SRE 实践手册

软件组织如何规模化实施站点可靠性工程

［德］弗拉迪斯拉夫·乌基斯(Vladyslav Ukis) /著

周靖/译

清華大學出版社

北 京

内容简介

本书基于作者在西门子医疗的 SRE 转型经历，为读者提供了一个全景式 SRE 落地路线，主题包括如何从基础设施、组织文化和流程等层面，在软件组织实际导入和实施站点可靠性工程过程。全书一共 15 章，实用性强，可操作性强，指导性强，适合想要落地 SRE 实践的软件工程师阅读和参考。

北京市版权局著作权合同登记号　图字：01-2023-0307

图书在版编目（CIP）数据

SRE 实践手册：软件组织如何规模化实施站点可靠性工程/（德）弗拉迪斯拉夫·乌基斯（Vladyslav Ukis）著；周靖译.—北京：清华大学出版社，2024.6

书名原文：Establishing SRE Foundations:A Step-by-Step Guide to Introducing Site Reliability Engineering in Software Delivery Organizations

ISBN 978-7-302-63308-2

I. ①S…　II. ①弗…　②周…　III. ①网站－开发－可靠性工程－手册　IV. ①TP393.092.1-62

中国国家版本馆 CIP 数据核字（2023）第 062740 号

责任编辑：文开琪
封面设计：李　坤
责任校对：方　婷
责任印制：
出版发行：清华大学出版社
网　　址：https://www.tup.com.cn，https://www.wqxuetang.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-83470000　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：
经　　销：全国新华书店
开　　本：178mm×230mm　　印　张：30　　字　数：637 千字
版　　次：2024 年 8 月第 1 版　　印　次：2024 年 8 第 1 次印刷
定　　价：128.00 元

产品编号：100550-01

推荐序

我和乌基斯相识于几年前在伦敦举办的某次 QCon 大会。他邀请我担任顾问，为西门子医疗内部的团队提供建议。在接下来一年多的时间里，我与他领导的西门子医疗 Teamplay 数字健康平台团队展开了密切的合作。在此期间，我们俩成了好朋友。

他的工作非常出色，帮助 Teamplay 团队以及整个西门子医疗取得了显著的进展。他和团队所获得的经验和教训来之不易，这些都充分体现在本书中。Teamplay 团队采用了一种先进的、以工程为主导的敏捷开发方法，该方法依托持续交付、DevOps 和 SRE，取得了显著的优势。他们的成功证明了这些理念的广泛适用性，超越了仅适用于谷歌等大型互联网公司的普遍认知。

我经常遇到一些组织否定大型互联网公司提出的重要理念，他们常说：“是的，但我们不是谷歌、亚马逊或奈飞呀！”大型互联网公司面临的问题并非总是独特的。实际上，规模的限制往往使它们更容易遇到常见问题很快发展成为瓶颈。这意味着对大公司来说，解决这些常见的问题就变得至关重要。大公司并不因为持续交付（CD）和 SRE 的流行就着急忙慌地落实这些理念。

作为这些理念的早期采用者和推动者，我观察到它们已经进入新的发展阶段，在广泛的软件开发组织中取得了显著的成效。汽车、航空航天、电信和医疗部门都有它们的身影。本书通过一个真实的复杂软件开发案例阐述了这一点，使人们不再认为“SRE 虽好，但我们又不是谷歌。”但除此以外，这样的例子还有更深远的意义。

在我看来，持续交付和 SRE 的持续发展有充分的理由和依据。两者都致力于通过度量和严谨、科学的推理方法解决当前的实际问题，不受软件开发的规模或问题性质的限制。我认为，持续交付之所以取得进步，就是因为我们采用了一种实验性方法来处理软件开发问题。SRE 同样在这方面发挥了重要的作用。

我写过如何将工程思维应用于软件开发的文章。我认为，深入学习和发展我们学科的这一方向至关重要。那么，为什么这本书特别强调这一点呢？重要的是要记住，SRE 中的 E 是指“工程”。对于软件工程，我最喜欢的是下面这个定义：

软件工程应用经验性和科学的方法，为软件中的实际问题找到高效和经济的解决方案。

Software engineering is the application of an empirical, scientific approach to finding efficient,economic solutions to practical problems in software.

SRE 的思想深深地扎根于这个定义的核心。此外，SRE 还采纳了实际工程中的一个基本原则：一开始就要假设自己可能会犯错。

是的，这个世界并不完美。软件的运行并不总是像我们希望的那样。大多数系统都有失败的时候。SRE 把这种考虑放在前面，迫使我们作为团队和组织去思考应该如何对待系统，比如，多长停机时间才算长？逼近这些极限时，又该怎么做？

本书主要完成了两项重要工作，而且执行得非常出色。

首先，本书核心阐述了 SRE 工程方法如何促进三个关键群体之间清晰和有效的合作：产品团队、产品开发团队以及产品运营团队。SRE 为他们提供了粘合剂，以一种合作的方式使其将精力集中在真正重要的事情上。与此同时，也为每个团队留下了足够的余地，允许他们在合适的范围内做出一些独立决策。本书不仅清晰定义了 SRE 的技术和原则，还以一种可持续的方式解释了它们背后的原则。这些定义并不晦涩难懂，是实际、可用而且容易理解的。

其次，本书详细介绍了如何实施变革才能将 SRE 的思考和技术应用于现实世界中复杂的开发组织。仅仅理论上的理解还不足以支撑写出如此深入和实用的内容。显然，书中提到的所有内容均来自行业内真正的践行者。

Teamplay 团队开发的软件并非简单的应用程序。他们开发的不是简单的网站或网店。他们构建的是真正重要到关系到人命的软件。它将医院内的先进医疗设备与云端信息系统整合，提供了新的洞察力，从而以新的方式来提供医疗方面的服务。他们采用这些领先的技术，并不是因为它们很时髦，而是因为它们比我们已知的其他任何方法都更有效。

本书帮助读者深入理解服务水平指标（SLI）、服务水平目标（SLO）和错误预算等关键概念、它们之间的联系以及如何有效地应用它们。本书详细探讨了组织如何对事故做出有效的响应以及如何在事故发生后进行良好的事后回顾以加强学习。本书阐述了何种组织结构是有效的。这本书的内容很全面。然而，它的价值远不止于此。

我是一个长期主义者，我的理念与本书高度一致。我花了好几年的时间研究 SRE，自认为对这个主题有比较深入的理解。但现在，我更加明白并打算将 SRE 理念应用到我的个人工作以及沟通和阐释方式上。为此，我要向弗拉迪表示感谢。

书中包含了许多深刻的见解，它们启发我思考如何将工程思维真正应用于软件开发中。我们都知道，工程实践离不开“权衡”或“取舍”。弗拉迪通过 SRE 实践案例对此进行了清晰的阐述并探讨了常见的权衡问题。

读到这句话“如果将 SLO 设为 100%，则意味着功能永远是次要的”的时候，我忍不住笑了。尽管我很早以前就理解这句话，但现在有更深刻的领悟，因为我有更精确的语言和模型来表达它。

我很高兴并且也很荣幸受邀为本书写序。我必须向大家推荐这本书。弗拉迪聪明，很有想法，他在 SRE 领域取得了卓越的成就。此外，他还是我的好朋友。当然，我之所以写这篇推荐序，并不只是因为他是我的好友。在这里，我要真心诚意地向大家推荐这本书，不带任何私心。

这本书深入浅出地探讨了 SRE 这个重要的主题，对推动该学科的发展具有重要的意义。希望大家能够像我一样喜欢这本书。

——大卫·法利

独立软件开发咨询师，Continuous Delivery 公司创始人兼 CEO

前言

本书基于一个真实的案例，展现了医疗行业一个软件交付组织的 SRE（Site Reliability Engineering，站点可靠性工程）转型之旅。该组织运行的云平台主要服务于医疗应用和服务。此平台部署在多个数据中心，全球各地的医院都在使用此平台上的应用。其中一些应用适用于危重病患，因而平台的可靠性相当重要。

但是，什么是可靠性？如何度量呢？如何营造一个环境，使开发团队有动力投入可靠性？这些问题是我几年来努力解决的。为了向应用程序和用户提供一个高可靠性的平台，我们组织可谓操碎了心。大规模的客户投诉升级极为常见。一方面，我们需要开发新的功能。另一方面，我们又要保障平台的可靠性。哪像工作优先级更高，人们看法不一。一方面，运营团队要煞费苦心，确保产品/服务能够正常运行。另一方面，开发团队又兴致勃勃，想要实现新的功能，很少关注现有功能在生产环境中的实际运行情况。项目管理计划因部署大量非预期的热补丁而受到严重影响。大量客户投诉升级，要求恢复服务或交付缺失的功能。大家都在发表高见，指出应该如何改善现状。然而，一旦再次出现故障，他们可能又会有发表新的意见。

我曾经连续几年参加 QCon 伦敦大会。通过这个会议，我了解了软件开发和运行的新趋势。SRE 是大会的主题之一。我虽然知道 SRE，但并没有真正开始了解它。在一次 QCon 大会上，整个分会场的主题是 SRE。我花了相当多的时间参加了相关的会议。在会议结束时，我清楚地意识到了 SRE 在业界的发展势头。

在大会结束后，我在回公司的路上翻看自己的笔记，意识到是时候尝试在组织内应用 SRE 来改善服务运行了。我没有见过其他结构化的运营方法。在没有采用 SRE 的情况下，我们自己的尝试并没有带来明显的改善。在大会上，许多公司在做报告的时候，都说它们成功应用了 SRE 服务运营方式——无论这些成功具体是指什么。开始的时候似乎很容易，只需要设定几个基本的指标（比如可用性和延迟），为每个服务定义可接受的目标，并在违反目标时发出警报。

回到工作岗位后，我开始思考如何在组织内推动 SRE。通过深入思考，我发现这需要整个组织的参与。我想到下面几个问题。

- 如何争取组织内部对 SRE 的支持？
- 如何让领导团队参与进来？
- 如何让运营团队参与进来？
- 如何让开发团队参与进来？这些团队的数量越来越多，很快就会达到 20 个甚至更多。如果是这样，如何在一个不断增长的组织中推动 SRE 并以一种能够随着团队数量增加而扩展的方式？
- SRE 发展到更高层次会怎样？
- 为什么它能发挥作用？
- 我怎样才能学到更多关于 SRE 的知识？
- 我怎样才能学到足够多的 SRE 知识，以便快速、轻松地向别人解释它？
- 是否有可以替代 SRE 的方法？
- 怎样才能与那些已经在其组织中引入 SRE 的人接触？
- 在组织中引入 SRE 的常见陷阱是什么，如何避免？

带着这些问题，我进行了深入的思考，最终成功地将 SRE 融入组织的开发团队和运营团队，并使其成为两个部门的核心学科，以可度量的方式极大地提高了我们运营全球化云平台的能力。

此外，组织与许多云端应用开发的团队保持联系。如何有效运行这些应用是这些团队共同面对的问题。我们现在将 SRE 选作首选运行方法。团队了解了之后，就导入 SRE 并使用我们提供的 SRE 基础设施。

在 SRE 转型期间，我们有机会拜访了柏林的 Delivery Hero 公司。他们的运营算得上是世界级水平。通过向他们学习，我们获得很大的启发。后来，看到我们自己的团队也逐渐接近世界级水平时，我们备受鼓舞。

回顾往事，我们学到很多经验。为了将 SRE 规模化导入从未做过运营的开发组织和一个从未让别人做过运营的运营组织，我们需要做许多事情。它要求开发团队长期深度参与并辅导团队，使其在运营能力方面越来越成熟。同时，它要求与运营团队长期接触并辅导他们成为 SRE 基础设施框架供应商，使开发团队也能参与运营。为了完成转型，需要在开发和运营两个方面，以独特的方式对技术、人员、文化和流程等领域发生的变革进行融合。

我们开始在 InfoQ 上发布系列文章（主题为数据驱动决策），分享我们在 SRE 方面的经验，后来又以电子杂志的形式发布。该系列文章中的 SRE 文章受到广泛的关注，有人找到我，邀请我写一本介绍 SRE 转型的书。至于剩下的事情，大家都知道了。

与 Addison-Wesley 合作对我来说是一种莫大的荣幸。在大学学习计算机科学时，我读了他们出版的很多专业书，以至于我在图书馆里隔得老远也能认出来。因此，当我有机会成为他们的作者时，必须是毫不犹豫地接受邀约。

在一个节奏非常快、经验并不总是特别宝贵的行业中，拥有一些可能值得在书中发表的知识，也是一种荣幸。当然，由于这个行业的节奏和对新事物的偏爱，我有点担心自己拥有的知识不够完整，没准儿很快就会过时。另外，我似乎是为数不多的且从未与谷歌合作却敢于写书来谈 SRE 的人。

不过，我的写作动机在广泛的阅读和丰富的专业实践中愈发强烈。我激励自己，这不仅是对软件工程社区的回馈，更是致敬过去十几年那些通过无数好书和讲座塑造我思想的作者和讲师。在这个数字干扰无处不在的世界，能够全身心投入一个需要高度集中注意力的项目，无疑是一种难得的特权。写书正是这样的过程。它教会我如何抵御数字干扰，让我迅速集中注意力，使我的专注力仿佛回到了互联网诞生之前。

我写这本书的目的是支持那些正准备导入 SRE（Site Reliability Engineering）理念的组织。这个旅程有极大的价值，但同样充满挑战，需要长期坚持，过程中难免遇到很多坑。导入 SRE 意味着在产品运营的文化、组织结构、责任分配、实践方法和技术应用上需要进行变革。对于用户和客户，产品运营至关重要，因为他们接触到的只有实际运行的产品。因此，优化生产流程直接关系到用户体验和客户满意度的提升。本书将探讨如何以可度量的方式改善用户体验和客户满意度、如何建立 SRE 基础设施来实现这一目标以及如何推动组织的 SRE 转型之旅。

本书分为三个核心部分。第 I 部分中，我要大致介绍 SRE 的概念、作用及其在软件运营领域中的地位。同时，我还要概述在刚接触 SRE 时组织所面临的挑战，解释如何从运营和 SRE 转型准备的角度评估组织的现状。

第 II 部分中，将着手启动并推进转型活动。确保 SRE 转型的成功，从一开始就需要获得组织的广泛认可。在这部分中，我将解释如何实现这种认可，如何在团队中启动转型活动，并确保组织能够建立警报系统、轮班制度和有效的事故响应流程。完成这些步骤，意味着组织已经为导入 SRE 做好了准备。

随后，继续探讨如何实施更高级的 SRE 实践，包括错误预算策略和基于错误预算的决策制定。完成这些实践后，便建成了一个成熟的 SRE 组织结构。在这部分结束时，组织不仅建立了基础和高级的 SRE 实践，还建立了长期可持续的组织结构。

第III部分中，将讨论如何度量 SRE 转型的成功以及如何保持 SRE 实践。在本书的最后一章，我将展望 SRE 转型在现有基础上的未来发展。

本书的结构化元素如下表所示。

元素	说明
要点	由书中的讨论所划出的重点，和别人闲聊 SRE 时要记住这些要点
SRE 误区	书中揭穿了行业中一些普遍存在的关于 SRE 的误区
SRE 参考	对一些 SRE 主题的简短解释，方便快速参考
实战经验	一些故事或见解，基于在 SRE 转型和实践过程中得到的经验和教训。描述了一个组织在特定情况下真正发生的事情

如果在阅读本书的过程中遇到任何问题，欢迎随时通过领英（LinkedIn）与我联系，我期待收到您的反馈和宝贵意见！

致谢

首先，我要感谢我的家人，是他们撑起了我的小宇宙，在情感、理智和精神上给予我坚定的支持。当然，我的妻子丽娜是这个小宇宙的中心。作为一名 UI/UX 设计师，她总是认真地听我谈论 SRE 这样的技术性话题，有时甚至还笑着提醒我：“我知道啦，你不用解释得这么详细！”正是因为她的热情、鼓励和耐心，才使得本书的写作成为可能。此外，还有我们俩的孩子，六岁的安妮卡和两岁的乔纳斯，我们很享受这种儿女承欢膝下的场景。乔纳斯已经习惯了翻阅家中那些封面或黑或白的 SRE 相关书籍。他很有可能已经从中学到一些关于可靠性的知识并能在他这个年龄段尝试应用。家里这种安宁使得本书的写作成为可能。此外，我的父母及兄弟姐妹、102 岁的外公、叔叔一家、姐夫一家、姻亲、妻子的叔叔一家以及远房亲戚和朋友们，都为本书的完成提供了宝贵的支持。

时间是一种宝贵的资源，我要特别感谢各位读者花时间阅读本书。我希望本书能够帮助大家重新思考软件运营，尤其是 SRE。我的目的是帮助大家顺利、迅速地实现 SRE 转型。请与我联系，我很想知道大家是如何实施 SRE 转型的。

西门子医疗是我个人职业生涯发展的重心。西门子医疗内部的 Teamplay 数字健康平台是我个人职业生涯的转折点。它为引入新的工作方式提供了一个必要的实验环境。在这里，新的工作方式得到尝试并被采纳，富有成效，甚至超过了最初引入 SRE 的团队。最后，整个公司都受益于 SRE。

特别感谢 Teamplay 的各位领导：前任主管托马斯·弗里斯博士和现任主管卡斯滕·斯皮斯。得知我在写书，他们非常支持我并以开放的态度对待我的写作。本书是对 Teamplay 团队实施 SRE 过程的全面复盘。

伦敦 QCon 大会为我个人的职业生涯留下了深刻的印记。事实上，Teamplay 团队在组织级别的重大技术变革都来源于那次伦敦 QCon 大会期间的交谈、对话、分论坛和回忆。那次大会是我深入理解持续交付和 SRE 转型的起点。

对我来说，一个重要的职业里程碑是我在伦敦 QCon 大会上认识了大卫·法利。他帮助我理解了持续交付对我个人和团队的价值、基本原理、战略和战术。他对持续交付的思考植根于科学的方法。所谓科学的方法，是指通过假设来回答问题，再通过验证来进行检验。在软件开发背景下应用科学的方法是持续交付取得成功的原因。有趣的是，当科学的

方法应用于软件运营场景时，SRE 取得了成功。为此，我要感谢大卫，不仅只是持续交付，还因为他把我介绍给了培生。

我要感谢本书执行编辑海茨·亨伯特，感谢她对我个人写作能力的信任。从我们建立联系到本书的出版，她的表现一直非常专业，我们的合作很愉快。此外，审稿人尼尔·墨菲为本书提供了很有价值的见解、发现了不少瑕疵并及时给出了改进意见。采纳这些意见之后，本书变得更完美。最后要感谢培生的团队，包括开发编辑马克·泰伯、文稿编辑奥黛丽·道尔、制作编辑朱丽叶·纳西尔及项目经理阿斯维尼·库马尔和其他许多人，这个高效的团队将我的初稿变成了一本高质量的书。在这里，我要向他们表达我诚挚的谢意！

至于 SRE，我要特别感谢 Teamplay 团队的运营工程师菲利普·冈迪施，感谢他的热情执着以及在 Teamplay 搭建的 SRE 基础设施。Teamplay 的 SRE 基础设施不仅非常可靠，使用体验也相当好。这一切离不开他的聪明才智。在搭建 SRE 基础设施的过程中，很多实习生为菲利普提供支持，在此，我也要向他们表示感谢。

当然，我还要感谢谷歌，因为它率先提出 SRE 的概念并将其变成一门新的计算机科学和软件工程学科。谷歌内部有些人说，在写第一本谷歌 SRE 书籍的时候，谷歌 SRE 部门甚至没有团队和其他团队采用相同的工作方式。也就是说，将不同的工作方式编入一套连贯一致的 SRE 原则和实践不亚于一项繁琐的任务。然而，为了推动 SRE 的发展，使其超越谷歌而惠及更多组织，这样做绝对有必要。因而在某种程度上，他们的努力传到我这里，于是就有了西门子医疗的 Teamplay 数字健康平台。剩下的事情大家都知道，我就不再赘述了。

通过与谷歌 SRE 先行者尼尔·墨菲以及 Equal Experts 的运营思想领袖史蒂夫·史密斯讨论 SRE，我对 SRE 的思考日渐成型。感谢两位的宝贵时间！

有趣的是，在我早期的个人成长和职业生涯中，我注意到有几个人特别重视个人工作流程的建立。比如我的祖父，他在一家化工厂担任过首席技术员。他花了很多时间向我解释这家工厂引进的新工艺，后者使工厂的运作随着时间的推移变得越来越高效。虽然我不明白底层的化学原理，但工艺改进的成果是显著和令人振奋的。

学生时代，在朋友推动下，我对计算机科学从星星之火发展为燎原之势。我们把计算器和个人电脑接到录音机和电视上，早期这些编程尝试点燃了我深入挖掘计算机学科的火花。

还有我的物理老师弗拉基米布·雅各布，他教全班同学公开讨论学习过程。学习过程的分享和改进是物理课的重点。这一点颇不寻常，但对学生的学习成果有非常积极的影响。它很早就让我明白一个道理：做对事情的过程和做对事情同等重要。

在我早期的职业生涯中，西门子医疗的卡尔海因茨·多恩让我认识到软件架构背景下一个严密的过程所具有的价值。

这里，我还要特别感谢斯特凡·雅布隆斯基教授，在他的指导下，我在德国埃尔朗根-纽伦堡大学大学完成了学士论文。我还要感谢吉拉德· 哈罗德，在他的指导下，我在

西门子医疗完成了我的硕士论文。这些重要的项目给我提供了独特的机会，使我在技术、人际和组织方面得到专业的成长。相关的论文写作让我认识到清晰的技术写作所具有的价值和影响。此外，非常感谢我在英国曼彻斯特大学的博士生导师刘公乔。他非常有耐心，在他的指导下，我的写作技巧得到了很大的提升。我记得当时经常在他的办公室开会，讨论我们的联合研究论文。我有些抓狂："怎样才能向一个计算机科学零基础的人解释这个问题呢？"在这样的会议上，这个问题出现的频次相当高。公乔老师始终坚持原则，直到零背景的人也能理解我们的学术论文。就这样，我的写作技巧和速度也随着时间的推移得到提高。

为了完成本书的写作，我自然需要在繁忙的职业和家庭生活中引入严格的写作时间表，在心理上需要准备好这样的工作日常并长期坚持。只有这样，才能真正按出版社提出的要求完成写作。

关于写作，埃德加·劳伦斯·多克托罗说过一句有趣的话："写作犹如夜间在迷雾中开车，眼前所见取决于雾灯，但即便如此，也能完成整个旅程。"我对此深有感触。令人惊讶的是，大脑中一个主题可以包含大量的信息，这些信息以一种高度浓缩的结构解压到几百页的篇幅内，最后以一种可供其他人学习的形式呈现出来。

在我开始攻读博士学位之前，许多人都说，做研究是一个独特的机会，可以专注于单一的主题，然而在未来的职业生涯中或许用不上这样的主题。我想，这个观点也不全对。写这本书当然可以让我专注于软件运营这个主题，就像我读研究生的时候专注于软件架构一样。但现在，我变得更熟练了，因为我已经掌握了具体的方式方法。

说到这里，还有一件趣事。我是在谷歌文档中写作的。在我写作的时候，我觉得可能会在写作时违反谷歌文档的 SLO。但考虑到谷歌对 SRE 过程的严格程度，我又感到安心，因为即便我违反谷歌文档的 SLO，谷歌文档的服务也会很快回到 SLO 的范围内。使用发明和实践 SRE 的公司以 SRE 方式来运营的字处理程序，旨在写一本介绍 SRE 的书，或许就是"吃自己的狗粮"的典型范例之一。

最后，这本书是对我女儿安妮卡开始写作的激励。她 2021 年开始上学，同年，我完成本书初稿。同样，这本书应该可以激励我的儿子乔纳斯，他同年开始学习字母，之后把它们组合成音节、单词、句子、段落、故事，最后是书。我很享受这本书的写作过程。写完这本书，我觉得意犹未尽认为将来还能再写一本。

前面提到的各位以及组织或团队在很大程度上影响了我。身处如此创新、专业且温暖有爱的环境，我感到非常幸运。千言万语化为一句话，感谢大家陪伴我走到现在！

简明目录

详细目录

第I部分　基础知识

第Ⅲ部分 度量和维持转型

第 I 部分

基础知识

以软件交付为中心的组织从三个方面来创建和维护软件产品：产品管理、产品开发和产品运营，其中，产品管理决定构建什么产品，产品开发决定如何构建产品，产品运营决定如何运营产品。

在软件开发的早期，产品管理、产品开发和产品运营往往各自为战。随着敏捷交付的兴起，三者开始以一种协同的方式开展工作。如此一来，软件交付组织在创建产品的时候更注重用户并以以渐进、迭代和快速的方式交付产品。

到目前为止，相比产品管理和开发之间的合作，开发和运营之间的合作还不够深入。事实上，在整个软件行业中，开发和运营依然各自为战，还有很大的改进空间。

一些新兴的软件交付理念，如 DevOps，正在推动产品开发和产品运营之间的深度合作。维基百科的词条指出，DevOps 是结合软件开发（Dev）和 IT 运营（Ops）的一整套工程实践，其目的是缩短系统的开发生命周期并提供具有高质量的持续软件交付。[1]DevOps 只是一个哲学理念，并没有具体规定产品开发和产品运营如何展开合作。也就是说，具体如何实现，需要由从业人员来决定。

站点可靠性工程（Site Reliability Engineering，SRE）则不然，它是实现 DevOps 理念的具象框架。事实上，正如谷歌在《Google SRE 工作手册》中说的那样：“SRE 实现了 DevOps。”[2]

DORA 的 State of DevOps 2021 报告指出：“SRE 和 DevOps 是相互补益的哲学。”[3] SRE 是谷歌开发的新兴软件工程实践，以规模化[4]、可靠的方式运行生产系统，有了 SRE，谷歌才得以实现可靠、低成本的运营。值得庆幸的是，谷歌最初出版的 SRE 丛书中对具体实践过程进行了精彩的阐述。[5]

受到谷歌的启发，很多公司开始采用 SRE 并从谷歌最初出版的 SRE 丛书中学习它们的具体实践。

实践证明，在一个习惯于以不同方式运营的组织中，要想顺利落地 SRE，需要在组织结构、技术和流程上进行重大的转变。具体怎么变，本书给出了详细的解释和说明。

本书第 I 部分旨在为 SRE 转型奠定基础。首先阐明产品运营为什么要做 SRE 转型。除了 SRE，运营还有其他选择吗？

有了这样的认识后，我们将讨论实现 SRE 的过程中会面临哪些挑战，具体从技术、人员、团队、文化、流程和组织结构展开讨论。

接下来，我们要定义一个大方向来应对这些挑战。如何让人们支持 SRE？需要哪些技术？如何推动文化变革？如何改革软件交付过程？

随后，我们要说明如何基于 SRE 活动和数据来改进业务。推动 SRE 是否有望减少故障的数量？如果是，这将如何降低因故障修复而产生的成本，并减少因故障导致的收入损失？SRE 有望减少故障修复过程中的沉没成本和故障所带来的收入减少。SRE 是否能够改善决策过程，特别是在决定何时投资于可靠性提升或新功能开发方面？这是否有助于优化资本分配，从而获得更高的投资回报？SRE 显然可以优化资本配置而获得更好的投资回报。

最后，我们要从一个新的视角审视 SRE 基础设施的建立——将其视为公司内部产品投资组合中的一个新产品，采用产品思维来进行开发。

注释

1 https://en.wikipedia.org/wiki/DevOps

2 Beyer, Betsy, Niall Richard Murphy, David K. Rensin, Stephen Thorne, and Kent Kawahara. 2018. *The Site Reliability Workbook: Practical Ways to Implement SRE*. Sebastopol, CA: O'Reilly Media.中译本《Google SRE 工作手册》

3 DORA. 2021. "State of DevOps 2021." . https://services.google.com/fh/files/misc/state-of-devops-2021.pdf

4 译注：at scale，指规模可大可小，却往往被误译为"大规模"。

5 Google. 2022. "Google SRE Books." https://sre.google/books

第 1 章

SRE 简介

本章将对 SRE 转型进行一个简要的描述。首先解释为什么要选择 SRE。接着探讨 SRE 转型会面临哪些挑战。最后对如何推动 SRE 转型进行展望。

1.1 为什么要选择 SRE

在考虑 SRE 转型时，需要认真思考一个关键的问题：“为什么我们要选择 SRE？”

事实上，现在的产品交付组织通常有其既定的运营方式，只不过这些方式可能因为没有产生预期的效果而催生了改进的需求。在这样的背景下，需要考虑所有可行的改进选项。除了 SRE，还可以考虑其他方法，比如 DevOps。DevOps 是一种产品开发与运维相结合的哲学理念，SRE 是 DevOps 理念的具体实践框架。DevOps 还有其他实现框架吗？业内是否还有其他方法来规模化且可靠地运行生产系统？我们先来审视一下现有的选项。

1.1.1 ITIL

除了 DevOps 和 SRE，还有 Axelos 提出的服务管理框架 ITIL——目前最新的版本是 ITIL 4。[1] ITIL 最初的全称为 Information Technology Infrastructure Library（信息技术基础设施库），但现在这个名称已经和它没有什么关系了。维基百科对 ITIL 的定义是“一套详细的 IT 服务管理实践，其目的是使 IT 服务与业务需求保持一致。”[2]

ITIL 描述的是 IT 过程、程序、任务和检查清单，用于证明合规性并衡量改进效果。它起源于 20 世纪 80 年代，当时，越来越多的 IT 组织开始采用多样化的实践，针对这些实践，英国中央计算机和电信局（The British Central Computer and Telecommunications Agency，CCTA）制定了一套标准。[3]

ITIL 4 定义了七大指导原则：[4]

1. 注重价值;
2. 始于足下;

3. 持续反馈稳步向前;
4. 合作并提高可视性;
5. 全盘思考和工作;
6. 保持简单实用;
7. 优化和自动化。

在 ITIL 4 中，服务管理的整体方法从以下四个方面展开：

1. 组织和人，员工队伍和组织文化、能力和胜任力;
2. 信息和技术，用于服务管理的信息、知识和技术;
3. 合作伙伴和供应商，与参与服务的设计、部署、交付、支持和持续改进的其他企业的关系;
4. 价值流和过程，组织单位的整合与协同。

ITIL 是一个用于设计企业 IT 职能的通用框架，在业界得到了广泛的应用。

1.1.2 COBIT

另一种 IT 治理方法是 COBIT。根据维基百科的解释，[5] COBIT 是 ISACA 创建的框架，ISACA 是专注于 IT 治理的国际专业协会。[6] COBIT 的全称是 Control Objectives for Information and Related Technologies（信息及相关技术的管理、控制与稽核）。[7]它是“一个信息技术管理和治理的框架，定义了一套信息技术管理的常规过程，每个过程都定义了输入和输出、关键过程活动、过程目标、绩效度量和基本的成熟度模型。”

COBIT 的一个核心原则是使业务目标与 IT 目标保持一致。这是基于 COBIT 的五大原则实现的：[8]

1. 满足利益相关者的需求;
2. 从端到端覆盖整个企业;
3. 应用单一的集成框架;
4. 启用一个整体的方法;
5. 将治理与管理分开。

ISACA 在 1996 年发布了 COBIT。最新版本 COBIT 2019 发布于 2018 年。它定义了六个治理系统原则：[9]

1. 为利益相关者提供价值;
2. 整体性的方法;
3. 动态治理系统;

4. 治理有别于管理;

5. 为企业需求量身定制;

6. 端到端的治理系统。

COBIT 和 ITIL 一样，是一个用于设计企业 IT 职能的总体治理框架。

1.1.3 建模

另一个可以在运营中采用的方法是建模。至于具体如何做到这一点，可以从软件安全学科中借鉴。在安全领域，人们通过建模来寻找威胁。威胁建模（threat modeling）是一种基于风险的安全系统设计方法。它基于对系统架构、实现和部署的分析来寻找安全威胁。一旦发现威胁，就定义并实现缓解措施。

与安全领域的威胁建模相似，可利用建模技术来寻找运营上的漏洞。可以分析系统架构、实现和部署，以找出会阻止系统在生产系统中良好运行的薄弱点。基于这些薄弱点，可以定义缓解措施。然后，可以在架构、设计、实现、部署、操作程序和组织过程等方面实现这些缓解措施。

以这种方式创建的模型需要定期更新，以适应新功能的开发、基础设施的变化以及从生产故障所获得的经验。总的来说，作为一种方法论，建模植根于从运营角度对系统架构、设计、实现、部署等进行定期分析。

虽然建模方法似乎可行，适用面也很广，但它并没有被业内广泛采用。它本身也不是一个已发表的且被广泛认可的运营方法。

1.1.4 DevOps

前面大致描述了 DevOps 是什么，现在让我们进行更深入的探讨。这有利于我们对 DevOps 和刚才描述的各种方法进行比较。

DevOps 定义了成功的五个支柱：[10]

1. 减少组织筒仓;

2. 接受失败是正常的;

3. 实施渐进式变革;

4. 充分利用工具和自动化;

5.一切皆可度量。

这是一种将开发和运营结合起来的通用哲学。自 2013 年以来，DevOps 哲学已经在软件行业的各个领域被广泛采用。具体的实现方式差别很大，其中之一便是 SRE 方法的应用。

企业的 DevOps 成熟度可以用 CALMS 框架来评估。CALMS 的全称是 Culture, Automation, Lean, Measurement, and Sharing（文化、自动化、精益、度量和分享）。这个框架由韩捷（Jezz Humble）创造，他写过几本关于 DevOps、持续交付以及其他主题的畅销书。

在文化方面，DevOps 要求责任共担，拆掉开发（Dev）和运营（Ops）之间的部门墙。DevOps 中的“自动化”是指围绕持续交付（Continuous Delivery，CD）的技术实践，在构建、基础设施调配、部署、测试和监视等方面尽可能地实现自动化。精益是指消除浪费和价值流优化的原则，它的具体实践包括在制品最小化（work-in-progress minimization）、批大小限制（batch size limitation）、减少交接复杂性（handoff complexity reduction）、队列长度管理（queue length management）和减少等待时间（wait time reduction）。

在度量方面，DevOps 组织需要收集关于其过程、构建、部署、失败、功能使用等方面的数据。这些数据被系统地应用于了解当前的能力并推动可度量的改进。最后，DevOps 中的共享是指开发和运营团队之间的共同目标、开放性和信息共享。

CALMS 有时也被用来消除 DevOps 与 ITIL 之间的差异。

1.1.5 关于 SRE

站点可靠性工程（Site Reliability Engineering，SRE）是做运营的最新方法。[11]根据维基百科，“SRE 是一门学科，包含了软件工程的各个方面，并将其应用于基础设施和运营问题。其主要目标是创建可扩展和高度可靠的软件系统。按照谷歌站点可靠性团队的创始人本杰明 • 特瑞诺尔 • 斯洛斯的说法，“当软件工程师得按要求设计运营团队时，SRE 便应运而生。”[12]

SRE 的原则由谷歌在《Google SRE 工作手册》[13]中提出，表 1.1 对它们进行了总结。

表 1.1 SRE 原则

#	SRE 原则	说明
1	运营是软件问题	SRE 使用软件工程方法来做运营
2	通过服务水平目标（SLO）进行管理	就服务的适当可用性目标达成一致
3	工作要尽量减少	如果一台机器可以执行所需的操作，就该由一台机器执行
4	工作自动化	确定在什么情况下，以什么方式对什么进行自动化
5	通过减少故障的成本来快速行动	减少常见故障的平均修复时间（MTTR）来加快产品开发的速度
6	与开发人员共享所有权	开发人员和 SRE 对整个栈有一个总体视图：前端、后端、库、存储等
7	使用相同的工具，不分职能或职称	管理服务的团队应该使用相同的工具，无论他们在组织中的角色如何

在实践 SRE 的过程中，还有另外三个原则：

1. SRE 需要有后果的 SLO;
2. SRE 必须有时间使明天比今天更好;
3. SRE 团队有能力调节他们的工作负荷。

因此，SRE 的原则是相当具象的，通常会直接规定需要做什么来实现可靠的运营。在最小化劳作、自动化和与开发人员责任共担方面，SRE 的原则与软件工程非常接近。

自 2014 年以来，SRE 在业界越来越受欢迎。它被认为是运营云原生系统[14]的首选。在全球范围内运行的云原生系统不断增加，这可能是 SRE 越来越受欢迎的原因之一。一般来说，SRE 可用于运营所有类型的系统，而不只限于云原生系统。

1.1.6 比较不同的方法

从 2014 年起，DevOps、SRE、ITIL 和 COBIT 的搜索热度逐渐上升。搜索趋势显示，DevOps 和 SRE 是大家更为关注的运营方法，ITIL 和 COBIT 的搜索相对较少。

此外，业内采用的运营方法也有显著的差异。ITIL 和 COBIT 提供用于设计企业 IT 职能的治理框架。建模方法则是基于对系统工件的分析来得到优秀的运营实践。SRE 扎根于软件工程，专门从这个角度来处理运营问题。DevOps 作为一种哲学理念，用于指导运营实践。

这几个方法各有千秋，可能互不排斥。然而，它们实际满足的是不同的需求。这意味着公司可能需要考虑从这几个方法中按需选用。例如，在处理客诉时，可能需要制定符合监管要求的 ITIL 程序。同时，为了确保开发人员和运营工程师能够有效地参与运营，可能需要采用 SRE 方法。作为 SRE 活动的一部分，在系统构建的初期阶段，建模方法中某些方面可能同样有用。

这些方法各不相同，所以直接对它们进行比较就显得比较困难。然而，从设想的 SRE 转型角度来看，这种比较又是必要的。为了推动 SRE 转型，我们需要说服组织内所有相关人员，使其相信 SRE 对整个组织是正确的选择。这里的“相关人员”是指产品交付所涉及的每个人。为了说服这么多不同的受众，我们需要明确并阐述选择 SRE 的第一性原理，而不能只是因为谷歌在做 SRE 就据此来说服人们转向 SRE。在这种情况下，我们需要比较所有可用的运营方法，然后找到 SRE 的切入点。

以谷歌为例来推广 SRE 可能适得其反。当组织出于本能抵制其他组织创造的方法时，就会表现出“非我所创”综合症。[15]根据维基百科的解释，研究表明，组织对来自外部的理念有强烈的偏见。[16]在 SRE 转型过程中，为了克服这种偏见，我们需要找出选择 SRE 的明确且根本性的原因。

下文尝试对各种运营方法进行比较，以便更好地了解它们在总体运营活动中的定位。我们最开始使用的比较标准如下：

- 是否代表了用于设计企业 IT 职能的治理框架；
- 是否明确支持 IT 监管合规；
- 是否植根于 IT。

基于以上三个比较标准，我们对所有方法的考察结果如表 1.2 所示。

表 1.2　运营方法论的第一次比较

方法论	是不是用于设计企业 IT 职能的一种治理框架	是否支持 IT 监管合规	是否植根于 IT
ITIL	是	是	是
COBIT	是	是	是
建模	否	否	否
DevOps	否	否	否
SRE	否	否	否

