软件开发和管理

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，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

本文档为软件开发和管理提供了实用指南。它分为五个主要部分，内容包括：

- 有关实践的一般信息
- 软件开发和管理的流程和活动及其在服务价值链中的作用
- 软件开发和管理中涉及的组织和人员
- 支持软件开发和管理的信息和技术
- 适用于软件开发和管理的适用于合作伙伴和供应商的注意事项。

1.1ITIL®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

软件开发和管理实践的目的是确保应用程序在功能，可靠性，可维护性，合规性和可审核性方面满足内部和外部利益干系人需求的要求。

软件开发和管理实践专注于应用程序软件的开发和管理。但是，许多原理也适用于作为开发和管理应用程序的基础结构的一部分的软件。

软件工程对于基础设施和平台管理越来越重要，例如在基础设施即代码的应用程序中。此概念使用机器可读的定义文件来管理和配置IT基础设施和平台，而不是物理配置硬件组件。

软件开发和管理涵盖了整个生命周期应用程序。这可能从几个月到几十年不等，平均为10到15年。从经济学的角度来看，从历史上看，应用程序的总体拥有成本中平均有20%花费在开发上，而不是管理，而软件管理的花费中有20%与纠正性维护有关。

在现代世界中，应用程序的总拥有成本中的较大份额转移到开发。由于不断的变化成为应用程序生命周期不可或缺的一部分，因此所有维护活动都可以成为开发的一部分，通常不称为维护。

2.2 术语和概念

软件

一组指令，告诉计算机的物理组件（硬件）如何工作。软件在针对最终用户的应用程序中表现出来，而且在开发和操作应用程序所需的基础架构中也表现出来。软件和基础结构是服务组件，它们与其他服务组件或资源结合在一起形成产品和服务。

软件是业务的关键部分。它可以通过技术使能业务服务为客户提供价值。由于大多数现代服务不是软件辅助，而是软件使能，因此软件开发变得至关重要。

软件开发

根据更改的职能型和非职能型的要求，设计和根据职能型和非职能型的要求构造的应用程序以及运行的应用程序的修正和改进。

近年来，软件开发服务外包的趋势发生了逆转，许多组织将关键的和战略性的开发收回内部。这包括银行，保险和零售公司。

维护

应用程序作为开发的一部分而进行的修改，同时用于修正和改进：

- 纠正：纠正应用程序中已引起事件的缺陷
- 预防性的：在应用程序出现缺陷之前就防止它们出现
- 自适应的：使应用程序适应变化的基础架构
- 完善：增强应用程序的功能，可用性和性能或绩效（有时称为“附加维护”，“增强”或“ 开发”）。

随着变更的普及，现代服务不断变化。通常在整个生命周期中都对现代应用程序进行修改。这意味着所有用于构成维护的活动现在都是开发流程的一部分。

管理软件是一个广义术语，可能是指应用程序策略以及规划，运维，应用程序制品和应用程序退役的安全性。

实践的目的在于，应用程序应在功能，可靠性，可维护性，合规性和可审核性方面满足内部和外部利益干系人需求的要求。提及的所有术语均描述软件质量。

软件质量

价值软件的鉴证作为生产及其使用。常见分类（ISO / IEC 25010：2011）为：

- 生产质量：职能型适用性，性能或绩效效率，兼容性，可用性，可靠性，安全，可维护性和可移植性
- 使用的质量：效果，效率，满意度，不受风险和背景的覆盖。

快速修复通常比适当但耗时的更改更可取。变更在软件中的高使用率可能会导致一定数量的返工，这些返工需要在某个时间点进行，称为技术债务。

技术债务

通过选择解决方法而不是需要更长的时间的系统解决方案，累计完成了待办项的返工。如果是软件开发和管理，则是修复次标准（更改）软件所需的返工总量。

对于参与软件开发和管理的许多从业人员而言，主要的分水岭在于所选软件开发生命周期（SDLC）模型的敏捷程度。

SDLC model

执行软件开发生命周期的顺序。主要阶段包括：

- 确定要求
- 设计
- 码
- 测试
- 运行/使用应用程序。
- 瀑布式模型：开发生命周期的每个阶段都是按顺序执行的，因此整个应用程序都可以一次交付使用。
- 增量模型：在确定了整个应用程序的要求和优先级之后，将应用程序分为几部分（构建）进行开发。对于每个构建，依次执行开发生命周期的每个其他阶段。可以并行（部分）构建构建，并且应用程序可用部分交付。
- 迭代或演进模型：在部分确定了整个应用程序的要求和优先级之后，应用程序是在单独的版本中（例如在增量模型中）开发的，但是由于无法在一开始就完全满足要求，因此设计进行了编码，测试或使用构建可能会导致要求的改进，从而导致另一个构建中的应用程序的部分得以改进。

敏捷方法和Scrum方法方法是增量方法和迭代方法的组合，着眼于紧密的协作与应用程序的所有者，以获得快速反馈并实现小增量的快速开发，所有者可以从中获得价值。

Scrum方法

生产交付的一种迭代时间盒方法，被描述为``一个框架，人们可以在其中解决复杂的自适应问题，同时以富有创造性的方式交付最高可能的价值产品"（Ken Schwaber和Jeff Sutherland撰写的Scrum方法指南，于2017年11月更新）。

