软件研发效能·2024中国年度调查报告

南京大学·软件研发效能实验室

CONTENTS

01

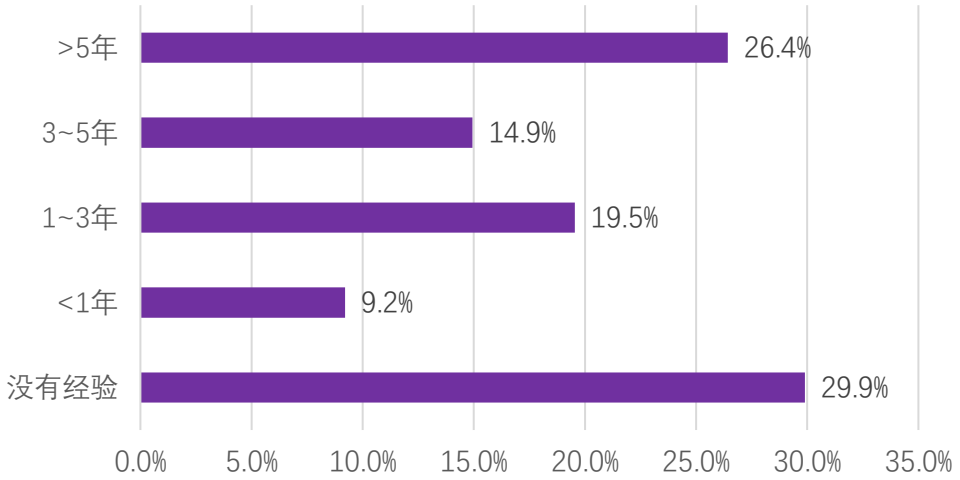
本年度受访对象情况概述

- **基础信息概况**
- **企业规模与安全团队**
- **流程模式与持续交付建设**

基本信息概况

本次受访对象的主要行业分布集中在**互联网/软件**行业，占比高达**63.2%**，远远领先于其他行业。2024年**DevOps工程师**占比**3.4%**，相较于2023年的**8.0%**和2021年的**7.0%**，有着明显的减少，但相应的受访对象组织DevOps实践经验时长则达到了新高，代表行业内DevOps实践能力已经比较成熟，而**AI/算法工程师**占比达到了**12.6%**，说明AI技术的兴起吸引了行业的广泛关注，除此以外受访对象的职位主要分布在**开发工程师**、**测试工程师**和**架构师**。