ITIL 是用于设计企业 IT 职能的一种治理框架。它支持 IT 监管合规，并植根于 IT。COBIT 同样如此。

建模则恰恰相反。它不是设计企业 IT 职能的 IT 框架，不支持 IT 监管合规，也并没有植根于 IT。DevOps 和 SRE 同样如此，前者是一种哲学，而后者只是实现了 DevOps 哲学。从这个比较中可以推断出，ITIL 和 COBIT 能很好地服务于首席信息官（CIO）。CIO 通常负责管理企业的总体 IT 职能。然而，其他直接有助于产品交付的企业职能（例如产品管理和产品开发）并不是 ITIL 和 COBIT 的重点。因此，ITIL 和 COBIT 不能很好地服务于首席技术官（CTO）和首席产品官（CPO）。

但是，它们应该很好地服务于 CTO 和 CPO 吗？我们的重点毕竟是运营。谁需要参与到运营中？在产品交付组织中，为了在生产中进行产品规模化的、可靠的运营，需要哪些人的参与？是产品运营吗？是产品开发吗？是产品管理吗？还是所有人都要参与？如何参与？以何种方式？如果只是基于 ITIL 和 COBIT，那么产品管理和产品开发需要做什么来促进可靠的、规模化的产品运营呢？这些还不够清楚。

考虑到这些问题，我们可以建立下一组标准，根据各种运营方法论在产品交付组织中对哪些人有吸引力来进一步比较它们。因此，可以选择以下 4 个比较标准：

- 一种方法论是否对 CIO 和运营工程师有吸引力；
- 一种方法论是否对 CTO 和软件开发人员有吸引力；
- 一种方法论是否对 CPO 和产品负责人有吸引力；
- 一种方法论是否作为软件产品交付的核心学科而植根于软件工程。

表 1.3 根据这些标准对运营方法论进行了比较。

表 1.3　运营方法论的第二次比较

方法论	对 CIO 和运营工程师有吸引力	对 CTO 和软件开发人员有吸引力	对 CPO 和产品负责人有吸引力	植根于软件工程
ITIL	是	否	否	否
COBIT	是	否	否	否
建模	否	是	否	否
DevOps	是	是	是	否
SRE	是	是	是	是

ITIL 框架对 CIO 和运营工程师有吸引力。植根于 IT 的它自带 IT 背景，因此对 CTO、软件开发人员、CPO 和产品负责人没有什么吸引力。它也不植根于软件工程。COBIT 框架与 ITIL 表现出相同的特点。

至于建模，ITIL 对 CIO 和运营工程师没有多大的吸引力，因为它只涉及 IT 领域的一小部分。建模对 CTO 和软件工程师有吸引力，因为它是一种借鉴于安全领域并应用于运营的分析方法。CPO 和产品负责人不会被建模吸引，因为它分析的是技术工件，例如架构、实现和部署，产品人员通常并不具备这方面的专业知识。许多时候，只有具有技术背景的人才能理解这些工件。最后，建模不是植根于软件工程，而是植根于产品安全。它代表产品安全领域的“安全威胁建模”。

所有涉及产品交付的人员都对 DevOps 哲学感兴趣：手下管理着运营工程师的 CIO、手下管理着软件开发人员的 CTO 以及手下管理着产品负责人的 CPO。手下管理着产品负责人的 CPO 认为，DevOps 能使软件发布更快。每个人都希望更快地获得新功能。DevOps 支持更快的功能交付。根据定义，对于手下管理着软件开发人员的 CTO 和手下管理着运营工程师的 CIO 来说，DevOps 是一套结合开发与运营的实践。这对两个群体很有吸引力，因为它试图弥合开发和运营之间的鸿沟，而这种鸿沟在业界是非常典型的。处于哲学高度的 DevOps 并没有植根于软件工程。作为一套结合了开发与运营的实践，它并没有明显包含产品管理，而产品管理却是软件工程的一个重要组成部分。

最后，当我们研究 SRE 时，会发现它对产品交付中涉及的所有人员都有吸引力。CIO 和运营工程师可以通过 SRE 来确保软件开发人员适当参与产品的运营。此外，有了 SRE，运营问题将在最初的产品架构和设计阶段得到解决。此外，运营问题甚至会上升到产品负责人都要关注的程度。对运营的考虑会影响到产品定义，而且或许最重要的是，会影响到开发团队的能力分配。

CTO 和软件工程师会对 SRE 感兴趣，因为它属于软件工程，主旨是软件工程师如何处理运营问题。事实上，SRE 是“软件工程师按要求设计一个运营团队时所发生的情况”。它涉及自动化、开发、度量、经验性证据、迭代以及根据生产中的度量结果分配工程时间。

CPO 和产品负责人会对 SRE 感兴趣，因为有了 SRE，他们能坐在指挥的位置，根据生产中的真实数据——而非技术人员的念叨——来决定工程能力分配。众所周知，工程能力分配（engineering capacity allocation）容易引起争议。在产品交付组织中，工程能力永远不够——与产品是否成功无关。在产品交付组织中，对于工程时间应该用在哪里，每个人都有自己独特的看法。同样，如果进一步问具体的原因，每个人都能说得头头是道。

运营工程师会收到很多来自客户的投诉。因此，从他们的角度看，大量由客户发起的售后支持工单必须首先由工程师来清零，因为它关系到客户的留存。客户一旦不满意，最终会掀桌子，停止使用，不再买账。

软件开发人员在产品的许多方面都有大量技术债务。因此，从他们的角度来看，在开发新功能之前，必须先偿还这些债务。因为生产中的系统可能在某些方面已经在“苟延残喘”，而且系统某些部分的维护可能需要付出非常多的努力。在技术上有基础缺陷的系统上增加功能，可能会成为压跨骆驼的最后一根稻草，造成客户完全无法使用该系统，从而进一步增加维护工作，技术债务会因此而暴涨，造成更高的利息，其表现是以后需要付出更多的努力来偿还债务。如此一来，明智的做法便是在向系统添加新功能之前清除技术债务。产品负责人与客户、用户、合作伙伴、利益相关者和公司管理层进行大量对话。每一方，包括产品负责人自己，都有很多关于新功能的想法。高级客户可能尤其苛刻。另外，高级客户的数量可能正在增长，但他们的要求可能需要更多的时间来实现——至少技术人员是这么认为的。一些来自客户的功能要求与公司管理层想在产品中增加的功能是冲突的。

合作伙伴也有不同的看法。他们中的一些人也许会和高级客户一样苛刻。在某种意义上，他们可能是对的。一些合作伙伴带来的收入比一些高级客户还要多。由于更接近产品，产品负责人自己也会产生许多想法，他们希望这些想法得以实现。这些想法有帮助但不一定可以反映客户、合作伙伴和管理层的意见。

总而言之，产品需要开发更多的功能。否则，高级客户可能会停止向同行推荐，甚至可能在某个时候停止付费。高级客户数量的增长可能会放缓。如果管理层没有得到他们认为对公司发展至关重要的功能，可能会缩减产品交付组织的预算。这可能导致合作伙伴开始寻找其他合作伙伴，从而导致公司来自合作伙伴业务的收入大幅减少。

在这种情况下，产品负责人应该听取谁的意见？自己的判断？还是客户、管理层、合作伙伴或技术团队的建议？如何解决这个难题？如何打破僵局？这恰好是 SRE 的优势所在。SRE 要求产品交付组织根据真实的生产数据进行调整。最终，根据这些数据来确定何时投资于可靠性，何时投资于功能。如果应用得当，SRE 将成为产品负责人的强大后盾。

他们可以忽略噪声污染，不会再出现“会哭的孩子有奶吃”的情况，甚至可以确保任何时间段都能在可靠性与功能的投资之间取得合适的平衡。

基于这一理念，可以用以下标准来比较前述的各种运营方法论：

- 是否适用于整个产品交付组织，包括产品管理、产品开发和产品运营等所有层面？
- 是否为运营工程师提供明确的指导，即规定他们的具体职责？
- 是否为软件开发人员提供明确的指导，即规定他们的具体职责？
- 是否为产品负责人提供明确的指导，即规定他们的具体职责？

根据这些标准对各种运营方法论进行比较，如表 1.4 所示。

表 1.4　运营方法论的第三次比较

方法论	与产品交付组织相一致	对运营工程师具象	对软件开发人员具象	对产品负责人具象
ITIL	否	是	否	否
COBIT	否	是	否	否
建模	否	否	是	否
DevOps	是	否	否	否
SRE	是	是	是	是

作为企业 IT 职能的管理框架，ITIL 不能与产品交付组织完全保持一致。对运营工程师而言，ITIL 是具体的，因为它明确指出了他们应采取的措施以确保服务的无故障运行。然而，对于软件开发人员和产品负责人，ITIL 就不是那么具体了。同样，作为企业 IT 职能的管理框架，COBIT 在给定的比较标准下与 ITIL 不相上下。

建模技术本身与产品交付组织并不完全一致。然而，对于软件开发人员，建模是具体的，因为它详细规定了分析运营风险所需要的技术工件和流程，如架构、软件设计和部署。

DevOps 虽然旨在促进产品交付组织中的协作，但并没有为运营工程师、软件开发人员和产品负责人提供具体的指导。

同样，SRE 能够协同整个产品交付组织并以具体的方式为所有相关方提供协作的框架。SRE 以一种具体的方式来实现 DevOps，要求运营工程师、软件开发人员和产品负责人在服务目标上保持一致。服务目标的定义需要能够体现用户对服务的满意程度。如果目标得到满足，用户的满意度就越高。如果没有达到目标，用户就会不满意。这些目标可以作为衡量用户满意度的代理指标。

实践 SRE 的一个核心原则是为目标设定具体的后果，即所谓的有后果的 SLO（SLOs with consequences）。基于此，产品交付组织还需要就未达成目标的后果达成共识。运营工

程师、软件开发人员和产品负责人需要事先商定，如果之前商定的目标没有被相应的服务实现，他们应该怎么做？

如果没有达到商定的目标，运营工程师会看到对用户的严重影响而导致肾上腺素水平飙升。他们想要尽快解决问题，恢复服务。软件开发人员立即能看清楚潜在的用户影响，明白自己的行为对用户造成了严重的影响而获得立即开始修复问题的动力。最后，如果商定的目标没有达到，产品负责人就要准备好接听来自客户的声讨。在这种情况下，他们会积极调配工程时间来恢复正常的服务，因为他们明白事故的重要性。不需要像以前那样，许多人花费大量无谓的时间来重新确定团队待办事项的优先顺序。各方达成共识：只有服务得到修复并达到之前定义的目标，才能继续处理待办事项。

1.2 使用 SRE 进行协同

因此，SRE 从三个方面对产品交付组织的运营进行协同，如表 1.5 所示。

表 1.5 SRE 对产品交付组织的协同

1. 共同定义服务目标	2. 共同定义未达到所定义的服务目标时的后果	3. 在未达到所定义的服务目标时，共同承担后果

首先，SRE 要求共同定义服务目标。其次，它要求明确当服务目标未被满足时的应对措施，以及各方应采取的具体行动。最后，它要求各方共同承担预先定义的后果。

换言之，SRE 迫使整个产品交付组织共同参与，并就各方对产品的可靠性期望以及他们愿意为此付出的代价达成一致。

这是 SRE 的真正强大之处。这种意义上的产品交付组织的协同在其他运营方法中找不到。这正是实现 DevOps 所需要的协同，以确保正确的服务运营，最终带来积极的客户体验。每一方——产品运营、产品开发和产品管理——都必须各尽其职。每个人都需要参与运营活动。对运营工程师而言，这是常规操作；对软件开发人员来说，这是一个新的领域；而对产品负责人来说，可能会感到意外。

SRE 的价值在于，它以一种具象的方式将三方聚在一起，追求持续的产品运营，进而提升用户的满意度。自从谷歌推出 SRE 以来，其服务受到了人们的广泛欢迎。根据 Statcounter 的数据，[17]在 2009 年至 2020 年期间，谷歌占据了超过 92%的搜索市场份额。虽然有许多因素促成了这种市场统治地位，但在确保用户搜索体验方面，谷歌 SRE 发挥了重要作用。

为了达到并保持 DevOps 的速度，也需要通过 SRE 来实现产品交付组织对运营的协同。如果产品交付组织没有就各种运营问题达成一致，是很难赢得速度的。如果仅由运营工程师

负责生产运营，他们可能难以在客户报告问题之前发现生产中的问题，因为警报参数的设置与产品开发和产品管理的目标不一致。运营工程师需要对常规的 IT 资源设置警报，例如内存消耗、CPU 占用率、队列填充水平、存储消耗等。虽然这些警报有时很有帮助，但它们不一定能反映出用户在使用系统时的满意度。因此，可能会出现用户虽然不满意但也不至于触发警报的情况，或者用户满意却触发了警报的情况。这是因为警报定义程序并没有以用户为中心，不包括所有必要的利益相关者，如运营工程师、软件开发人员和产品负责人。

一旦运营工程师遇到无法解决的问题，就需要软件开发人员介入。这是一个持续的斗争。为什么呢？因为软件开发人员的工作是由产品负责人排好优先级的待办事项。

一旦运营工程师发现自己无法解决的问题，就需要软件开发人员介入。这是一个持续的挑战，因为软件开发人员的工作是由产品负责人根据优先级安排的。他们专注于处理这些待办事项，并希望工作过程中不被打断。当运营工程师要求他们解决生产中的问题时，开发人员可能会表现出不悦。首先，开发人员通常专注于开发环境，而非生产环境。其次，解决生产问题可能导致对产品负责人承诺的待办事项延期。第三，这可能会打破对产品负责人的承诺。然而，是否应将解决生产问题优先于待办事项，这一点并不明确。更重要的是，我们需要决定是立即解决生产问题以满足当前用户的需求，还是继续实现新功能以兑现销售和市场的承诺。

为了打破僵局，产品负责人需要提供意见来决定优先级。为了做出这个决定，产品负责人需要首先了解生产问题的具体情况：有多少数据中心受到影响？有多少用户受到影响？这个问题是否阻碍了产品中经常使用的工作流？是否有任何临时方案？对收入、成本、商誉、客户支持有什么影响？提供修复的紧迫性如何？实施修复需要多少努力？部署修复方案又需要多少努力？监测修复方案的部署呢？为了决定正确的优先顺序，这些问题只是一个开始。另一方面，如果优先解决生产问题，可能会影响待办事项的交付时间，这会带来机会成本。如果待办事项延迟交付，那么对销售承诺、营销承诺、合作伙伴和管理层利益相关者会产生什么影响？考虑到待办事项延迟交付的连锁反应，能否以合理的方式重新确定待办事项的优先顺序？如果因为修复生产问题而延迟交付，那么待办事项是否应该缩减交付规模？[18]

在运营工程师、开发人员和产品负责人之间，确定正确优先级的过程是艰难且漫长的，这影响了生产中的用户体验。如果其中一方因休假而缺席，是否会延误确定正确优先级的过程？整个过程是否实现了 DevOps 承诺的速度？

如果谷歌的每个生产问题都与此类似，谷歌搜索还能有如此高的市场占有率吗？显然不会。

图 1.1 展示了在功能开发和运营生命周期中，组织内的运营工程师、软件开发人员和产品负责人在运营问题上缺乏协同和一致性。在功能开发期间，大家各自为阵。而在生产

运营期间，一旦出问题，运营工程师、软件开发人员和产品负责人得以作为一个团队方式展开协作。

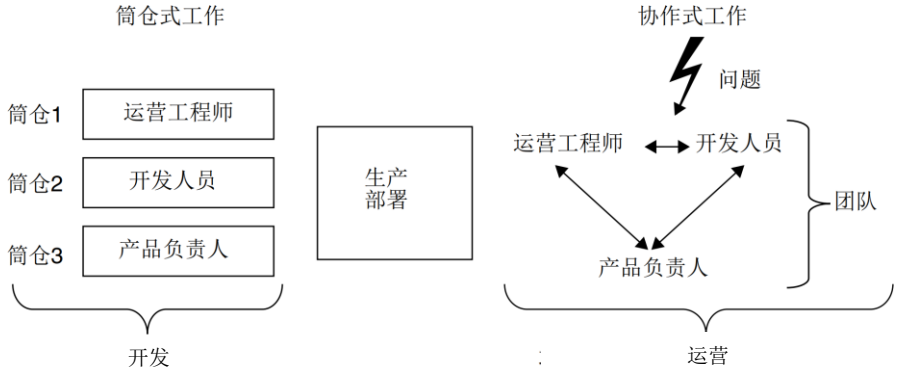

图 1.1　开发阶段和运营阶段的筒仓式工作与协作式工作

图中左侧展示的是功能开发期。运营工程师为生产做准备，但不参与功能开发过程。开发人员专注于功能开发，他们与产品负责人互动以澄清功能需求。产品负责人忙于细化功能需求，回答开发人员的问题，协调利益相关者，引导设计师参与，以及做一些用户研究。在运营方面，三方的工作通常是相互隔离的。运营工程师全力确保已部署功能的顺利运营。虽然开发人员在功能开发过程中会考虑生产问题，但这通常不是他们需要优先解决的最重要的问题。产品负责人的思考和行动通常不涉及运营。

在生产部署之后，功能就会进入运营阶段。运营工程师把他们的注意力放在功能上，因为刚进入生产，所以他们会尽量多了解它。突然间，客户开始报告问题。运营工程师发现，最近部署的功能出现了问题。他们与开发人员取得联系，开发人员又与产品负责人取得联系。他们三方碰头并共同商定下一步的工作计划。他们作为一个团队，以一种非常协作的方式工作，这很好。但是，他们必须针对每个生产问题重复这一过程，因为他们之间缺乏对如何处理功能运营问题的共同理解和一致性。

上述讨论表明，组织在运营问题上缺乏共识和真正的协同。在没有 SRE 的情况下，产品运营、产品开发和产品管理未能在整个产品生命周期中就运营问题进行有效的合作，以确保良好的用户体验。SRE 的作用是在产品生命周期中促成三方在运营上达成共识。这种一致性是产品交付组织中交付速度的重要推动因素。通过适当的协同，我们可以同时充分关注运营和开发，避免频繁且繁琐的上下文切换（图 1.2），从而满足双方的需求。SRE 对产品生命周期每个阶段需要达成的协议都有规定。执行这些协议确保产品运营、产品开发和产品管理能够以高度一致和松散耦合的方式协同工作，从而确保产品运营能够提供积极的用户体验。

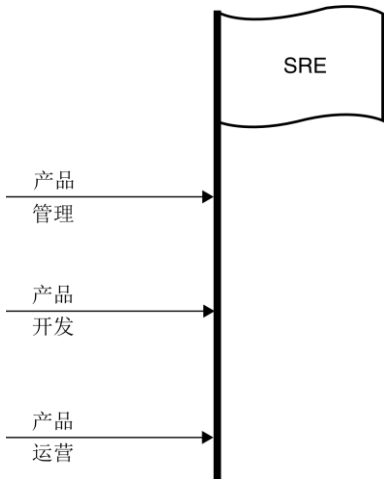

图 1.2　通过 SRE 实现真正的协同

尽管 SRE 在促进产品交付组织在运营问题上实现协同有优势，但它并不能直接支持 IT 监管合规。正如 1.1.5 节所述，谷歌 SRE 创始人本杰明 • 特雷诺 • 斯洛斯如此描述：“当软件工程师得按要求设计运营团队时，SRE 便应运而生。”由于 SRE 不提供 IT 监管合规支持，所以不足以全面管理企业的 IT 职能。因此，SRE 常被整合到 ITIL 或 COBIT 的框架中。这样一来，总体的 IT 监管合规可以通过 ITIL 或 COBIT 来实现，而产品运营、产品开发和产品管理在运营方面的一致性可以通过 SRE 来达到。换句话说，不应认为 SRE 与 ITIL 或 COBIT 互斥。

前面的比较表明，SRE 是支持产品交付组织的三个部门——产品运营、产品开发和产品管理——在运营问题上达成一致的最佳方法。理解这一点后，在推动 SRE 转型时，你就有了有力的论据。基于这个论点，多方可以取得共识，选择 SRE 作为核心运营方法。

要点 SRE 使产品运营、产品开发和产品管理三方在运营上达成一致，实现真正意义上的协同。

接下来，将探讨 SRE 为何有效、其核心原理是什么以及它为何能在谷歌之外的组织中同样发挥作用。

1.3　SRE 为什么有用

为了理解 SRE 为什么能发挥作用，我们首先需要更详细地探讨它是如何工作的。以下过程是 SRE 的核心：

- 从用户的角度为服务定义所谓的“服务水平目标”（SLO）；

- 在生产中度量服务对 SLO 的实现情况；
- 如果 SLO 在生产中被违反，就努力使服务回到 SLO 的范围内，或者调整 SLO 本身。

例如，设想一个负责验证用户身份凭据的服务。该服务公开了一些端点，用户通过这些端点来验证身份凭据。在这种情况下，服务端点的可用性至关重要，因为如果无法验证身份凭据，任何用户都无法登录。基于这一点，服务的可用性 SLO 就需要相当高。例如，所有用于验证用户身份凭据的端点都可能被设置为 99.99%。这意味着在限定时间内，对端点的 99.99%的调用必须成功，才能保证服务在其可用性 SLO 内。

在服务部署到生产之前，完成对可用性 SLO 的定义。在服务部署到生产之后，就可以着手度量 SLO 的实现情况。具体如何度量，需要由 SRE 基础设施提供支持。该基础设施必须保证可用。除了完成度量，SRE 基础设施还要在发现违反 SLO 时发出警报。收到警报后，需要对 SLO 违反情况进行分析。如果发现违反 SLO 会对用户造成真正负面的影响，就表明用户验证服务需要在技术上进行改进，确保其端点在定义的可用性 SLO 内运行。另外，还可能需要制定更严格的可用性 SLO，从而在未来能更早地发现违反 SLO 的情况。另一方面，如果 SLO 违反仅表明技术上的不足而并未对用户体验造成显著影响，这可能意味着需要设定更为宽松的可用性 SLO。

上述过程本质上涉及创建假设、测试假设、从测试结果中学习，然后根据所学采取行动。这一过程反映了科学家们用了几个世纪的科学发现方法。科学发现方法在所有工程行业中得到应用。正如“站点可靠性工程”这一名称所暗示的，SRE 也是一门工程学科。和其他所有工程学科一样，SRE 的核心也是科学发现方法。事实上，在运营中应用发现科学方法是 SRE 能够真正发挥作用的根本原因。

这个过程反映了科学家沿用了几个世纪的科学发现方法。维基百科将此描述为一种经验性的知识获取方法，自 17 世纪以来，这一直是科学发展的核心特征。具体包括仔细观察，并对所观察到的事物采取严格的怀疑态度，因为认知假设可能会扭曲人们对观察的解释。该方法基于观察，通过归纳法提出假设；对假设中得出的推论进行实验和基于度量的测试；根据实验结果完善（或消除）假设。这些都是科学方法的原则，有别于适用于所有科学企业的一系列明确的步骤。[19]

科学方法之所以有效，部分原因是它模仿了自然界通过食物链自动调节物种种群的反馈机制。这种方法久经考验。所有工程学科都以独特的方式应用科学方法。[20]


要点 SRE 之所以独特，原因在于它在软件产品运营中应用了科学方法。

SRE 之所以独特，原因在于它在软件产品运营中应用了科学方法的机制。和其他所有工程学科一样，SRE 的核心也是科学方法。在运营中应用科学方法是 SRE 能够真正发挥作用的根本原因。

1.4 小结

SRE 是在运营问题上实现产品交付组织一致性的最佳方法。它促进了产品运营、产品开发和产品管理在生产运营中的适当参与。SRE 可以使产品交付组织在运营问题上达成一致，可以嵌入到对企业进行 IT 总体治理的 ITIL 或 COBIT 实现中。

正如 SRE（站点可靠性工程）名称所暗示的，SRE 也是一门工程学科。SRE 将科学方法应用于软件产品的运营。这是 SRE 有效的原因，而不只是因为它在谷歌得到了成功。这是要记住的另一个要点。为了在整个组织内引入 SRE，必须基于这个要点来说服所有人。

在澄清了 SRE 为什么确实能发挥作用之后，接下来的话题是 SRE 转型过程中哪些地方会有哪些挑战。这就是下一章的主题。

注释

1 Axelos. n.d. “ITIL”, https://www.axelos.com/certifications/itil-service-management

2 https://en.wikipedia.org/wiki/ITIL

3 Davis, Jennifer, and Ryn Daniels. 2016. “Foundational Terminology and Concepts.” In Effective DevOps. O’Reilly Online Learning. https://www.oreilly.com/library/view/effective-devops/9781491926291/ch04.html

4 Gallacher, Liz, and Helen Morris. 2012. *ITIL Foundation Exam Study Guide*. West Sussex, UK: John Wiley & Sons.

5 https://en.wikipedia.org/wiki/COBIT

6 https://en.wikipedia.org/wiki/ISACA

7 ISACA. “COBIT.” n.d. https://www.isaca.org/resources/cobit

8 Davis, Jennifer, and Ryn Daniels. 2016. “Foundational Terminology and Concepts.” In Effective DevOps. O’Reilly Online Learning. https://www.oreilly.com/library/view/effective-devops/9781491926291/ch04.html.

9 Nissen, Christian F. 2019. “Introduction to COBIT 2019 and IT Management.” SlideShare IOS. https://www.slideshare.net/ChristianFNissen/introduction-to-cobit-2019-and-it-management-140511572

10 Hazrati, Vikas. 2019. “The Difference Between DevOps Engineers and SREs.” DZone, July 29, 2019. https://dzone.com/articles/the-battle-of-devops-and-sre

11 Google Cloud Tech. 2020. “The History of SRE.” YouTube, July 15, 2020. https://www.youtube.com/watch?v=1NF6N2RwVoc

12 https://en.wikipedia.org/wiki/Site_reliability_engineering

13 Beyer, Betsy, Niall Richard Murphy, David K. Rensin, Stephen Thorne, and Kent Kawahara. 2018. *The Site Reliability Workbook: Practical Ways to Implement SRE*. Sebastopol, CA: O’Reilly Media.中译本《Google SRE 工作手册》

14 译注：应用程序从设计之初即考虑到云环境，这样的系统就是“云原生系统”。

15 译注：非我所创，指的是社会、公司和组织中存在的一种文化现象，人们不愿意使用、购买或者接受某种产品、研究成果或者知识。他们的理由并不是与技术或者法律等因素有关，而只是因为它源自其他地方，通常带有贬义。

16 https://en.wikipedia.org/wiki/Not_invented_here

17 Statcounter Global Stats. n.d. “Search Engine Market Share Worldwide.” https://gs.statcounter.com/search-engine-market-share

18 译注：de-scope 是指减少要完成的可交付成果的数量。例如，如果一个项目预定有三个可交付成果，但在截止日期前只完成其中两个，就可以对该项目进行 de-scope，使其显得“成功”。

19 https://en.wikipedia.org/wiki/Scientific_method

20 Neutel, Anje-Margriet. 2014. “Feedback Loops: How Nature Gets Its Rhythms.” YouTube, August 25, 2014. https://www.youtube.com/watch?v=inVZoI1AkC8

第 2 章

面临的挑战

1.2 节举例说明了产品交付组织在没有就运营进行协同的情况下如何运作的。结果显示，没有协同会导致运营问题在生产出现问题后才能得到解决，通常是在产品运营、产品开发和产品管理部门召开的临时紧急会议中。这显然不是以产品为中心的思维方式。

缺乏运营上的协同意味着在整个产品周期中，产品交付组织不能持续和一致地整合运营的各个方面。由于产品运营是产品管理、产品开发和运营流程的最后一环，人们往往将其视为最后要处理的事项。这不是一种以产品为中心的思维方式。用户接触的是生产中的产品。因此，这个接触点需要以产品创造生命周期中的所有活动为中心。事实上，产品运营的重要性需要提升与用户研究、用户故事地图、用户体验设计、架构和开发同等地位。

可以通过杂货店的例子来说明在整个产品生命周期中不考虑生产后果的情况。设想一下，一家连锁杂货店在全国各地拥有设计精美的店铺，铺面中陈列着各种产品，唯独忽视了销售点的收银台。顾客可能无法快速完成付款以至于结账的队伍越来越长。POS 设备由运营团队提供支持，该团队收到了大量支持请求。危机发生时，开发人员正忙于为 POS 设备开发新功能。运营工程师联系开发人员，但开发人员不确定是该优先满足运营工程师的要求还是继续开发新功能。开发人员联系产品负责人，要后者来决定优先级。最终，运营工程师、开发人员和产品负责人共同决定优先修复最紧迫的产品问题。

基于此例，图 2.1 展示了在未就运营进行协同的前提下产品交付组织的运作方式。

图中左边是产品开发团队，后者根据产品管理排定的优先级来完成待办事项中的任务。总体而言，产品开发团队通常会忽略生产环境中的实时动态。他们对系统在生产中的表现没有持续的可见性，也未设置警报系统来及时接收异常情况的通知。产品开发的重点完全放在新功能的开发上，没有把产品运营并未列入他们的待办事项清单。

图中右边是产品运营团队。该团队致力于维护产品在生产中的稳定运行。然而，他们缺乏对产品内部运作的深入了解，这种知识与产品开发紧密相关。随着新版本频繁部署到生产中，这些知识也在迅速发生变化。由于缺乏对运营产品的深入了解，运营团队就只是对外部可见的技术资源设置警报，如内存消耗、CPU 占用率、队列填充水平、磁盘存储填充水平以及网络监控等。一旦这些参数超过预设的阈值，就触发警报。此时，运营团队会尝试了解系统是不是出了什么问题，通常需要咨询产品开发团队，分析潜在

的问题。随着积压的问题增多，产品运营团队会有挫折感。他们不理解产品开发团队对解决生产中产品问题是怎样的态度。既然生产是客户真正使用产品的地方，为什么它们的重要性那么低呢？

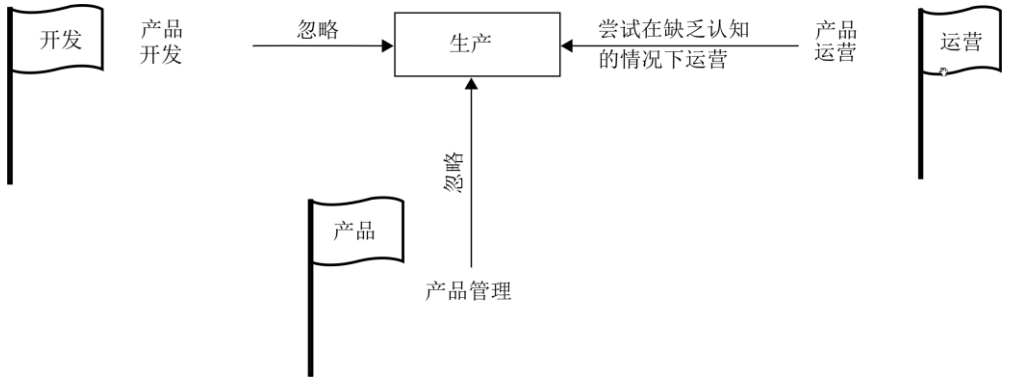

图 2.1　产品交付组织未就运营问题进行协同

这种挫折感暴露了不擅长运营的产品交付组织的核心问题。在这样的组织中，“以产品和用户为中心”对不同的人意义不同。从产品运营的角度来看，这意味着生产问题会得到最优先的处理。从产品开发的角度来看，这意味着产品负责人要求的功能会被尽快开发出来。从产品管理的角度来看，这意味着客户要求的用户故事会尽快转化为生产中的功能。在产品过程中，这种以产品和用户为中心的严重错位是导致产品难以在生产中满足客户需求的原因之一。在对产品交付组织的各方进行协同上，SRE 能发挥巨大的作用。

图 2.1 底部描绘的是产品管理团队的角色。产品管理团队通常与生产思维保持距离。他们忙于与管理层、利益相关者、客户、合作伙伴和用户对话，试图弄清楚产品在市场中的定位，确定遗漏的用户旅程，指出优化工作流的方法等。产品管理部门维护着一个包含待实现功能的待办事项清单。尽管待办事项有优先级，但它并没有考虑到产品运营的要求。产品管理团队期望产品开发团队负责开发，产品运营团队负责运营。这正反映了这些部门名称所暗示的职责划分。

实际上，这种设置导致没有人真正对生产运营承担责任或拥有所有权。那么，究竟由谁负责呢？是产品运营团队吗？不是，因为他们缺乏必要的知识来真正负责生产运营。从产品开发和产品管理到产品运营，没有适当的持续知识迁移，反之亦然。是产品开发负责吗？肯定不是。他们的重点是搞定待办事项。待办事项中是没有产品运营的。偶尔在产品运营升级后交付必要的生产热修复补丁，并不表示他们真的就在“负责”生产运营。那么，是产品管理负责吗？肯定不是。他们的重点是定义产品。他们的期望是，产品开发实现产品，产品运营在生产中运营它。虽然其职位描述中有“负责人”（owner）一词，但产品负责人并不全权负责或拥有产品，包括生产。

在这种情况下，难怪产品最终在生产中被忽视。没有“所有权”的地方就不会有“承诺”。这就需要产品交付组织中的所有各方做出承诺，为生产中的产品运营做出贡献。但具体怎么做呢？谁需要承诺什么才能为产品运营确立有意义的部分所有权？那么，产品运营的所有权是一种集体所有权吗？接下来让我们详细探讨这些问题。

2.2 集体所有权

根据维基百科的解释，集体所有权是指一个集体的所有成员为其共同利益而拥有对生产资料的所有权。[1]这个定义表明，每个人都需要从所有权中获益。在产品运营的背景下，这意味着如果要在产品交付组织中建立集体所有权，那么所有权需要使所有相关者从中获益。具体来说，如果要在产品运营、产品开发和产品管理之间建立生产运营的集体所有权，各方都需要从中获益。这是一个值得深入研究的问题。

在产品运营团队看来，他们负责或拥有产品运营。然而，很难让产品开发和产品管理团队参与其运营活动。因此，如果产品开发和产品管理团队能对生产运营拥有部分所有权，产品运营团队绝对会拍手称快。

在产品开发团队看来，他们是功能的开发者。将新功能快速交付到生产中是其活动的核心。如果能够部分负责或拥有生产运营，他们将获得哪些好处？待办事项是否包含与运营相关的用户故事？然而，由于运营工作相较于功能开发更加不可预测，所以将运营用户故事纳入待办事项可能不太实际。但如果部分拥有生产运营的权益能够提升开发人员对运营的洞察力而改善开发过程并改善整个运营环境，则是有益的。此外，如果改进的开发过程能够减少那些频繁干扰功能开发的生产问题，同样有益。

在产品管理团队看来，他们是定义产品的人。功能应该由产品开发团队来开发，由产品运营团队来运营。如果产品管理部门部分负责或拥有生产运营，会有什么好处？为了回答这个问题，需要研究一下客户投诉升级[2]的问题。产品管理部门尤其不喜欢客户投诉升级。每次客诉升级都与生产中的问题有关。这种投诉升级扰乱了他们的工作，需要立即关注。尽管客户不满意，产品管理部门仍需花费大量时间向各利益相关者解释产品的合理性，这可能会损害他们的信任。利益相关者的信任度降低，可能导致产品的预算减少。这是每一个产品负责人都要努力避免的困境。如果部分拥有生产运营的权益能够减少客诉升级，对产品管理来说将是一个巨大的优势。

表 2.1 总结了在产品交付组织中建立产品运营集体所有权可以为各方带来的好处。

在明确生产运营的集体所有权可能为产品运营、产品开发和产品管理带来的好处后，接下来要探讨的问题是如何获得这些好处。一个相关的问题是参与各方都可以获益的话需要投入多少成本。换言之，在 SRE 转型的背景下，如何使用 SRE 来实现生产运营的集体所有权并有一个可观极的投资回报率？

表 2.1　产品运营的集体所有权所带来的好处

	学科	好处
产品运营的集体所有权	产品运营	产品开发和产品管理根据需要适当参与运营活动。不用再追着产品开发团队和产品管理团队问生产中的每个问题应该如何处理
	产品开发	对产品运营有了适当的洞察，进而改善开发过程，并通过一个完整的运营环境来加强这个过程。开发功能时，若能充分了解使功能在生产中获得技术上的成功所需的前提条件，则可以减少客户投诉升级。这进而使开发团队免受干扰，有了更多不间断的时间进行新功能的开发
	产品管理	减少客诉升级和处理这些状况的时间投入

2.3　SRE 应用场景下的所有权

SRE 应用场景下生产运营的部分所有权是什么意思呢？这个问题需要针对产品交付组织中的各方进行具体解答。

2.3.1　产品开发

产品开发团队负责或拥有生产运营的好处在于，可以借此了解系统在实际用户、数据和基础设施负载下的生产行为。为了对生产中的系统有一个持续的了解，最有效的方法就是在生产中观察它。这可以通过实施轮流值班制度[3]来实现。传统上，产品运营负责生产中的服务值班。这样的话，来自生产的洞察就不会直接传递给产品开发团队。因此，产品开发需要参与生产中服务的轮流值班。每个开发团队都应对一些服务负有责任或所有权。对于这些服务，相关开发人员应参与轮流值班，以便从实际运营中获得宝贵的见解。

这些见解可以促成产品开发和运营做出以下改进。

- 由具有产品实现知识的开发人员开展产品故障调查。
- 从问题发生到修复的链条中，步骤可以简化到一个。问题可以直接交给实现服务的开发人员，他们能以最快的速度修复问题——只要警报的目标足够明确，并与产品负责人达成立即修复生产问题的协议。开发人员可以将从故障本身、故障分析和修复中获得的经验带到新功能的开发过程、支持基础设施以及调试工具中。这将提高产品的未来‘可运营性’，并减少其投入运营所需要的时间。这可以进一步确保开发人员有更多的连续时间专注于功能开发。

- 开发人员在生产现场进行测试，可以在现实世界中体验到产品的质量。一个系统的内部测试很少像在生产中那样密集。看到真实世界的场景为自动化测试套件的开发提供参考，并有助于缩小内部测试和生产场景之间的差距。因此，一旦看到内部测试套件的测试结果是绿色的，就会增强将产品部署到生产中的信心。这应有助于减少因内部未测试的场景而导致的生产失败。此外，这也有助于缩短修复生产问题的时间，从而为功能开发留出更多时间。
- 开发人员可以掌握运营产品和排除故障所需的必要知识。这为开发过程提供了见解，至少能有更好的运营工具。此外，它还有助于减少花在诊断生产问题上的时间，留出更多时间来进行功能开发。
- 开发人员在开发新功能时可以利用来自产品运营的知识。例如，可以了解到可扩展性和性能要求，这往往会导致架构的变更。虽然进行这样的变更需要做大量工作，但确实有必要根据生产中观察到的负载情况来实现架构的弹性。只有这样，系统的运行负担（也称为劳作或 toil）才能减少，留出更多时间来进行功能开发。
- 开发人员对提供良好运营的产品所需的测试和工具会有一个更好的理解。需要设计测试场景、测试级别、测试运行和测试环境，使所有测试活动有机地结合起来，解决系统在生产中遇到的重要情况。可以肯定的是，生产本身也能成为测试环境之一，只不过这种测试是以 24/7 的方式运行的。洞察产品在生产中的运行情况，可以为整个测试管理过程提供大量信息。这会导致测试套件和测试在运行时更贴近生产中的场景，避免浪费时间去测试或捕捉生产中不容易重现的错误，同时避免浪费时间对这种测试进行返工和维护。理顺测试管理之后，就有更多的时间来进行功能开发。
- 开发人员更有动力去实现可靠性功能和工具，以获得良好的产品运营体验。这是因为，如果开发人员负责值班，实际上是希望花尽可能少的时间来处理生产问题。在这种情况下，他们可以完全掌控局面。他们有能力在实现产品时就考虑到生产运营的问题。这样一来，在生产问题上花的时间更少，多留些时间来开发信功能。这对客户和产品管理都有好处。客户也不希望在生产中处理产品的故障。相反，他们希望现有的功能可以在生产中发挥作用，并迅速为产品添加新功能。在客户要求的驱动下，产品管理团队希望能够专心开发新的功能。