DevOps通过使用诸如持续集成/ 持续交付之类的技术来（部分）使部署流水线自动化来加速转换从编码到运行/使用的速度，从而进一步接近改进吞吐量。

许多敏捷方法都使用“ 完工定义”作为工具来同意在考虑完成生产或生产增量/ 待办项项目之前要满足的准则集。

完工定义

拟议的生产或服务的议定准则，反映了职能型和非职能型的要求。

2.3 范围

软件开发和管理的范围是根据活动和受活动影响的资源定义的。

软件开发和管理实践支持的活动包括：

- 应用程序开发
- 软件和软件制品管理
- 操作应用程序（与基础设施和平台管理紧密结合使用协作）。

软件开发和管理实践的范围中的资源是所使用的各种环境中的具体应用程序制品。主要的应用程序制品是规格，设计，源代码，目标代码和文档。

在职责方面，软件开发和管理实践位于：

- 应用程序的所有者，这些所有者确定对开发和/或管理的要求
- 基础架构管理，其中（a）提供软件开发和管理的环境，以及（b）管理生产环境，其中软件管理在其中运行应用程序
- 需要有关应用程序使用的支持的用户
- 管理软件组织，该组织：
  - 以前负责应用程序的管理，并且涉及应用程序的引入
  - 负责应用程序的未来管理，并涉及应用程序的撤销。

软件开发和管理实践中未包含许多活动和职责范围，但它们仍然密切相关。这些在表2.1中列出，并引用了可以找到它们的实践。重要的是要记住，ITIL实践将价值链活动到价值流结合在一起以交付价值。

表2.1其他实践指南中描述的相关活动

实现价值	实践指南
软件架构	架构管理
功用和功效要求	业务分析
应用程序的部署制品从一个环境到另一个	部署管理
提供反馈界面来自用户	服务台
应用组合管理	组合管理
使应用程序可用于使用和启用用户	发布管理
验证应用程序是否满足要求 测试潜在的可发布应用程序	服务验证和测试

协调整体设计	服务设计
应用领域	
应用程序监控	监控和事态管理

2.4 实践成功因素

实践成功因素（PSF）

实践的复杂职能型组件，是实践实现其目的所必需的。

PSF不仅仅是任务或实现价值；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。

软件开发和管理实践包含以下PSF：

- 同意改进和组织对软件的开发和管理的方法
- 确保软件在其生命周期中始终满足组织和质量准则的要求。

第一个PSF与选择合适的方法有关，第二个与应用方法有关。

2.4.1 同意改进和组织对软件的开发和管理的方法

如SLDC 模型（2.2）中所述，有多种开发和管理软件的方式。这些是瀑布式，增量式和迭代式（或渐进式）。敏捷方法和Scrum方法方法是增量方法和迭代方法的组合。

软件开发和管理实践的前提是必须有多种方法。这些反映了预期发生的各种情况。此战略决策还涉及其他实践，例如架构管理，业务分析，变更使能，发布管理，部署管理，信息安全管理，组合管理，风险管理，服务验证和测试以及战略管理。因此，将在（服务）价值流的背景中做出决定，在这些流中应用了这些实践。

针对软件开发和管理的实践成功因素涉及的战术决策是，根据组织对生产的要求，从这种预定义的方法集中选择每种软件生产的最佳方法。

这种战术选择取决于有多少信息可用于完成工作以及执行工作所需的资源。可以将要完成的工作细分为必须满足的要求和优先级。根据有关需求，优先级和所需资源的可用信息量，可以选择适当的方法。

一些例子：

- 当知道要求和优先级，知道如何开发软件以及需要哪些资源时，瀑布式方法可能是一种有效的选择。
- 如果知道需求和优先级，但是尚不知道如何开发软件以及需要哪些资源，则首先开发最重要的工作项的时间排序方法可能是更好的选择。
- 当需求和优先级在较高的水平上已知但难以最终确定时，线性迭代方法将允许产品负责人至体验并在多个迭代中完善生产。
- 并行实验可以为产品负责人提供原型，以在需求模棱两可甚至不明确时帮助制定需求。

不同的方法需要不同的资源组合。这些资源涵盖所有服务管理四维模型，并在本文档的第3至6节中介绍。

常见的资源组合：

- 小型，相对独立的，多个基于生产的职能型，基于开发/ 维护的团队，其中产品负责人管理要完成的优先级的工作。
- 基于平台的团队通过以下方式支持开发/ 维护团队：
  （自行）提供开发/ 维护和生产所需的基础结构。
- 跟踪所有生产制品的版本控制工具（例如代码和文档）。
- 自动化的流程，用于持续集成和交付/部署软件。

实现此PSF涉及多种实践。软件开发和管理实践的方法要求以组织性能或绩效信息和改进点机会的形式出现。这些已转换为改进倡议和计划（持续改进实践）。执行计划（组织变革管理实践），根据可以完成的工作的特征（软件开发和管理实践），可以使用各种方法和资源。

