake (2003), INVEST in Good Stories, and SMART Tasks, xp123.com/articles/invest-in-good-stories-and-smart-tasks