有产品运营经验的开发人员在行业中更受欢迎。通过值班，他们可以掌握更多技能来提高薪资。

让开发人员来值班，这样的想法引出了大量的问题。

- 开发人员是否总是需要值班？不需要。
- 开发人员能否只在办公时间值班？是的。
- 负责产品运营的人员可以参与值班吗？可以。

- 对于一个特定的组织，什么是最好的设置？这要视情况而定。
- 开发团队的设置是否需要进行调整以进行值班？是的。
- 团队中的开发人员可以轮流值班吗？可以。
- 尽管有值班任务，重点功能的开发仍然可以完成吗？是的。
- 具体如何实现呢？这要视情况而定。
- 开发人员都去值班了，他们还是开发人员吗？是的。他们会成为更好的开发人员。他们的技能将在就业市场中获得更高的评价。

这些问题及其他相关问题将在本书中适时深入探讨，此处不再赘述。现在，我只想概述一下 SRE 转型所带来的挑战。目前最重要的是理解产品开发团队的人员需要适当参与值班，这具体取决于为 SRE 导入所选择的组织设置。如果没有开发人员适当参与值班，产品开发将无法充分利用生产运营的集体所有权带来的好处。没有生产和开发团队之间的实时反馈循环，功能开发就很难从运营的角度进行改进。如果功能开发团队不能很好地利用实现功能的人所经历的、来自生产的实时反馈循环，就不能持续改善生产中的故障。换言之，如果没有开发人员进行某种程度的值班，无论运营出了什么问题，产品开发团队中的事情都不会有太大变化。

要点 开发人员必须要适当参与值班，从很少的时间到几乎全部的时间都可以。

图 2.2 中，左端展示了这一要点。

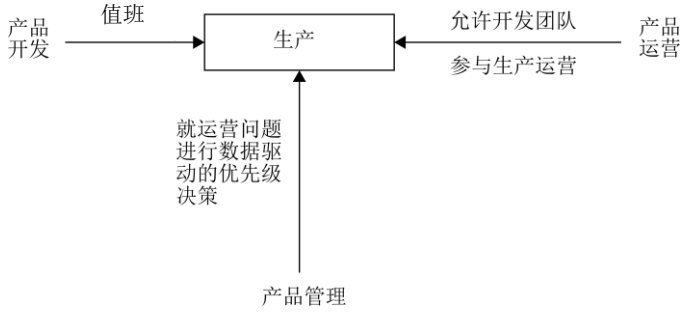

图 2.2　使用 SRE 的产品运营的集体所有权

2.3.2　产品运营

图 2.2 中，右端展示了产品运营。在开发人员参与值班的情况下，产品运营团队需要提供必要的支持，以确保开发人员能够顺利参与运营活动。

那么，开发人员需要哪些支持呢？他们可能从未涉足运营，所以这一领域对他们来说可能相当陌生。是否有这方面的培训？运营团队是否需要提供一些培训？在产品运营

中，“好”是如何定义的？是否有任何现成的文档？这些都是开发人员首次值班时要考虑的问题。

产品运营团队掌握所有关于产品运营的知识。这些知识包括哪些方面呢？基本上，就是将产品作为一个黑盒子，把它放到生产环境中，激活对 IT 资源的监控，并对一些超出阈值的行为发出警报。开发人员可以学习和理解这些知识。凭借对产品内幕知识的了解，他们还能找到更多可以监测和报警的场景。开发人员对产品架构、实现、配置和部署的了解，是提升生产监控能力的宝贵资产。如何实现 DevOps 所倡导的开发与运营的有效沟通呢？

开发人员和架构师不仅了解架构的优势，还清楚其劣势。他们知道架构的局限性在哪里导致了性能和可扩展性问题。他们知道在什么情况下性能和可扩展性问题可能会表现出来并可能影响到客户。他们知道系统在架构上存在的债务，以及其中的哪一部分计划在不久的将来偿还。他们知道任何必须进行的重大的架构重构，这些重构原本因为牵扯的精力太大而未被纳入计划。

开发人员还掌握了许多其他知识。他们了解运行产品所使用的基础设施的限制。他们知道每个服务是如何影响其他服务的；例如，他们知道如果服务网络中的某个服务在基础设施的特定区域内消耗了大部分的内存会怎样。他们可能知道运行服务的容器集群的一些参数，并会根据不断变化的数据和用户负载情况来预测可能发生哪些问题。

换言之，开发人员对系统内部的运作了如指掌。然而，如何帮助他们运用这些知识来提升产品运营的效率呢？为了解决这些问题，需要稍微转移一下注意力，关注开发人员如何让外界了解系统内部正在发生的事情。这是通过日志记录来实现的。开发期间，开发人员决定在什么情况下记录什么。这样一来，一旦产品在生产中运行，就会不断生成包含日志信息的日志条目。然后，分析存储在日志文件或其他存储系统中的日志条目，了解系统运行时发生了什么。这就是开发人员向外界公开系统运行过程的基本过程。该过程得到了工具的支持，这些工具提供了各种开箱即用的运行时功能。也就是说，开发人员关于产品的知识可以编码在日志中，可以在系统外进行分析。

接下来需要考虑的问题是，为了改善产品运营，应该记录哪些信息？假设这些问题都能得到妥善解决。一旦问题得到回答，就要考虑如何以统一的方式记录相关信息。日志的格式如何？哪种日志格式适合自动化日志处理？运营的不同方面是否需要不同的日志格式？例如，是否需要用一种日志格式计算服务的可用性，另一种计算服务的延迟？异步操作又该怎么办？怎么记录它们？在哪里存储日志？日志应该存储在区域数据中心，还是应该集中存储？日志应该存储多长时间？我们假定所有这些问题都会得到解答。

有了这些问题的答案，接着考虑如何检测异常情况。什么应该被视为可用性违反（broken availability）？什么应该被认为是延迟的违反（broken latency）？什么应该被认为是吞吐量不足（insufficient throughput）？除了可用性、延迟和吞吐量，还有哪些方面需要考虑？同样，我们假定所有这些问题都会得到解答。

接着，想知道在异常情况下如何报警。警报应该在检测到异常情况后立即生成，还是稍后生成？警报应该进行采样吗？如何避免警报疲劳，即那些收到警报的人被太多的警报所淹没，并停止对它们做出反应？在频繁提醒人们以至于造成警报疲劳和很少提醒以至于造成事故不被注意到之间，如何平衡？警报中需要包括什么样的信息？即使这些问题也都不是问题，也还有更多的问题。

接下来的问题是向谁发出警报——具体地说，哪些开发人员会收到警报？如何提醒开发人员，使其不至于从当前的功能开发工作中分心？在不会导致警报疲劳的前提下，如何提醒开发人员并使其真正对警报做出反应？是不是任何开发人员都能接收警报？开发人员需要具备什么样的知识，才能在合理的时间范围内，付出合理的精力对警报做出反应？

问题清单还可以继续。它表明，需要一个全面的框架使开发人员能参与产品运营。但什么是框架呢？维基百科上的词条如此定义：“软件框架是一种抽象，可以通过额外的、由用户编写的代码，选择性地改变提供了通用功能的软件，将其转化为特定于应用的软件。”因此，在运营框架的背景下，需要一些可以选择性改变的通用功能。在 SRE 的背景下，这样的框架可以称为 SRE 基础设施（SRE infrastructure）。它需要提供通用的功能，以支持前面示范的各种用例，并在 SRE 的背景下实现。通用功能要求能选择性地改变，使基础设施适应整个 SRE 活动中的特定用途。

运营工程师需要提供一个框架使开发人员能参与服务运营。在 SRE 背景下，这样的框架可以称为 SRE 基础设施。本书写作时，已经有一些现成的商用工具可以支持 SRE 基础设施，但还不够全面，缺的部分还需要进行定制开发。因此，为了建立 SRE 基础设施，十有八九还是需要一些定制的软件开发与现成的工具相结合。这意味着产品运营也需要学习软件开发。

产品运营团队面临的挑战是，缺少提供一个框架以便其他团队参与运营的经验。产品运营一直都在使用现成的工具，以动手实践方式开展运营工作。现在，产品运营还要支持产品开发对服务运营的参与。这个支持是通过设想中的 SRE 基础设施来实现的。SRE 基础设施需要使用一流的软件开发技术来构建。

这与 SRE 创始人本杰明 • 特雷诺 • 斯洛斯的观点相一致，他说：“当软件工程师须按要求设计运营团队时，便有了 SRE。”基于此，为了使产品开发团队能参与运营，需要软件开发团队建立一个合适的 SRE 基础设施。这不奇怪。构建框架在软件开发中很常见。使用框架对软件开发人员来说也不陌生。但对于来自产品运营学科的运营工程师来说，无论构建框架还是使用框架，都不熟悉的。

以下是目前我们所理解的 SRE 转型挑战：

软件开发人员需要学习如何通过值班来参与产品运营。

图 2.3 对此进行了展示。两个箭头类似于击剑的动作。听起来很讽刺，但这正是 SRE 转型期间发生的事情。

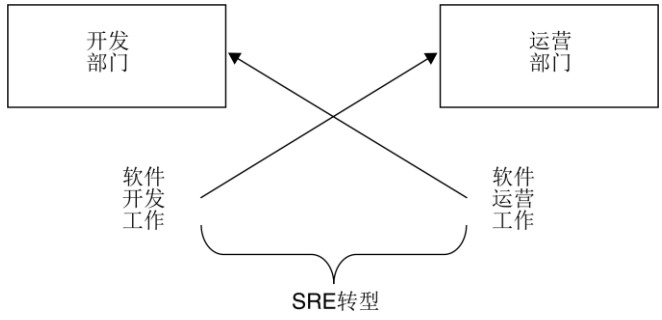

图 2.3 关键的 SRE 转型挑战

这两个箭头都不容易实现。然而，正如世界各地越来越多的软件交付组织所证明的那样，这是完全可能的，本书将对此进行详细探讨。

图 2.3 展示了所谓实现 DevOps 的真正含义和要求。它是指开发人员做运营工作，而运营工程师做开发工作。它涉及产品开发和产品运营这两个长期存在的学科的核心，动摇了它们的基本职责。为了真正实现 DevOps，要求的远不限于实现产品开发和产品运营之间的紧密合作。

在传统软件交付组织中，困难尤其大。那些从未涉足过运营的开发团队，以及从未让其他团队参与运营的运营团队，都缺乏建立 SRE 的基础。开发人员不理解自己为何要做运营，运营工程师没有提供让开发人员来参与运营的框架。管理者不倡导这样的努力，更别说提供经费了。

虽然存在这些困难，但着手进行 SRE 转型的努力是非常值得的。使用 SRE 来实现 DevOps 的好处在于，一方面能为开发人员留出最多的时间来开发功能，同时产品能在生产中很好地服务于客户。如果没有 SRE，虽然开发人员能最大限度地增加时间开发新的功能，但生产被忽视了。

此外，借由 SRE 来实现 DevOps 理念，运营工程师能向开发人员提供 SRE 基础设施，使其可以参与生产运营，从而获得良好的可扩展性。没有 SRE，运营工程师就会成为瓶颈，纯粹靠自己进行生产运营，无暇顾及产品质量和关于产品的内幕知识。

2.3.3 产品管理

在阐述了生产运营的集体所有权对产品开发和产品运营的重要性之后，还需要在产品管理的背景下进行澄清。产品管理团队如何参与才算得上是部分负责或拥有产品运营？

传统上，产品管理与产品运营之间的联系并不紧密。如 2.2 节所述，产品管理参与生产运营的主要好处是减少客诉升级。如果客诉均涉及技术问题，产品管理应如何减少投诉升级？产品负责人并非技术专家，他们不懂产品的实现和部署。

为了解决这一问题，需要考虑导致客诉升级的因素。在客户拿起电话向客户支持部门投诉之前，会发生一系列事件。客户在使用产品时，可能遇到一些令人不悦的情况，如数据显示迟缓、完成任务的步骤繁琐、操作无预期结果或产品直接崩溃并导致数据丢失。无论原因是什么，都直接关系到客户损失了大量时间或金钱，促使他们联系客户支持以表达不满并寻求帮助。

技术专家——产品开发团队和产品运营团队——能否发现并提前解决产品中的任何问题？他们是否已建立了相应的机制来检测和解决此类事故？再次强调，这些都属于技术问题的话，与产品管理有何关联呢？

让我们做一个更进一步的探讨，假设产品开发和产品运营希望建立事故检测和解决机制，以便在客诉升级之前发现并修复问题。开发人员掌握了大量产品技术方面的知识，而运营工程师在客诉升级方面有丰富的经验。他们能够汇总以往的投诉事件并预测产品的薄弱环节，因为产品开发还没有进行修复。产品开发和产品运营知识紧密结合，产品开发团队带来技术实现方面的知识，而产品运营团队带来生产实际问题的知识。结合这些知识，可以创建一个植根于技术实现和客户历史投诉升级的事故检测与解决过程。这是基于临时的、非系统化的事故响应的一个巨大飞跃，能有效减少客诉升级。

但我们的目标更为远大，即建立一个事故响应和解决流程，能够及早发现每个现有功能和新功能的异常，使产品开发团队能够及时修复，并在客诉升级前部署这些修复。此时，产品管理在产品运营集体所有权中的作用开始明确。需要强调的是，该过程应适用于所有现有的和新功能，而不仅仅适用于产品运营部门根据过去客诉升级的经验而了解到的功能。此外，开发人员开始以这种方式分配时间：在客户愤怒到投诉升级之前修复已经发现的问题，并将其部署到生产中。这意味着，开发人员不再只是完成产品负责人制定好的优先级待办事项。现在，我们有了另一个优先级驱动因素，即事故响应过程中发现的产品可靠性问题。

就这样，产品管理对产品运营集体所有权的贡献开始明确下来。

- 产品负责人需要将用户旅程的知识贡献给事故检测过程。碎片化的用户旅程应该是事故检测的核心。哪些是进行事故检测时最重要的用户旅程？在一个给定的用户旅程中，使用户旅程保持意义的最重要的、必须有用的步骤是什么？反过来说，

用户旅程的哪些步骤可以失败以及失败的程度如何而不会造成整个用户旅程的崩溃？总的来说，事故检测过程的有效性应与其定义的事故检测能力相匹配。为了很好地定义可检测的事故，需要结合产品负责人的用户旅程知识、开发人员的实现知识和运营工程师的运营知识。

- 产品负责人还需要理解并同意建立一个待办事项管理程序的重要性。在这个程序中，开发人员可以灵活地分配时间来修复事故检测过程中发现的生产问题。传统上，是由产品负责人对用户故事的待办事项进行优先级排序，他们希望开发人员能够专注于待办事项。为了减少客诉升级，产品负责人希望开发人员立即对事故检测所报告的问题采取行动。

现在，我们清楚了这一切对产品负责人的意义。他们有责任参与定义事故检测。他们知道事故检测要检测什么。要检测真正的中断用户旅程，而不仅仅是一些技术上的偏差。现在，产品负责人更容易接受花时间在解决事故上。为什么？因为花这些时间能直接减少客诉升级。如果开发人员没有在合理期限内修复生产事故，即使尽早发现了正确的事故，客诉也会升级。

也就是说，为了减少客诉的升级，需要满足以下标准：

- 事故检测可以检测到由运营工程师、开发人员和产品负责人共同定义的被违反和碎片化的用户旅程；
- 开发人员在检测到受损和中断的用户旅程时，优先修复它们，而不必每次都与产品负责人协商工程时间的分配；
- 开发人员在规定期限内，在客户生气和沮丧到投诉升级之前，修复生产中受损和碎片化的用户旅程。

这个过程如图 2.4 所示。图 2.4 中，左上方展示了事故检测的定义过程。它将开发人员的实现知识、运营工程师的运营知识以及产品负责人的用户旅程知识作为输入。事故检测定义过程的结果是对生产中要检测的事故加以理解。它检测的是以下几个方面存在的不健康的模式：从运营关键性的角度看的用户旅程、实现用户旅程的关键服务依赖项、实现用户旅程的关键基础设施组件及其可扩展性。

定义了生产中待检测的事故后，即可开始事故检测。图 2.4 中，生产环境在运行时，会对真实用户旅程的实现情况进行监控，而不再只是监控技术参数是否达标。监控真实用户旅程的实现情况更有针对性，有助于减少客诉升级，这是产品管理参与产品运营的一个优势。

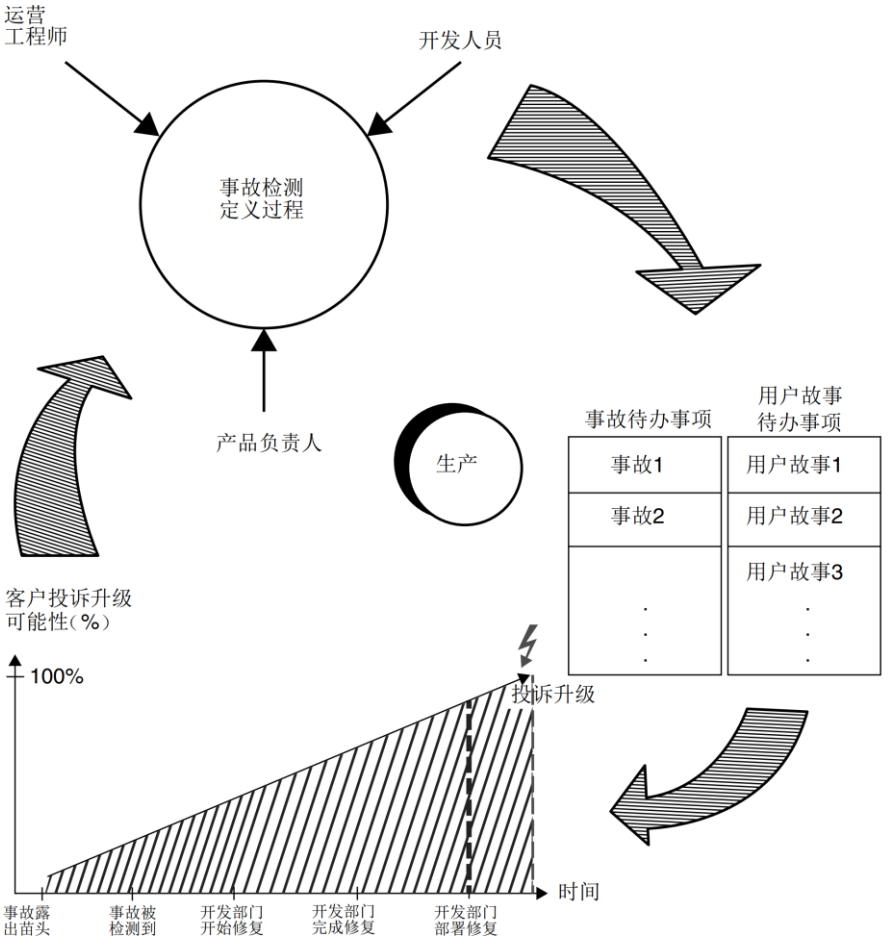

图 2.4　减少客户投诉升级的过程

事故检测过程一旦识别到事故，就会将它们加入待办事项清单，如图 2.4 右侧所示。事故待办事项将与用户故事待办事项并列。用户故事待办事项已经由产品负责人确定了优先级。事故待办事项同样需要进行优先级排序。确定优先级的工作应在检测到事故后立即开始，并迅速完成。运营工程师、开发人员和产品负责人之间不应进行冗长的协商来确定待办事项的优先级。这意味着三方需要事先达成共识。最佳地点是在哪里就事故优先级达成协议呢？一个好地方是在事故检测定义的过程中。作为该过程的一部分，不仅要定义事故本身，还要就其相对优先级达成协议。这些协议应该能使所有值班人员——尤其是开发人员——对大多数事故自主做出优先级判定。

由于事故待办事项清单中的事故需要及时处理，所以处理事故的开发人员无法同时处理用户故事待办事项清单中的用户故事。此外，频繁在事故待办事项和用户故事待办事项之间切换，会产生大量的上下文切换成本。这不仅效率低下，还会给开发人员带来沉重的心理负担。为了解决这一问题，我们可以通过实施策略来构建开发团队，以在事故待办事项清单中的持续值班和用户故事待办事项清单中的关键用户故事之间实现良好的平衡。我们将在后面具体讨论这些策略。

图 2.4 底部展示了事故处理的时间轴。时间轴从事故初现端倪开始（最左侧）。在事故刚刚发生时，客诉升级的可能性非常低。随着时间的推移，这种可能性最终可能增长到 100%，需要避免这种情况。我们的目标是在客户变得非常愤怒和沮丧且不得不联系客户支持进行投诉之前，解决这个事故。

事故初现时，可以通过事故检测来识别。在时间轴上，接下来的步骤是开发人员开始修复的时间点。问题一旦修复，就需要部署到生产环境。部署完成后，需要监控修复效果，确保事故得到彻底解决。我们的目标是在客户投诉升级的红线之前完成修复部署并进行监控，确认事故已得到解决。

可能未能及时留意事故的苗头，或者事故检测可能过早或过晚发现它们。这些事故可能是误报——即报告的事故并未实际导致用户体验下降。所有这些情况都应作为后续调整事故检测定义的依据。这个反馈回路是基于数据的。这使得产品运营、产品开发和产品管理三方能够基于数据中立地决定事故的定义。

在 SRE 应用场景下，这样的事故检测和响应过程是使用特定的机制和术语来设置的，例如服务水平指标（SLI）、服务水平目标（SLO）和错误预算策略。我们很快就要开始探究这些概念。但在开始探索之前，先总结一下使用 SRE 的集体生产运营所有权的效益和成本。

2.3.4 效益和成本

经过分析，我们知道产品运营、产品开发和产品管理如何以团队方式实施 SRE。使用 SRE 实现 DevOps 时，三方需要进行深度整合，而非只是三方密切合作。表 2.2 显示了效益和成本。

表 2.2　使用 SRE 进行产品运营时，集体所有权的效益与成本

	学科	效益	成本
使用 SRE 的产品运营的集体所有权	产品运营	产品开发和产品管理根据需要适当参与运营活动。不必每遇到一个生产问题，都要追着产品开发团队和产品管理团队问应该如何处理	将 SRE 基础设施作为一个框架来实现，使其他人也能参与运营
	产品开发	对生产运营有了适当的洞察，以达到改善功能开发过程的目的，并通过完整的运营环境来加强这个过程。开发功能时，若能充分了解使功能在生产中获得技术上的成功所需的前提条件，那么可以减少客户投诉升级。这进而使开发团队免受干扰，有更多不间断的时间进行新功能的开发。另外，这还升级了开发人员的技能，使其在就业市场更占优势	要在规定时间内值班以参与产品运营
	产品管理	减少客户投诉升级和处理这些投诉的时间投入。对众多生产问题的临时参与也减少了。另外，现在还能以数据驱动的方式对工程能力在功能与运营问题上的分配做出权衡	参与对事故检测的定义和基于生产数据的、数据驱动的优先级决策

现在，使用 SRE 的生产运营的集体所有权的效益和成本已经清楚了，让我们看看 SRE 试图达成怎样的总体目标，如图 2.5 所示。

SRE 使产品交付组织在其运营问题上保持协同。产品开发对生产运营的贡献在于，通过值班获得产品如何在生产中满足客户需求的第一手经验。这样，既确保了产品在生产中满足客户需求，又最大限度地延长了功能开发的时间。

产品运营的贡献在于使开发人员能够独立进行生产运营。这需要开发一个 SRE 基础设施框架，供开发人员使用。

产品管理的贡献在于，他们现在可以基于数据，对需要进行事故检测的关键用户旅程做出优先级决策。此外，他们还与值班人员就事故待办事项的自主优先级排序达成了共识。最终，借助数据驱动的优先级决策，现在可以准确地确定哪些事项应纳入用户故事待办事项中，以确保可靠性。

SRE 转型并非没有成本。产品交付组织必须投入时间、金钱和精力，通过 SRE 协同解决运营问题。这就是管理层也需要参与的原因。管理层可以在 SRE 转型中发挥两方面的作用。首先，他们需要从态度上表示支持。这可以在全体员工会议、小型会议或通过其他沟

通渠道来实现。其次，管理层需要确保为 SRE 转型提供必要的资源和支持，包括资金、人员和工具等。

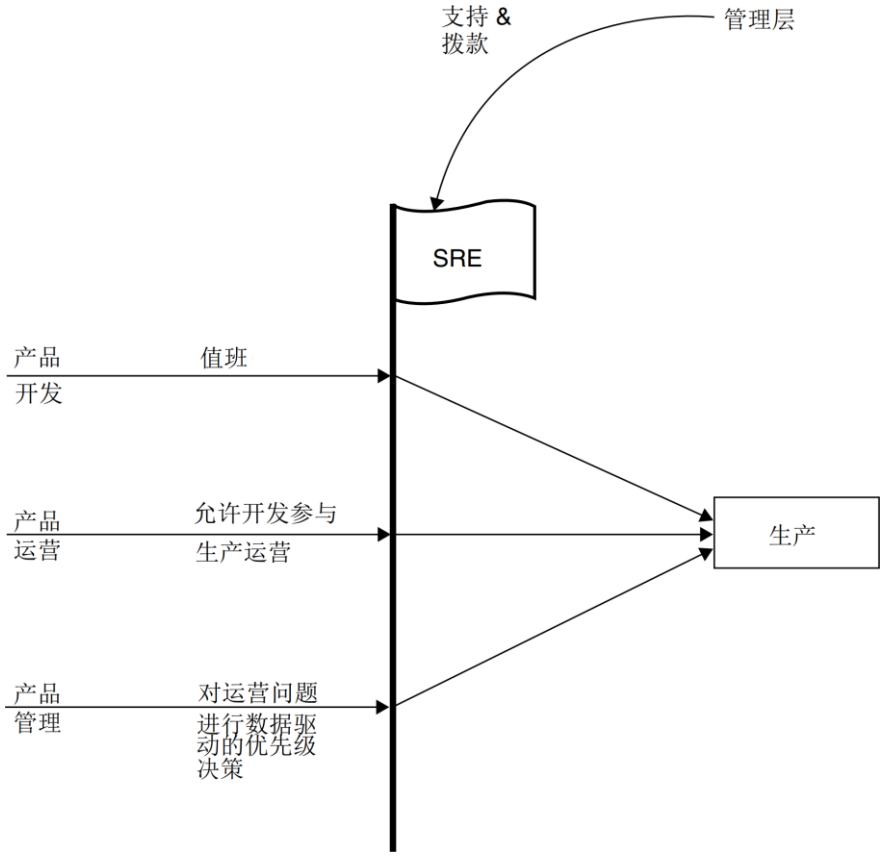

图 2.5　使用 SRE 做产品运营，通过集体所有权将三方整合在一起

通过这些措施，产品交付组织可以确保 SRE 转型的成功，从而提高产品的可靠性和客户满意度，同时也提升整个组织对运营问题的处理能力。在大企业，由于供应商的选择和数据保护过程，可能需要花较多时间来下单。不过，这是值得的，用钱来投票是一个强烈的信号，表明管理层支持 SRE 转型。

深入了解 SRE 转型过程中需要争取什么之后，可以简要提炼出挑战说明。下面描述所面临的挑战。

2.4 挑战声明

SRE 是一种运营方法，旨在帮助产品交付组织实现运营协同。传统软件交付组织面临的主要挑战是生产运营上的错位。在这样的组织中，出现以下情况是正常的：

- 开发人员不知道自己为什么要参与运营；
- 运营工程师不知道开发人员为什么对运营不感兴趣；
- 产品经理认为运营工作是由运营工程师完成的；
- 管理层不倡导这样的努力，也不会为此提供经费。

在这样的软件交付组织中，缺乏基础设施来建立将 SRE 实践，所以需要先打好基础。这将是 SRE 转型的主要部分。转型需要通过以下方式改变利益相关者的思维方式：

- 开发人员应希望参与值班过程，以获得足够的最新运营知识，开发能在生产中发挥良好作用的功能；
- 运营工程师应希望提供 SRE 基础设施作为框架，以便开发人员能在该框架上参与服务运营，从而在整个产品交付组织中以最佳方式分配运营工作；
- 产品经理应希望参与运营，基于生产数据来做出决策，以帮助减少客诉升级。要对待检测的故事排优先级、决定事故待办事项如何处理以及涉及可靠性的功能如何排优先级；
- 高管希望通过促进 SRE 并及时拨款来实现有效和高效的产品运营。

软件交付组织中的各方都从 SRE 中受益。这些好处使它值得经历 SRE 转型。这些好处可以成为人们向往的灯塔，并在 SRE 转型过程中为人们带来乐趣。本书将带领你把软件交付组织转型为以 SRE 方式进行运营，并真正乐在其中。为了开始逐步 SRE 转型之旅，下一节先来看 SRE 转型的一般方式。

2.5 教练

产品交付组织由多个人组成，这些人属于不同的团队。SRE 转型过程有一个明确的目标，即在团队中将 SRE 作为生产运营的核心方法。然而，和运行项目不同，SRE 转型没有预定义的里程碑需要跟踪。相反，SRE 转型过程是一个诱导变革和反馈变革的网络，变革在不同团队和个人之间平行推进。许多团队将在同一时间转型，但变革和反馈回路对每个团队和个人来说都是独特的。图 2.6 对此进行了展示。

如何以图 2.6 所示的方式建立 SRE 转型过程呢？

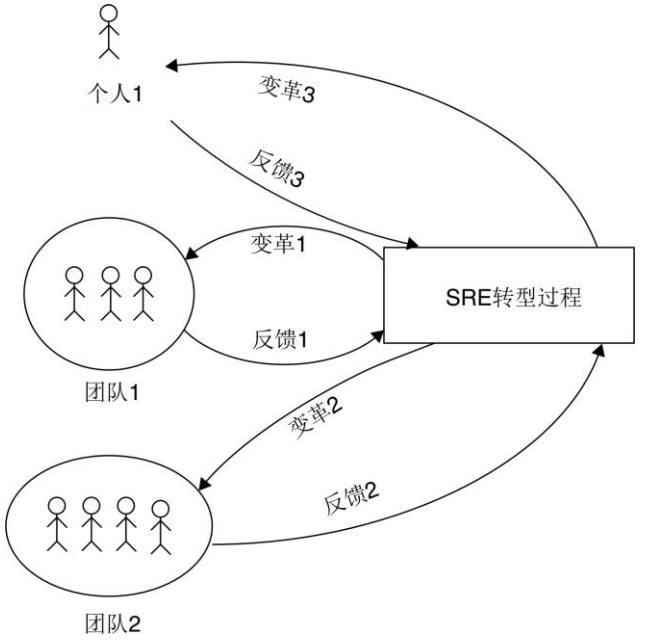

图 2.6　SRE 转型过程

SRE 转型通过教练来完成。根据维基百科的定义，教练（coaching）是指有经验的教练通过培训和辅导的方式来帮助学习者或客户实现特定的个人或职业目标。[4]由此可见，教练是在个人的基础上与人和团队合作。转型需要教练，要求有同理心以及有特定的结构。

在 SRE 转型的背景下，使教练这种方式特别有趣的地方在于，它已经作为一门学科存在于组织和团队层面。根据教练协会的说法，组织教练的目标是促进组织内部积极的、系统化的转型。[5]在教练这一大的主题下，组织教练（为组织提供的教练服务）是较为成熟的。

另一方面，团队教练或团队辅导相对较新，其结构化程度相对较低。在过去十年中，该学科逐渐受到重视。根据 TPC Leadership 的定义，团队教练是一门艺术，旨在推动和挑战团队在追求组织重要目标的过程中尽可能提升业绩和乐趣。[6]。

教练学科明确区分了组织教练和团队教练两个细分领域。在进行 SRE 转型时，团队教练是较为合适的选择。这两种教练方法应同步实施，并采取连贯的工作流程。然而，在选择组织和团队教练作为 SRE 转型策略时，关键问题在于教练的人选。对于刚开始接触 SRE 的组织，合适的教练人选是谁？是否需要聘请外部教练？是否可能培养内部教练？如果可行，又由谁来负责培养这些内部教练？

外部教练能够为组织带来丰富经验的 SRE 视角，并有助于大家快速全面理解 SRE 基础知识。然而，业内难以找到具有成功 SRE 转型经验的教练。这在一定程度上是因为谷歌

2016 年出版的《SRE：谷歌运维揭秘》[7]。考虑到 SRE 转型在大型组织中通常需要数年时间才能完成，因此可用的教练资源相对有限。

鉴于 SRE 转型在大型组织中通常需要好多年的时间，因此在转型期间长期聘请外部教练在经济上可能不可行。因此，寻找并培养内部的 SRE 转型教练变得至关重要。

对于教练来说，学习 SRE 并达到能够指导他人的水平，有多种途径。谷歌的《SRE：谷歌运维揭秘》[8]为潜在的教练提供了宝贵的起点。这两本书展示了实现复杂和规模化 SRE 需要哪些步骤。此外，前谷歌员工的 *Implementing Service Level Objectives: A Practical Guide to SLIs, SLOs, and Error Budgets*[9]和《SRE 生存指南：系统中断响应与正常运行时间最大化》[10]提供了更深入的 SRE 实践观点。

我们这本书旨在向潜在的教练展示如何在未曾采用 SRE 的组织中实施 SRE。此外，教练可以参与相关会议和行业活动来建立联系。USENIX 的 SRECon[11]和 IT Revolution 的 DevOps[12]企业峰会值得关注。这两个大会汇聚了从事 SRE 工作或负责 SRE 转型的专业人士。通过这些联系，可能有机会参观在 SRE 转型中取得进展的其他公司。亲眼见证其他公司熟练运作 SRE 流程，可以显著促进自身的转型。

最后，教练可以在组织内部进行 SRE 转型的同时，积累学习经验。在大型组织中，不同团队可能以不同的速度采纳 SRE。从领导 SRE 转型的团队中吸取经验，并将这些经验传递给正在迎头赶上的团队，这是教练过程中极具价值的一部分。这不仅能够帮助团队的教练积累经验，还能促进不同团队间的相互学习。

在团队层面进行 SRE 转型最有效的方法是实施长期的、基于团队的教练计划，确保所有团队成员——包括产品负责人、架构师、开发人员、运营工程师，以及设计师——参与其中，共同积累实战经验。这种经验对于确保团队成员全面理解并实施 SRE 的原则和实践至关重要。

2.6 小结

引入 SRE 时，需要对产品运营、产品开发和产品管理进行变革。产品运营最大的变化在于采用 SRE 基础设施作为开发框架，使开发人员能够通过它进行值班，在生产环境中运营服务。产品开发的主要变化在于实际参与值班，并在真实的生产环境中运营自己开发的服务。开发人员和运营人员参与值班的程度将根据组织的具体情况而有所不同。

SRE 转型的挑战在于，传统软件交付组织中的产品运营团队从未提供过让其他人员参与运营的框架。同样，产品开发团队也不曾涉足运营工作。因此，它们缺乏建立 SRE 的基础。开发人员不理解为何需要参与运营，运营工程师未能提供框架让开发人员参与运营，而管理人员也没有提倡这一议题，更遑论必要的资金支持。为了推动整个组织的 SRE 转型，培养和发展 SRE 教练至关重要。

本书后续章节将详细介绍软件交付组织的 SRE 转型之旅，并在下一章中探讨 SRE 的基本概念。

注释

1 https://en.wikipedia.org/wiki/Collective_ownership

2 译注：customer escalation 是客服行业的术语，一线客服解决不了的疑难问题会 escalate 给专家坐席或者二级客服，由他们进行专业处理。

3 译注：on call 是指随时待命，在一定时间内随叫随到。本书采用“值班”的译法。

4 https://en.wikipedia.org/wiki/Coaching

5 David, Susan. 2015. “Introduction to Organizational Coaching.” Institute of Coaching, January 13, 2015. https://instituteofcoaching.org/resources/introduction-organizational-coaching

6 Cardillo, Andrea. 2019. “How Is Team Coaching Different from Group Coaching?” TPC Leadership, uly 10, 2019. https://tpcleadership.com/how-is-team-coaching-different-from-group-coaching

7 Murphy, Niall Richard, Betsy Beyer, Chris Jones, and Jennifer Petoff. 2016. *Site Reliability Engineering: How Google Runs Production Systems*. Sebastopol, CA: O’Reilly Media

8 Beyer, Betsy, Niall Richard Murphy, David K. Rensin, Stephen Thorne, and Kent Kawahara. 2018. The Site Reliability *Workbook: Practical Ways to Implement SRE*. Sebastopol, CA: O’Reilly Media

9 Hidalgo, Alex. 2020. *Implementing Service Level Objectives: A Practical Guide to SLIs, SLOs & Error Budgets*. Sebastopol, CA: O’Reilly Media

10 Welch, Nat. 2018. *Real-World SRE: The Survival Guide for Responding to a System Outage and Maximizing Uptime*. Birmingham, UK: Packt Publishing Ltd

11 “SRECon.” 2017. USENIX. August 25, 2017. https://www.usenix.org/srecon

12 “SRECon.” 2017. USENIX. August 25, 2017. https://www.usenix.org/srecon

第 3 章

SRE 基本概念

SRE 的基本概念较为有限且易于理解，为我们的 SRE 转型之旅提供了一个良好的开端。这些基本概念包括服务水平指标（Service Level Indicator，SLI）、服务水平目标（Service Level Objective，SLO）、错误预算（Error Budget）[1]和错误预算策略。第 1 章和第 2 章已经简要提及了这些概念。在本章中，我们将更详细地探讨它们。

3.1 服务水平指标

服务水平指标（Service Level Indicator，SLI）是 SRE 领域的基础概念，其他相关概念均基于 SLI 构建。*Site Reliability Engineering: How Google Runs Production Systems*[2]一书对 SLI 给出了简洁的定义：“一种针对所提供服务水平特定方面的精心定义的量化度量。”

服务相关的 SLI 需要明确界定。典型的 SLI 包括可用性、延迟和吞吐量。为服务制定一套相关的 SLI 是一个经验性的过程，依据前文提及书籍作者的观点，SLI 的定义基于“直觉、经验以及对用户需求的深入理解”。在微软技术社区的一次演讲中，杰森·汉德提出了一个 SLI 层次结构，由多个相互依赖的 SLI 共同构成服务的整体可靠性。如图 3.1 所示，最基本的可靠性层次涵盖可用性和延迟。换言之，服务的可靠性首先要求其可用性。在确保可用性的基础上，服务的响应速度，即低延迟，成为衡量其可靠性的另一关键指标。