2.4.2 确保软件在整个生命周期中持续满足组织的要求和质量准则的要求

质量软件用于将软件描述为生产及其使用形式，通常以诸如以下术语表示：

- 生产质量：职能型适用性，性能或绩效效率，兼容性，可用性，可靠性，安全，可维护性和可移植性
- 使用的质量：效果，效率，满意度，不受风险和背景的覆盖。

在ISO / IEC 25010：2011 标准中，这些特征中的每一个都包含多达六个子特征，可帮助指定所需的软件属性。也可以将它们视为功用需求（例如职能型适用性和可用性）和功效需求（例如性能或绩效效率和可维护性）。

软件开发和管理相关的活动影响力或受这些特性影响。例如，可维护性很大程度上取决于源代码的结构和文档编制方式（在维护的文档和源代码中

本身）。在该软件的初始开发的开始处，将决定要花多少时间在可维护性上进行投资。这取决于需要多少维护，以及是否值得投资。在软件的初始开发中，可以满足这些要求。因此，初始开发会影响软件的可维护性。

最初的开发之后，维护受已实现的软件可维护性的影响。理解软件通常只占维护的一半，因此可维护性影响维护的速度和成本。

维护不仅受可维护性的影响，而且也影响它。除非努力进行维护，否则软件的质量倾向于降级（“软件熵”）。这项投资限制了技术债务（对修复不合标准的软件更改所需的返工）。

这些软件质量特性的许多要求是软件开发和管理的输入。它们由软件的所有者决定。但是，某些特性不是软件开发和管理的主要关注点。例如，效果由用户对软件的理解以及他们如何使用它以及软件启用的信息或设备来确定。

该实践Success 因素的最重要组件是：

- 了解源代码，各个模块之间的相互关系以及应用程序架构
- 了解要求以及使用应用程序的背景
- 确保完工定义包含非职能型（功效）要求
- 在编码之前创建测试
- 所有应用程序制品的有效版本控制
- 充分认识到编码的巨大困难并尊重人脑固有的局限性，从而完成编码任务[1]
- 遵守编码约定
- 同行评审
- 来自测试的快速反馈，例如通过使用自动化测试，以及快速采取补救措施性能或绩效。

软件开发和管理不是唯一涉及的实践。要实现此PSF，还需要确定软件的正确要求（业务分析），测试软件是否符合这些要求（服务验证和测试），在生产基础架构上运行软件（基础设施和平台管理），制定问题报告（问题管理）等。因此，在更广泛的背景中必须考虑指标。

2.5 关键指标

ITIL实践是管理产品和服务的手段或工具。与任何工具的性能或绩效一样，实践性能或绩效只能在该工具的背景中进行评估

[1]迪克斯特拉（EW The Humble Programmer）（1972）
www.cs.utexas.edu/~EWD/transcriptions/EWD03xx/EWD340.html [2019年10月29日访问]

application.但是，质量中的工具可能有所不同。这种差异定义了工具的潜力或能力

表2.2 实践成功因素的示例指标

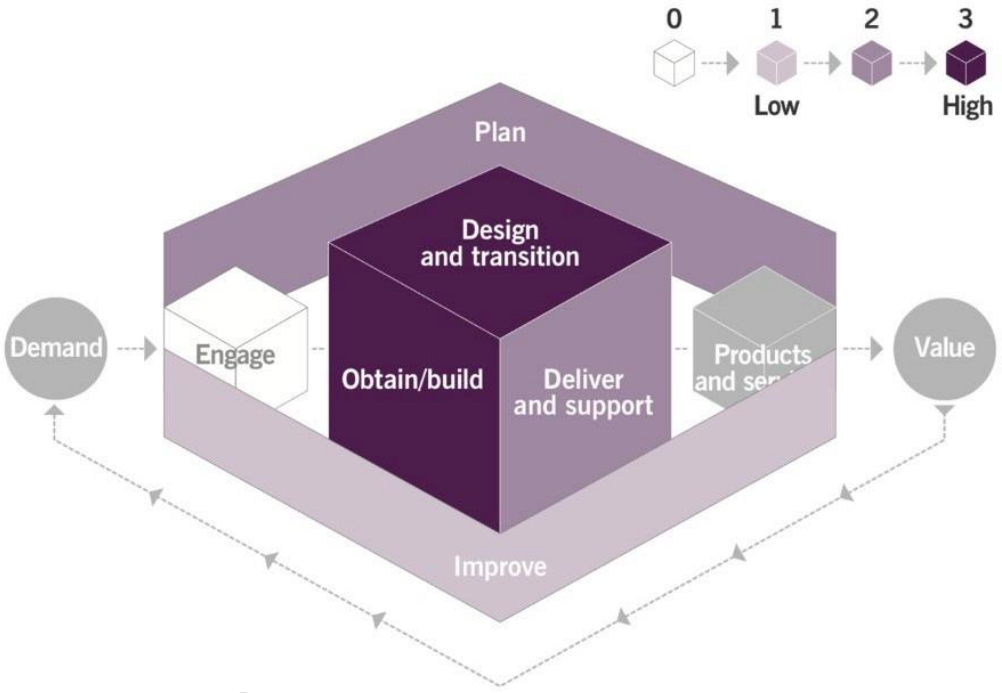
实践成功因素	指标指标
同意改进和组织对软件的开发和管理的方法	● 利益相关者的满意度和为软件开发和管理选择的方法 ● 遵循所选方法的开发团队所占百分比 ● 所选方法允许的利益相关者满意度和变更的费率 ● 适用于软件开发和管理实践的改进倡议吞吐量 ● 使合规性符合内部和外部要求，政策和法规。
确保软件在整个生命周期中持续满足组织的要求和质量准则的要求	● 利益干系人满意度及其交付价值的应用程序 ● 具有内部和外部要求和政策的应用合规性 ● 软件交付频率（用于新功能或更改功能） ● 软件交付的速度（从收到规范到提交给存储库并为部署发布的代码） ● 软件交付的可靠性（在发布之后检测到的部署缺陷） ● 成本（按职能点或其他大小单位；递减的成本） ● 技术债务（对修复次标准（更改为）软件的返工估计成本） ● 资源利用率（计算，网络，存储） ● 可用性软件（MTTR，MBTF） ● 安全违规和与审核等相关的费用