并非每个服务都需要吞吐量 SLI，但对某些服务来说，却必不可少。覆盖率 SLI 也只适合某些服务，它衡量服务所处理的数据集是否被完整处理。接下来的 SLI 包括正确性和保真度，它们的适用性取决于服务所涉及的业务和技术领域。正确性是指验证处理过程是否做了它应该做的事。保真度是指向用户提供的功能的真实程度：是否所有功能都像预期的那样提供给用户，或者有的功能因为系统侧的故障而关闭了一段时间？换言之，服务或许仍然可用，但提供给客户的功能集可能在一段时间内被降级。

新鲜度和持久性 SLI 主要适用于某些服务，并非所有服务都需要这两个指标。新鲜度是指所处理的数据的新旧程度。如果存在延迟，所提供的数据可能不是数据源中的最新数据。持久性 SLI 度量的是数据存储中的数据能否在以后检索。

《SRE：Google 运维解密》一书依据相关 SLI 对服务进行了分类，具体分类见表 3.1。我们的目标是将 SLI 作为客户体验到的可靠性的一种代理度量。处于压力下的系统在技术体验上不是 SLI 要度量的。相反，客户在使用系统时的体验才是 SLI 所要表达的

表 3.1　系统分类和相关的 SLI

系统类型	相关的一套 SLI	SLI 所回答的问题
面向用户的服务系统	• 可用性 • 延迟 • 吞吐量	• 我们可以响应请求吗？ • 多久响应？ • 能处理多少请求？
存储系统	• 延迟 • 可用性 • 持久性	• 花多长时间读取或写入数据？ • 我们能按需访问数据吗？ • 数据在需要时可用吗？
大数据系统	• 吞吐量 • 端到端延迟	• 处理多少数据？ • 数据从输入到完成需要多久？

3.2　服务水平目标

SLI 关注客户的期望，SLO 则关注如何满足这些期望。《SRE：Google 运维揭秘》如此定义 SLO：“由 SLI 度量的服务水平的目标值或值范围。”换言之，SLO 是根据 SLI 来定义的。

例如，对于可用性 SLI，可以如下设定：

- 特定某个端点在 4 个日历周内 98%的可用性；
- 另一个端点在 4 个日历周内 99.99%的可用性。

对于延迟 SLI，可以如下设定：

- 95%的端点在 4 个日历周内的请求延迟为 400 毫秒；
- 90%的端点在 4 个日历周内的请求延迟为 250 毫秒。

图 3.2 描述了 SLI 和 SLO 之间的关系[3]。

SLI 代表被度量的目标，如可用性或延迟。SLO 定义服务的最低度量阈值。如图 3.2 所示，SLO 界定服务水平的可接受范围——最小值 0%和最大值 100%。SLO 要基于客户的角度来设定与其相关的 SLI。

此外，为了确保 SLO 反映客户的需求，还必须对目标客户群体进行明确的定义。缺乏清晰的客户定义，SLO 将无法准确反映特定客户群体的期望。因此，若 SLO 未与客户体验紧密关联，则本质上只是一个技术度量，不能充分反映客户群体的实际体验。

如果 SLO 设定不当，就不能准确反映客户体验受影响的程度。在这种情况下，工程团队可能会犹豫是否应投入时间修复服务以便确保其性能恢复到 SLO 规定范围内。

例如，图 3.3 中，SLI 为可用性，SLO 规定服务端点的可用性应达到 98%。换言之，在给定时间范围内，服务端点的可用性不应低于 98%。这可以通过确定在该时间范围对端点请求的成功率来度量。成功率至少为 98%。这意味着，在给定时间范围内，实施和运营团队需要确保端点的可用性不低于 98%。团队需要主动监控端点并采取措施，确保其可用性不会下降到 98% SLO 以下。

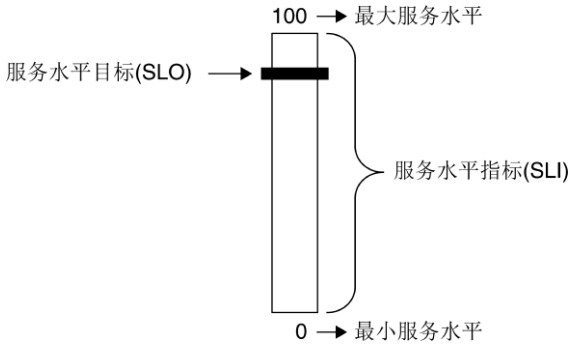

图 3.2　SLI 与 SLO 的关系

图 3.4 展示了另一个例子，其中的 SLI 是延迟。该服务端点的 SLO 要求 95%的端点请求须在 400 毫秒内返回。在这个例子中，实现和运营团队需要确保在给定时间范围内，端点 95%以上的请求都在 400 毫秒内返回。团队需要主动监控端点并采取措施，确保 95%的端点访问延迟都在 400 毫秒以内。

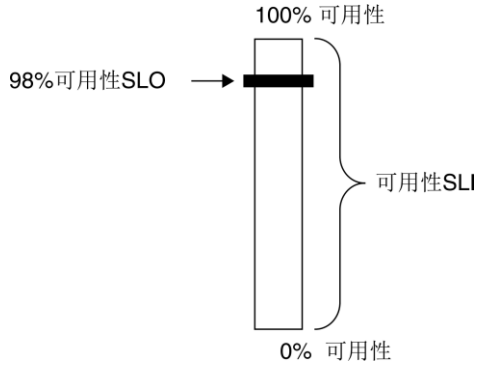

图 3.3　98%可用性 SLO

SLO 应根据客户的需求来设定。若 SLO 没有充分考虑客户视角，将无法准确反映客户的期望，从而导致客户体验降级。违反 SLO 时，可能难以判断客户是否受到影响及其受影响的程度。这导致难以确定应投入多少工程资源以及紧迫程度如何（使其恢复到 SLO 规定的标准）。

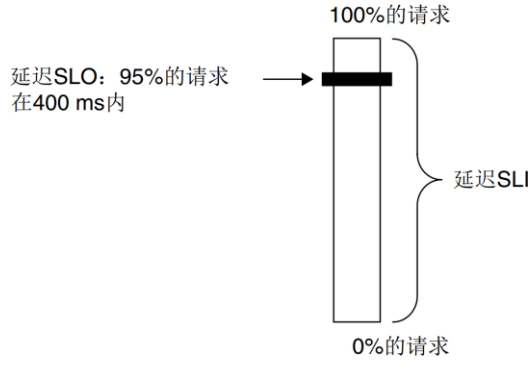

图 3.4　95%延迟 SLO

要将 SLO 的违反情况与客户体验相关联，首要步骤是对客户群体进行明确界定，并基于客户视角设定 SLO。最好是所有相关方——运营工程师、开发人员和产品负责人——可以初步达成 SLO 共识。SLO 的度量应从这一步开始。

如果 SRE 基础设施报告违反 SLO 的情况，而客户仍然满意且无投诉，这可能意味着 SLO 设定过于严格，需要适当放宽。相反，如果客户有投诉而 SRE 基础设施未报告违反 SLO，可能表明 SLO 设定过于宽松，需要收紧。

3.3 错误预算

错误预算（error budget）是一个颇具争议的概念。软件开发通常须避免错误，一旦发现错误或 bug，便需要修复。然而，错误预算似乎与此相悖，它为开发人员设定了一个可以容忍错误的限额，这在直觉上似乎有悖于避免错误的常规目标。为了进一步阐明，我们可以回顾第 3.2 节中对 SLO 的讨论。根据 SLI 设定的 SLO 用于定义能够反映客户满意度的服务水平。

要点 错误预算等于最大服务水平与 SLO 阈值之间的差值。SLO 是基于 SLI 设定的阈值，位于最小服务水平 0%和最大服务水平 100%之间。图 3.5 展示了这一概念。

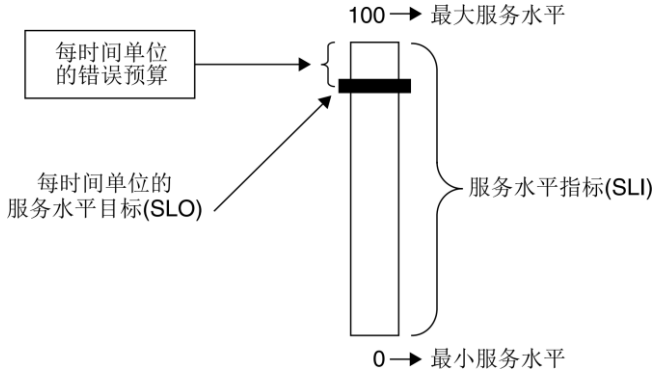

图 3.5　每时间单位的错误预算

错误预算是最大服务水平（例如 100%）与 SLO 阈值之差。与货币预算类似，错误预算是按照时间单位来分配的。这意味着服务端点在时间单位内可以消耗错误预算，但应避免在时间单位结束之前完全耗尽。换句话说，“保持在错误预算内”指的是在指定时间单位内不完全使用完错误预算。

在时间单位结束时，错误预算会得到补足，类似于电脑游戏中的机制，玩家升级后，其虚拟货币（如生命、精力等）会完全恢复。图 3.6 提供了错误预算补足过程的示意图。

无论一个时间单位内的错误预算消耗率（error budget depletion rate）如何，错误预算在每个时间单位的开始都会得到补足。服务端点的目标是在当前时间单位结束前不要过早耗尽给定的错误预算。

时间单位本身可以是一个固定的跳跃性日历窗口，例如，从一年的第一个日历周开始的 4 个日历周的期限。也可以是一个连续的滑动窗口，例如，从当前日期向后的 28 天窗口。

SRE 基础设施应负责跟踪并记录所选时间单位内的错误预算消耗情况。这些数据将用于基于错误预算消耗率做出数据决策。

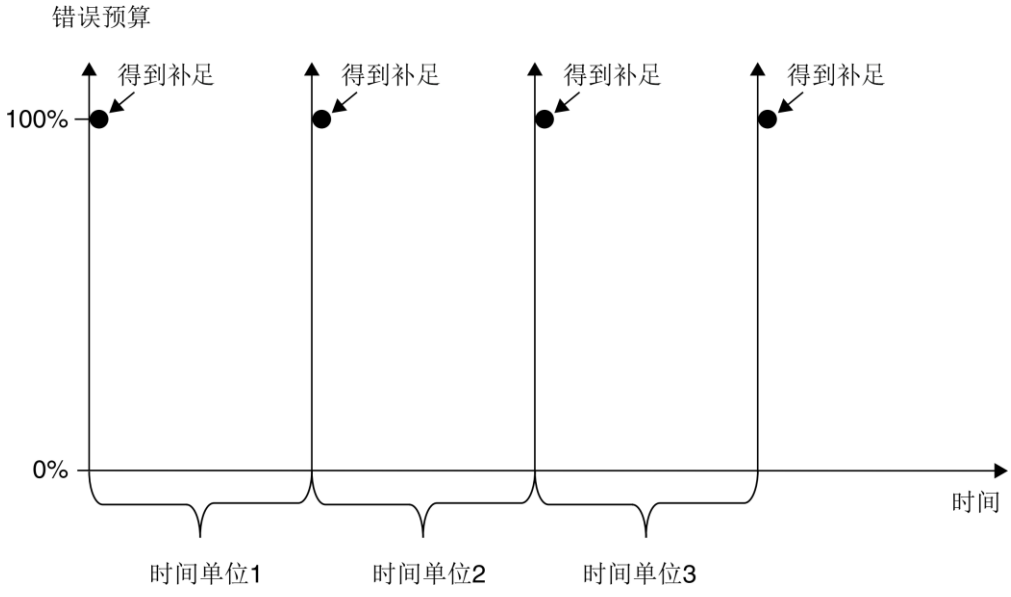

图 3.6　错误预算补足

3.3.1　可用性错误预算的例子

下面举例说明可用性 SLO。在图 3.7 中，可用性 SLO 被定为 98%。因此，错误预算的计算方法如下：

可用性错误预算= 100%可用性 – 98%可用性 SLO = 2%错误预算

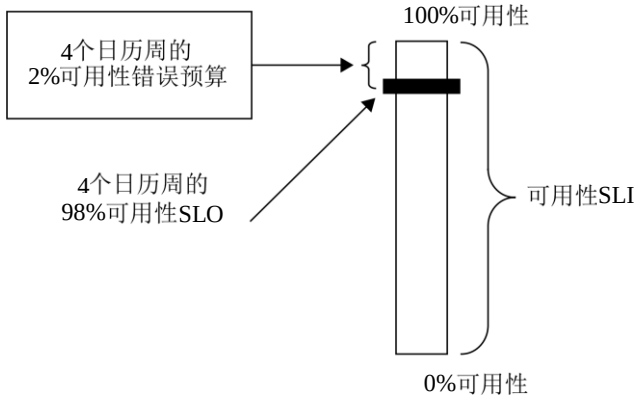

图 3.7　2%可用性错误预算

在本例中，服务端点在 4 个日历周的时间单位内被分配 2%的可用性错误预算。这意味着对于具有该错误预算的服务端点，在 4 个日历周内收到的所有请求中，最多允许 2%请求发生“端点不可用”的情况。

如果服务端点作为公共 API 对外开放，那么错误预算信息就对客户至关重要。基于错误预算，客户应采取适当的技术措施，在服务间预留余量（即建立适应能力），以准备应对潜在的端点不可用极端场景。此外，客户在设定自己的 SLO 时，依赖的服务端点的 SLO 起着重要的作用。这是因为高度依赖的情况下，客户的服务 SLO 通常不能比其依赖服务 SLO 更严格（尽管这有时也是可能的），就像史蒂夫•麦吉尔所说的那样，在可靠性较低的基础之上，仍然可以建立更可靠的系统。[4]

3.3.2 错误预算为零

借此机会，让我们回顾本小节开始所提出的问题：是否应将错误预算设为零以实现服务的持续可用性？如果将错误预算设为零，则相当于将 SLO 设为 100%。在前面的例子中，可用性 SLO 被设为 100%。对服务客户而言，100%的可用性 SLO 是否理想呢？乍一看，似乎如此。

然而，即便达到 100%的可用性 SLO，是否真正意味着服务客户会始终感受到服务的可用性？遗憾的是，并非如此。服务部署在某个地方，服务的客户却部署在别的地方。在服务和服务的客户之间，是一个由许多网络设备和电缆组成的网络，而且，数据包的路由是动态进行的。这些网络组件中的任何一个都可能发生故障。在网络故障的情况下，即使服务本身在其部署的地方实现了 100%可用性，服务客户也无法体验到同等水平的服务可用性。

如果我们暂时忽略网络因素，仅考虑服务的部署点，是否能够实现 100%的可用性 SLO 呢？如果可以，又需要满足哪些条件？100%的可用性 SLO 意味着对应的错误预算为 0%，即不允许有任何出错的余地。对于设定 100%可用性 SLO 的服务端点，在规定的时间内，所有对其发出的请求都是要成功响应的。这要求负责开发和运营服务的团队必须竭尽全力满足 SLO 和错误预算的要求。表 3.2 总结了团队为实现 100%可用性 SLO 可能采取的措施及其目的和潜在的后果。

第一个团队目标是最大化投资于系统的弹性（可伸缩性），确保系统在压力条件下也不至于消耗完错误预算。众所周知，工程师总是致力于技术改进。因此，在追求 100%可用性 SLO 的情况下，他们会投入大量时间来增强系统的弹性（可伸缩性）。根据服务目标，增加工程投入以提升服务对客户的效用可能是合理的。在定义 SLO 时，团队必须在客户实际体验到的服务可用性与团队在部署点上实现的服务可用性之间找到平衡。100%的可用性 SLO 很可能不能为客户增值（以证明为此目标所增加的工程投入是合理的）。

第二个团队目标是增加服务值班人员的数量，以便更快从事故中恢复。更快地从事故中恢复，意味着因为事故而造成的错误预算消耗更少。由于错误预算设为零，因此没有可供消耗的余地。团队需要提供额外的值班人员。同样，根据服务的目的，增加运营投入来提升服务对客户的效用可能是合理的。团队在协商值班设置时，必须在可能减少的事故恢复时间和可用于功能开发的工程时间之间找到另一种平衡。

表 3.2　为了保证零错误预算，团队的目标和行动

#	团队目标	为了满足 100%可用性 SLO 所采取的团队行动	对行动的评估
1	增大系统的弹性，使系统有足够的适应能力来满足 SLO 的要求	实现冗余、分布式系统的稳定模式、自动健康检查、快速故障切换、零停机部署、基础设施的防御性（过度）配置等	客户体验与工程上付出的努力的比例是否合适？客户感受到的服务效用的增强，能否证明服务工程成本的增加合理？
2	增加专门处理事故的人手，通过减少事故恢复的时间来减少错误预算的消耗	实现轮流值班，几个人同时值班，以减少恢复时间	客户感受到的服务效用的增强，能否证明运营服务成本的增加是合理的？
3	减少因生产部署期间或之后的故障而导致的错误预算消耗的可能性	避免在生产中更新服务；避免部署新功能、错误修复和安全补丁	虽然向生产推送更新是故障的一个主要原因，但为了使得服务随着时间的推移一直对客户有用，还是有必要的。一潭死水的服务会失去客户

因此，第一个团队目标和第二个团队目标要求团队（尤其是产品负责人）深入考虑他们期望服务的可靠性水平以及为此愿意承担的成本。第三个团队目标是减少生产环境中的服务更新，以降低故障发生的风险。尽管听起来可能不切实际，但当可用性 SLO 设为 100%时，团队至少会在形式上采取这种做法。确实，致力于开发创新功能的开发团队可能避免将这些功能部署到生产环境中，以维持零错误预算。这表明，在实践中零错误预算没有意义。如果预留错误预算，开发团队将开始采取传统运营团队那样的做法。“千万不要去碰正在运行的系统”，在软件行业，此言不虚也！然而，避免对系统的任何变更可能导致系统逐渐远离客户。设定 100%可用性 SLO 而不预留任何错误预算，显然有悖于产品成功的目的。

要点　设为 100%的 SLO，意味着理论上不允许服务中断，但这并不会直接使得错误预算为零。当错误预算为零时，开发人员不会在生产中更新服务，以免出故障。每次故障都会减少错误预算——错误预算设置为零，意味着根本没有错误预算。不更新服务会导致服务停顿并影响到服务体验。也就是说，错误预算为零有悖于成功交付产品或者服务。

这并不意味着服务端点无法在给定时间单位内运行而不耗尽错误预算。这表明，在实践中，零错误预算是没有意义的。出色的服务往往能够在不耗尽错误预算的情况下持续运行并满足客户的需求。如果没有预留错误预算，开发团队可能无法达成两个目标：

- 实现客户实际经常使用的功能；
- 在不过早耗尽错误预算的情况下推出新的功能并使其服务于客户。

为了成功交付服务，这两个目标必须同时实现。为了激励团队实现这些目标，一开始就要设置错误预算。这是通过将 SLO 设置为一个低于 100%的值来实现的。这个适度的错误预算能让团队更自信地进行频繁且可靠的生产发布。

3.3.3 延迟错误预算的例子

再来看另一个错误预算的例子，即延迟 SLI，如图 3.8 所示。在这里，延迟 SLO 被设定为在 4 个日历周内，对一个端点的 95%的请求在 400 毫秒内返回。

因此，错误预算的计算方法如下：

延迟错误预算 = 100%的请求 – 95%的请求在 400 ms 内= 5%的请求没有 400 毫秒返回时间的上限

错误预算设了 4 个日历周的时间。因此，开发和运营服务的团队目标是不要在 4 个日历周结束前过早用完错误预算。

在充分理解错误预算后，接下来探讨错误预算策略。

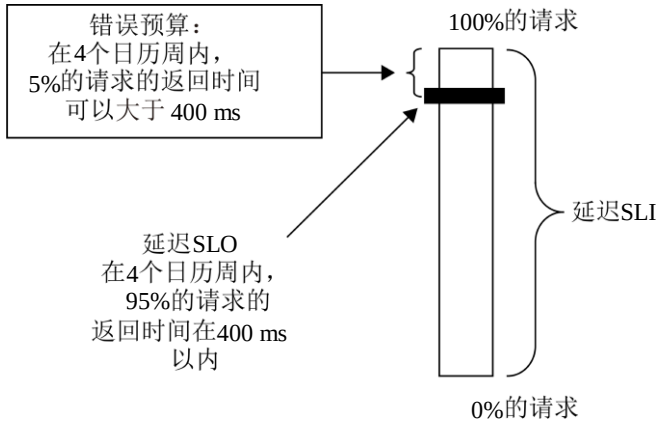

图 3.8 5%延迟错误预算

3.4 错误预算策略

简单地说，错误预算策略（error budget policy）描述了服务过早耗尽其错误预算时服务开发团队和运营团队会采取哪些措施来提高可靠性。在这里，“过早”是指发生于指定时间单位结束之前。服务的计划内事故或计划外事故导致错误预算被大量消耗以至于当前时间单位结束之前已经没有错误预算——换言之，错误预算被过早耗尽了。图 3.9 说明了这种情况。

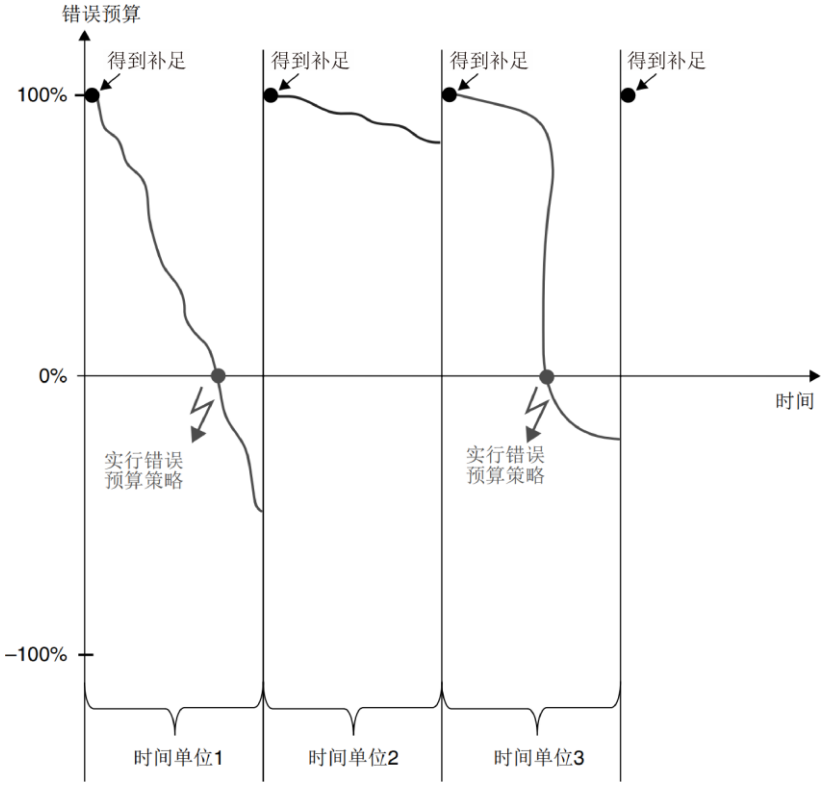

图 3.9　针对错误预算过早耗尽的“错误预算策略”

纵轴代表错误预算。错误预算的范围可以从 100%的全额预算到−100%的预算透支。确实，错误预算透支很常见，尤其是在生产环境中服务可靠性较低的情况下。错误预算透支并不限于−100%，实际上可能超出这一数值，导致更严重的后果。为了不产生错误预算透支，应当在错误预算策略中明确列出团队将采取哪些具体措施来提高服务可靠性。

横轴代表时间轴，划分为三个等长的时间段：时间单位 1、时间单位 2 和时间单位 3。在时间单位 1 中，错误预算消耗得非常快。这意味着有事故不断发生，服务正在消耗错误预算。每次对应的 SLO 被违反，错误预算的一小部分都会被消耗。违反 SLO 的频率越高，错误预算的消耗频率越高。在时间单位 1 的中间位置，大约消耗了 50%的错误预算。在时

间单位 1 的末尾，错误预算继续迅速减少。它消耗得如此之快，以至于在时间单元 1 的 80%时错误预算就已经趋于零。此时，相应的服务端点已经没有任何错误预算了。这时，团队需要采取错误预算策略。策略中应包含具体的措施以便于团队确保错误预算未来不至于被过早耗尽。

在时间单位 2 开始时，SRE 基础设施会自动重新补足错误预算。在整个时间单位 2 中，只有大约 20%的给定错误预算被消耗。这意味着服务中有一些小事故，但它们不会导致错误预算过早耗尽。在时间单位 1 的错误预算消耗率非常陡峭之后，时间单位 2 的错误预算消耗率比较稳定，表明团队根据错误预算策略采取的措施产生了积极的效果。换句话说，实行错误预算策略得当的话，可以提高服务的可靠性。

然而，在时间单位 3 中，错误预算消耗率再次上升。一开始，错误预算消耗率是温和的。然而，在时间单位 3 的 60%左右，错误预算消耗曲线断崖式下跌，在很短时间内从 80%掉到 0。这是典型的重大事故，其中一些大量使用的主要功能突然停止工作。一旦错误预算达到零，就会进入负值区域。团队必须想方设法显著减缓消耗的速度。在时间单位 3 的 90%左右，错误预算的消耗得到控制。时间单位 3 结束时，错误预算的透支约为−20%。

SRE 基础设施应能够生成图表并针对错误预算消耗率发出警报，以便监控和预警。来自 SRE 基础设施的警报应该说明违反 SLO 的原因以及给定时间单位内剩余的错误预算。基于这些信息，接收警报的值班人员可以快速评估分配给警报的优先级。

与金钱债务不同，错误预算透支不会累积，而是在每个时间单位开始时被重置。然而，错误预算透支会对服务的客户产生影响而可能导致客户体验下降，甚至造成财务上的损失。客户可能会“用脚投票”，放弃这样不靠谱的服务。

这就是错误预算策略的意义所在。它以最简单的方式描述运营工程师、开发人员和产品负责人之间的协议——当错误预算过早耗尽时，团队应该做哪些事情。错误预算策略可能包含以下描述：

- 团队将进行无责事后回顾，了解错误预算被过早消耗甚至耗尽的原因；
- 团队将停止向生产部署新的功能，只部署技术可靠性改进措施，直到服务在一段时间内稳定地回到其 SLO 内；
- 团队将审查服务的实现、架构和依赖项，确定应采取哪些可靠性增强措施；
- 团队评估是否需要向监管实体提交监管通知，报告服务的可靠程度及其后果；
- 除非服务在一段时间内稳定在其 SLO 内，否则团队将停止执行生产部署，覆盖部署工具对仍然可用的低水平错误预算的警告。

错误预算策略由团队成员达成的团队协议，适用于团队负责或拥有的所有服务。《Google SRE 工作手册》以案例方式给出了谷歌的错误预算策略。[5]

3.5 SRE 概念金字塔

图 3.10 用一个金字塔模型概括了前面几个小节所讨论的 SRE 概念。在这个金字塔中，每一个较低的层级都必须在到达顶部的层级之前实现。

金字塔中三个较低的级别在《实现服务水平目标》（*Implementing Service Level Objectives*）一书中被称为可靠性堆栈（reliability stack）。[6]《SRE 运维之道》一书描述了若干个公司的 SRE 实现。[7]有的公司（如谷歌）能一直升到 SRE 概念金字塔的顶端。其他公司则停留在较低的级别上。以金字塔的形式说明 SRE 概念，体现了每上升一个级别，采用者的数量就会相应减少。在金字塔中的位置越高，说明行业中的应用越少。

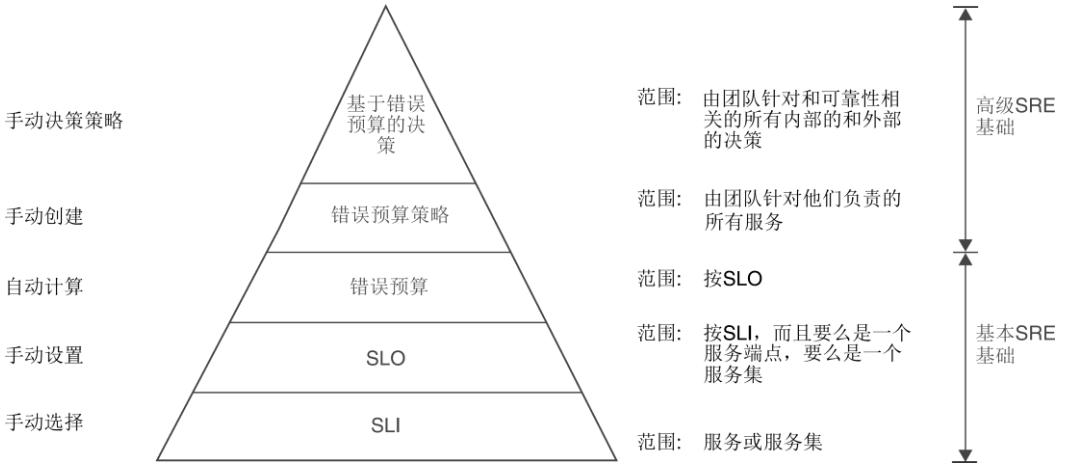

图 3.10　SRE 概念金字塔

这便是推动 SRE 转型的有效方法，自下而上，登上 SRE 概念金字塔并在此过程中稳步进行改进。要想充分体验 SRE 的好处，显然需要全面理解和应用金字塔中的所有概念。

为了方便大家深入理解 SRE 概念金字塔，金字塔左侧给出了指示，用来标明各个概念是依赖手动设定还是自动计算。与此同时，金字塔右侧也提供了对范围的详细说明。

金字塔的基础是服务水平指标（SLI），它们根据服务选择，从客户的角度反映服务可靠性的重点。例如，可用性和延迟普遍适用于所有服务领域，但 3.1 节讨论的其他 SLI，如吞吐量和正确性，则需要根据服务的具体领域进行选择性应用。SLI 需要手动选择，并适用于单个的服务或一成套的服务。

SLI 之上是服务水平目标（SLO），它们在所有 SRE 对话中占据核心地位。SLO 根据 SLI 从客户的角度设定，它确定了运营服务所需要的阈值。服务表现一旦低于或等于 SLO 阈值，客户通常会感到满意；如果超过阈值，客户可能会开始投诉。因此，SLO 作为客户满意度的一个代理度量。根据为服务设定的 SLO，开发和运营团队分配相应的开发和运营

资源。SLO 越严格，为达到规定服务可靠性水平分配的技术时间就越多；SLO 越宽松，分配给新功能开发和推向生产的时间就越多。

SLO 需要手动设置，根据 SLI 被应用于服务端点、单项服务或成套的服务。在某些情况下，可以为单个服务端点、单项服务或服务集设置多个 SLO。例如，对于延迟 SLI，为一个服务端点设置两个不同的延迟 SLO，这样做可能是有意义的。一个 SLO 可能设为较高的延迟阈值，要求大多数请求在此阈值内返回（例如，95%的请求在 700 毫秒内返回）；而另一个 SLO 则设为较低的延迟阈值，要求一部分请求在此阈值内返回（例如，75%的请求在 350 毫秒内返回）。

在 SRE 概念金字塔中，错误预算位于 SLO 之上。错误预算根据 SLO 自动计算，是服务最大可靠性（100%）减去 SLO 阈值的结果。如果为服务端点设定了 SLO，那么错误预算就是该端点在一个单位时间内允许的错误数量。

错误预算的范围取决于 SLO。这意味着，如果一个服务端点设了多个 SLO，那么在该端点的一个单位时间内就有相应数量可以消耗的错误预算。虽然每个错误预算都是独立管理的，但某些故障可能导致多个错误预算同时减少。SRE 基础设施需要对所有错误预算及其消耗水平、速度以及在当前时间单位结束前剩余的水平进行精确跟踪。这些信息应实时提供给值班人员，以便他们迅速确定事故的优先级。所有错误预算都使用统一的单位时间来度量。

SLI、SLO 和错误预算代表 SRE 的基本要素。若缺少这些基本要素，就说明组织并没有真正在做 SRE。在这些基础要素之上，是更高级的 SRE 要素。下面详细讨论这些高级要素。

SRE 概念金字塔的次顶层是错误预算策略。策略由服务的开发团队和运营团队制定，描述错误预算被过早耗尽的时候应该如何响应。错误预算是按单位时间授予的，以免在单位时间结束前耗尽错误预算。如果错误预算被过早消耗或耗尽，团队就会执行错误预算策略，采取措施来提高服务的可靠性。随着服务可靠性的提升，服务未来应保持在错误预算之内。错误预算策略至关重要，因为没有它的指导和实施，其他概念将无法在组织中落地。错误预算策略由团队制定，适用于团队负责或拥有的所有服务。

位于 SRE 概念金字塔最顶端的是基于错误预算的决策（error budget‐based decision‐making）。这个概念超越了错误预算策略（描述团队在服务过早消耗或耗尽错误预算时所采取的措施）。它高屋建瓴，是一种全面的可靠性决策。所有与可靠性有关的内部团队和外部团队都基于团队达到 SRE 成熟度水平之后的错误预算来制定决策。例如，在使用 API 之前，团队要用历史 SRE 数据来了解该 API 的 SLO 并检查它在目标环境中是否能一直保持在其 SLO 之内，然后再用这些数据来实现服务，缓解 API 不可用或响应慢等情况。服务部署完成后，团队还需要验证服务是否满足 API 的 SLO。如果不满足，团队就要核实 API 的 SLO 是否需要收紧。如果需要，就要找到负责或拥有 API 的团队，解释基于错误预算数据的新应用场景并要求他们收紧 SLO。

其他基于错误预算的决策的应用场景是根据预测的错误预算消耗（error budget depletion）来决定要实现哪些技术功能。例如，团队需要考虑是替换缓存还是交换数据库以及这些操作对错误预算的影响。哪个会减少错误预算的消耗？在部署这两种功能时，还剩下多少错误预算？这两种功能的部署方式是否可以使错误预算不至于过早耗尽？这些问题都很好，可以纳入基于错误预算的决策加以考量。

实际上，对错误预算消耗的预测也可以应用到功能部署中。团队需要评估功能部署的停机时间、错误预算消耗量以及部署前后的错误预算余额，以确定最佳部署时机。此外，基于错误预算的决策还包括通过混沌工程实验对不同环境下的错误预算消耗做出假设。一旦团队到达 SRE 概念金字塔的最顶端，不仅使用错误预算策略来指导行动，还可以根据自己和其他团队的错误预算做出所有与可靠性相关的决策。错误预算成为一种普遍的优先级工具，广泛适用于与可靠性相关的所有场景。这样的组织已将其错误预算制度化，在组织的很多层面实践基于错误预算的决策。

SLI	服务可靠性的哪些方面与客户相关？
SLO	服务可靠性的特定方面的阈值是多少？（该阈值是客户的“痛点”，会导致客诉升级）
错误预算	在达到客户的“痛点”，他们开始升级投诉之前，我们还有多少错误空间？
错误预算策略	如果无法在服务的错误预算内运营它以避免客户投诉升级，我们该怎么做？
基于错误预算的决策	我们用哪些数据来做出团队内部和外部的可靠性优先级决策？

在 SRE 概念金字塔中，各种概念通过特定的“基数”[7] 相互关联。如图 3.11 所示。

图的顶部是几个服务的一个集合。每个服务都可以有几个服务端点。每个服务端点、服务或服务集都可以定义几个 SLO。每个 SLO 是为一个 SLI 定义的。反过来说，一个 SLI 可以定义几个 SLO。另外，几个 SLI 可以应用于一个服务。SLO 和错误预算之间是 1:1 的关系。这是因为错误预算根据 SLO 来自动计算。最后，错误预算策略是团队形成的单一协议。一个团队可以实践一种基于错误预算的决策方式，并“负责”或“拥有”几个服务。当然，“基于错误预算的决策”可由几个团队来实践。

深入理解 SRE 概念金字塔之后，2.3.4 节所讨论的组织协同就可以用 SRE 概念来解释。这是下一节的主题。

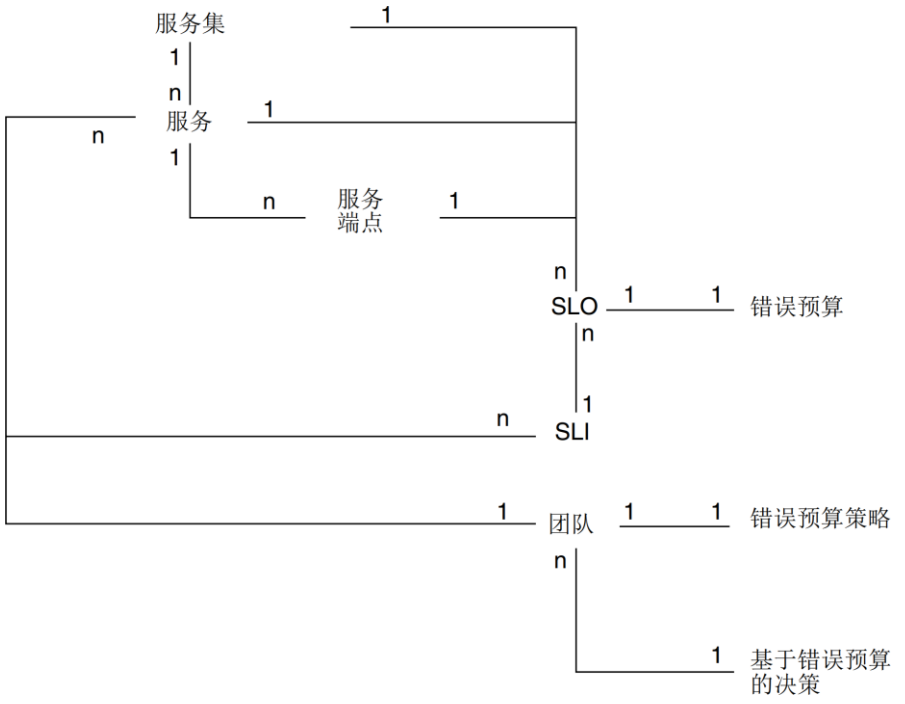

图 3.11　SRE 各个概念之间的关系

3.6　使用 SRE 概念金字塔进行协同

SRE 概念如何协同工作以确保产品交付组织与精益产品运营达成一致？这些概念使组织能够在实际生产部署前建立起协同关系。组织在生产部署前达成协同，是高效产品运营的关键。在生产部署之前，需要达成哪些共识，需要哪些人参与？

图 3.12 展示了在生产部署前需要达成的共识。顶部并排列出 4 个协同点——即要取得共识的目标。第一个协同点是定义与服务相关的 SLI，以客户的视角来度量服务的可靠性。第二个协同点是为每个 SLI 设定符合客户期望的 SLO 以明确可靠性目标。第三个协同点是制定错误预算策略，明确错误预算过早消耗或耗尽时需要采取的应对措施。最后，第四个协同点是建立轮流值班制度，确保服务有持续有效的运营支持。