3 价值流和流程

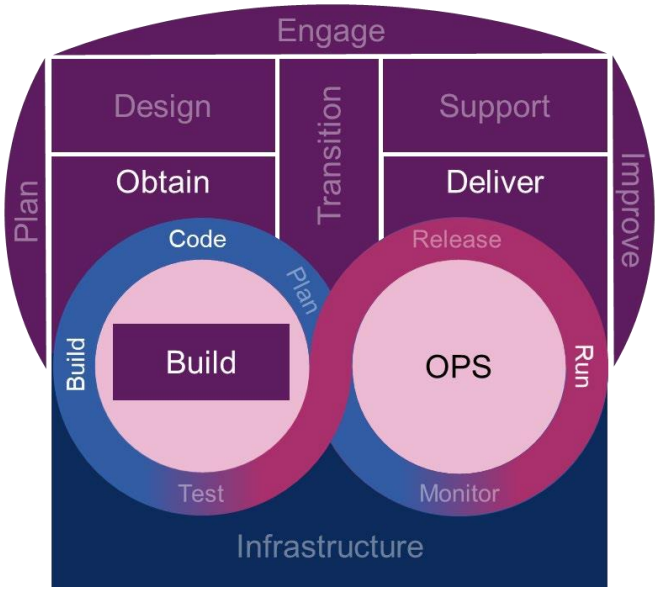
3.1 价值流的贡献

像任何其他ITIL 管理实践一样，软件开发和管理也贡献了多个价值流。请记住，没有价值流由单个实践组成。软件开发和管理与其他实践相结合，可以为消费者提供高质量服务。软件开发和管理贡献的主要价值链活动是：

- 获取或构建
- 交付和支持。



图片3.1热图，显示软件开发和管理参与的主要价值链活动



图片3.2代码，构建和运行对应于服务价值链活动获取或构建和交付和支持

3.2 流程

每个实践可能包含一个或多个流程和活动，它们对于实现该实践的目的可能是必需的。

流程

一组相互关联或交互的活动，可将输入转换为输出。流程定义动作的顺序及其依赖性。

有许多型号可以构成软件开发和管理实践的流程。这些跨越几十年，范围从瀑布（例如V-模型或Winston W. Royce的模型）到迭代式和增量式（如敏捷方法和螺旋方法）。

服务提供者组织通常结合了多种不同的方法，以实现最有效和可管理的可重复流程组。但是，对于本出版物而言，可以确定大多数实践方法中共有的一组活动。

表3.1 容器化的输入，活动和输出

输入值	活动	输出
• 商业案例，业务逻辑要求，服务模型，架构文档，用户故事，任务，现有待办项和项目计划中的缺陷 • 相关待办项\ 项目项目 • 现有环境配置 • 现有的开发工具集和版本跟踪方法 • 用户对应用程序的反馈 • 应用程序开发的技术标准。	• 生产规划和优先级 • 软件设计 • 新代码制作 • 代号评审 • 缺陷处理 • 技术债务管理 • 代码重构 • 研究与开发 • 定期会议和改进点活动 • 运维和维护软件自动化 • 管理开发环境 • 版本控制。	• 待办项/ 项目新任务，项目或变更交付计划 • 新软件或更改软件的技术要求。 • 应用程序代码，测试案例，自动化的单元测试 • 更新的代码，新的待办项项目 • 会议议程，会议记录，时间表，会议记录，决定和新规则，性能或绩效计划 • 流水线，监控和维护自动化软件 工具 • 更新了开发环境配置 • 为部署准备了新版本，保留了软件变更记录 • 新的/拟议的架构决策变更 • 有关该软件的价值的信息 • 发布关于正在开发的软件的注释：技术文档和用户文档（如何使用，安装，配置）；管理文档（如何管理） • 对技术标准的新的/提议的更改。