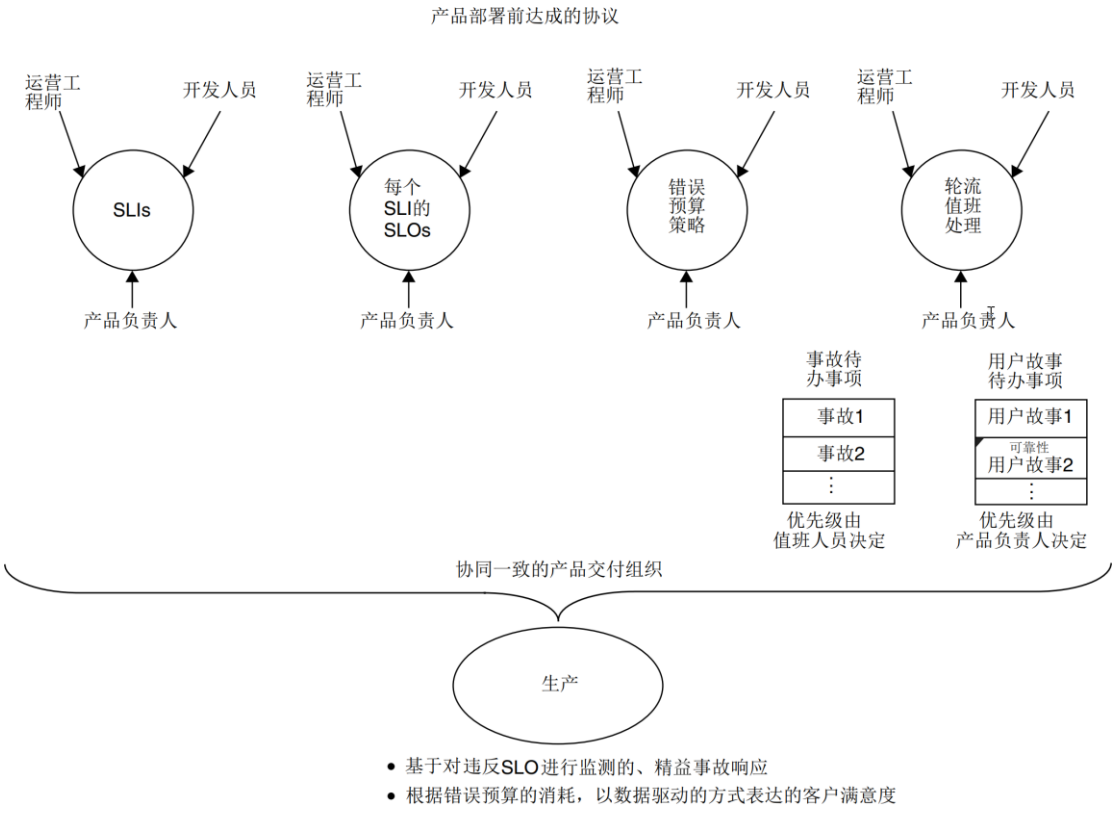

图 3.12　使用了 SRE 概念金字塔的 4 个协同点

这 4 个协同点需要由相关的运营工程师、开发人员与产品负责人达成共识和执行。虽然这看似耗时，但其实不然。也就是说，团队初步达成共识后，对于后续的增量式变革，可以很快达成一致。无论投入时间多少，如果没有组织协同就贸然进入生产部署，极有可能出现 2.1 节描述的那种乱象。所以，在生产部署之前，合理投入时间就运营问题建立组织协同，对团队来说是最好的一项投资，能确保客户有愉快、可靠的产品使用体验。

接下来探讨如何通过 SLI、SLO、错误预算策略以及轮流值班等概念来实现团队协同。

如前所述，运营工程师、开发人员和产品负责人需要共同商议并就 SLI 和 SLO 达成共识。

在讨论过程中，开发人员分享他们在服务与基础设施实现、日志记录、系统追踪和错误处理方面的经验与知识。运营工程师提供他们在生产环境中基础设施的可伸缩性、处理客户投诉、历史事故分析、值班轮换制度以及 SRE 基础设施能力的专业知识。产品负责人则分享他们对用户体验、关键利益相关者的需求、销售与营销的承诺、销售管道、用户最期望的功能、客户反馈处理以及整体产品战略的深入见解。

SLI 和 SLO 的优势在于，能够将各类知识以结构化的方式应用于解决可靠性问题。要想为 SLI 设定合适的 SLO，就必须采取这种方法，恰当地平衡客户满意度与达成这一目标所需要的工程资源。具体过程可参考图 3.13。

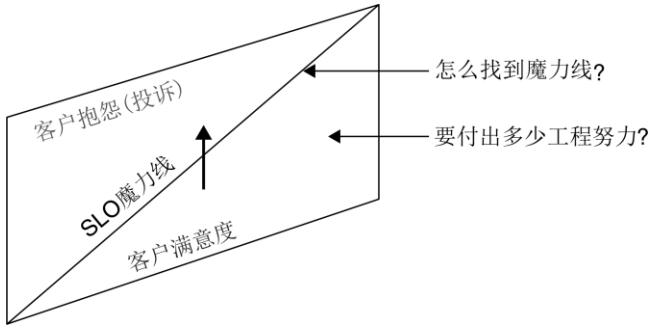

图 3.13　客户满意度和客户投诉之间的魔力线

SLO 魔力线介于客户满意度和客诉之间。当服务表现达到或超过 SLO 标准时，客户满意度通常比较高。相反，如果服务未能满足 SLO，客户可能就会开始抱怨或投诉。在初始阶段，尤其是在数据不够充分的情况下，确定这条 SLO 魔力线的位置需要开发人员、运营工程师和产品负责人共同运用各自的专业知识和集体智慧，通过限定时间的方式进行设定。

至于 SLO 的定义，需要大家进行充分的讨论，尤其是所有人都需要清楚 SLO 设定可能的后果。SLO 的设定对以下几个方面有决定性作用：

- 作为服务或服务端点之客户满意度的代理指标；
- 每个时间单位内可用的错误预算；
- 开发人员为实现服务弹性和可靠性而分配的工程资源和时间；
- 开发人员和运营工程师在轮流值班期间为服务投入的工程资源和时间。

综上所述，设定 SLO 对团队的时间分配有着深远的影响。如图 3.14 所示，SLO 设定越严格，工程投入就越大。

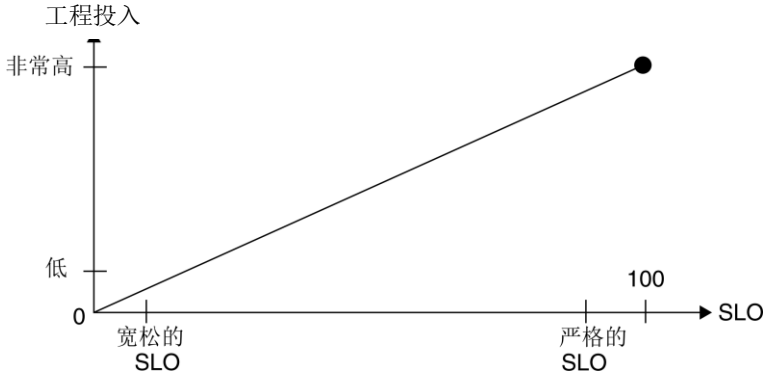

图 3.14　SLO 与工程努力

尽管设定 SLO 对团队的时间分配会产生深远的影响，但团队仍然应该达成共识——SLO 的设定并非不可以更改。最初设定的 SLO 应在生产环境中进行测试，以便发现并解决实际的客户体验问题。对 SLO 的设定应形成以生产为基础的强大反馈循环——从中验证决策的有效性。具体可参见图 3.15。

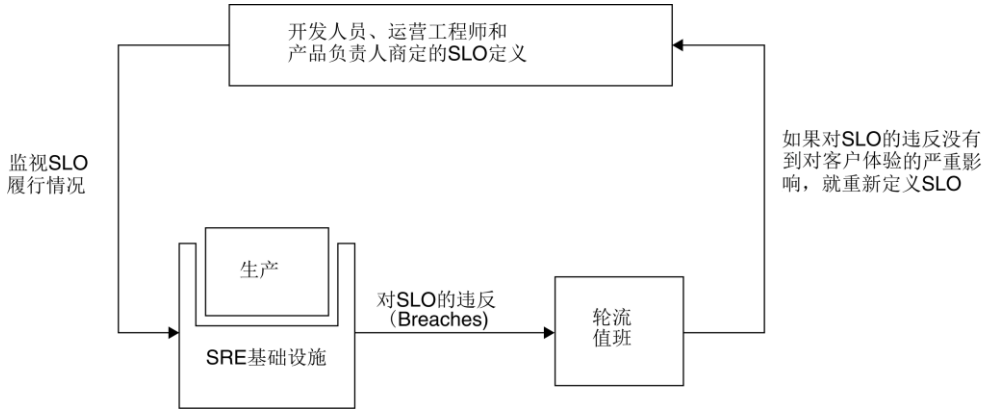

图 3.15　利用来自生产的反馈来测试 SLO 定义决策

由开发人员、运营工程师和产品负责人共同商定的 SLO 定义属于集体智慧，目的是找到那条能够平衡客户满意度与实现这种满意度所需工程努力的 SLO 魔力线。因此，刚开始的时候，我们只能初步假设一条 SLO 魔力线。

接下来是验证假设。这需要 SRE 基础设施在生产环境中监控 SLO 的实现。如果 SLO 被违反，SRE 基础设施就会通知值班人员，由后者来分析具体情况并调查这些情况是否真正严重影响到客户体验。如果 SLO 被违反但并没有对客户体验造成严重的影响，就需要重新定义 SLO 并依据生产中的真实数据和见解来做出相应的调整。重新定义 SLO 继续由运营工程师、开发人员和产品负责人共同进行，并达成共识。

回到前面的图 3.12，值班轮换制度的确立需要三方达成共识，具体的细节可以参见表 3.4。

表 3.4　三方就轮流值班达成共识

要协商的主题	说明
分配人员进行轮流值班的一般策略	一般情况下，谁可以值班？为哪项服务值班？在什么时候值班以及值班多长时间？
事故待办事项的优先级规则	值班人员如何自主对事故进行优先排序？应该使用什么标准来确定事故的优先级？
用户故事待办事项可靠性方面的优先级规则	如何做出基于错误预算的决策，将可靠性工作与客户功能开发工作进行优先排序？错误预算过早耗尽是否会导致可靠性工作的优先级高于客户功能开发？

最后，错误预算策略也需要得到开发人员、运营工程师和产品负责人的三方认可。它需要说明服务过早消耗或耗尽其错误预算的场景下团队会采取哪些措施来提高服务的可靠性，而且该策略适用于服务可能有的所有错误预算。

在生产部署之前，如果已经就 SLI、SLO、错误预算策略和值班达成共识，组织中各个部门就可以齐心协力，为持续提供高满意度的客户体验打下良好的基础。将服务部署到生产中之后，SRE 基础设施会持续检测是否由违反 SLO 的情况。相关信息会被转发给值班人员，由值班人员及时进行优先排序和分析并尽快修复事故。如果发现违反 SLO 却不至于影响到客户体验，值班人员就与运营工程师、开发人员和产品负责人开会，重新定义 SLO。

另外，错误预算消耗由 SRE 基础设施持续监控。当错误预算被过早消耗甚至耗尽时，值班人员会实行错误预算策略，其中包含为提高服务可靠性而采取的措施。在这些措施中，提升可靠性的优先级高于为客户开发新的功能。

与 2.1 节中描述的乱象相比，缺乏 SRE 基础设施和相应的系统知识，不建立违反 SLO 的监控机制、错误预算、策略以及事故和用户故事待办事项的优先级规则，情况可能更糟——场面失控、困惑、重大事故等一连串问题，甚至可能同时发生。

在这样的组织中，如何有效实施 SRE 才能为其赋能呢？这是本书要探讨的核心问题。

3.7 小结

本章探讨了 SLI、SLO、错误预算、错误预算策略以及基于错误预算的决策等 SRE 基本概念。这些概念按金字塔结构排列，引领我们在 SRE 转型过程中持续前行。在接下来的章节中，将正式启动软件交付组织的 SRE 转型之旅。

对于如何引入 SRE、如何组织 SRE、如何运维 SRE，业内尚未取得共识。本书旨在抛砖引玉，探讨我是怎么实施 SRE 的。我期待并欢迎大家提出不同的看法。本书展示了如何系统地实施 SRE 导入项目以推动软件行业实施 SRE。本书的目标帮助大家顺利导入 SRE，尽量少去试错，同时全面概述 SRE 导入项目需要避免哪些坑。

本书的主题是从头构建 SRE 基础设施，因而聚焦于基础，并不涉及太高级的 SRE 实践，其他书有这方面的介绍。“守破离”（Shu Ha Ri）是源自日本剑道的一种哲学，阐述了通过学习来逐步精通技艺的不同阶段。本书主要关注“守”这个阶段并在一定程度上探讨‘破’，但肯定不涉及“离”阶段。

大致了解 SRE 各个学习阶段之后，我们要从软件交付组织现状评估开始启程 SRE 转型之旅了。

注释

1 译注：错误预算是指技术系统在不产生约定后果的情况下可以出现故障的最长时间。

2 Murphy, Niall Richard, Betsy Beyer, Chris Jones, and Jennifer Petoff. 2016. *Site Reliability Engineering: How Google Runs Production Systems*. Sebastopol, CA: O'Reilly Media

3 Rundeck. n.d. "SRE for Everyone: Making Tomorrow Better Than Today." SlideShare IOS. Slide 33. https://www.slideshare.net/Rundeck/sre-for-everyone-making-tomorrow-betterthan-today-devops-days-austin-2019

4 McGhee, Steve. 2021. "SLO Math." YouTube, May 16, 2021. https://www.youtube.com/watch?v=-lHPDx90Ppg

5 Beyer, Betsy, Niall Richard Murphy, David K. Rensin, Stephen Thorne, and Kent Kawahara. 2018. *The Site Reliability Workbook: Practical Ways to Implement SRE*. Sebastopol, CA: O'Reilly Media.

6 Hidalgo, Alex. 2020. *Implementing Service Level Objectives: A Practical Guide to SLIs*, SLOs & Error Budgets. Sebastopol, CA: O'Reilly Media.

7 Blank-Edelman, David N. 2018. *Seeking SRE: Conversations about Running Production Systems at Scale*. Sebastopol, CA: O'Reilly Media.

第 4 章

评估现状

假设有这么一家产品交付组织，其场景就像 2.1 节描述的那样。在这样的组织中，开发人员可能并不理解自己为什么要参与运营工作。运营工程师可能不会为开发人员提供框架来支持他们参与运营。管理者不会倡导运营和开发协作，更不会为此提供必要的经费。然而，共同的目标——减少客诉升级——可能是让组织团结在一起的关键。

组织中有一些人可能知道 SRE 是掌控生产运营的关键，因而可能有一个非常小的群体想要实现 SRE。他们可能会在某个时候成为 SRE 教练。组织 SRE 转型的第一步是准确评估组织的基本面。为此，有几个重要的维度需要分析，其中包括组织机构及其人员、技术、文化和过程。接下来依次讨论每个维度。

4.1 组织现状

就组织而言，需要考虑以下几个方面：

- 组织结构；
- 组织协同；
- 正式领导和非正式领导。

4.1.1 组织结构

要看清组织现状，首先需要明确产品运营是如何根据组织结构图中代表各部门和团队的方框来组织的。可以通过以下问题来指导组织现状分析：

- 组织结构图中哪些方框中的人有生产运营的责任并为生产运营中出现的问题承担管理责任？组织结构图是否清楚描述了这一责任？
- 图中是否包括实线和虚线来显示组织中的报告层级？
- 领导团队是否与实线和（如果有的话）虚线报告保持一致？领导团队还有其他的人员吗？

- 组织结构图中的哪些方框实际负责产品的运营？
- 哪些方框参与热修复补丁的交付？
- 哪些方框负责决定热修复和可靠性工作的优先级？

SRE 教练首先需要了解组织现状。组织当前是如何应对生产运营的？基于这种理解和 SRE 引领组织走向协同的愿景（参见 1.2 节），组织的 SRE 转型路径将逐渐明朗起来。

通过分析可能发现，生产运营的责任在组织并没有明确的定义。然而，更常见的情况是，生产运营的工作职责与管理层的责任分配不一致。这些负责人可能缺乏足够的控制力来有效执行工作。

在传统的软件交付组织中，通常会成立产品开发、产品运营和产品管理部门，产品运营部门要承担生产运营的正式责任和管理责任。然而，由于组织的筒仓式结构，这样的责任很难得到落实。图 4.1 描述了这种情况。

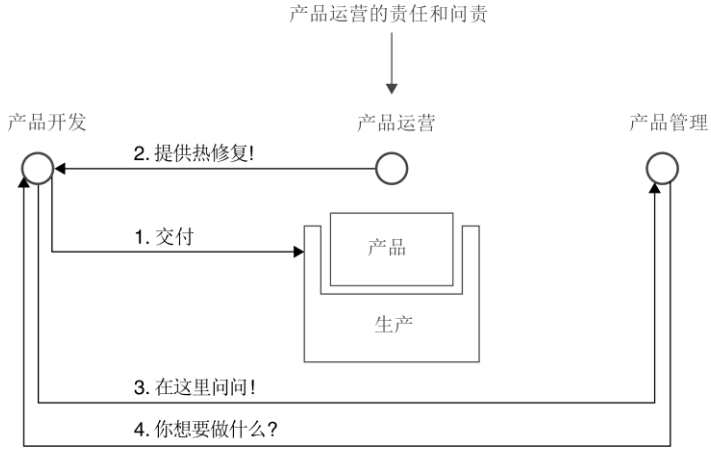

图 4.1　未就运营问题协同的组织

产品运营团队负责运营，是生产环境的责任人。产品开发团队负责把产品交付到生产环境。对产品运营来说，因为缺乏产品实现的细节，就只能用一些可以反映 IT 资源的参数来监控产品。一旦生产环境出现严重问题，产品运营团队就要求产品开发团队提供热修复补丁，此时开发团队如果正在完成产品负责人制定的高优先级用户故事待办事项，就需要重新确定优先级，必须先完成热修复补丁。优先级由产品负责人确定，所以开发团队就会回复运营团队，称自己需要先问产品管理部门如何调整待办事项的优先级。然而，产品管理部门并不在这个反馈循环中，所以他们的首要任务是摸清现状，然后再商讨如何设置待办事项的优先级。

由此可见，产品运营团队并不能完全掌控生产运营，以至于基本上无法履行自己的职责。如 2.2 节所述，生产运营的责任由产品开发、产品运营和产品管理部门共同承担，三方各司其职并从中受益。

4.1.2 组织协同

接下来，深入了解组织协同。无论正式的组织结构如何规定，都要先探究组织当前究竟是如何执行生产运营的？尽管生产运营可能交接不畅，但仍然需要明确当前的运营方式和状态。可以借助于以下问题来展开分析：

- 客户支持是如何进行的？
- 如果有的话，组织中谁来提供一级支持？
- 如果有的话，组织中谁来提供二级支持？
- 如果有的话，组织中谁来提供三级支持？
- 是否存在三个以上的支持级别？有多少？谁负责提供每一级的支持？
- 客户支持请求通过各种支持级别的典型路径是什么？
- 在过去 12 个月里，客户支持请求数量的趋势是什么？
- 在过去 12 个月里，客户支持请求的平均处理时间是多少？
- 版本发布和全面推出过程是如何进行的？
- 推出热修复补丁的决策是如何做出的，由谁做？
- 推出功能发布的决策是如何做出的，由谁做？
- 在热修复或功能发布中，是否涉及监管合规？
- 谁负责制定发布计划和推出计划？
- 进行生产推出需要进行怎样的协调？
- 是否有发布经理来协调生产推出工作？
- 日常是否有团队或个人能在开始走客户支持流程前检测到生产中的问题？

在这里，目标是了解组织内各方目前处理运营问题时如何协作。这包括客户支持、发布、热补丁推出、功能推出和主动生产监控等。需要了解这些流程和活动是如何相互作用的，暂时不考虑它们的效果和效率。

基于这样的理解，可以开始思考如何推进 SRE 转型。如何解决组织当前运营中各个问题以便为 SRE 转型奠定基础。

通过分析组织现状，我们可能注意到过多的客户支持级别可能导致客户支持请求的处理时间变长。此外，支持级别越多，开发团队越不可能基于极致用户体验来实现全部功能，因为他们基本上不参与运营而不可能将相关认知纳入功能开发的考量中。另外，支持级别越多，消息在传递给开发团队的过程中越有可能衰减或者丢失。在极端情况下，开发团队

如果不值班的话，基本上没有意愿在产品实现过程中兼顾运营团队的问题，因为他们更注重开发客户所看重的功能。为此，SRE 转型要想取得成功，必须鼓励开发团队适当参与值班，以便在服务实现过程中兼顾运营问题。

要点 SRE 转型的艺术在于确定开发团队参与值班的合理程度，以最大限度地学习和激励，使他们能在服务开发过程中兼顾运营问题。具体参与程度在每个组织中是不同的。

分析可能还表明，热补丁和功能的发布和推出涉及多个团队与人员的协作。从运营角度来看，快速发布和推出热补丁尤为重要。对当前过程有更好的了解后，才可能确定 SRE 转型期间需要怎么做以加快热修复补丁的发布和推出。特别是受监管的行业，任何生产发布和推出都必须合规。在 SRE 转型过程中，可能需要利用自动化技术来确保监管合规并加快发布和推出服务。

4.1.3 正式和非正式领导

在评估组织的运营现状时，领导层的作用不可忽视。领导可以分为正式和非正式两种。正式领导由 4.1.1 节讨论的组织结构来确定，因此，看看组织结构图，就会明白正式领导是谁。

与此同时，非正式领导也在组织中扮演着重要角色。他们虽然没有正式的权力，但对组织成员有较大的影响。一些非正式领导甚至可能比正式领导影响力更大，特别是从第一性原理[1]来解释采取某种行动的理由时。他们通常是出色的沟通者，而且之所以影响力大，并非依靠权力，而是通过逻辑和情感来赢得同事的信任。这种做法增强了他们的真实性和可信度。人们选择跟随非正式领导，因为他们真的相信他们提出的行动方案。

以下问题有助于分析组织中的正式领导和非正式领导：

- 组织中哪是正式的领导？
- 组织中哪些人是非正式的领导？他们的影响力发挥在哪些领域？
- 组织中哪些人是公认的专家？他们擅长哪些领域？
- 哪些正式领导在组织中被认为是优秀的？
- 总的来说，组织中人们听从哪些人？不信服哪些人？
- 在处理所有生产事故中发挥关键作用的是哪些人？他们是化腐朽为神奇的英雄。

这些问题的回答对 SRE 转型至关重要，因为所有的正式领导和非正式领导都需要适当参与转型。在非正式领导中，有些人可能适合担任教练以推动 SRE 实践。正如 2.5 节所述，SRE 教练要在组织内部进行培养。因此，要想推动 SRE 转型成功，就要利用一切机会物色合格的 SRE 教练。

通过对组织结构、协同和领导的深入了解，我们能够清晰地认识到组织如何执行生产运营。此外，这个过程还能帮助我们思考如何以 SRE 为核心来改进生产运营。

4.2 人员现状

接下来，探索组织中人员的知识、心态和态度。对于运营现状以及应该如何执行运营，大家有哪些看法？下面这些问题有助于摸清人员的现状。

针对运营工程师和运营经理：

- 日常工作有哪些？
- 产品的总体质量如何？
- 如何决定设置哪些警报？
- 知道这些警报是否报告了实际存在的用户体验问题？
- 一般值班吗？如果是，会是什么时候？
- 团队是否有轮流值班制度？
- 如果需要发布热修复补丁并在生产环境中推出，会怎么做？
- 开发人员如何参与生产运营？
- 产品负责人如何参与生产运营？
- 是否提供了一种手段，使其他人能够自己参与生产运营？
- 组织中有哪些客户支持级别？
- 组织中谁在负责哪个支持级别？
- 生产中的危机是如何管理的？
- 客诉数量在一段时间内的趋势如何？

针对开发人员、 架构师和开发经理：

- 如何衡量产品的可靠性？
- 如何决定在产品中实现哪些针对可靠性的功能？
- 如何确定可靠性功能的优先级？
- 谁来确定可靠性功能的优先级？
- 如果需要推出热修复补丁，会怎样？
- 团队自己做生产部署吗？
- 是否有过值班？如果有，是什么时候？
- 团队是否有轮流值班的制度？

针对产品负责人和产品经理：

- 产品愿景是什么？
- 产品在实现这个愿景方面目前处于什么位置？
- 谁是产品的用户？
- 谁是产品的客户？
- 对于用户和客户来说，最重要的用户旅程是什么？
- 产品的可靠性对用户和客户的重要性如何？
- 客户投诉数量在一段时间内的趋势如何？
- 如何进行待办事项的优先级排序？
- 如何进行可靠性功能的优先级排序？
- 是否参与了生产运营？

针对副总裁和管理层（高管）：

- 针对产品的可靠性，客户是如何报告的？
- 组织如何管理生产运营？
- 组织是否为良好的生产运营进行了协同？
- 是否有可能为团队分配一些时间进行 SRE 转型？
- 产品交付组织的经费情况能否支持实现所需的产品可靠性？
- 是否分配了一些预算给额外的工具以改善生产运营？

可以肯定的是，要想摸清人员现状，只需要对组织中极少数人进行采访。在 SRE 转型的这个阶段，我们不鼓励大型研讨会和冗长的会议，因为这样做可能使项目看似艰难而导致人们不愿意支持 SRE。

回答完这些问题，基本上就有了足够多的背景知识，了解不同岗位对生产运营的感受。基于这种理解，思考 SRE 转型过程中需要如何转变心态。

4.3 技术现状

为了充分了解组织的现状，还要评估技术，从技术层面检视生产运营所涉及的所有技术。评估依据是米奇 • 迪克森在《SRE：Google 运维揭密》[2]一书中提出的“服务可靠性层次结构”。如图 4.2 所示，该层次结构描述了实现服务可靠性需要具备的条件及其采用的顺序。

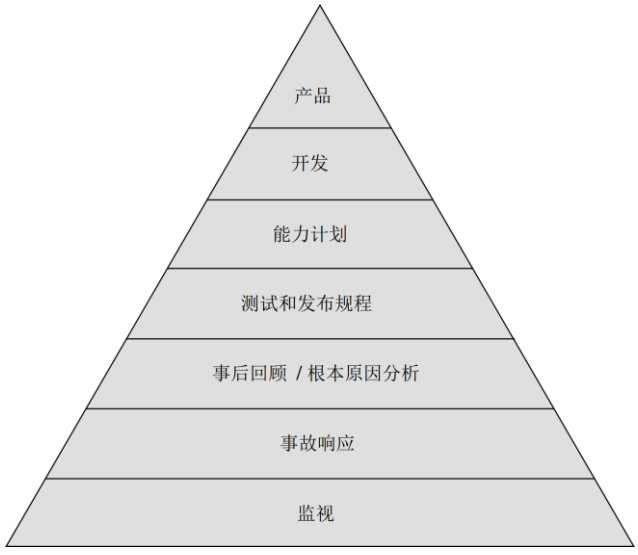

图 4.2　服务可靠性层次结构

在服务可靠性层次结构中，技术方面包括监视、测试和发布程序、能力计划和开发。进行技术评估时，要留意以下问题并逐一进行检视。最好以团队形式进行技术评估，因为团队往往以一种独特的方式来管理技术。不过，在 SRE 转型的这个阶段，只需要采访少数团队中的少数人。对每个团队单独进行教练留到以后进行。目前，有一个总体的理解即可。这种理解旨在评估 SRE 转型的复杂性、持续时间和成本。在对技术进行评估时，可以考虑以下问题。

1. 日志和监视问题

- 服务是否有日志记录？
- 日志是否以统一的方式进行吗？
- 所有服务是否使用相同的日志基础设施进行日志记录？
- 服务是否通过生产部署将日志记录到日志基础设施的单一实例中？
- 日志基础设施提供了哪些开箱即用的工具（例如，运行时依赖关系图、调用持续时间等）？
- 日志基础设施是否提供了一种查询语言来程序化的方式查询日志并将查询结果制成图表？
- 日志有人看吗？在哪些情况下看？
- 是否有任何关于日志的指标？
- 是否基于日志和指标生成一些警报？
- 异步操作如何进行日志记录？
- 根据日志来分析异步操作有多容易？

- 是否用到了分布式跟踪？

2. 测试问题

- 有没有针对服务的测试？
- 哪些测试是手动的？
- 对于自动化测试，对服务进行测试有哪些级别？
- 测试套件的自动测试执行的触发器或节奏是什么？
- 自动测试套件的测试执行时间是多少？
- 手动测试套件的测试执行时间是多少？
- 哪个部署环境执行了哪个测试套件？
- 在功能发布前执行了哪些测试？
- 在热补丁发布前执行了哪些测试？
- 在部署完成后，是否会运行自动部署检查来验证部署程序？
- 自动化测试是否在生产中全天候运行？
- 测试和测试的执行是否值得信赖？

3. 发布问题

- 开发团队是否负责或拥有部署管道？
- 部署管道的范围是什么？通常使用一个部署管道部署多少个服务？
- 开发团队是自己进行生产推出还是由运营团队集中进行？
- 进行生产发布的时候，需要哪些手动步骤？
- 哪些手动步骤可以确保生产发布的监管合规（文档创建、审查和签名、审批等）？
- 每个团队平均多久进行一次生产发布？
- 是否有金丝雀发布过程3？
- 向所有生产环境推出一项服务需要多长时间（包括金丝雀发布）？
- 是否有标准作业程序（Standard Operating Procedure，SOP）来描述如何在生产中完成标准化工作（例如，如何手动扩充内存等资源）？

4. 能力计划问题

- 服务是否在组织拥有的服务器上运行？是基础设施即服务（IaaS）还是平台即服务（PaaS）解决方案？
- 是否有几个服务在一个共享的基础设施单元中运行而只能将该单元作为整体进行扩展？
- 运行在容器中的服务是否部署在集群中，如 Kubernetes 集群，使基础设施的扩展可以由容器按需完成？
- 基础设施的扩展是手动、自动还是半自动的？

5.开发问题

- HTTP状态码的使用是否恰当？例如，500错误码是否真的意味着内部服务器出错了？
- 日志基础设施是否提供了一种日志查询语言？如果是的话，开发人员是否经常使用该日志查询语言？
- 服务是否建立在 12 要素应用（Twelve-Factor App）[4]的原则之上？开发人员是否基本了解这些原则？
- 服务是否实现了《发布!》[5]一书所描述的分布式系统的任何稳定性模式？开发人员是否大体了解这些模式？

有趣的是，上述问题与团队使用的技术本身并没有密切的联系。分析团队的技术现状时，应更多关注概念性因素而非具体的技术细节。例如，使用哪种日志基础设施并不重要。重要的是，团队是否以一致的方式使用所有日志基础设施。如果是，就可以建立一个统一的 SRE 基础设施，让所有团队从中受益。如果不一致，就必须先取得一致（这是实施 SRE 转型的前提）。

另外，还需要知道所用的日志基础设施是否支持程序化查询。具体使用的查询语言并不重要。关键是得有日志查询语言，因为它可以使 SRE 基础设施的某些部分更容易建立。如果值班人员能快速以程序化的方式查询日志，就能以更快的速度对事件做出响应。

对于测试，重要的是测试的可靠性和效率，而不是具体使用的测试框架和工具。需要关注测试结果的可信度、测试运行的速度、在什么环境下执行以及针对的是哪个版本。基于这些因素，可知可以以多快的速度把变更部署到生产环境。如果存在瓶颈，必须在 SRE 转型期间予以解决。

发布、能力计划和开发问题也不例外。具体使用什么发布框架不重要。例如，关键在于是否做了金丝雀发布，先测试生产中的变更对少数用户群体的影响。如果做了金丝雀发布，团队发布过程就会更完善。在 SRE 转型过程中，基本上不需要重点关注这些细枝末节的问题。

虽然某些问题看似无足轻重，但对 SRE 却有重大的影响。例如，在所有服务中使用正确的 HTTP 错误码就非常重要。因为服务可用性的计算是基于 HTTP 错误码进行的。错误的服务可用性计算会随着时间的推移逐渐积少成多，导致利益相关者感到困惑。团队可能为了提高可用性而把资源错投到并不重要的地方。API 的消费者可能采取不恰当的措施来提高可靠性，试图避免 API 退化可能带来的影响。

综上所述，明确前面讨论的问题有助于充分理解提升技术水平来适应 SRE 的重要性。毫无疑问，这是需要付出努力的，具体取决于团队的具体情况。这也表明 SRE 活动的优先级需要在组织的项目组合层面上进行排序。如果不明确 SRE 活动在组织中的优先级，SRE 转型可能无法取得成功。其结果可能是在多个地方进行局部的增量改进，但 SRE 的目标是以系统化的方式协同整个软件交付组织的运营（参见 1.2 节），为了实现这个目标，技术达标是唯一的前提。

4.4 文化现状

众多书籍和出版物中，都认为文化是影响组织各个方面的重要因素。SRE 及其转型特别受组织文化的影响。SRE 的推行速度并不取决于项目计划、行政人员的意愿或 SRE 教练的一厢情愿，在很大程度上由组织文化决定。

如此说来，什么是组织文化呢？它是如何影响 SRE 转型的？现在，我们要深入探讨这些问题。

《朗文词典》将“文化”定义为“一个特定社会中人们共同接受的信仰、生活方式、艺术和习俗。”[6]社会学家罗恩·韦斯特鲁姆提出了一个流行的组织文化拓扑结构，即韦斯特鲁姆模型（Westrum），[7]该模型根据组织处理信息的方式将文化划分为病态型、官僚型和生机型。病态型组织文化以权力为导向，官僚型组织文化以规则为导向，而生机型文化以绩效为导向。根据 DevOps Research and Assessment（DORA）的评估，[8]以绩效为导向的生机型文化能够促成高效的软件交付。

根据该模型，生机型组织文化涵盖以下六个方面：高度合作、训练信使、风险共担、鼓励交流、失败时追根溯源和接纳新的想法。所有这些方面都与 SRE 直接相关，表 4.1 对此进行了总结。显然，以绩效为导向的生机型文化能够带来 SRE 高效转型。

表 4.1　韦斯特鲁姆生机型文化和 SRE 的关系

	韦斯特鲁姆生机型文化	和 SRE 的关系
1	高度合作	SRE 的目的是使软件交付组织在运营方面的问题上保持协同。只有产品开发、产品运营和产品管理之间的高度合作，才能实现
2	训练信使	其服务过早耗尽错误预算的服务负责人需要接受培训，以实现可靠性措施来保持错误预算。解决事故的值班人员在进行无责事后回顾时应得到支持，这应被视为组织中每个人学习可靠性的宝贵机会
3	共担风险	产品运营、产品开发和产品管理需要就 SLI、SLO、错误预算和轮流值班达成一致（3.6 节），从而共担联合决策的风险
4	鼓励交流	服务 SLO 和错误预算随时间推移的消耗率需要公开，以便在团队之间建立对话，针对所依赖的服务的可靠性问题，形成由数据驱动的决定。此外，在 SRE 转型期间，需要促进团队之间的定期交流，例如， SRE 实践社区、午餐会和学习会议
5	失败时追根溯源	事故解决后，需要进行无责的事后回顾
6	接纳新想法	从生产运营中获得的新见解要使可靠性功能能够在产品中及时实现

可以从生产运营的视角评估组织文化的现状，分析距离表 4.1 描述的行为还有多大的差距。这种评估有助于理解组织文化对 SRE 转型的潜在影响，并为实现更高效的 SRE 实践提供指导。通过深入了解和积极塑造组织文化，可以为 SRE 转型创造一个更好的环境。

4.4.1 是否高度合作

在生产运营领域，产品开发和产品运营之间有一面臭名昭著的部门墙。墙的一边是产品运营，其目标是保持生产稳定。基于这个目标，他们并不希望生产环境频繁变动，因为每一次变动都可能导致不稳定（或根据他们的经验，通常是这样）。墙的另一边是产品开发，其目标是尽快实现、部署和发布产品管理部门要求的新功能。DevOps 试图打破壁垒。SRE 也不例外，它是 DevOps 的具体实现。

可以通过以下问题来探索组织在运营问题上的合作质量：

- 产品运营和产品开发的关系如何？
- 产品运营和产品开发的关系是否紧张？如果是，根源在哪里？
- 产品管理参与生产运营的情况如何？
- 产品管理和产品运营之间是否有工作上的联系？
- 运营工程师对产品的可靠性怎么看？
- 运营工程师对可靠性工作的优先级怎么看？
- 对于刚发生的生产事故，让不同开发团队的人按要求参与有多容易？
- 是否有产品运营、产品开发和产品管理三方都认同的 SLI 和 SLO？
- 是否有产品运营、产品开发和产品管理部门都认同的错误预算策略？
- 错误预算策略是否会在错误预算过早消耗的时候实行？
- 轮流值班设置是否已经成为产品运营、产品开发和产品管理三方的共识？

4.4.2 训练信使

可以通过以下问题来探索是否以及如何共担运营风险：

- 团队是否因生产事故而受到管理层打击？
- 生产事故发生后，是否会做回顾或者复盘？
- 人们是否害怕在回顾或复盘中受到指责？
- 提出可靠性问题的人是否被忽视？
- 组织中是否有人成为生产缺陷的背锅侠？
- 错误预算过早耗尽的话是否安排相应的可靠性培训？
- 事后回顾或者复盘是否被视为了解可靠性的机会？

4.4.3 是否共担风险

可以通过以下问题来探索是否以及如何共担生产运营的风险：

- 生产运营的责任是否有明确的描述？
- 运营工程师是否了解生产运营的书面描述？开发人员和产品负责人呢？
- 开发人员是否共担生产运营的风险？如果是，又如何共担？
- 产品负责人是否共担生产运营的风险？如果是，又如何共担？
- 产品运营、产品开发和产品管理是否共同决策定并共担有关 SLI、SLO、错误预算策略和轮流值班设置的后果？

4.4.4 是否鼓励交流

可以通过以下问题来探索组织内团队和个人如何沟通运营问题：

- 组织中是否任何人都可以随时查阅事后回顾报告？
- 如果随机挑一个开发人员，他会知道事后回顾报告存放在哪里吗？
- 如果随机挑一个产品负责人，他会知道事后回顾报告存放在哪里吗？
- 事后回顾报告是不是从用户影响的角度出发并以组织中非技术人员都能理解的方式来写的？
- 是否定期与更广泛的人交流事后回顾中的内容（如精益咖啡）？
- 是否有人阅读事后回顾并从中学习？
- 是否有针对运营——或更准确地说，SRE——的实践社区（Community of Practice，CoP）？
- 运营工程师是否参与了任何产品创建活动（即在产品投入生产环境之前）？
- 产品运营、产品开发和产品管理是否定期交流，探讨如何完善 SLI、SLO、错误预算策略、基于错误预算的决策和轮流值班设置？