下表3.2给出了实现活动的两种不同的场景：在传统瀑布项目环境和以生产为中心的敏捷开发团队中。

软件开发和管理实践的表3.2 活动

实现价值	项目管理示例	生产管理示例
生产规划和优先级	请求者向相关的项目经理或开发团队负责人提交新的一批工作。	产品负责人收集新的外部需求，包括待办项的发现缺陷，并且可能与开发团队一起从待办项选择要执行的任务。 在下一个迭代中提供。
软件设计	开发人员或分析人员根据业务逻辑文档提供要在软件中实现的技术代码要求。	基于软件的详细信息和编码约定，可以直接在代码中构建技术规范和算法描述，而无需单独提供的文件。
新代码制作	软件开发人员将软件代码与单元测试一起提供，并确保单元测试通过完成。然后，他们提交代码进行测试并 验证并获得批准。	软件开发人员提供软件代码，并确保单元测试通过完成。然后，他们提交代码以进行自动或手动测试。
缺陷处理	软件开发人员分析缺陷任务以验证缺陷。他们向项目管理提出项目问题，以确保计划用于修复缺陷的资源。并修改 相应的软件代码。	软件开发人员分析缺陷任务以验证缺陷。然后，他们修改软件代码以修复缺陷。
技术债务缓解	软件开发人员分析技术债务任务并进行修改 相应的软件代码或架构。	
代号评审	软件开发人员通过查看或阅读代码来检查代码。最好至少一位审稿人不是 代码的作者。	
代码重构	重构是在不更改外部行为的情况下重构源代码，目的是改进， 可维护性，效率等。 软件开发人员分析代码重构任务，然后 相应地修改软件代码或架构。	
研究与开发	一位软件开发人员在待办项中分析了研究和开发任务，并提出了要添加到该任务中的新任务。 待办项。	

定期会议和改进点活动	软件开发人员，或开发团队负责人	开发团队执行定期迭代评估，用于

	参与项目的通信并与其他项目团队进行交互，以确保及时交换信息，并与风险和管理。	该示例可以确保在计划的下一个工作阶段有效完成任务，并着重指出障碍。
运维和维护软件自动化	在实施项目的过程中，软件开发人员提供了一个工具集来自动化软件的操作，例如诊断收集，弹性增强功能，监控和警报系统，例行公事维护等。 软件开发人员维护 并与软件操作一起发展工具集。	软件开发人员优化通过开发和发展操作工具集来操作软件所需的人力资源。
管理开发环境	开发团队负责人确保开发环境配置已提供给开发团队。	
版本控制	开发团队负责人实施版本控制规则和工具集，以确保团队成员之间一致的代码跟踪。	

4 组织和人员

4.1 角色，能力和责任

实践指南没有描述实践管理的角色，例如实践所有者，实践主角或实践教练。实践指南着重于每个实践的专门角色。每个角色的结构和命名都可能与组织和组织不同，因此ITIL中定义的任何角色都不应被视为强制性的，甚至不建议使用。请记住：角色不是职务。一个人可以担任多个角色，一个角色可以分配给多个人。

流程和活动的背景中描述了角色。每个角色都具有基于以下模型的能力概况：

能力代码	描述
L	领导。活动和与此能力相关的技能包括决策，委派，监督其他活动，激励和动机以及评估结果。
A	管理员。活动和与此能力相关的技能包括任务的分配和优先级，记录的保存，正在进行的报告以及基本 改进倡议。
C	协调员/沟通者。活动和与此能力相关的技能包括多方协作，利益相关者之间的沟通，和认知销售活动的运行。
M	方法和技术专家。活动和与此能力相关的技能包括设计和工作技术的实施，程序文档，有关流程的咨询，工作分析以及持续性 改进点。
T	技术专家。该能力侧重于技术（IT）专业知识和基于专业知识的任务。

4.1.1 软件开发和管理活动中涉及的角色

表4.1中列出了软件开发和管理实践活动中可能涉及的角色示例，以及相关的能力概况和特定技能。

表4.1 软件开发和管理活动涉及的角色

实现价值	负责角色（示例）	能力简介	具体技能
生产规划和优先级	项目经理产品负责人	CMLT	对业务目标的了解 熟练掌握项目管理惯例和其他相关的交付方式
软件设计	业务分析师或软件开发人员	Tm值	技术开发和特定于软件的分析工具
新代码制作	软件开发人员	Tm值	技术开发和特定于软件的分析工具
缺陷处理	软件开发人员	Tm值	技术开发和 特定于软件的分析工具
技术债务缓解	软件开发人员	Tm值	技术开发和特定于软件的分析工具
代号评审	软件开发人员	Tm值	技术开发和特定于软件的分析工具
代码重构	软件开发人员	Tm值	技术开发和 特定于软件的分析工具
研究与开发	软件开发人员	TMC	技术开发和特定于软件的分析工具
定期会议和改进点活动	软件开发团队负责人，产品负责人，软件开发人员，业务分析师，测试	t	技术开发和特定于软件的分析工具

工程师，Scrum主管

运维和维护软件自动化	软件开发人员	TMC	技术开发和特定于软件的分析工具 了解软件操作以及维护和操作活动手册的性质软件。
管理开发环境	软件开发团队负责人，软件开发人员，基础架构工程师	MTC	对受控环境的良好了解
版本控制	开发软件团队负责人，软件开发人员	MTC	熟悉版本软件的跟踪方法