4.4.5 失败后是否可以追根溯源

可以通过以下问题来探索组织内如何处理生产运营中的故障：

- 热修复补丁是否会导致管理层对团队和个人进行惩罚？
- 谁来发起事后回顾？
- 是否有一套明确的标准规定只有发生特定事故才能启动事后回顾？
- 参加事后回顾的人在心理上觉得安全吗？他们怎么知道是心理安全的？

- 是否有来自事后回顾的行动事项来促成对 SLI、SLO 和错误预算策略的重新考虑？如果有，这些行动事项如何跟进？由谁来负责？

4.4.6 是否接纳新的想法

可以通过以下问题来探索组织如何处理生产运营的相关新想法：

- 有了服务过早耗尽错误预算的统计数据后，是否会在组织中找人背锅？
- 有了服务过早耗尽其错误预算的统计数据后，是否会基于先前商定的错误预算策略来决定用户故事待办事项的优先级？
- 生产版本是否被看作是一个可以用来测试之前既定功能假设的实验机会？
- 从一个生产版本中学习，以便为下一个生产版本的开发提供见解，有这样的过程吗？这个过程是否是结构化的？
- 在团队的用户故事待办事项中，事后回顾中发现的技术创新排入优先级的平均时间是多少？

通用和丰田合资企业 NUMMI 的约翰·舒克在公司内部进行了一次重大的文化转型。他在文章“如何改变文化：NUMMI 的教训”[9]中如此描述：“改变文化的方法不是先改变人们的思维方式，而是先改变人们的行为方式——也就是他们做什么和不做什么……”基于此，SRE 转型需要致力于引入 SRE 的工作方式。一旦开始以不同的方式做运营，组织文化就会随着时间的推移而改变。不要指望这是一个一蹴而就的过程。但无论速度快慢，都需要稳定，由 SRE 教练定期与所有团队合作。

4.5 过程现状

产品交付组织最后要评估的维度生产运营过程。这个过程由许多子过程组成，涉及若干个团队。以下问题有助于了解该过程及其子过程的现状：

- 客户支持
  - 客户支持请求如何传达给开发团队？
  - 客户支持请求如何传达给开发团队？
  - 客户支持和开发团队之间的接口是什么？工单？定期会议？ChatOps？
- 值班过程
  - 服务是否有人值班？
  - 值班覆盖率是多少？是 24/7 吗？
  - 是否安排了轮流值班？如果有的话，

        - 哪些角色负责？
        - 交班时，知识交接如何组织？
        - 每班是否有主要和辅助值班人员？
        - 轮班的典型持续时长是多少？
    - 是否有运行手册？如果有，谁负责更新？
- 事故响应
    - 对于涉及若干个团队的事故，是否有人对事故进行指挥？
    - 如何根据事故类型来指派合适的人选？
    - 事故得到解决的平均时间是多少？
- 生产访问控制
    - 是否要由人进行生产访问？如果需要，
        - 谁有访问权限？
        - 如何申请访问权限？
        - 在提出申请后，如何快速提供访问权限？
- 监管合规
    - 对于已经部署的服务，健康检查的频率如何？
    - 进行生产部署的时候，需要用到哪些工件？
- 生产部署
    - 团队能否以自主的方式自行安排生产部署？
    - 在进行生产部署之前，是否需要通知利益相关者或客户？
    - 生产部署期间是否有停机时间？平均多长时间？
    - 生产部署的平均频率是多少？
    - 生产部署失败是否有一个明确的定义？
    - 在单位时间内，生产部署失败的平均值是多少？
    - 生产失败的平均恢复时间是多少？
    - 为了完成生产部署，是否需要手动步骤？
- 生产发布
    - 面向客户的生产发布与数据中心的生产部署是否是脱钩的？如果是，谁可以向客户发布以及代表谁发布？
- 热修复补丁部署
    - 热修复补丁和功能部署在部署过程上有哪些区别？
    - 为了完成热修复补丁的部署，是否需要手动步骤？
- 热修复补丁发布
    - 热修复补丁和功能部署在发布过程上有哪些区别？

- 排定优先级
  - 是否有一个结构化过程可以用来管理可靠性工作与面向用户的功能的优先级排序？

4.6 SRE 成熟度模型

前面描述的对产品交付组织中的生产运营进行全面评估后，便可以建立一个 SRE 成熟度模型。SRE 教练可以利用这个模型来评估组织在 SRE 转型之前的状态。一旦启动转型，SRE 教练就可以重新评估组织，检查不同领域是否有进步、停滞或退化。这样的评估，合理的重新评估频率大约是每年两次。

SRE 成熟度模型也有常见成熟度模型的通病，那就是它假设有一个线性进展路径，有一个固定的卓越指标。然而，由于每个团队各有其独特的技术场景，所以其 SRE 转型各有不同。在 SRE 成熟度模型中，作为最高目标的卓越并不是巅峰，而是一个孜孜以求的过程。每个团队持续完善 SRE 过程和实践，使其能够以可持续的方式顺利落实 SRE。

尽管有这些不足，但 SRE 成熟度模型仍然可以帮助 SRE 教练在 SRE 转型过程中确定方向并为团队提供行动指南。具体说来，在旅程开始时，SRE 教练可能无法立即基于数据做出转型决策，因为还没有打好基础。此外，SRE 教练可能也不太熟悉 SRE 转型。因此，SRE 成熟度模型可以为 SRE 教练提供一个宝贵的总体指导。

图 4.3 展示了 SRE 成熟度模型，其中定义了三个级别：退化、初级和高级。注意，每个团队都严格遵循需要这些级别。尽管存在普遍的不足，但 SRE 成熟度模型仍然是一个有用的工具，可以帮助组织了解自己在 SRE 转型之旅中处于何处，还可以指导组织发展到更高的成熟度。

对团队而言，花时间做自我评估或参与耗时的评估并不能使自己从中获得明显的好处。SRE 成熟度模型的目的是帮助 SRE 教练识别 SRE 转型过程中亟待关注的关键领域。

SRE 教练应记录并保存评估结果作为后期参考。SRE 基础一旦建成，就可以开展后续的评估。重新评估的结果可以与初始评估结果进行比较，衡量 SRE 转型的进展情况。

成熟度模型清晰地揭示了 SRE 实现的多面性，SRE 转型本身也涉及很多个方面。下一小节将探讨对 SRE 转型的期望。在启动转型之前，有必要对齐这些期望，以免日后导入 SRE 的幻想落空。

SRE成熟度模型		打分标准：“0” = 退化成熟度 ，“1” = 初级成熟度, “2” = 高级成熟度						
		退化		初级		高级		
组织								结果
	结构	筒仓型产品交付组织	0	部门合作	1	独立的跨职能团队	2	
	就运营问题达成一致	不透明	0	通过临时会议来达成一致，有一些运营数据的支持	1	使用商定的SLI、SLO、错误预算策略和基于错误预算的决策来达成一致	2	
	正式领导	不知道SRE	0	对SRE的一定支持	1	完全支持SRE	2	
	非正式领导	生产英雄	0	整个组织内有一些懂得运营知识的人	1	实践SRE教练	2	
								平均
人员								结果
	开发, 架构师, 开发经理	从不值班	0	对生产运营有一定见解	1	按约定共同值班	2	
	运营工程师, 运营经理	总在值班，但在筒仓里	0	总在值班，但能影响优先级了	1	按约定共同值班	2	
	产品负责人, 产品经理	生产不关我的事	0	一定程度参与生产运营	1	基于错误预算的决策	2	
	副总裁, 管理层	不知道SRE	0	一定程度支持SRE	1	完全支持SRE	2	
								平均
技术								结果
	日志记录	非结构化	0	部分日志是结构化的	1	结构化以便机器处理	2	
	监控	无警报	0	基于技术资源(参数)的警报	1	警报反映对用户体验的影响	2	
	测试	不值得信任	0	有的测试套件可以信任	1	值得信任	2	
	发布	所有服务一起发布	0	有的服务单独发布	1	独立的服务发布	2	
	能力计划	不存在	0	比较随意	1	弹性能力	2	
	开发	没有适应能力	0	一定适应能力	1	适当的适应能力	2	
								平均
文化		病态型		官僚型		生机型		结果
	合作	低度合作	0	适度合作	1	高度合作	2	
	信使待遇	怪罪信使	0	忽视信使	1	训练信使	2	
	共担风险	推卸责任	0	狭义责任	1	风险共担	2	
	交流	不鼓励交流	0	容忍交流	1	鼓励交流	2	
	对失败的态度	失败会找替罪羊	0	失败会被公正追责	1	失败会共同找原因	2	
	对新想法的态度	压制新想法	0	新想法会带来问题	1	新想法会被实现	2	
								平均
过程								结果
	客户支持	比较随意	0	专门的角色	1	在运营团队中轮流	2	
	值班	不存在	0	出于好意	1	由开发和运营团队按约定共同进行	2	
	事故响应	比较随意	0	按经理的要求	1	由值班人员集中解决	2	
	生产访问控制	不透明	0	人和脚本都可以	1	只有脚本可以	2	
	监管合规	手动	0	一定程度自动化	1	很大程度自动化	2	
	部署	手动	0	一定程度自动化	1	完全自动化	2	
	可靠性的优先级判定	不透明	0	由关键意见领袖做出	1	使用错误预算策略，由数据驱动	2	
	对可靠性的决策	不透明	0	由关键意见领袖做出	1	使用SRE指标，由数据驱动	2	
								平均

图 4.3　SRE 成熟度模型

4.7 提出假设

对于生产运营，组织内部各方观点不同。尤其是产品交付组织因持续故障和大量客户投诉升级而面临挑战时，更是各执一词。

对 SRE 转型的看法和期望也不同。开始启动 SRE 转型之前，一个重点是调整不同角色和利益相关者的期望。这不仅可以有效避免失望，而且还有利于加强 SRE 转型的支持和深入理解 SRE。SRE 转型的主要利益相关者包括管理层、经理、运营工程师、开发人员和产品负责人。各方利益相关者可能都有一些不切实际的期望，如表 4.2 所示。

表 4.2 各个利益相关者对 SRE 不切实际的期望

利益相关者	可能不切实际的期望
管理层	可靠性问题所导致的客户流失将在几周内停止
经理	产品运营和产品开发之间的冲突将在几个月内结束
产品负责人	生产故障所导致的持续大量客诉升级将在几个月内停止
开发人员	产品负责人最终会把可靠性置于功能之上。运营工程师将停止干扰开发工作过程
运营工程师	开发人员最终会实现具有生产级质量的软件。产品负责人最终会把可靠性置于功能之上

SRE 教练需要现实一些，对可能不切实际的期望进行调整。通常，SRE 转型的目的是为运营工程师、开发人员和产品负责人建立一个共享的可靠性决策流程。确定的错误预算策略用于设定可靠性工作优先级的准则。SRE 转型的初期速度不可预测。但几个月之后，SRE 教练可以提供合理的预估。

随后，SRE 教练应该邀请利益相关者以结构化的方式提出假设，以此来明确他们的期望。这些假设可以根据假设驱动开发[10]方法来定义。假设的提出应该先于产品交付团队开始建立产品功能之前，并采用三段式的描述，包括<产品功能>/<客户成果>/<可度量的信号>，具体如下：

- 我们相信这个<产品功能>；
- 会达成这个<客户成果>；
- 在看到这个<可度量的信号>时，我们就知道成功了。

通常情况下，启动 SRE 转型的动机可以作为假设的起点。动机由生产故障、客户投诉升级、可靠性工作优先级设定以及运营问题协同解决等关键痛点驱动。表 4.3 展示了利益相关者可能对 SRE 转型提出的一些假设。

表 4.3　SRE 转型假设的例子

利益相关者	示例假设		
	能力	成果	可度量的信号
管理层	在组织中建立起 SLI、SLO、错误预算和错误预算策略	降低因可靠性问题而导致的客户流失率	SRE 转型 12 个月，因可靠性问题而导致的年度客户流失率比前 12 个月减少 50%。
产品负责人	在组织中建立起 SLI、SLO、错误预算和错误预算策略	减少客户投诉升级	第一，对客户投诉升级明确的、无歧义的定义。第二，与前 6 个月相比，SRE 转型 6 个月后，客户投诉升级的数量减少了 50%
开发人员	在组织中建立起 SLI、SLO、错误预算和错误预算策略	更快地确定可靠性工作的优先级	第一，可靠性工作在团队待办事项中可以明确地识别。第二，在 SRE 转型的第 4 季度，可靠性工作优先级排定的平均准备时间（lead time）比 SRE 转型的第 2 季度至少缩短 25%
运营工程师	在组织中建立起 SLI、SLO、错误预算和错误预算策略	在运营问题上更精简的组织协同	第一，SRE 转型 8 个月，开发人员在规定的时间和规定的情况下值班。第二，SRE 转型 6 个月，任何开发团队都能在请求的两小时内参与到一个正在发生的生产事故的解决中。第三，SRE 转型 12 个月，过去 3 个月的生产部署失败率和生产部署恢复时间的中位数比转型开始前的三个月减少了 50%
经理	组织中建立 SLI、SLO、错误预算和错误预算策略	产品开发和产品运营之间的冲突减少	在 SRE 转型的第 4 季度，提请经理注意的有关生产推出的问题比 SRE 转型的第 2 季度至少减少 40%
SRE 教练	在组织中建立起 SLI、SLO、错误预算和错误预算策略	在生产的可靠性方面，以优化的成本提供卓越的客户体验	第一，SRE 转型 24 个月，团队可以自行维持 SRE 活动，不需要持续进行教练。第二，SRE 转型 24 个月，客户就同一问题的重复投诉不超过一周。第三，SRE 转型 18 个月，团队定期根据需要调整他们的 SLO 和错误预算策略

有趣的是，表 4.3 中所有假设都来自组织的 SRE 概念金字塔（参见 3.5 节）。很多预期的成果都由此而言。不同的成果在 SRE 转型的不同阶段以相应的可度量信号进行评估。

启动转型之前，有必要与产品运营、产品开发和产品管理的少数关键利益相关者共同定义对 SRE 转型的假设。这个过程应在 SRE 教练指导下进行，会带来以下三个主要的优势：

1. 利益相关者达成共识后，能够以结构化的方式阐明自己的期望，并思考如何度量实现的这些期望；

2. 表明 SRE 教练提倡 SRE 并非出于一时冲动，而是作为一项基于数据的结构化实验；

3. SRE 转型假设定义应当在组织内部公开（例如，通过建一个公开的 SRE 维基页面），让每个人都能看到转型想要达成的成果，以及成果具体如何度量。

对 SRE 转型过程以及 SRE 教练的信任由此而来。通过这种方式，组织可以确保 SRE 转型是基于各方共识的、目标明确的和成果可量化的集体举措。

4.8 小结

本章展示了如何从运营角度来评估软件交付组织的现状。评估可以从 5 个维度进行：组织结构、人员、技术、文化和过程。为了了解组织产品运营实践的现状，评估是必不可少的。它可以回答这个问题：当前的产品运营是怎么执行的？

度量成果的话，会引发人们深入思考组织如何导入 SRE。一个好的开端是运用“假设”来定义 SRE 转型的最终成果。这些假设定义了希望通过新的组织能力来达成哪些成果。

关键的一点是，这些假设定义了如何度量这些成果。SRE 转型的目标是持续测试既定的假设，为取得长期性成效而及时调整 SRE 转型实践。

注释

1 译注：第一性原理（first principle），哲学与逻辑名词，是一个最基本的命题或假设，不能被省略或删除，也不能被违反。第一原理相当于是在数学中的公理。最早由亚里斯多德提出。

2 Murphy, Niall Richard, Betsy Beyer, Chris Jones, and Jennifer Petoff. 2016. *Site Reliability Engineering: How Google Runs Production Systems*. Sebastopol, CA: O’Reilly Media. Reprinted by permission of O’Reilly Media, Inc.

3 译注：一种基于流量比例的发布策略，部署一个或者一小批新版本的服务，将少量（比如 1%）的请求引流到新版本，逐步调大流量比重，直到所有用户流量都被切换新版本为止。

4 Wiggins, Adam. n.d. “The Twelve-Factor App.” https://12factor.net

5 Nygard, Michael T. 2018. *Release It! Design and Deploy Production-Ready Software*, 2nd edition. Raleigh, NC: The Pragmatic Bookshelf.

6 Longman Dictionary of Contemporary English. 2014. “Definition of culture.” https://www.ldoceonline.com/dictionary/culture

7 Westrum, R. 2004. “A Typology of Organisational Cultures.” Quality and Safety in Health Care 13 (suppl_2): ii22–27. https://doi.org/10.1136/qshc.2003.009522

8 Google. n.d. “Cloud Architecture Center.” https://cloud.google.com/architecture/devops/devops-culture-westrum-organizational-culture.

9 Shook, John. 2010. “How to Change a Culture: Lessons From NUMMI.” MIT Sloan Management Review 51 (2), 66.

10 Thoughtworks. n.d. “How to Implement Hypothesis-Driven Development.” https://www.thoughtworks.com/insights/articles/how-implement-hypothesis-driven-development

第 II 部分

启动转型

如前所述，我们已经介绍了 SRE 的基本概念，摸清了组织的现状，清楚了 SRE 为什么是改进产品运营方式的首选。同样，也清楚了组织现状与 SRE 运营目标之间的差距。接下来，我们要鼓起勇气，启动 SRE 转型！

第 5 章

取得组织的认同

SRE 转型要求组织内所有团队从多个方面进行改革。为此，需要取得整个组织的认同和支持。如何取得组织对 SRE 的认同呢？本章将给出具体的指导。

在深入探讨这个主题之前，需要注意一点：需要根据组织文化合理选取本章介绍的内容。组织文化在很大程度上决定着组织会以什么样的方式接受 SRE。本章阐述的内容尤其适用于韦斯特鲁姆（Westrum）模型中描述的规则导向型文化和效能导向型文化[1]。

在以权力为导向的文化（病态型文化）中，SRE 以什么样的方式得到组织的认同，则较为复杂，在很大程度上取决于掌权者的行为。为了在这种文化中获得整体认同，需要仔细分析具体行为、不易察觉的差异、关系、功能障碍、合作方式、心态以及掌权者的意愿，进而在此基础上制定相应的协调策略。

5.1 取得组织内部对 SRE 的认同

3.6 节讨论了如何通过 SRE 概念来实现组织协同。现在探讨的问题是如何实现这种协同。人们为什么要关心 SRE？如何激励？如何以一致的方式获得人们对 SRE 的支持？换言之，如何在一个从未听说过 SRE 的组织中发起一场可持续的 SRE 运动？

为此，需要先来理解什么是“运动”。《朗文词典》如此定义：“一群有相同想法或信仰的人共同努力以实现特定的目标。”[2]如此说来，SRE 运动便是一群人共同努力，推广他们对软件运营秉持的共同理念。

这意味着一开始就需要组建一个信守 SRE 理念的团队。这个团队至少有两个人，他们将共同致力于推广 SRE 软件运营理念。

2.5 节强调了培养内部教练来推动 SRE 转型的重要性。因此，要发起 SRE 运动，最初至少需要两名 SRE 教练。

最初这个 SRE 教练团队需要采取哪些措施来获取人们对 SRE 的支持呢？首先，需要在组织内推动对 SRE 的支持。需要在组织内开展讨论，就像 1.1 节描述的那样，让大家都能

大体上理解运营方式变革的必要性，尤其是启动 SRE 转型的必要性。在组织中，需要以自上而下、自下而上和横向的方式合力推动大家对 SRE 的认同。

为了促进自上而下的认同，应将 SRE 活动纳入项目组合管理活动中。因为一个产品交付组织中的所有团队都需要积极参与 SRE 转型。为确保这些措施产生价值，应在组合管理活动中适当回应组织对 SRE 的诉求，引导大家一起讨论 SRE 转型相对组织中其他主要倡议的优先级。这个优先级可能评定为“紧急”，需要立即实施“边干边转型”（transform while you perform），或者可能为 SRE 活动分配一个较长的时间周期，后者更为常见。

在任何情况下，为了在项目组合层面确定 SRE 的优先级，都必须尽早与大领导取得联系。组织的领导具有不同的背景，需要在战略和日常工作中处理不同的问题域，因此，为了使这样的 SRE 对话取得成效，需要根据每位领导的特定背景适当进行调整。

为了促进自下而上的认同，在介绍 SRE 活动时，要把它定义为一种可以解决当前问题的方法，当前的问题包括产品运营中的故障和混乱等。如 2.5 节所述，团队教练在这里显得尤为关键。团队辅导会议要持续进行，直到团队能够完全独立地持续完成 SRE 相关活动。对团队进行长达数年的 SRE 辅导会议并非例外，而是一种常态。

为了促进横向认同，开发主管和运营主管及其团队之间的所有正式和非正式沟通都应不断强调预期目标的 SRE 设置。必须明确谁负责值班、哪些服务需要值班、在什么情况下值班及其相对组织中其他事务的优先级。与管理者接触是 SRE 转型的关键。

通过自上而下、自下而上和横向的组织引导，可以推动 SRE 运动的进展。可以将整个过程视作一种营销活动，优秀的营销活动会推演到所有可能的渠道。例如，线上渠道可以通过 Instagram、YouTube、Facebook 和 Twitter 等平台分享 SRE 转型故事。需要注意的是，所有渠道都有自己的特点和参与方式，因此，叙事的方式需要适应每个渠道的场景和用户参与方式。

这种思维模式对试图创建 SRE 运动的 SRE 教练来说是必要的。对于技术人员，从市场角度思考并采取相应的行动刚开始的时候可能会觉得不自然，但这对推动 SRE 转型有必要。在这种情况下，最合适的渠道是整个组织内部自上而下、自下而上和横向的对话。这些对话可以在线下达成，也可以通过组织的任何在线沟通工具进行。重点在于，要同时适当参与所有三个渠道，发展一个能够长期自发持续的 SRE 运动。

图 5.1 展示了一个传统软件交付组织的架构图，开发、运营和产品管理这三个部门界限分明。有些组织结构和部门之间的界限较为模糊，筒仓现象不太突出。在这样的组织中，可能更容易实现组织对 SRE 的认同。图 5.1 中各部门之间界限分别是为了突出问题，因为许多传统企业采用的都是这种结构。

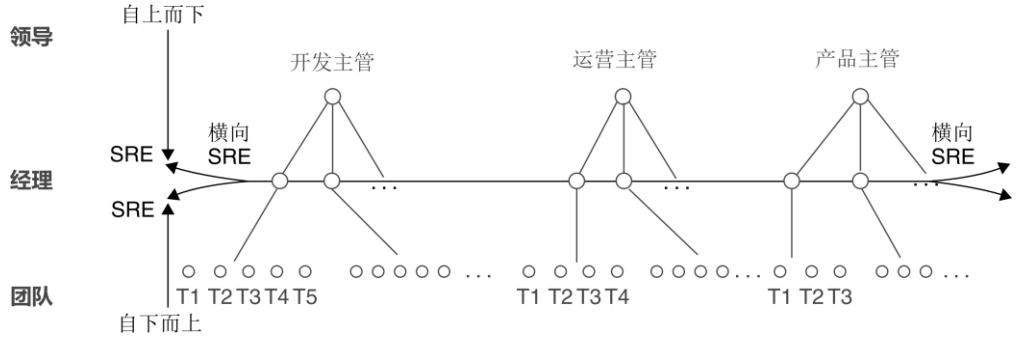

图 5.1　传统软件交付组织

传统软件交付组织通常由三个主要部门组成：产品开发、产品运营和产品管理。各部门均由各自的领导负责，包括开发主管、运营主管和产品主管。为了确保 SRE 在组织的项目组合倡议中得到适当的优先级，需要这些领导的支持。此外，领导们需要推动跨部门以及各自部门内部对 SRE 进行沟通。

在典型的组织结构中，下一个级别由开发经理、运营经理和产品经理构成。经理通常是推动各项流程变革的中坚力量。因此，可能从中涌现出一批 SRE 教练。经理需要参与建立 SRE 理念，确保支持该理念并促进它在组织内的横向沟通。经理通常负责为其团队成员设定目标。因此，让经理参与推动 SRE 至关重要，这可以确保 SRE 相关活动可以纳入他们为团队设定的目标。

最后，在典型的组织结构中，基层由若干个团队组成。这些团队需要持续接受辅导，以便在日常工作中引入、发展和深化 SRE 实践。这需要 SRE 教练提供长期的指导和支持。自下而上推进 SRE 是 SRE 转型中最消耗人力的环节。谁说不是呢？这正是决定转型成败的关键。

5.2　SRE 营销漏斗

市场营销中，有一个流行的 AIDA[3]模型，它通过一系列认知和情感步骤引导消费者，从而影响其购买决策。AIDA 模型包括四个阶段：意识（Awareness）、兴趣（Interest）、欲望（Desire）和行动（Action）。首先，要让消费者对产品有所“认识”，这通常可以通过广告来实现。随后，消费者需要展现出进一步了解产品的“兴趣”，这可以通过访问产品网站等方式来实现。在“欲望”阶段，消费者对产品有了积极的态度。最终，“行动”阶段促使消费者产生购买行为。

为了取得组织对 SRE 的认同，组织需要经历一系列类似的认知和情感步骤，直到 SRE 在组织内部得到广泛的理解和支持。在这个背景下，对 SRE 的“认同”意味着在认知和情感层

面准备好进行 SRE 转型。图 5.2 对通用的 AIDA 营销漏斗加以调整，以适应 SRE 转型的特定背景。在 SRE 营销漏斗中，包括意识（Awareness）、兴趣（Interest）、理解（Understanding）和共识（Agreement）四个步骤。

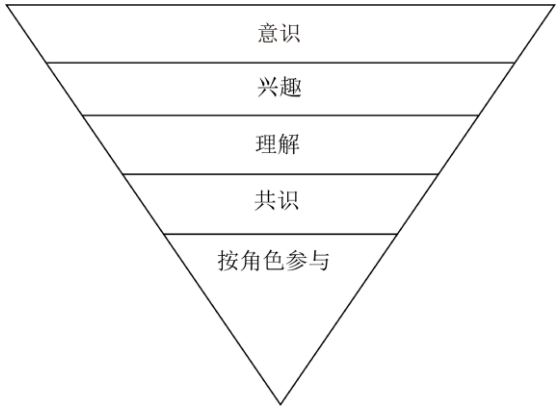

图 5.2　SRE 营销漏斗

随着 SRE 营销漏斗的建立，SRE 教练需要考虑如何让 SRE 转型作为一个行动倡议，在众多引发利益相关者注意力的倡议中脱颖而出。如何使 SRE 转型的“营销”比其他倡议更醒目？如何以一种吸引人的方式呈现 SRE 转型？如何激发团队成员对 SRE 的兴趣？组织中是否有其他倡议与 SRE 相近或者可以得到 SRE 的支持？是否可以将一些活动与其他倡议结合，以加强信息的传播并引发更多的关注？这些问题需要 SRE 教练深入思考。

5.2.1　认识 SRE

为了在组织内进行推广，让大家都知道 SRE 是一种运营方法，需要开展一系列宣传活动。在本书写作期间，许多公司和业界人士对 SRE 仍然不太了解或只是略有耳闻。SRE 教练可以通过组织一些小的演讲活动来帮助组织了解 SRE。如果组织有设施可以举办这类轻量级学习活动，就更好了，因为这意味着已有潜在的听众，并且他们已准备好接受新的知识。精益咖啡、午餐会和学习会等活动都是 SRE 宣讲活动的理想场合。

为了帮助大家认识 SRE，SRE 教练无需亲自制作演示材料。实际上，使用公开可用的资源更加高效。网上可以免费获取并利用的资源如下：

- 2018 年伦敦 DevOps 企业峰会，Stephen Thorne 发表的演讲 Getting Started with Site Reliability Engineering[4]及幻灯片；

- 2019 年奥斯汀 DevOps 开放日，Damon Edwards 发表的的演讲 SRE for Everyone: Making Tomorrow Better Than Today；[5]
- 2019 年微软 Ignite Tour，Jason Hand 发表的演讲 Monitoring Your Infrastructure and Applications in Production。[6]

这些演讲的目的是传递一个信息：“有一种新的有趣的运营方法，越来越多的软件行业公司正在采用。”这些演讲可以在非正式的场合，如饮水机旁、午餐时间或在线聊天中激发额外的讨论。为了最大化演讲的影响力，应确保演讲内容有记录，并在名称中明确显示“SRE”标签，同时在组织内部的维基页面上进行适当的引用，以便在组织中赢得更多曝光的机会。

这些努力至关重要，不宜低估。目标是尽可能广泛地传播 SRE 知识，激发人们对 SRE 的兴趣。应抓住每一个机会来促进对 SRE 的讨论。这些讨论最初可能会吸引开发人员、架构师和运营工程师的注意，也可能引起开发和运营经理的兴趣。尽管这种方法可能会引起一些具有技术背景的产品负责人的关注，但它可能不会立即触及一般意义上的产品经理。然而，最初引发的热潮暂时还无法传递到管理层。

5.2.2 兴趣

确认一部分人对 SRE 有了认识之后，下一步是激发他们进一步了解的兴趣。为此，可以通过在线聊天分享前面提到的视频，分发电子书，购买 SRE 相关纸质书并分发给大家，对于居家办公的员工，则把书籍送到他们家中。此外，如果有读书会或技术实践社区，可以在这些场合对书籍进行讨论。推荐阅读以下书籍：

- *Site Reliability Engineering: How Google Runs Production Systems*[7]，作者 Niall Richard Murphy, Betsy Beyer, Chris Jones, and Jennifer Petoff，中译本《SRE：Google 运维揭秘》；
- *The Site Reliability Workbook: Practical Ways to Implement SRE*[8]，作者 Betsy Beyer, Niall Richard Murphy, David K. Rensin, Kent Kawahara, and Stephen Thorne，中译本《SRE：谷歌运维揭秘》；
- *Implementing Service Level Objectives: A Practical Guide to SLIs, SLOs, and Error Budgets*[9]，作者 Alex Hidalgo，中译本《实施 SLO》；
- *Real-World SRE: The Survival Guide for Responding to a System Outage and Maximizing Uptime*[10]，作者 Nat Welch，中译本《SRE 生存指南：系统中断响应与正常运行时间最大化》。

值得注意的是，SRE 教练要推荐 SRE Weekly[11]的文章。此外，技术社区是分享 SRE 博客文章和新闻的理想场所，因为人们通常愿意订阅这类内容。如果有技术实践社区（例如，开发人员、架构师、运营工程师或者 DevOps 这样的主题），那么上一节提到的演讲应该在这样的小圈子中反复进行。小圈子特别适合设计问答环节来进一步广泛引发人们对 SRE 的兴趣。

此外，如果组织举办过类似 Scrum of Scrums 这样的研讨会，那么邀请参与者加入 SRE 专题会议将是一个值得尝试的举措。这样的讨论可能会有 Scrum Master 参加，他们是组织中的意见领袖。此外，偶尔可能会有项目经理、运营经理和开发经理参加，他们在其他会议上都有机会和管理层对话。因此，和这些团体对话讨论 SRE，便是为 SRE 这个话题提供机会，使其能够得到管理层的关注。

参加 Scrum of Scrums 研讨会的人往往很有兴趣了解组织动向。因此，在和他们对话的时候，需要解释以前是怎样的、SRE 的观点又如何以及组织如果采用 SRE 的话有哪些好处。对话的目的是邀请参与 SoS 的人员参与讨论 SRE 转型。这一点尤其重要，否则他们会本能地抵触这样的变革，即没有任何思想准备就落地 SRE 转型。

随着 SRE 教练团队不断讨论、展示和分发 SRE 相关内容，人们可能逐渐意识到是这些教练在推动 SRE。随着时间的推移，他们可能开始将 SRE 教练与 SRE 联系起来。这是一个很好的迹象，因为突然间组织中就有了这么一些人，任何人只要有 SRE 相关问题，就可以去找他们。

5.2.3 理解

SRE 营销漏斗的下一个阶段是推动人们深入理解 SRE。这可以通过深入讨论 SRE 核心概念来实现。例如，什么是 SLI（服务等级指标）？什么是 SLO（服务等级目标）？什么是错误预算？什么是错误预算策略？SRE 如何适用于我们的工作场景？其他公司是怎么做的？这些话题都可以推动人们深入理解 SRE。

通过 SRE 宣讲活动，我们可以识别出哪些人对 SRE 特别感兴趣。他们会主动借阅办公室里分发的书，提出问题，与同事讨论 SRE 相关话题。我们可以为他们举办几场专题会议，深入探讨他们对 SRE 的共同兴趣，了解他们对 SRE 的了解程度，参与过哪些讲座，读过哪些书，与其他公司的 SRE 实践者交流经验。这些会议的目的是加深人们对 SRE 概念的理解并在组织内部培养 SRE 社区意识。

5.2.4 共识

SRE 营销漏斗的下一阶段是在组织内部达成广泛的共识。换句话说，需要确保大家都清楚一点：通常情况下，采用 SRE 能够改善产品运营并给组织带来好处。我们应该明确，与目前的产品运营方式相比，导入 SRE 能够带来哪些具体的优势。例如，在当前阶段，应该清楚地认识到，在投资可靠性工作与开发新功能之间做出由数据驱动的决策可能有哪些好处。同时，我们应该达成共识，认识到运营工程师、开发人员和产品负责人共同参与运营所带来的好处。

为了推动这个共识，SRE 教练可以准备一个内容丰富的演讲。与 5.2.1 节所描述的那种宣讲不同，这次演讲应该详细总结自最初宣讲以来组织内发生的 SRE 相关活动。演讲应该列出组织内大家普遍认同的观点，同时指出那些引起讨论但尚未得出结论或未开始讨论的开放性问题。在演讲结束时，提出一些具体的后续步骤，例如：

- 与管理层讨论 SRE 转型启动计划；
- 在某个团队中尝试一些新的方法，如引入新的日志查询语言来构建日志基础设施；
- 安排会议，进一步理解如何在特定生产领域中应用 SRE，例如为新的 ETL 管道引入 SRE，准备数据供新的 AI 算法学习。

结束演讲时，以积极的语气强调 SRE 为组织带来的巨大潜力，例如“正如大家所见，SRE 为组织提供了巨大的潜力，我们决心继续深入挖掘。”千万不要在接近尾声的时候传达消极信息，比如“还有很多问题我们没有讲到。SRE 转型是一个漫长且不确定的过程，不要指望一蹴而就。”SRE 教练必须始终保持积极和乐观的态度，这将在很大程度上帮助团队应对转型过程中的起伏。与此同时，这样的态度也能帮助教练在组织中顺利推行产品运营方式的改革。

5.2.5 参与

SRE 营销漏斗最后一个阶段是结束营销活动，确保人们已经准备好实际参与 SRE 活动。如果营销漏斗的前几个阶段都顺利，则完全可以实现最后这个阶段。如果人们认识了 SRE，有了兴趣，对 SRE 有了自己的理解，并且一致认为它对运营有帮助，就说明他们已经准备好躬身入局了。

不同角色以不同的方式参与 SRE。表 5.1 展示了人们准备参与 SRE 活动时思考的问题。

表 5.1 不同角色参与 SRE 转型时关于就绪程度的问题

角色	问题
管理层	我怎么支持 SRE 转型？
经理	我需要做什么来推动 SRE 转型？我怎么推动人们参与其中？
开发人员	我需要做什么来参与 SRE？
运营工程师	我需要做什么来参与 SRE？
产品负责人	我需要做什么来参与 SRE？

针对准备不够充分的人，还需要重复或者加强 SRE 营销漏斗中的一个或几个阶段。也就是说，并不是每个人都认同 SRE。在组织中，看到任何实际的成果之前，尤其如此，毕

竟眼见为实。但这在当前阶段不可能。重要的是在组织中达成广泛的共识：SRE 可能是一个好的运营方式。如果大多数人都认同这个观点，就说明营销漏斗是成功的，SRE 教练可以胸有成竹了。

5.3 SRE 教练

如前所述，SRE 教练在推动 SRE 转型中扮演着关键角色。在深入转型旅程之前，我们需要探讨 SRE 教练应该具备的领导力和其他特质。为了有效推动转型，关键在于建立团队对 SRE 教练的信任。

5.3.1 特质

如何识别能够赢得团队信任的转型领导人？需要具备哪些特质才能够赢得团队的信任？以下是一个有代表性的关键特质清单，不过并不完整：

- 在产品交付领域至少工作三年以上，最好是在计划 SRE 转型的组织内；
- 有定期交付产品的成功记录；
- 有在组织内引入新工作方式的经历；
- 成功推广并持续实施新工作方式的经验，这一点可以通过团队成员改变工作方法得到证明；
- 有让不同利益相关者参与并推动倡议的记录；
- 有对改进进行有效度量的良好记录；
- 在工作中展现出良好的人际关系技巧；
- 与组织中的大多数成员保持良好关系；
- 具备坚韧、同理心、持之以恒、反馈迭代、实验精神、毅力、耐心、可靠性、耐力、热情、抱负、目标导向、决策力、沟通能力、创造力、自我激励和激励他人的能力、开放心态、学习能力、诚实、真诚和善良等人格魅力。

一旦确定转型领导人，维护与其团队成员之间的持续信任关系就变得尤为重要。然而，这种信任也很容易被破坏。以下行为就极有可能导致信任被破坏：

- 设定不现实的 SRE 转型目标，并将其作为既定目标呈现给团队；
- 没有认识到转型需要在持续的产品交付压力下进行，对转型速度缺乏耐心；
- 在向上级报告时，过份夸大转型的成功，使团队成员觉得转型领导人脱离实际；
- 好大喜功，不提真正实现成功的人，而是归功于自己；

- 只让团队中的某些群体参与影响整个团队的决策，例如，只让架构师参与技术决策，而不考虑他们是否能代表整个团队；

总之，为了确保转型的成功，推动转型的人必须赢得所有相关人员的信任。SRE 转型领导人不一定需要在组织中拥有正式的权力地位，这样反而有益，因为团队之所以改革，是基于自己所获得的结论和信念，而非对权力的服从。这有助于变革能够以一种可持续的方式坚持下去。

如前所述，在组建 SRE 教练团队方面，最佳配置是从产品运营和产品开发抽调出少数几个人。因为大多数变革都发生在这些部门。来自产品管理部门的人不太可能成为 SRE 教练，而且也没有必要，因为在产品管理中导入 SRE 不会像产品运营和产品开发中那样影响深远。通常，最好是由一名运营工程师、一名开发人员和一名产品负责人来组成 SRE 教练团队，只不过这样的可能性不大。

5.3.2 责任

SRE 教练虽然身兼多职，但也有其重点领域。需要分担的责任清单如下。

- 管理：
  - 管理项目；
  - 安排会议/课程；
  - 向管理层报告；
  - 游说 SRE 活动预算（以支付工具、培训、考察会议的费用）；
  - 管理新工具的采购和后续许可证。
- 策略：
  - 确定项目组合的优先级；
  - 为每个团队创建 SRE 采用策略，指导团队的工作；
  - 管理 SRE 待办事项；
  - 管理 SRE 基础设施的功能请求。
- 技术：
  - 定期召开团队辅导会议；
  - 将团队辅导会议的记录保存在一个共享的地方；
  - 了解 SRE 基础设施、即将推出的功能以及未来的方向；
  - 在 SRE 基础设施上对团队进行培训。
  - 支持团队的 SLI 和 SLO 定义过程；

- 支持团队建立值班过程和轮流值班/轮值；
- 管理故障排除请求；
- 确定最佳实践，并使团队相互交流。

- 营销：
  - 宣传成功经验，可以通过博客文章、新闻通讯、演讲/演示等方式进行；
  - 寻找机会将 SRE 思维方式注入团队和人员的日常工作中；
  - 参加公司内部的活动和会议，传播 SRE 相关信息。
- 人员：
  - 根据需要和人员开展一对一的交流；
  - 促进团队建立 SRE 待办事项立；
  - 推动建立 SRE 实践社区；
  - 将 SRE 纳入员工培训过程；
  - 为新工具的培训提供便利。

尽管 SRE 教练有一系列责任，但这不一定是一个全职工作，可以是兼职。例如，负责 SRE 基础设施运营的工程师也可以是 SRE 教练，并承担上述清单中的“技术”责任。类似地，担任 SRE 教练角色的开发经理可以同时承担项目管理、会议/课程安排、管理层汇报、采购和项目组合管理等相关责任。

通过这种方式遴选出来的 SRE 教练组成一个非正式的跨职能小组，在整个产品交付组织中工作。在某些组织文化中，类似跨职能小组受重视的程度可能不如筒仓式项目。SRE 教练需要留意并在必要时确保领导层在整个产品交付组织内正式公告发指定 SRE 教练，明确他们未来需要与所有团队合作。在实现组织对 SRE 的认同方面，SRE 教练需要以自上而下、自下而上和横向的方式与组织合作。在接下来的小节中，我将对具体细节进行深入的解释。

5.4 自上而下认同

为了促进自上而下对 SRE 的认同，需要明确与行政领导团队（管理层）中的哪些人进行接触。SRE 提倡大量的时间投入和适度的金钱投入。基于此，可以编制一份利益相关者的名单。《50 招改进用户故事》（*Fifty Quick Ideas to Improve Your User Stories*）[12]一书建议创建一个利益相关者图表。利益相关者图表有助于识别会受此倡议直接和间接影响的人。间接受影响的人很容易被忽视，但他们和直接受影响的人一样，都要参与。因为他们可能会成为倡议的支持者或反对者，而这两种人都是 SRE 教练需要了解的。

5.4.1 利益相关者图表

为了创建一个利益相关者图表，首先要编制一份可能涉及的所有利益相关者的名单。这份名单应该包括产品交付组织的所有高层领导。这些人可能是运营主管、开发主管和产品管理主管。在大型组织中，这些领导可能看起来与 SRE 相距甚远，因此可能不被认为是 SRE 的利益相关者。但这个结论显然是不正确的。如果领导远离日常工作，则说明更需要与他们接触，因为他们负责倡议，负责部门的时间和预算分配。他们必须在适当的层面上理解 SRE 转型需要投入时间和金钱才能做出合适的判断。SRE 教练的工作是尽可能将这些领导转变为 SRE 转型的支持者。

另一个极其重要的方面是，SRE 试图在产品运营、产品开发和产品管理之间建立运营协作（见 1.2 节）。如果运营主管、开发主管和产品管理主管之间未能达成一致，便无法实现。如果他们在 SRE 的重要性上没有取得共识，SRE 在组织倡议清单中的优先级也不可能很高。由此可见，产品交付组织的最高领导人必须列入利益相关者名单，无论他们的行政级别如何。

一旦建立所有可能利益相关者的名单，就可以根据他们在组织中的正式权力和对倡议的兴趣进行分类（见表 5.2）。

表 5.2　利益相关者图表

正式权力类别	兴趣较低的利益相关者	兴趣较高的利益相关者
高级别利益相关者	保持满意	充分参与
低级别利益相关者	监控	保持沟通

在表格的左侧，正式权力分为两类：高和低。在表格的上方，兴趣也分为两类：低和高。因此，根据这个表格，利益相关者可以分为四类。对于高级别位但对 SRE 兴趣较低的利益相关者，需要保持其满意度。这意味着需要确定哪些因素来使其对 SRE 转型感到满意。对于低级别且对 SRE 兴趣较低的利益相关者，需要持续关注。除非他们的权力或兴趣发生变化，否则不需要对这个利益相关者群体采取任何行动。

对于高级别且对 SRE 兴趣较高的利益相关者，需要让他们完全参与 SRE 转型。的确，一些高级领导，尤其是有技术背景的，可能愿意深入了解转型的细节。SRE 教练应确保这些领导能够参与转型。和他们讨论何种程度的参与对他们最为合适。一方面要满足他们的兴趣，另一方面又不能过多占用他们的时间。

最后，确保 SRE 转型中低级别但兴趣较高的利益相关者能够了解情况。这可以通过以下方式实现：定期通过电子邮件保持联系并在组织的新闻通讯中更新最新进展，邀请他们

参加为较多人准备的演讲等。

建议 SRE 教练创建一个 SRE 利益相关者图表来识别转型的直接和间接利益相关者。特别是为了实现自上而下的认同，必须将图表创建过程中可能不明显但最终会被识别出来的利益相关者纳入其中。SRE 利益相关者图表不应仅限于 SRE 教练个人使用，可以分享给部分利益相关者，向他们征求意见和反馈。毕竟，在一个大企业里，很容易忽略一些人或不清楚他们的职责。

表 5.3 展示了一个 SRE 转型利益相关者图表示例。表格的右上象限代表产品交付组织的最高领导人。这些人通常包括运营主管、开发主管以及产品管理主管。SRE 转型会直接影响到他们的部门。

表 5.3　示例 SRE 转型利益相关者图表

正式权力类别	兴趣较低的利益相关者	兴趣较高的利益相关者
高位利益相关者	保持满意： • 法务主管 • 合规主管 • 财务主管 • 市场主管 • 销售主管 • 产品交付组织主管	充分参与： • 运管主管 • 开发主管 • 产品管理主管
低位利益相关者	监控： • 采购主管 • 项目组合管理主管	保持沟通： • 合伙人管理主管 • 公司中的其他产品交付组织

左上象限代表一大群利益相关者。这些人有很高的正式权力，但对 SRE 的兴趣不高。例如，法务部主管起初对 SRE 没有兴趣。然而，当组织发展了有意义的 SLI、SLO 和错误预算策略时，法务部也可能开始关注。因为 SRE 数据可用于制定服务水平协议（Service-Level Agreement，SLA）。

SLA 是服务提供方和服务消费者之间的合同协议，因此法务部门通常会参与 SLA 的合同谈判。在《实施 SLO》[13]一书中，法务部门被识别为 SRE 的利益相关者，因为 SLO 数据可以用来支持 SLA 谈判。

在左上象限，还有合规主管，他们最初对 SRE 的兴趣通常也很低。然而，随着 SRE 实践在组织中的展开，SRE 数据会获得合规主管的关注。从监管的角度来看，监控生产中服务的健康状况可能是一项合规要求。例如，ISO/IEC 27001[14]是一个管理信息安全的国际标准。它要求对合规的服务健康监控进行存证。采用 SRE 方法来实现这一目标，是满足监管要求的有效方式。

左上象限还包括财务、市场、销售以及整个组织的主管。这些领导在组织中拥有很高的正式权力，但对 SRE 的兴趣可能不高。目前可以维持现状，但需要先确保这个象限的领导对 SRE 转型是满意的。换言之，SRE 转型不会引入任何新的法律或监管问题、使组织超出预算或者影响到市场与销售。

左下象限是采购和项目组合管理的领导，这些领域并不直接管理 SRE。因此，没有必要立即与这些领导接触并商讨如何推动 SRE 转型。

右下象限包含需要了解转型进展的利益相关者，其中包括负责合作伙伴管理的领导。他会对 SRE 感兴趣，因为 SRE 有望以更有效的方式管理合作伙伴关系。例如，使用公共平台 API 的内部和外部合作伙伴有可能看到相关数据中心的一些汇总的错误预算消耗统计数据。随着时间的推移，这可能有助于增进合作伙伴之间的信任。事实上，谷歌为类似目的专门组建了一个团队，即客户可靠性工程（Customer Reliability Engineering，CRE）[15]。

右下象限另一个重要利益相关者群体是组织内的其他产品交付团队。他们可能也有运营方面的困难，尤其是刚开始以 SaaS 模式运营软件的团队。因此，这些领导可能对了解 SRE 转型的成果和所付出的努力非常感兴趣。对他们来说，看到 SRE 转型成功会增加他们的信心，因为他们同属一家公司。其他部门对 SRE 感兴趣并最终采用 SRE，还有一个好处是会在公司内部形成一个更大的联盟，为相同的共享资源出谋划策，这些资源包括工具和运营预算等。

利益相关者图表应该定期重新审视，以适应组织变化和 SRE 成熟度的提升。一般说来，每 6 个月审视一次似乎较为合理。

运营主管、开发主管和产品管理主管具有较高的正式权力，能够从 SRE 获得更多好处，所以他们需要充分参与 SRE 转型。因此，获得这三类利益相关者的认同最为重要。在产品交付的范围内，这些领导各自拥有独特的利益和关注点。接下来，探讨如何有效地让他们参与进来，使其认可 SRE 转型。

5.4.2 与开发主管接触

开发主管通常更关注功能交付的质量和速度。他们最关注以下问题：

- 功能交付的质量；
- 功能交付的速度；
- 招聘；
- 预算；
- 持续的技能提升；
- 与产品管理、运营和产品交付组织的关系；
- 过程改进。

从产品运营的角度来看，由于需要关注质量问题，所以 SRE 可能成为优先考虑的解决方案。开发主管通常会感受到来自四个方面的质量交付压力：客户、产品交付组织主管、运营主管和产品管理主管。似乎每个人都在吐槽质量。在这种情况下，虽然“质量没有商量的余地”在理论上正确，但在实践中，真正的刚需是在高质量保证和保持功能交付速度之间取得平衡。

如何取得这种平衡呢？一个典型的答案是：改进招聘过程，聘用重视质量的软件工程师；投资于持续提升技能，确保组织掌握最新工具和方法；建立一个指标体系来指导决策。指标体系的建立可能很棘手，因为不能直接度量软件的质量。

5.4.2.1 改进指标度量

《加速》[16]一书提出了一套越来越受欢迎的指标体系，该体系包括四个指标：部署频率、变更准备时间、变更失败率和服务恢复时间。如表 5.4 所示。

表 5.4 指标

指标	解释[17]
部署频率	组织多久一次将代码部署到生产环境
变更准备时间	从提交到进入生产所需的时间
变更失败率	在生产中导致失败的部署的百分比
服务恢复时间	生产中出问题时，需要多长时间恢复

此外，《加速》一书的几位作者发起了 DORA 研究项目[18]，发布了 DevOps 2021[19]趋势报告，其中新增了一个运营效能指标：可靠性。报告中如此定义可靠性：团队对运营软件的承诺以及认定的遵从度。作者要求受访者对一些可靠性目标进行打分——可用性、延迟、性能和可扩展性。

这是一个很好的开始。相较于 DORA 以前的 DevOps 趋势报告（即历史上只衡量可用性的报告），2021 年的这份报告有了进步。从本书前几章的讨论可以看出，SRE 能提供可用性和延迟度量，并为运营效能增加许多度量指标。事实上，在 SRE 术语中，可用性和延迟都是 SLI。还有更多 SLI，例如吞吐量、正确性、新鲜度和持久性等，详情可参见 3.1 节。所有这些都能用相应的 SLO 来度量。

为了说服开发主管启动 SRE，可以指出 SRE 是一些有用运营指标的额外来源，这些指标比《加速》一书中提到的更丰富，可以媲美甚至超越 DORA 研究项目所提到的。使用这些新指标，可以更精确地实现高质量软件交付。

5.4.2.2 改善关系

开发主管非常注重与运营主管及产品管理主管建立良好的工作关系。对此，SRE 也能提供帮助，因为它在实现协同性方面具有独特的优势（参见 1.2 节）。但如何有效地让开发主管领会这一点呢？可以借助于 SRE 提供的额外运营指标，这些有理有据的指标可以用来取得对方的信任。前面讨论的指标都是用来指导产品开发改进的。然而，SRE 指标是从用户角度来报告生产的状态。这对产品交付组织、运营和产品管理的主管来说显然更有吸引力。

也就是说，可以依托于 SRE 指标，与产品交付组织主管、运营主管和产品管理主管一起讨论生产状态。不应仅将负面信息作为生产状态的滞后指标呈现给开发主管，SRE 指标还能够作为前置指标来促成大家及时沟通生产问题。这样一来，所有人都可以在正确的时间针对保持生产顺利运行所需要的措施做出共识决策，以免问题恶化而遭到客户投诉。这显然可以改善与产品交付组织主管、运营主管和产品管理主管三者的关系。各方都要接触产品运营并做出更优的决策来改进生产状态。

图 5.3 展示了没有 SRE 的情况下开发主管收到客诉的流程。客户在生产环境中使用产品，他们发起的客诉先后被转给运营主管和产品管理主管，最后转给开发主管。

运营主管在进行生产监控的过程中，也会将相关客诉反馈给开发主管。但有了 SRE 之后，就有了全新的决策机制。有了这个机制，主管就可以在客诉发生之前根据生产数据及时做出与可靠性指标相关的决定，如图 5.4 所示。

SRE 基础设施监控生产状态并生成适当粒度的 SRE 指标，使其能够被各团队的主管理解。这样一来，运营主管、开发主管、产品管理主管以及产品交付组织主管就能够作出共识决策。决策涉及时间、预算和人员的分配。例如，他们可能会发现，在最近三个季度，特定的部署服务（如支付处理服务）几乎耗尽了错误预算。同时，他们可能注意到，客户对支付的投诉数量正在增长。利用这些数据，他们便可以审视组织中负责支付处理的部门“负责”或“拥有”多少服务，并且可能由此发现服务数量与人手配置不合理。这就可能引发讨论，大家共同商议是为支付处理服务分配更多人手来扩充团队还是新建一个团队。这可能引发进一步的讨论，比如是从其他团队调派人员来支付处理服务，还是需要为招聘新人制定预算。

除了促进数据驱动的决策，SRE 的引入还能显著改进流程。有了它，整个组织可以更懂生产，使组织能以数据驱动的方式提前应对运营问题，推动开发人员主动为生产环境中的工作成果承担责任。

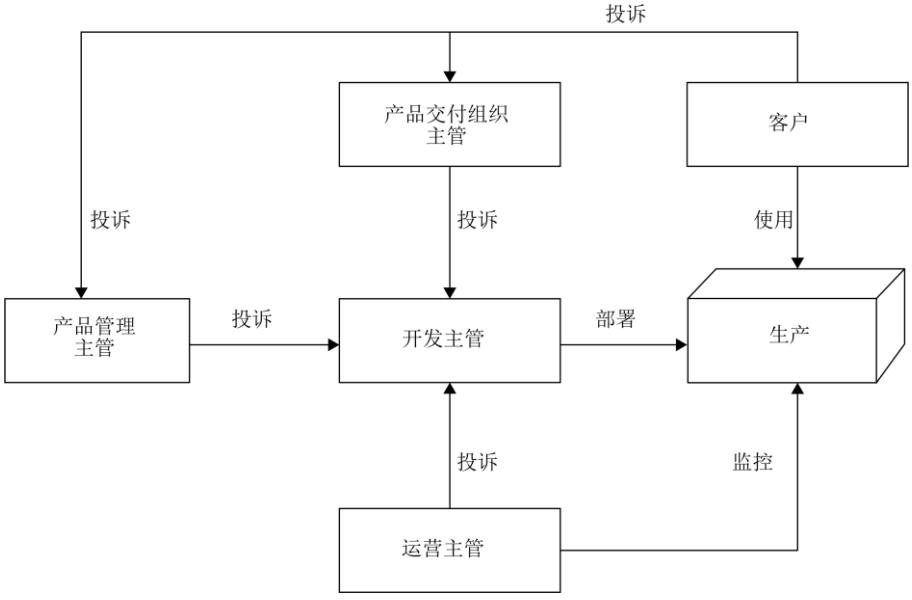

图 5.3　投诉到达开发主管的路径

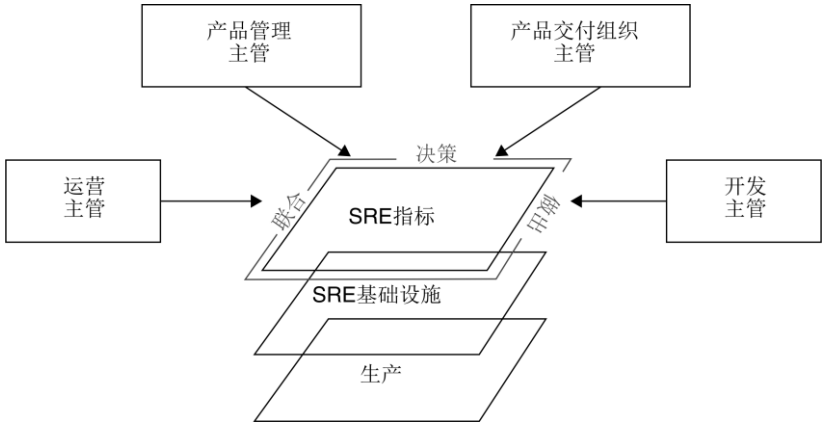

图 5.4　使用 SRE 指标进行联合决策

5.4.2.3 评估投入产出

综上所述，一切都很好。但最后，开发主管可能会问 SRE 转型要牵涉多少成本。我们应尽量利用现有的基础设施来进行成本计量。为了基于 SRE 概念金字塔来计算指标，并为开发人员提供便利，必要的 SRE 基础设施插件将由产品运营部门负责开发和维护。可能需要一些新的工具许可证，但不会大幅增加成本。就预算而言，这足以让开发主管放心了。

此外，引入 SRE 的话，需要重新分配工作，以确保为生产中的服务提供值班支持。产品运营、产品开发和产品管理部门之间需要就适合本组织的值班设置进行讨论。这种讨论可能导致不同的结果，从开发人员始终为其服务值班，到仅在服务未达到一定服务水平时才进行值班。在任何情况下，开发人员都需要在 SRE 基础设施的支持下，根据事先的约定进行值班。如 2.3.1 节所述，让开发人员根据约定进行值班，可以提供必要的实时反馈回路，将产品运营的经验直接反馈到功能开发过程中。对于产品开发来说，SRE 转型需要真正付出人力。

开发主管对安排开发人员值班的反应可能有所不同。如果开发主管遵循行业 DevOps 的最佳实践，会早就预料到这一点。他们自然很高兴，因为 SRE 转型最终明确地表达了让开发人员在约定范围内值班的必要性，这样可以缩短开发人员和生产之间的距离。他们将更多地致力于满足客户需求，而不仅仅是满足产品负责人的要求。

另一方面，如果开发主管对 DevOps 的最佳实践不够了解，他们可能会将开发人员的值班安排视为产品运营部门的威胁。在这种情况下，需要向其详细解释基于 SRE 概念金字塔的整个 SRE 系统，以及让开发人员值班的优势（参见 2.3.1 节）。在 SRE 转型的初期阶段，开发主管的坚持是必要的。但当开发人员的值班任务成为一个必须讨论的话题时，SRE 教练一定要坚持下去。这个讨论需要由 SRE 教练引导，因为它暂时还非常不成熟。虽然有许多选项可以讨论，但当前无法做出最终决定。

开发主管还可能关注的一个问题是谁来推动导入 SRE。为了回答这个问题，需要传达的是，确定和培养 SRE 教练，做一些艰苦的工作，与每个团队会面，介绍工具和方法。可以准备好一份希望或已经同意做这件事情的人员的名单，这有助于提高 SRE 倡议在产品开发主管眼中的可信度。

开发主管最后可能提出的一个问题是，如何衡量 SRE 导入是否成功。答案是，对于 4.7 节描述的那些假设，它们将由来自产品运营、产品开发和产品管理的人员共同创建。这些假设将包含可量化的、可度量的信号，这些信号将指导转型，并在规定的时间以数据驱动的方式检查进展。这些假设的实现也将决定 SRE 转型的时间线。另外，应该传达出这样的印象：SRE 倡导基于持续的反馈，运用科学方法来驱动。

为了获得开发主管的支持，可能需要召开很多次会议。在第一次会议之后，开发主管可能想和他们的直接上司讨论 SRE。他们可能希望与 SRE 教练一起讨论，也可能选择独立

进行。SRE 教练对这两种情况应该保持开放态度。这同样适用于与开发主管的同僚进行讨论的情况。主持会议的 SRE 教练需要确保开发主管最终表态支持 SRE 转型，而且与组织在项目组合层面上的其他倡议相比，SRE 转型能排在一个较优先的位置。

5.4.3 与运营主管接触

运营主管通常最关注生产系统是否运行良好以及是否会有客户投诉。因此，在运营主管看来，最重要的是下面四个问题：

- 生产状态；
- 客户投诉升级；
- 客户支持请求；
- 向执行管理层报告。

在这些优先事项中，SRE 将参与并改进全部四个领域。特别是生产状态——SRE 需要直接且优先解决的问题。这也是说服运营主管参与 SRE 转型的起点。在转型前的设置中，生产状态可能由客户投诉升级和运营团队设置的基于资源的警报来决定。

5.4.3.1 改善生产状态报告

有了 SRE，生产状态将由产品运营、产品开发和产品管理共同商定的服务目标达成来决定。这些目标更侧重于客户导向，而不只是技术导向。因此，警报将更好地反映负面的客户体验。通过提前收到这样的警报并采取适当行动，可以减少客户投诉升级，因为更多问题可以在客户注意到之前得到解决。此外，报告的生产状态将更可靠地反映受影响的客户体验，从而使得向执行管理层的报告也能更好地反映客户体验。

此外，SRE 基础设施将生成 SRE 指标，显示服务在一段时间内实现既定目标的情况（错误预算消耗）。运营工程师、开发人员和产品负责人之间要达成一些约定，根据目标的实现情况来管理对可靠性改进的投资（错误预算策略）。团队将根据这些约定来进行可靠性投资（基于错误预算的决策），而不是像以前那样，总是将明显的可靠性改进“永久性地”推迟，优先考虑实现新的面向客户的功能。

5.4.6.2 改进值班设置

在目前的状态下，运营工程师可能承担了所有的值班工作。引入 SRE 后，值班设置将由运营工程师、开发人员和产品负责人之间的协议来创建。无论服务表现出的服务水平如何，运营工程师都不再运营自己没有参与开发的产品。此外，产品负责人也将参与值班设置，确保值班人员和解决事故所需要的其他人员能够迅速响应，无需每次事故都重新协商处理事项。

理想状态下的 SRE 值班设置是，开发团队按协议参与生产监控。这样一来，开发团队就要成为率先发现生产问题的人。另外，开发团队应该先于其他人注意到修复生产中出现的问题。图 5.5 对此进行了展示。

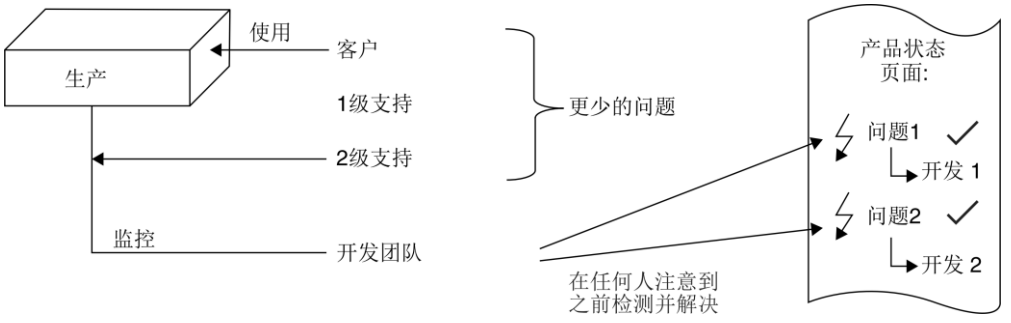

图 5.5　开发团队的目标是率先检测到生产中的问题

此外，开发团队应该让组织上下都能看到当前存在的问题及其状态以及谁在处理这些问题。如果销售团队和市场团队在组织外部，还需要让他们能够看到图中右侧的产品状态页面。这样一来，可以减少组织内部围绕生产状态展开的沟通和对话。Microsoft Azure [20] 或 Amazon AWS [21] 那样的专业状态页面是 SRE 转型的目标。

5.4.4　和产品管理主管接触

通常情况下，产品管理主管主要关注产品与市场的契合度、产品的收入以及客户要求的产品功能。以下是他们最关注的问题：

- 功能交付日期；
- 功能规格；
- 用户体验；
- 销售和市场承诺。

在以上列表中，SRE 扮演什么角色？看起来用户体验会受到它的影响。运营和用户体验又如何呢？它们与产品管理紧密相关，通常相互交织。在他们的观点中，用户体验相关的活动应在开发人员着手开发之前完成。这些活动涵盖了设计思考、设计冲刺、信息架构开发、用户研究、与 UI/UX 设计师共同探讨用户交互以及 UI 屏幕设计等。这些活动至关重要，它们为开发人员提供了必要的信息，帮助他们开始考虑如何实现用户体验和技术界面。

首先，让我们探讨“用户体验”这一概念。维基百科的词条如此定义：“用户体验（UX）是一个人在使用特定产品、系统或服务时的情感和态度。”[20]重点是使用产品，而不是思考、

设想或设计产品。用户会使用设计冲刺阶段、产品负责人与 UI/UX 设计师会议中产生的工件吗？不会。用户实际使用的是那些量产的产品。

由此可见，在产品开发过程中，用户体验作为设计的一个输入环节，与输出环节同等重要。所谓的“输出”，指的是部署到生产环境中的产品。此外，产品管理部门应将其对用户体验的理解延伸至产品的生产阶段，特别是确保产品的可靠性。图 5.6 对此进行了展示。

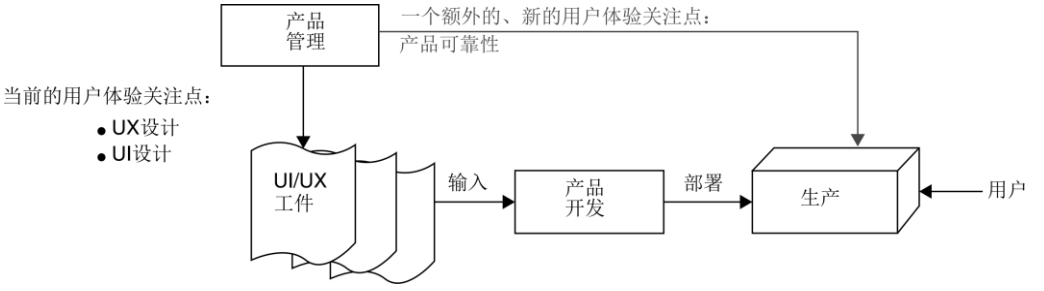

图 5.6　将产品可靠性包括在内之后扩展的用户体验观点

在用户体验方面，产品管理目前关注的是 UX 和 UI 设计。UX 和 UI 设计过程会产生一些工件作为产品开发的输入。产品开发过程的输出则是部署到生产中的产品。这就是用户使用产品的地方。他们的整个用户体验以生产中的产品为基础。因此，产品管理需要一个额外的、新的用户体验关注点，即产品在生产中的可靠性。将 UI/UX 设计和产品在生产中的可靠性这两个关注点结合，才能在生产中为用户带来最好的体验。

但是，如何扩展用户体验的关注点？是将生产中的产品的可靠性也包括进来？这就是 SRE 的优势所在。它允许产品负责人与运营工程师/开发人员共同参与一个结构化的过程，从而获得足够高的产品可靠性。产品负责人需要做什么？

1. 参与定义生产中服务的指标和目标（SLI 和 SLO）;
2. 参与定义违反目标时的策略（错误预算策略）;
3. 以策略为基础来决定优先级（基于错误预算的决策）;
4. 参与生产中服务的值班设置。

产品管理主管可能会质疑产品负责人参与 SRE 活动的时间投入。之所以提出这个问题，是因为产品负责人通常都忙于处理产品管理的许多工作。增加一个新的工作类型，不管它有多重要，都得从其他工作领域抽出一些时间。针对这个问题，回答是参与 SRE 活动的时间相当有限。以下是时间投资的分类描述。

1. SLI 和 SLO 定义

- 与团队举行几次会议，从用户的角度建立最初的 SLI 和 SLO。
- 根据在违反 SLO 后的反馈，对 SLO 进行持续的调整。

2. 错误预算策略定义

- 与团队举行会议，制定初步的错误预算策略。
- 根据事故发生后的反馈，不定期地调整错误预算策略。

3. 基于错误预算策略的优先级决定

- 根据 SRE 指标来决定可靠性方面的待办事项的优先级。

4. 值班设置

- 召开会议，规定如何确保服务的值班支持：哪些角色要去值班，在一天中的哪些时间，以及什么样的轮换周期。
- 在会议上，与运营工程师和开发人员就值班人员自主对事故待办事项的优先级判定达成一致。

因此，产品负责人其实不需要投入太多时间，就能取得很多成果：

- 影响服务的可靠性目标（SLO）——按产品重要性、关键性、客户细分等；
- 影响当服务目标没有实现时所推行的策略；
- 基于生产报告中涉及用户体验影响的真实数据，决定可靠性工作的优先级；
- 影响值班设置（包括其成本）——按服务的重要性、关键性、客户细分等。

和开发与运营主管一样，产品管理主管可能需要召开多次会议，直到认可 SRE。和其他领导一样，他们会在部门内部讨论这个问题。关于进行 SRE 转型的问题，可以用与开发和运营主管相同的方式来回答。SRE 教练需要与产品管理主管接触，直到达成一致，在项目组合层面上将 SRE 转型列为优先事项。

5.4.5 实现联合认同

SRE 教练可能认为，只要运营主管、开发主管和产品管理主管各自认同就足够了。但事实并非如此。为了取得自上而下的认同，最后还有一个决定性的部分，即全部三位领导的一致认同。达成个人协议与达成联合协议可能是完全不同的挑战。这是因为在讨论 SRE 和 SRE 转型的过程中，可能出现多种误解和细微分歧。此外，这可能是三位领导首次共同讨论部门间运营问题上的新型合作模式。

会议应简洁且由 SRE 教练主导。会议邀请的标题可以是“确保 SRE 在项目组合层面获得优先权的最终联合准备”。另外，会议邀请应该说明三个议程：

- 摘要，大家之前在 SRE 方面取得了哪些共识？
- 讨论，产品运营、产品开发和产品管理使用 SRE 进行产品运营方面的合作会是什么样子？
- 问答。

若 SRE 教练能使会议基调更加容易理解，将大有裨益。这可以通过会议的开场白来实现，即说明这次会议的目的是将组织的运营提升到一个全新的水平以及为什么需要这样做。之后，SRE 教练需要重申分别与各位领导达成的协议。这可能带来一些新的问题，也可能澄清一些问题。这是因为就目前来说，SRE 肯定不是领导人心目中首要的目标。他们手头上还有其他好多事情。

另外，他们可能有各种工作和人际关系方面的问题，彼此之间还有其他话题。所有这些都要求 SRE 教练有共情能力。一旦之前单独达成的协议在会上成功得到再次确认，SRE 教练就需要申明未来会在领导及其部门之间建立的新合作模式。首先应该说明新的合作模式相较于目前的产品运营方式的好处，如下所示。

- 目前，由于组织的不同部分在运营问题上的错位，所以无法反映出对生产中的产品提供卓越的客户体验的需求。SRE 提供了一个框架，使组织在运营问题上取得协同。
- 在 SRE 框架下，运营工程师、开发人员和产品负责人共同定义服务要实现的重要的可靠性指标和可靠性目标。运营工程师提供基础设施来度量可靠性目标的实现情况。运营工程师、开发人员和产品负责人共同遵守商定的策略，对商定的可靠性目标的实现负责。
- 此外，运营工程师、开发人员和产品负责人之间就服务值班设置达成了联合协议，从而加强了组织的协同性。

SRE 教练需要就自由和责任的定义陈述进行讨论：

- 运营团队将提供 SRE 基础设施，使其他人也能进行产品运营，并对可靠性进行数据驱动的决策；
- 在运营团队和开发团队如何排班，将在 SRE 转型过程中共同决定；
- 做出数据驱动的可靠性决策的 SRE 指标将被团队和领导层使用；
- SRE 教练与所有产品运营/产品开发团队共同进行 SRE 转型。

在会议中，无需过多深入 SRE 概念金字塔的专业术语。SRE 教练不要试图去教育领导团队，让他们了解 SRE 的概念。会议的目标是为 SRE 建立共同愿景，并确保其在项目组合层面获得优先地位。

问答环节可以探讨 SRE 转型的持续时间，尤其是各种改进所需要的时间。讨论应明确告知每位参与者，SRE 转型是一项长期工作，过程中会取得多个短期胜利。

会议结束时，SRE 教练应强调导入 SRE 是所有人的共识。接下来，教练将把 SRE 作为优先议题纳入项目组合管理议程。他们将通知领导层何时在项目组合管理会议上讨论 SRE 的优先级。

5.4.6 让 SRE 进入项目组合

根据项目管理协会（Association for Project Management）的定义，项目组合管理涉及根据组织的战略目标和交付能力，对方案和项目进行选择、优先级排序和控制。[21]在特定组织中，让 SRE 在项目组合层面上获得优先考虑的起点是组织的一系列战略目标。值得注意的是，并非所有组织成员都清楚战略目标。SRE 教练的任务是识别这些战略目标，并将 SRE 定位为支持这些目标的手段。项目组合管理的负责人会有当前的战略目标清单。

在某些情况下，组织的战略目标清单可能未直接包含可靠性、稳定性等目标，这正好为 SRE 转型提供了切入点。在这种情况下，SRE 可以被定位为支持任何产品相关目标。根据《Google SRE 工作手册》[22]，可靠性是任何系统最重要的特征。该书的作者根据以下两个论点做出了这一声明：

- “如果系统不可靠，那么用户就不会信任它。”
- “如果用户不信任系统，那么一旦有选择，他们就不会使用它。”

换句话说，可靠性是一个系统最关键的特性，因为没有可靠性就无法赢得用户的信任，最终可能导致用户流失。对于已经有用户基础的系统，可靠性无疑是最重要的特性，因为它已经取得了用户的信任。这个论点使我们很容易找到一个或几个 SRE 能提供支持的组织目标。

一旦发现一个或多个适合作为 SRE 转型起点的战略目标，接下来的任务是了解组织如何通过项目组合管理系统处理 SRE 等技术项目。在包含多种倡议的项目组合待办事项中，有些组织将面向客户的商业机会与技术项目一并考虑。技术项目被视为支持这些商业机会的一部分。在这种情况下，商业机会的优先级自动决定了相关技术项目的优先级。但在这种情况下，技术项目不会显现在项目组合待办事项的商业机会层面上。

无论何种情况，SRE 教练都应与项目组合管理主管面谈，解释 SRE 转型的相关内容。项目组合管理主管需要理解，SRE 的意图是通过加强产品运营、产品开发和产品管理间的协同性，以可度量的方式提升系统的可靠性。反过来，项目组合管理主管也能简单解释项目组合管理在组织中的运作方式。特别是，他们能阐述技术项目在项目组合待办事项中的处理方式。

若技术项目在项目组合待办事项的商业机会层面有所体现，项目组合管理主管则可向 SRE 教练提供建议，指导如何构建 SRE 项目，完善它并提交以取得较高优先级，同时将其与一个或多个支持的战略目标建立关联，如图 5.7 所示。

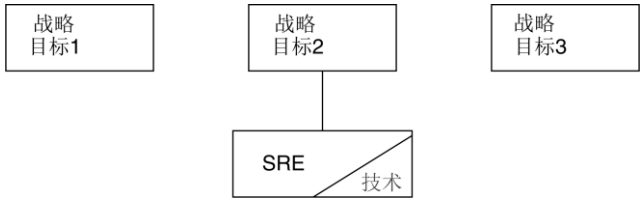

图 5.7　将 SRE 作为一个专门的技术项目插入项目组合

注意，SRE 项目被明确标记为“技术”类别，以区分其作为技术项目的性质。如果技术项目未体现在项目组合待办事项的“商业机会”层面，项目组合管理主管应指导如何将 SRE 整合进其支持的面向客户的商业机会中。这可以通过为项目组合待办事项中的相关项目添加标签和链接等方式实现。在图 5.8 中，SRE 作为战略目标 2 相关商业机会的一部分被列出。具体而言，该商业机会旨在将平台扩展至额外的 10 个地区，以增加收入和市场份额。

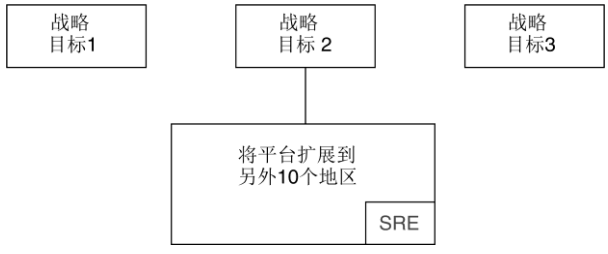

图 5.8　将 SRE 作为面向客户的商业机会的一部分插入项目组合

该商业机会添加了 SRE 标签，表明 SRE 转型是该特定机会工作的一部分。这也表明 SRE 的优先级与该商业机会相绑定，并与其他商业机会一同排序。

在任何情况下，项目组合待办事项中都应能轻松识别 SRE。如果项目组合管理系统具备该功能，应能通过超链接直接查看作为主题的 SRE 及其当前优先级（如果有的话）。这样可以方便地分享超链接，对协调工作至关重要。

SRE 教练承诺，一旦 SRE 的优先级在项目组合会议上确定下来，就要与产品运营、产品开发和产品管理的主管进行讨论。这个日期可以从项目组合管理主管那里获得。对于专门的项目组合待办事项，领导可在优先级排定会议前审查、评论并调整项目。

SRE 教练可能不参加优先级排定会议，但应在会后对决策进行投票。理想情况下，SRE 项目的排名应通过同一个超链接查看。

一旦 SRE 的排名确定，就相当于 SRE 转型获得了在组织内推进的许可。SRE 教练已经达到一个伟大的里程碑，有理由庆祝一下！庆祝活动开启，便是发邮件感谢运营、开发和产品管理的主管。与他们保持良好的关系对 SRE 教练成功实施 SRE 转型至关重要。此外，SRE 教练应通知所有对 SRE 特别兴趣的人。

5.5 自下而上认同

实现自上而下的认同，并且确保 SRE 在项目组合层面获得优先级之后，团队对 SRE 的时间分配方案已有共识和验证。现在，团队能够参与 SRE 活动。参与的团队分为两组：运营团队和开发团队。开发团队的目标是参与构建 SRE 概念金字塔（参见第 3.5 节）。运营团队的目标是构建 SRE 基础设施，以便开发和运营团队开展 SRE 活动。

SRE 教练需要协调各方参与，确保 SRE 基础设施的发展与开发团队的需求同步。与自上而下的认同不同，自下而上的认同是通过时间的推移在一系列辅导会议中逐步建立的。这是因为运营和开发团队需学习并适应新的工作方式，以便将其应用于日常工作。这需要他们改变在职业生涯中养成的一些工作习惯。