4.1.2 软件开发人员/团队成员

软件开发和管理实践的密钥角色是开发人员或工程师。这是IT领域中最常见的知识工作者类型。

对于软件开发人员来说，算法思维是技能组的核心。核心软件开发人员知识和技能集的其他方面是：

- 编程语言，环境和技术
- 面向对象的软件设计
- 当代的系统架构，例如事态驱动的架构（EDA）
- 软件测试方法和方法
- 问题解决技术。

但是，现代软件开发人员需求具有广泛的技术和通信能力的强大命令：

- 与他们工作的技术栈相邻的技术知识
- 计划中的技术以及控制范围中的优先级活动，决策和风险缓解措施，无论是他们自己还是他们管理的团队
- 人际，双向和广播通信方面的技能，包括突出问题，传递思想和观念以及记录和提出解决方案的能力
- 不断学习的能力，以跟上技术发展的步伐。

软件开发人员还必须维护和利用一些个人特征：

- 愿意在问题和解决的旅程中快速前进到探索的新可能性，进行试验并承担合理的风险（例如，参见OODA Loop方法，Highvelocity IT）。
- 急于完成评审，例如错误修复，代码重构，旧版软件维护和技术债务任务。

- 系统地执行操作任务，并具有使部署，维护，备份，监控和其他与软件操作相关的日常琐事自动化的前景。
- 团队合作方法的敏捷性，软件开发人员在团队，产品或项目之间迁移，这在商业和大型服务提供者组织中尤其重要。

4.1.3 软件团队负责人

软件开发人员通常会经历从初级开发人员开始的职业发展过程，该初级开发人员处理风险较低的基本任务，再到围绕特定生产和基础技术拥有更多体验的高级开发人员。高级开发人员可以是开发团队其他成员寻求建议的关键知识载体。

团队负责人通向高级开发人员的一种职业道路，俗称“团队负责人”角色。团队负责人可以在某些组织环境中担任管理职务（尤其是在传统团队中），但首先是他们的开发人员团队的仆人负责人（有关更多信息，请参见High-speed IT and Create 服务型领导力，交付和支持）。

团队负责人的主要任务是在更广泛的业务或服务提供者背景中与其软件开发团队保持有效联系。除了列出给软件开发人员的技能和行为，团队领导者还应注意以下几点：

- 对软件正在解决的业务和问题的理解
- 了解他们的团队面临的障碍
- 对开发团队所属的服务提供者拥有的服务提供者背景和价值流的了解
- 对业务背景的了解，该软件将用于
- 了解其他学科，例如管理，生产，开发，市场营销等。
- 谈判技巧
- 激励和激励人事技术。

在软件开发和管理组织中，随着技术人员，开发管理人员等的逐步扩展，软件开发和管理还具有其他角色。请参阅下面的4.2组织结构。

4.1.4 Scrum主管/敏捷教练

Scrum主管是一个口语术语，源自Jeff Sutherland和Ken Schwaber撰写的《 Scrum方法指南》（请参阅https://www.scrumguides.org).该角色通常代表敏捷环境中的教练学科。Scrum主管可以确保所有相关人员都遵循敏捷的工作方式。从纯粹的咨询和通信任务到服务型领导力和管理任务的子集，都可以通过多种方式实现目的。在后一种情况下，团队负责人自然要承担Scrum主管的责任。

指导的重要性源于以下事实：敏捷方法要求对围绕服务和产品交付方式的习惯和传统进行根本性的改变。指导可帮助团队成员发展和维持新的行为和态度，并促进可视化提升所有团队活动的成果。例如，教练是建议的丰田Kata 持续改进实践的基础（请参阅3.4.3高速IT）。

阅读更多有关现代敏捷服务提供者环境要求的文化转变的信息，这些变化在Highvelocity IT（3。High-speed IT 文化）和Create 交付和支持（2.3开发团队文化）中进行。

4.1.5 产品负责人

这是软件开发和管理团队外部的角色，但对其成功至关重要。产品负责人通常在某些敏捷框架中定义为最终负责业务结果的人员。传统组织设置中存在相似之处，例如项目管理人员，甚至服务所有者。

软件开发和管理实践可以通过多种方式在服务提供者组织中实现，包括诸如Scrum方法之类的敏捷框架。因此，至关重要的是为开发团队定义一个单一的权限，以便其优先进行工作和外部升级。因此，此角色是在开发团队与服务提供者和服务消费者组织的其他部分之间实现正确接口的关键点。

4.2组织结构和团队

无论有多才多艺或富有成效，一个开发团队都很难满足所有需求的新服务或更新服务。由于团队的规模通常是根据直接管理的自然能力来定义的，即每个经理需要5至7名员工，因此开发团队的数量是主要的组织变量，这决定了容器化的人力投入。

尽管所有软件开发和管理团队都执行类似的活动，但是可以根据服务产品中软件产品的相对重要性以及整个组织机构设计的不同将它们分组在一起，例如：

- 软件的目的和功能
- 软件功能
- 运行软件的平台
- 使用的技术类型或品牌。

一个例子是生产团队：一个相对独立的，多学科和多任务的开发/ 管理团队，其存在时间与应用程序（“ 生产”）存在的时间一样长。这是为执行项目而被解散的临时基于项目的团队的替代方案。与基于项目的方案相比，工作执行的一致性更高。

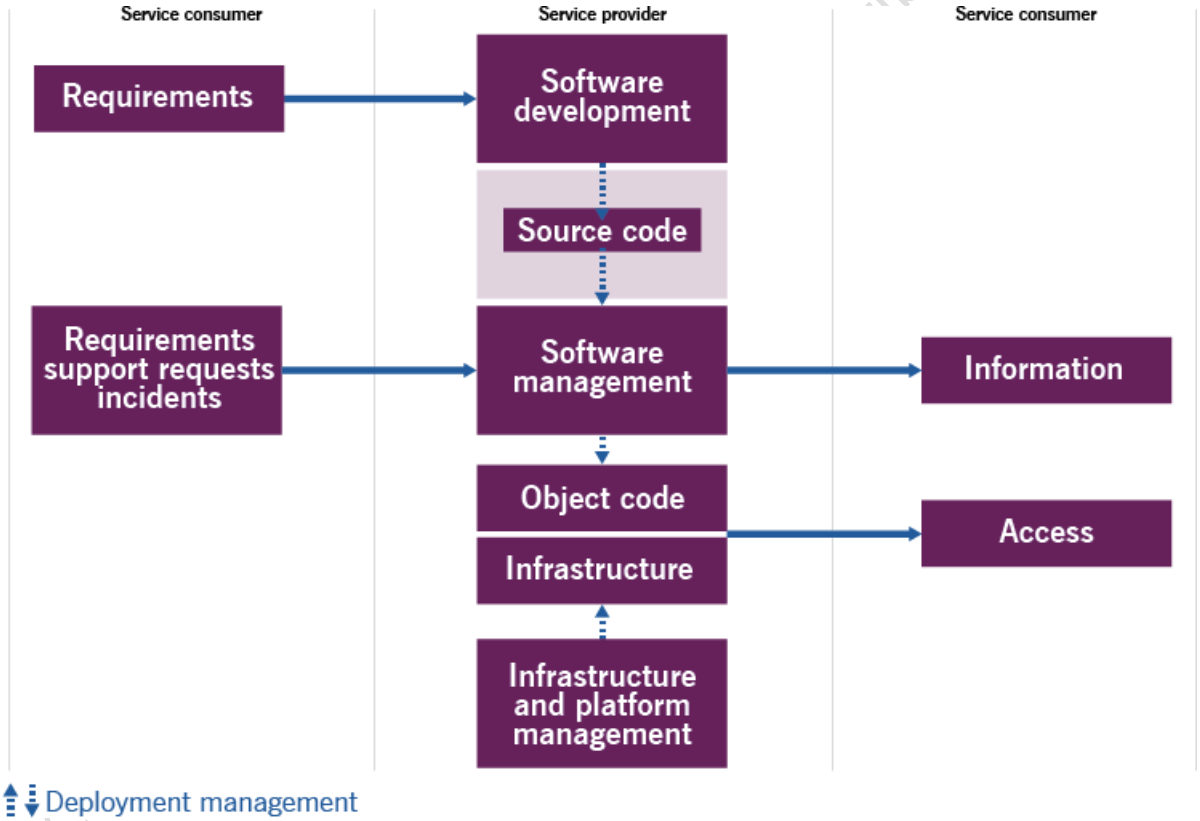
到20世纪末，内部和商业IT服务提供程序中的应用程序开发和管理部门主要充当独立单位，有时甚至充当不同实体的一部分。两者之间的断开与传统项目和操作竖井并无不同。但是，最近这两个部门合并在一起，承受了来自业务更大的压力，要求它们做出更快的响应和合作。DevOps 模型满足了这样的集成的要求，这表明开发人员有责任确保自动化且易于错误的部署软件及其正在进行的运维和维护。

5 信息和技术

5.1 信息交流

有效的信息交流对于软件开发和管理实践的成功至关重要。重要的是要注意，每个开发团队很少通过提供全面的软件生产来独立行动，而是为服务产品提供组成部分。这样的团队取决于其他团队（例如基础设施和平台管理）的输出，并产生其他开发团队（例如可重用代码库）可能使用的结果以及价值流（例如质量保证或部署）中的进一步步骤。

需求，支持请求和事件是软件开发和管理的主要输入，对运行的应用程序的访问和信息是主要的输出。



图片5.1 软件开发和管理输入和输出

重要的是信息交换清晰可靠。它应该被设计成既可以在团队成员之间（可能在地理上分布广泛），也可以在外部团队之间高效。

软件开发和管理实践中几乎每个实现价值的核心都是一个项目的概念。

根据特定的工具集或使用的方法，这些项目可以使用不同的名称和分类法。分层汇总分组（例如“ 史诗”或“主题”）可以在同时处理成千上万项的开发环境中使用。在某些情况下，不同性质的物品甚至可能具有不同的生命阶段和进度条件。大型开发和生产重点组织可以采用积压的层次结构来分发控制和流动项目。