5.5.1 与运营团队接触

运营团队可能比开发团队更早采纳 SRE。他们需要学习构建一个可供他人使用的框架，以便于开展 SRE 活动。对运营团队而言，使其他团队能参与 SRE 是一项令人振奋的变革，因为他们一直期望开发人员能够深度参与产品运营，而非仅仅交付新版本。核心挑战在于运营团队开发 SRE 基础设施的能力，而非 SRE 哲学本身，这要求他们掌握新技能和积累经验。

针对运营团队，可以设定若干辅导会议主题，具体内容见表 5.5。

表 5.5 运营团队辅导会议主题

会议主题	简短解释	成功度量
SRE 介绍	介绍团队环境下的 SRE 概念金字塔（3.5 节），需要什么样的 SRE 基础设施，以及 SRE 框架下的责任	运营工程师理解 SRE 概念金字塔，准备着手开发 SRE 基础设施，并知道他们在 SRE 框架下需要做什么
日志基础设施	选择具有功能齐全的日志查询语言的日志基础设施	已经选择、采购一个日志基础设施（功能齐全的查询语言），供开发和运营团队使用
为可用性和延迟 SLI 设置 SLO 的基础设施	实现能存储来自开发团队的可用性和延迟 SLO 定义的基础设施	开发团队使用该基础设施来设置可用性和延迟 SLO
针对可用性和延迟 SLI，检测 SLO 违反情况的基础设施	实现能将可用性和延迟 SLO 定义与端点的实际可用性和延迟进行比较的基础设施	能可靠地检测违反 SLO 的情况

续表

会议主题	简短解释	成功度量
警报基础设施	实现能及时有效地对违反 SLO 的情况发出警报的基础设施	开发团队为警报的及时性（警报会很快发出，但不会过早）和有效性（不要有太多警报）提供了正面反馈
图表基础设施，用于显示和各个 SLI 对应的 SLO 的实现情况	实现各种仪表盘，显示和 SLI 对应的 SLO 在一段时间内的实现情况。	开发团队使用 SLI/SLO 仪表盘来查看 SLO 在不同时间单位内的实现情况
生成错误预算消耗图的基础设施	实现显示不同时段的错误预算消耗的仪表盘	开发团队使用错误预算消耗仪表盘来检查给定时间单位内的剩余错误预算
为基于错误预算的决策生成图表的基础设施	为产品负责人和经理实现仪表盘	产品负责人和经理使用仪表盘来做出基于错误预算的决策
用于自定义 SLI 的基础设施	实现一种设施，可以接受基于预定义日志结构的常规输入，并从中制作一个完全体的 SLI	开发团队使用基础设施来定制除了可用性和延迟之外的其他 SLI。
自助式 SLO 调整工具	实现一个工具，使开发团队能自行调整 SLO	开发团队使用该工具按需调整其 SLO，而无需联系运营团队。
SRE 基础设施的自助式配置	在 SRE 基础设施中实现配置点，实现整个团队和服务范围内的调整	开发团队使用提供的配置点来调整警报算法、仪表盘等
值班管理基础设施	选择一个值班管理基础设施	已经选择、采购了一个值班管理基础设施，并提供给开发团队使用
为 SRE 基础设施提供支持	实现请求紧急错误修复、一般错误修复和新功能的一个过程，并提供培训支持	没有产生太多对 SRE 基础设施的支持请求。根据使用该基础设施的团队的反馈，能尽快予以满足

5.5.2 与开发团队接触

开发团队需要了解的是，为什么说使用运营团队提供的 SRE 框架是个好主意？如何使用？如何通过值班来参与产品运营？如何对可靠性方面的投入做出数据驱动的决定？

在这里，为了获得开发团队的认同，挑战的核心是让开发人员转向 DevOps，让他们参与运营。为此，他们的思维方式需要发生转变。可以肯定的是，产品负责人也需要一种新的思维方式。他们需要让开发团队率先在生产中发现产品问题，通知利益相关者正在处理的问题，并在客户投诉开始升级之前解决这些问题。这是不是与产品负责人一直以来只要求开发新功能这一臭名昭著的形象不一样？

图 5.9 展示了新的思维方式。开发团队要为自己设定新的标准，即率先发现事故，通知利益相关者，并在事故导致客户投诉升级之前完成修复。

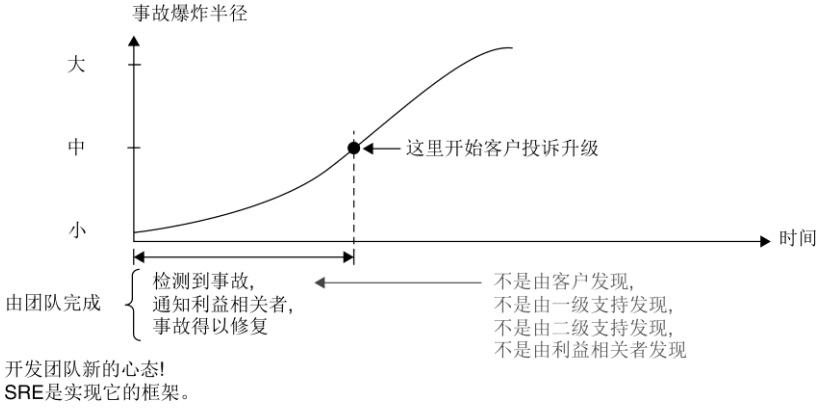

图 5.9　开发团队在运营方面要有的新心态

也就是说，开发团队对自己的要求转变了——生产事故不再是由客户、一级或二级支持或利益相关者发现，而是由他们自己发现。为了实现这个目标，开发团队需要有自己的监控。这就是运营团队提供的 SRE 基础设施能起的作用。

表 5.6 展示了为开发团队设置的辅导会议主题。

表 5.6　开发团队辅导会议主题

会议主题	简短解释	成功度量
SRE 介绍	介绍团队环境下的 SRE 概念金字塔（3.5 节），需要什么样的 SRE 基础设施，以及 SRE 框架下的责任	开发人员和产品负责人理解 SRE 概念金字塔，准备在他们的团队中开始实现 SRE，并知道他们在 SRE 框架下需要做什么
日志记录	确保服务使用由运营团队推荐的基础设施来进行日志记录	所有服务都使用由运营团队推荐的同一个日志基础设施，并生成适合生产环境的日志记录
日志查询	学习日志查询语言，能以编程的方式自定义查询	开发人员使用日志查询语言来回答有关生产中系统状态的问题
可用性 SLO	从客户的角度确定最重要的工作流，并为端点设置可用性 SLO	开发团队就初步的可用性 SLO 以及后续审查和调整它们的过程达成一致
SLI/SLO 仪表盘	了解 SLI/SLO 仪表盘	开发团队使用 SLI/SLO 仪表盘来检查 SLO 在一段时间内的实现情况

续表

会议主题	简短解释	成功度量
错误预算消耗仪表盘	了解错误预算消耗仪表盘	开发团队使用错误预算消耗仪表盘来查看剩余的错误预算及其消耗趋势
基于错误预算的决策仪表盘	了解基于错误预算的决策仪表盘	开发团队使用基于错误预算的决策仪表盘，对可靠性投入做出基于错误预算的决策
延迟 SLO	从客户的角度确定最重要的工作流，并为端点设置延迟 SLO	开发团队就初步的延迟 SLO 以及后续审查和调整它们的过程达成一致
对违反 SLO 的情况做出反应	在开发团队中建立一个规程，对违反 SLO 的情况做出恰当的反应	开发团队按照团队的约定，定期对违反 SLO 的情况做出反应
确定 SLO 的违反是否真的意味着发生了糟糕的客户体验	检查违反事先定义的 SLO 时是否真的意味着客户体验糟糕	开发团队定期调整 SLO，确保在违反这些 SLO 时真的意味着发生了糟糕的客户体验
基于错误预算的决策	了解如何利用错误预算来进行数据驱动的可靠性投资决策	开发团队实践基于错误预算的决策，为可靠性方面的投资决策提供指引
错误预算策略	为团队定义错误预算策略	开发团队定义好错误预算策略，并根据需要实行
轮流值班	通过定义要负责值班的角色和人员，在什么时候值班，要具有什么样的技术知识，并建立交接程序，从而为服务设置好轮流值班	开发团队按照运营工程师、开发人员和产品负责人之间的约定，实行值班制度
额外的 SLI	定义并设置除了可用性和延迟之外的其他 SLI（例如，对队列消息进行处理的正确性，参见图 3.1）	开发团队确定了除了可用性和延迟之外的其他 SLI，并为其设置了 SLO
利益相关者	确定服务的利益相关者	利益相关者名单得到了所有利益相关者的认同
事故优先级排定	从客户的角度定义事故的优先级；为每个事故优先级分配一组任务	值班人员根据事故优先级采取相应的行动
利益相关者通知	确定对利益相关者重要的场景；设置向利益相关者发出的通知	利益相关者收到通知，并提供正面反馈，认为这些通知有意义，而且及时送达

5.6 横向认同

随着自上而下和自下而上的认同变得明确，组织的中层管理者也要参与进来。在获得自上而下的认同后，SRE 教练应与运营经理、开发经理和产品经理沟通，解释已达成的协议，并通报即将进行的项目组合优先级排序计划。一旦 SRE 在项目组合层面获得优先级，经理就需要负责在团队中建立 SRE 相关流程。

只有运营和开发团队在自下而上的认同过程中明确 SRE 对各自团队的意义并获得了 SRE 实践经验，建立流程才显得有意义。也就是说，SRE 首先以临时方式实施，积累经验之后，才能将实践规范化为组织流程。

一旦制定组织流程来制度化 SRE 的时机成熟，SRE 教练就要根据与各团队的互动经验，为这些流程提供建议。通过接触，SRE 教练将识别出哪些做法在哪些情况下可行及其原因，以及哪些做法在哪些情况下不可行及其原因。此外，SRE 教练可能发现，在制度化 SRE 流程之前，组织需要优先解决人员配置问题。

这些流程应被记录并与所有受影响的团队共享和审查，这些团队共同构成整个产品交付组织。收集到的审查反馈应整合到流程中。根据反馈引起的变更程度，可能需要进行额外的审查轮次。过度沟通优于误解。因此，面对疑问时，应优先选择充分沟通，以降低误解风险。

一旦流程确定，就要通过口头和书面方式进行广泛的沟通。表 5.7 概述了这些流程应该包含的关键要点。

表 5.7 SRE 过程要点

过程要点	解释
值班设置	哪些角色在哪些时间负责值班哪些服务？
基于错误预算的决策	哪些决策是团队要基于错误预算来做出的？
责任	在产品运营领域，谁负责什么？
自由和责任分割	团队享有的自由和团队要履行的不可协商的责任是什么？
监管合规	哪些 SRE 工件与监管合规有关？哪些 SRE 工件在审计期间是相关的？
升级策略	如果涉及的人解决不了问题，应该由谁来破局？
变更管理	如何请求对 SRE 过程的变更？

中层经理的横向认同体现在他们建立的流程中。对 SRE 教练而言，最重要的是推动这一流程，并确保在组织内取得正式 SRE 实践与非正式 SRE 实践的平衡。

5.7 交错认同

自上而下、自下而上和横向的 SRE 认同不必同时发生。例如，在以规则为导向的稳定产品交付组织中，三种认同可能依序进行。针对这种情况，图 5.10 展示了说服流程。

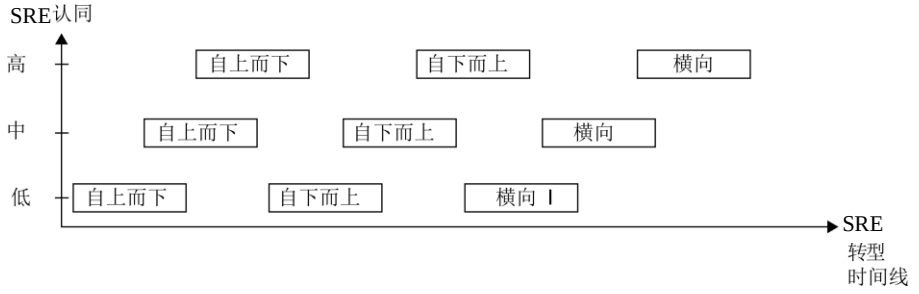

图 5.10　在稳定且以规则为导向的文化中的 SRE 交错认同流程

首先，作为 SRE 营销漏斗（5.2 节）的一部分，SRE 教练与整个组织的人员广泛接触。在这个时候，对 SRE 的认同度在整个组织中是相当低的。接着，SRE 教练开始与产品交付组织的领导层进行大量接触。如果成功，这将提升领导层自上而下的认同。最终，SRE 将成为项目组合层面的优先事项，获得许可进入整个组织。

此后，SRE 教练开始接触运营和开发团队。经过一系列长期的辅导会议，团队对 SRE 的理解和经验逐步增长。如果成功，将在适当的时候实现自下而上的深度认同。在组织已经有了足够多的 SRE 经验后，可以把各种 SRE 流程制度化。为此，SRE 教练与开发、运营和产品经理接触，建立和沟通过程，在组织内落实 SRE。如果出现多个“负责”或“拥有”SRE 流程管理的经理，那么表明已经达到了规模化 SRE 高度认同。

在不稳定的企业环境中，说服流程可能无法线性进行。可能需要同时进行一系列复杂且结果难以预测的讨论。这些讨论可能涉及政治因素，部分协议可能只是初步达成。这些临时协议可能需要作为正式协议向其他人宣布，否则难以推动进展。最终，一旦所有人都同意，临时协议将转变为正式协议。在某些组织文化中，可能确实需要这种有时令人不快的“灵活性”，以促进组织上下对 SRE 的认可。SRE 转型更像是一门艺术，而非科学。

5.8 团队辅导

如 2.5 节所述，团队教练或团队辅导是推动产品交付组织实施 SRE 转型的主要手段。SRE 教练要制定日程表，以辅导不同团队。团队会议的议程要提前发送。

在运营团队中，负责实现 SRE 基础设施和提供第三方工具的运营工程师应持续参与辅导会议。此外，运营部门中应指定一人担任 SRE 基础设施的产品负责人，并参与持续辅导。

在开发团队中，开发人员、商业分析师和产品负责人应始终参与辅导。UI/UX 设计师无需频繁参加会议。然而，SRE 教练应创造一个包容的环境，鼓励设计师选择性地参加 SRE 辅导会议。从一开始就应明确商业分析师和产品负责人的参与。表 5.8 解释了商业分析师和产品负责人为什么必须持续参与 SRE 辅导会议。

表 5.8　商业分析师和产品负责人参加 SRE 辅导会议的理由

理由	解释
团队文化	SRE 旨在引入新的工作方式和决策方式。通过以数据驱动的方式共同定义 SLI、SLO、错误预算策略、值班设置以及在可靠性上的投入，在团队中建立了一种新的关于可靠性的文化。然而，只有当所有团队成员都参与 SRE 导入过程，这种文化才能很好地孕育并最终开花结果
组织协同	SRE 的优势是在运营工程师、开发人员和产品负责人之间建立协同（参见 1.2 节）。然而，只有商业分析师和产品负责人都持续参与 SRE 活动，这种协同才能真正建立起来
用户视角	在产品的用户身上，商业分析师或产品负责人花的时间比团队中其他任何人都多。为了从用户角度定义 SLI/SLO，把它们作为用户满意度的代理度量（代用指标），只有足够接近用户的人能做出正确判断。否则，SLO 就不能真正反映用户的满意度，而只能反映系统的技术完善程度。这样一来，即使出现违反 SLO 的情况，也无法判断它的真实意义，修复它的动机也就不那么明确了
客户视角	这和用户视角相同，只是针对的是客户。客户 SLO 的定义可能需要比用户 SLO 更严格，因为可能需要为付费客户提供更高的服务水平。同样地，只有足够接近客户的人才能确定客户满意的服务水平
SLI/SLO 的可理解性	如果商业分析师或产品负责人在 SRE 辅导会议中没有出现，就只有技术团队的成员（如开发人员和运营工程师）参与 SLI 和 SLO 的定义。这样一来，所定义的 SLI 和 SLO 会扎根于技术。最终，只有技术人员才能理解这种 SLI 和 SLO。然而，这与 SLI 和 SLO 作为用户满意度的代理度量标准的目标相矛盾，它们应该是每个人都能理解的

如果商业分析师或产品负责人经常缺席 SRE 辅导会议，SRE 教练就要安排一次非正式的会面。根据《非暴力沟通》[23]一书中的指导，SRE 教练应按照观察、感受、需求、请求的顺序引导对话。对话首先指出商业分析师或产品负责人缺席上次 SRE 辅导会议的事实。接着，教练要表达自己的感受，如沮丧、不快、不安或担忧。之后，教练依据表 5.8 阐明商业分析师或产品负责人持续参与 SRE 辅导会议的重要性。最终，SRE 教练委婉地邀请他们参加未来举行的 SRE 会议。

商业分析师或产品负责人可能会解释，他们认识到了 SRE 的重要性和自己的角色，但之前因为其他事务无法参加。在这种情况下，他们可与 SRE 教练商定后续参与的日期。然

而，如果他们的行为未有改变，SRE 教练应继续与团队其他成员保持联系。尽管如此，SRE 的实施可能未能完全发挥潜力，但其效益仍足以证明这样的努力是合情合理的。

实战经验 在实际工作中，产品负责人有时会主动要求参加 SRE 会议，可能是团队成员持续要求的，或业务需求签署 SLA（服务水平协议）。

SRE 教练应在辅导会议中负责记录，并确保记录在公共区域保存。随着团队越来越熟悉 SRE，应逐步承担记录工作，目的是逐步减少 SRE 辅导会议，使 SRE 活动能够自动持续。记录人员应在每次辅导会议后向整个团队分发会议记录。通过历史记录，可以追踪团队随时间取得的进展，这是非常有成就感的。

5.9 组织穿越

在大型产品交付组织中，可能会有若干个运营团队和很多个开发团队。面对这种情况，应该首先关注哪些团队？随后跟进哪些团队？如何最有效地在组织内推广 SRE？这个进程可分为三个大组：

- 运营团队；
- 与产品运营密切相关的开发团队；
- 与产品运营较远的开发团队。

5.9.1 组织的分组

起点是负责开发 SRE 基础设施的运营团队。该团队最先需要理解 SRE 概念金字塔（参见 3.5 节）。他们应评估现有的日志、监控和警报基础设施，确定其适用性以构建预期的 SRE 基础设施。哪些技术将用于开发？团队中谁有开发能力？谁有开发框架以供他人使用的经验？

一旦确定适合 SRE 目的的日志基础设施，运营团队就可以启动开发团队的辅导会议。开发团队可以分为两组。一组由在 SRE 转型开始前就接近产品运营的团队组成。这些开发团队需要根据具体情况创建和设置警报，以及按计划响应警报等。尽管他们的做法可能并不完全符合 SRE 的结构和严谨性要求，但他们的工作已接近于产品运营。

这类开发团队在 SRE 引入前已对 DevOps 有所了解。对这些团队而言，SRE 转型意味着从传统产品运营过渡到标准化的 SRE 实践。这些开发团队适合作为 SRE 转型的起点，因为他们已具备基本心态。

然而，与其他开发团队相比，这类团队可能更快向运营团队提出对 SRE 基础设施的要求。在这种情况下，SRE 教练需平衡开发团队对 SRE 基础设施的要求与运营团队实施这些要求的能力和速度。这要求做出合理的平衡：

- 确保 SRE 基础设施的功能缺失不会使开发团队处于不满状态，从而影响其对 SRE 的积极性；
- 运营团队不应因为同时开发过多功能而劳累过度。

需要注意的是，运营团队可能缺乏开发框架的经验，而这些框架需要供其他人使用。因此，不应假设 SRE 基础设施功能的首次成功率。

另一组开发团队由那些不直接参与产品运营的团队构成。这些团队不直接在生产中运营服务，而是依赖运营团队，并且只接收缺陷报告。在这类情况下，可进一步细分为两个小组。一个小组对现状不满，希望深入了解产品运营，不希望运营团队作为生产环境的中介。这些团队欢迎 SRE，因为它提供了一种结构化的方式，使开发团队能够更接近产品运营。另一个小组则对现状满意。他们完全专注于功能开发。产品运营是运营团队的事。运营团队报告来自生产中的缺陷，并由开发团队修复。开发团队觉得没必要更多地参与产品运营。在这种团队中，在引入任何 SRE 实践之前，首先需要转变基本的思维方式（心态）。这些团队可能将觉得 SRE 的价值主张可能危及其宝贵的时间，担心影响功能开发的时间。

图 5.11 提供了一个产品交付组织的示例图。

组织初期的 SRE 推广应采用深度优先策略，从左至右进行，直至 SRE 转型能在所有团队中同步执行。有趣的是，对 SRE 基础设施的需求呈相反趋势增长。相对于左侧团队，右侧团队对 SRE 基础设施功能的需求较少。下一节将详细阐述这一点。

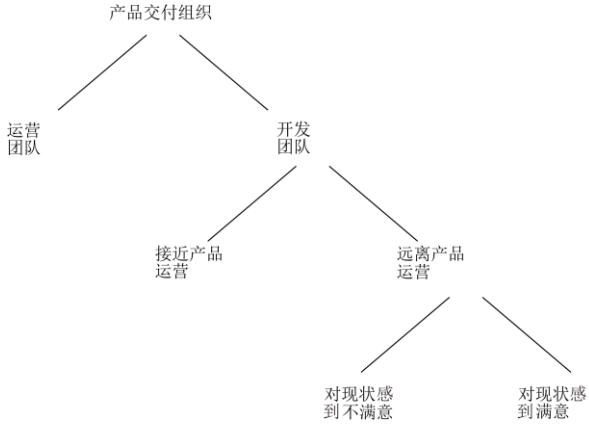

图 5.11　SRE 转型的团队分类

5.9.2 组织穿越与 SRE 基础设施需求

图 5.12 展示了组织穿越的流程。在 SRE 导入的前两个月，首先接触运营团队，负责开始构建 SRE 基础设施。接下来与接近产品运营的开发团队接触。这些团队已具备正确的心态，能迅速投入实践。然而，他们的需求迫切性与运营团队建立基础设施的能力之间需要保持平衡。

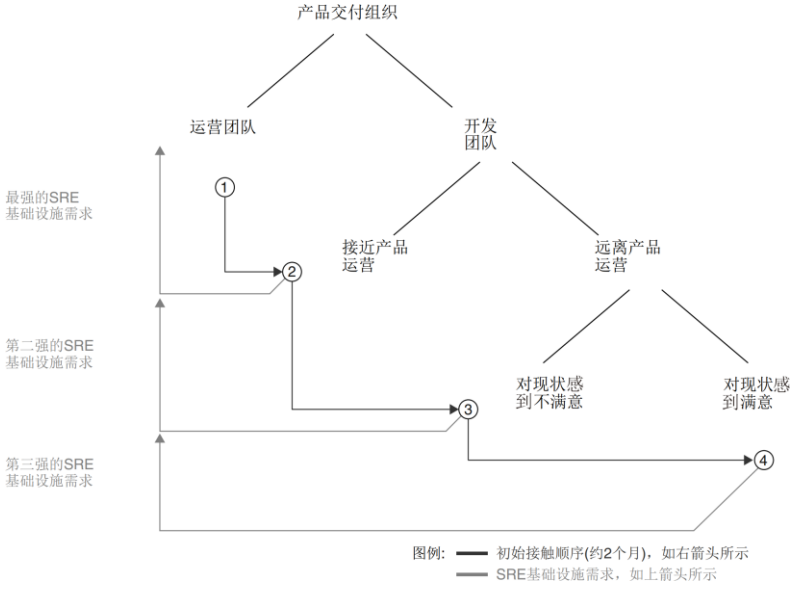

图 5.12　穿越组织以及不同团队对 SRE 基础设施的需求

要接触的第三类团队是对远离产品运营现状感到不满的。他们期待 SRE 的引入能带来改善。尽管他们的心态还需培养，但预期的变革终将实施。技术变革将逐步启动。因此，他们对 SRE 基础设施的需求不如早期接触的团队迫切。

最后接触的是那些对现状感到满意的团队。在进行任何技术变革之前，首先要进行必要的思维方式的转变，这需要一定的时间。这些团队对 SRE 基础设施的需求最弱。很可能的情况是，当这些团队准备请求 SRE 基础设施功能时，这些功能已基于其他团队的需求开发完成。

5.9.3 接触各个团队的时机

图 5.13 展示了 SRE 教练随着时间的推移与各个团队进行接触的情况。在 SRE 转型的前两个月，SRE 教练开始接触有限数量的团队。首先接触运营团队，然后是接近产品运营

的开发团队。对远离产品运营不满意的开发团队排在第三，对远离产品运营满意的开发团队排在最后。

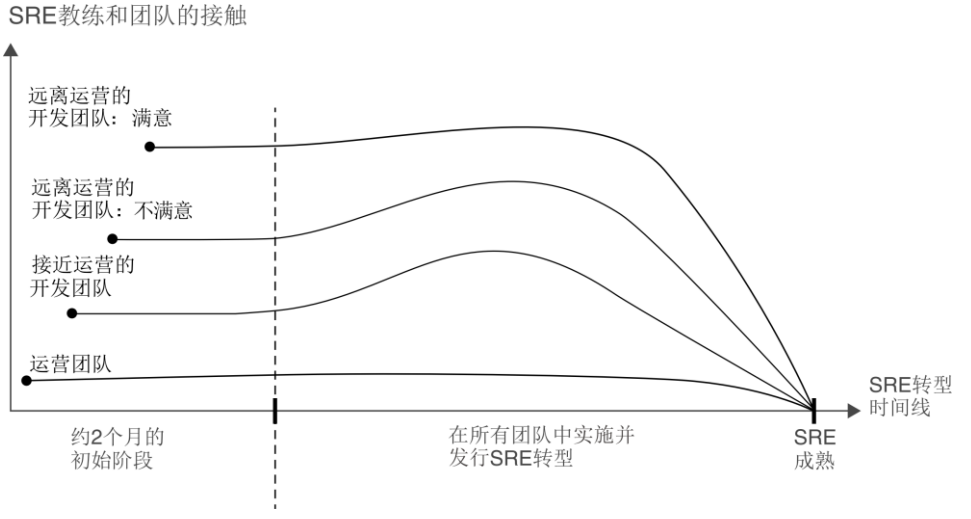

图 5.13　SRE 教练随着时间的推移与团队的接触情况

在转型初期，SRE 教练调整新团队的参与，使其可以量力而行——个人的能力、运营团队构建 SRE 基础设施的能力以及开发团队对基础设施的需求。一旦 SRE 教练确认基础水平的 SRE 基础设施已准备就绪，就可以决定在所有团队中同时推进 SRE 转型。这必然会极大地推动 SRE 运动在组织内的发展。

在所有团队中同步推动 SRE 转型，须具备以下条件：

- SRE 基础设施需要支持可靠性的两个基本 SLI：可用性和延迟（见 3.1 节）；
- SRE 基础设施需要支持 SLO 定义；
- SRE 基础设施需要基本警报来支持一些违反 SLO 的情况；
- SRE 基础设施的使用需要被几个开发团队所证明；
- SRE 基础设施的所有权及其维护是明确的，并不含糊；
- SRE 运动正在兴起，越来越多的团队想要了解 SRE，并等着加入 SRE 转型。

经由这些条件，SRE 教练可以打开闸门，邀请其他所有团队加入 SRE 转型。SRE 教练可能会感到不确定，因为他们可能需要面对更多的团队（相较于初期）。然而，SRE 教练应认识到，他们已积累了引导团队参与 SRE 的经验。此外，SRE 教练需要意识到，各团队采用 SRE 的速度会有不同。设法安排会议，避免 SRE 教练一次性承受过大的压力。毕竟，他们负责安排会议、调整团队参与的节奏，并确保开发团队对 SRE 基础设施的需求与运营团队的交付能力相匹配。

此外，目前 SRE 运动仍然处于起步阶段。成功的案例尚属少数。因此，当前目标是支持 SRE 运动，增强其可持续性。为此，SRE 教练值得冒险，采取更积极的策略并掌控 SRE 推动过程。

实战经验 少数 SRE 教练能够同时指导数十个团队实施 SRE 转型。在 SRE 转型的前两个月，会议频率将显著增加，因为所有团队都在同步推进。然而，由于各团队采纳 SRE 概念金字塔中的概念速度不同，每周会议次数将逐渐减少。团队可能要求安排辅导会议以满足其他需求，某些会议可能需要团队完成当前任务后进行，而一些进展可能依赖于尚待开发的 SRE 基础设施功能等。最终，一些团队可能选择在 SRE 概念金字塔的较低层级停止攀登，而其他团队可能迅速达到顶端。这两种情况都可能减少 SRE 教练的进一步参与需求，从而减少会议次数。换句话说，SRE 教练无需担心指导的团队太多。随着 SRE 转型的推进，辅导工作自然可以落实到位。

一旦启动，潜在好处将远远超过风险。随着更多团队加入 SRE 转型，知识共享的机会也随之增多：

- 更多成功故事将在精益咖啡会和午餐会中被分享；
- 新建的 SRE 实践社区（CoP）日趋成熟；
- 团队可能展开关于其服务运营的新对话，这是之前缺乏 SRE 基础数据而不可能进行的。

换言之，精益方法中限制工作进展的原则适用于 SRE 转型最初的两个月。然而，一旦过了初始阶段，应尽可能加快工作进度，全力推进组织内的 SRE 运动。若运动缺乏活力，整个倡议可能无法在组织内成功推广。

在 SRE 转型过程中，实现组织 SRE 成熟之前，最大化教练的工作进度不会延长总工作时间。这是因为 SRE 教练的终极目标是变得不再必要。每次与团队的 SRE 辅导会议都是朝这个目标迈进的一步。实际上，加快 SRE 教练的工作进度有助于整个组织更快实现 SRE 成熟。这对项目组合管理至关重要。

将 SRE 纳入组织的倡议项目组合本来就有挑战，维持其地位同样困难。其他倡议持续被评估，并与现有倡议竞争项目组合中的位置。在实现 SRE 成熟前，若缺乏明确的成功信号，SRE 倡议可能面临在项目组合层面被降级或取消的风险。

一旦 SRE 成熟，这种风险将消失，SRE 理念和工作方式将成为组织内处理业务问题的标准组成部分。实际上，SRE 应从倡议项目组合列表中移除，因为不再需要推动 SRE 转型。SRE 已成为组织运营的标准方式。SRE 的基础设施已经建立起来，而且有人维护。每次成立新的团队，都可以利用所有这些资源，基于 SRE 来建立产品运营，因为其他团队都是这么做的。

5.10 组织辅导

通过本章的讨论，可以看出通过信息分享和交流来推动 SRE 运动的重要性。这些活动体现的是组织辅导，因为它们跨越团队和角色引发了各种各样偶然的对话，使人们在理解 SRE 的过程中不断成长。这样的对话很重要，因为 SRE 涉及产品交付组织中的所有角色，而不同的人以各自的方式实践相同的角色。可以应用表 5.9 总结的分享方法在整个组织内分享特定的信息。

为促进除精益咖啡、午餐学习和 SRE CoP 会议等讨论会之外的信息共享，应尽量记录会议内容。如果会议是在线进行的，应默认进行记录。在视频分享服务可用的情况下，为每个记录添加 SRE 标签。这样一来，便可以通过单一超链接轻松访问所有与 SRE 相关的记录。

为展示 SRE 的整体进展，可使用看板（Kanban），它展示了所有团队在 SRE 成熟过程中经历的典型阶段。表 5.10 提供了一个看板示例。值得注意的是，每个组织应根据自身情况定制类似的看板。

表 5.9　在整个产品交付组织内共享 SRE 信息的方法

信息	分享方式	解释
SRE 成功故事	精益咖啡、午餐学习、工程博客	在精益咖啡中，参与者需要对主题进行投票。如果 SRE 被选中，说明大家对这个话题普遍兴趣很高。午餐学习是一个很好的载体，可以在轻松的氛围中传播正面信息。如果组织建立了工程博客，成功故事的分享就是一个很好的主题。如果没有这种博客，可以考虑开始记录关于 SRE 的博客
更大型的事后回顾	精益咖啡	在精益咖啡中，投票给事后回顾的人越多，从别人犯的错误中学习的文化氛围越浓厚。随着团队的 SRE 成熟度越来越高，对事后回顾的兴趣应相应地增长
关于 SRE 整体进展的报告	精益咖啡、信息公告板	团队之间要展开良性的竞争
技术讨论	SRE CoP	在 CoP（实践社区）中，技术工作方式可以跨团队有效地分享。很容易邀请人加入 SRE CoP
SLA	全体员工会议	组织通过合同来约定的 SLA 可以在全体员工会议上有效地广播。应该强调，SLA 是基于各团队之前的 SRE 工作定义的。通过这样的沟通，经理们会认同之前在项目组合层面上将 SRE 定为优先事项的决定

表 5.10　用于可视化 SRE 转型进展的看板

1	2	3	4	5	6	7	8	9	...
引介	启用日志	设置初始 SLO	响应 SLO 违反	持续调整 SLO	定义错误预算策略	执行错误预算策略	定义利益相关者	启用利益相关者通知	...
	团队 A			团队 D			团队 I		
			团队 B	团队 E		团队 H			
			团队 C	团队 F	团队 G			团队 J	

表 5.10 所示的看板具有以下重要的优势：

- 清晰展示从起步到 SRE 成熟的过程（位于看板顶部的两条横线）；
- 能够迅速识别出参与 SRE 转型的所有团队（位于看板底部的三条横线）；
- 标示各团队在 SRE 引入过程中的具体位置（看板中的各个单元格）；
- 揭示了团队集中的阶段，即组织中大多数团队所处的阶段（看板的第四步和第五步）。这些信息有助于识别 SRE 转型中最关键的教练和团队工作重心。

使用看板来展示组织范围内的统计数据，能够在人们日常经过时提醒他们，从而增强 SRE 倡议的重要性。尽管这种做法看似微不足道，但能促进更多关于 SRE 的讨论，为 SRE 运动提供进一步的支持。

5.11　小结

在组织层面上获得对 SRE 的认同，标志着 SRE 转型的初步成功。自上而下的认同意味着让产品交付组织的领导层支持 SRE，并在组织的倡议项目组合中优先考虑 SRE。自下而上的认同涉及与运营和开发团队共同构建 SRE 概念金字塔（参见 3.5 节），除旧布新，摒弃旧的工作方式，学习新的实践。横向认同涉及与运营经理、开发经理和产品经理沟通，正式使其能够负责或拥有 SRE 过程管理。

获得组织认同后，团队和个人可着手开展具体的 SRE 转型活动。

假设有一个产品交付组织与 2.1 节描述的相似。在这样的组织里，开发人员可能不明白自己为什么需要参与运营。运营工程师可能未能提供合适的框架来支持开发人员参与运营。经理们可能不会支持这一努力，更不会提供资金。然而，最有可能团结组织的力量在于共同的愿望——减少因生产故障导致的客户投诉频繁升级。

组织内部有一些人可能了解 SRE 能够控制生产运营。因此，该组织内可能有一些人，致力于实现 SRE。他们有望成为 SRE 教练。因此，该组织的首要步骤是明确自己的 SRE 基础现状。为此，需要分析多个关键的维度，包括组织本身、人员、技术、文化和流程。接下来，我们将逐一讨论这些维度。

注释

1 Westrum, R. 2004. “A Typology of Organisational Cultures.” Quality and Safety in Health Care 13 (suppl_2): ii22–27. https://doi.org/10.1136/qshc.2003.009522

2 Longman Dictionary of Contemporary English. 2014. “Definition of movement.” https://www.ldoceonline.com/dictionary/movement

3 https://en.wikipedia.org/wiki/AIDA_(marketing)

4 Thorne, Stephen. 2018. “Getting Started with Site Reliability Engineering.” YouTube, July 11, 2018. https://www.youtube.com/watch?v=c-w_GYvi0eA

5 “SRE for Everyone: Making Tomorrow Better Than Today.” SlideShare IOS, May 2, 2019. https://www.slideshare.net/Rundeck/sre-for-everyone-making-tomorrow-better-thantoday-devops-days-austin-2019

6 Microsoft. 2019. “Monitoring Your Infrastructure and Applications in Production.” YouTube, April 2, 2019. https://www.youtube.com/watch?v=Si6ehIr6kjw

7 Murphy, Niall Richard, Betsy Beyer, Chris Jones, and Jennifer Petoff. 2016. *Site Reliability Engineering: How Google Runs Production Systems*. Sebastopol, CA: O’Reilly Media

8 Beyer, Betsy, Niall Richard Murphy, David K. Rensin, Stephen Thorne, and Kent Kawahara. 2018. *The Site Reliability Workbook: Practical Ways to Implement SRE*. Sebastopol, CA: O’Reilly Media

9 Hidalgo, Alex. 2020. *Implementing Service Level Objectives: A Practical Guide to SLIs, SLOs, and Error Budgets*. Sebastopol, CA: O’Reilly Media

10 Welch, Nat. 2018. *Real-World SRE: The Survival Guide for Responding to a System Outage and Maximizing Uptime*. Birmingham, UK: Packt Publishing Ltd

11 SRE Weekly. Weekly online newsletter published by Lex Neva. Available at https://sreweekly.com

12 Adzic, Gojko, and David Evans. 2014. Fifty Quick Ideas to Improve Your User Stories. Woking, UK: Neuri Consulting LLP.

13 Hidalgo, Alex. 2020. *Implementing Service Level Objectives: A Practical Guide to SLIs, SLOs & Error Budgets*. Sebastopol, CA: O’Reilly Media

14 https://zh.wikipedia.org/wiki/ISO/IEC_27001

15 Rensin, Dave. 2016. “Introducing Google Customer Reliability Engineering.” Google Cloud, October 10, 2016. https://cloud.google.com/blog/products/devops-sre/introducing-a-new-era-of-customer-support-google-customer-reliability-engineering

16 Forsgren, Nicole, Jez Humble, and Gene Kim. 2018. *Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations*. Portland, OR: IT Revolution Press. 中译本《加速：企业数字化转型的 24 项核心能力》

17 “Use Four Keys Metrics Like Change Failure Rate to Measure Your DevOps Performance.” n.d. Google Cloud Blog. https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance

18 “DORA Research Program.” n.d. Accessed January 18, 2022. https://www.devops-research.com/research.html

19 DORA. 2021. “State of DevOps 2021.” Google Cloud. https://services.google.com/fh/files/misc/stateof-devops-2021.pdf

20 https://en.wikipedia.org/wiki/User_experience

21 APM. n.d. “What Is Portfolio Management?” Accessed April 14, 2022. https://www.apm.org.uk/resources/what-is-project-management/what-is-portfolio-management

22 Beyer, Betsy, Niall Richard Murphy, David K. Rensin, Stephen Thorne, and Kent Kawahara. 2018. *The Site Reliability Workbook: Practical Ways to Implement SRE*. Sebastopol, CA: O’Reilly Media.

23 Rosenberg, Marshall B., and Deepak Chopra. 2015. *Nonviolent Communication: A Language of Life: Life-Changing Tools for Healthy Relationships*. Encinitas, CA: Puddle Dancer Press.