但是，通常公认待办项的项目应以统一的精益方式进行处理，就像价值流方法建议的那样。开发团队应该计划的工作，寻找瓶颈，并将活动和信息交换的重点放在它提供的价值上。

必须提供约定数量的文档以支持：

- 持续的开发
- 运作
- 用。

该文档是设计临时文档的补充，该文档描述了要创建或修改的需求。

5.2 自动化和工具

与软件相关的活动受益于支持它们的信息管理工具。下表5.1建议了每种活动的特定工具。

表5.1 软件开发和管理活动的自动化解决方案。

实现价值	自动化手段	关键功能	实践上的影响
生产规划和优先级	任务和工作流跟踪工具集 项目管理工具集	工作安排和可视化	高
软件设计	开发工具集，开发环境	协作和自动化设计	高
新代码制作	开发工具集，开发环境	码管理	高
代号评审	开发工具集，开发环境	码管理	高

缺陷处理	开发工具集， 开发环境	码管理	高
技术债务缓解	工作流程和任务跟踪系统，已知错误数据库，开发工具集，开发环境	码管理	高
代码重构	开发工具集，开发环境	码管理	高
研究与开发	开发工具集， 开发环境	码管理	高
定期会议和改进点活动	开发工具集，开发环境	协作和调度；记录保持	介质
运维和维护软件自动化	远程管理工具，配置管理工具，自动化部署系统，开发工具集，开发环境	脚本化任务自动化和调度，基础架构流程	高
管理开发环境	配置管理工具集开发环境	基础架构流程	高
版本控制	开发工具集， 开发环境	代码库管理	高

6 合作伙伴和供应商

仅使用组织自己的资源提供的服务很少。许多组织通常依赖于第三方提供的服务（对于服务关系的模型，请参见ITIL®Foundation：ITIL 4版本2.4节）。

开发团队代表了高度专业化的能力，大多数组织都没有合适的方法来进行管理。当代内部服务提供者通常将软件开发和管理功能外包给第三方。当业务应用程序的开发和管理或其他软件之一或两者都从商业来源获得时，服务提供者管理应该考虑所有复杂性，这些复杂性伴随着从外部提供者定义好的输出看起来必须是什么样。

有几个重要的质量准则需要在相应的软件发包合同中进行协商，达成一致并进行定义：

- 服务的定义。显而易见，对于开发和管理合同的服务产品的确切定义是绝对必要的。实践中的活动提供了开发团队可以执行的全部工作，以确保软件质量。该列表不仅限于简单的新编码和错误修复，还包括整个软件生命周期。各方应有意识地协商范围内所需的活动，并考虑对两者最重要的定价和合作关系收益。
- 人员和组织。各方必须就组织结构，预期的知识转移机制，预期的流失率，审查原则和地域性达成一致
  开发团队的工作人员位置。
- Value streams and processes.各方必须就外部开发团队应如何与其他团队进行交互达成一致。这一点特别重要，因为服务提供者内保留了一些开发和管理功能，而外部团队则需要遵守两套控件：
  供应商组织内的一组控件，即它们的控件。
  雇主，以及服务提供者中的另一个雇主，即客户组织。可能发生冲突的例子很多：从双重保留记录和修饰待办项（边界浪费）的简单负担，到可用性规划的并发症。服务所有者（或敏捷环境中的生产所有者）的工作对于分析价值流中的外部开发团队参与度至关重要。
- Information and technology.各方必须为合同的目的明确定义记录的单个系统，如上所述，制造开发没有什么好处。
  团队花费时间在其“本机”和外部系统中复制缺陷记录。协议还明确涵盖了新员工的信息安全注意事项和引入。

可以说，系统的成本收益分析得出的结果与直观的预期完全不同，即外部专业开发团队总是比内部团队“更便宜”和“更有效”。仅仅由于软件开发的速度和新兴性质，短期内预期的其他好处并不能长期保证。变更在实践，体系结构和用户中的期望值可能需要外部供应商可能不愿意提供的功能。

7 重要提醒

实践指南的大部分内容都应作为组织在建立和培养自己的实践时可能考虑的领域的建议。实践指南是组织可能考虑的事情的目录，而不是答案的列表。使用ITIL 实践指南的内容时，组织应始终遵循ITIL 指导原则：

- 聚焦价值
- 从你所处的地方开始
- 基于反馈迭代推进
- 协作和提升可视化程度
- 通盘思考和工作
- 保持简单实用
- 优化和自动化。

有关指导原则及其应用程序的更多信息，请参见以下内容的第4.3节。

ITIL®Foundation：ITIL 4版出版物。

8 致谢

AXELOS Ltd非常感谢为本指南的开发做出贡献的每一个人。这些实践指南融合了ITIL社区前所未有的热情和反馈。AXELOS特别要感谢以下方面：

8.1 作家

康斯坦丁·纳里兹尼（Mark Konstantin Naryzhny）马克·斯莫利（Mark Smalley）

8.2 审稿人

奥列格·斯金尼克（Oleg Skrynnik）