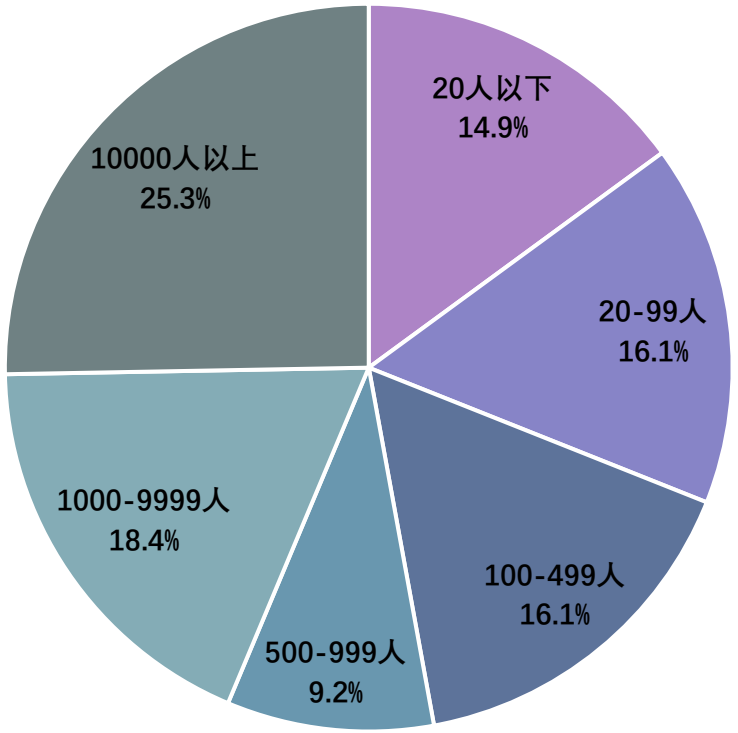
受访对象DevOps实践经验时长分布



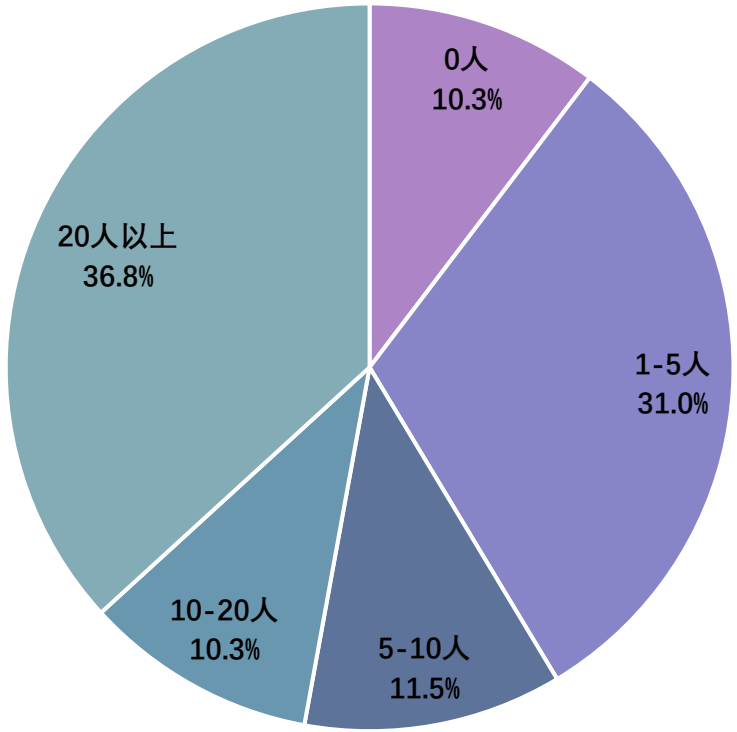
受访对象中有相当一部分已经拥有**至少3年**的DevOps实践经验（占比**41.3%**），包括一些超过5年的组织，这反映出DevOps作为一种软件开发和运维的实践方法，在许多组织中已经较为成熟。同时，DevOps实践处于起步阶段的组织占比显著减少，这表明DevOps在行业内的普及已经达到一个瓶颈。

企业规模与安全团队

受访对象企业规模分布



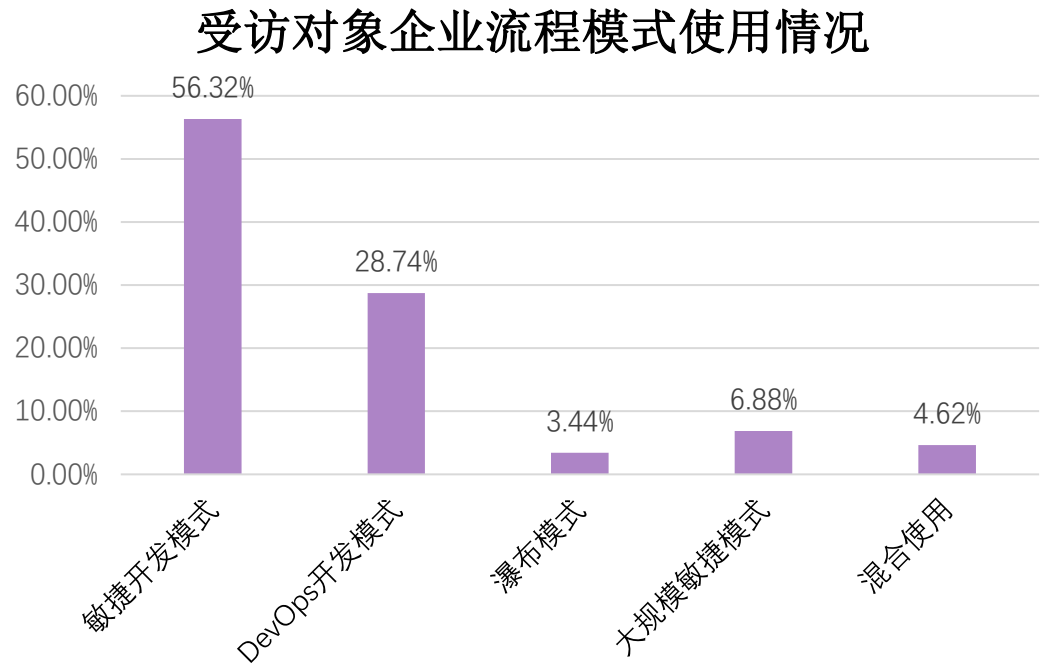
受访对象企业安全规模分布

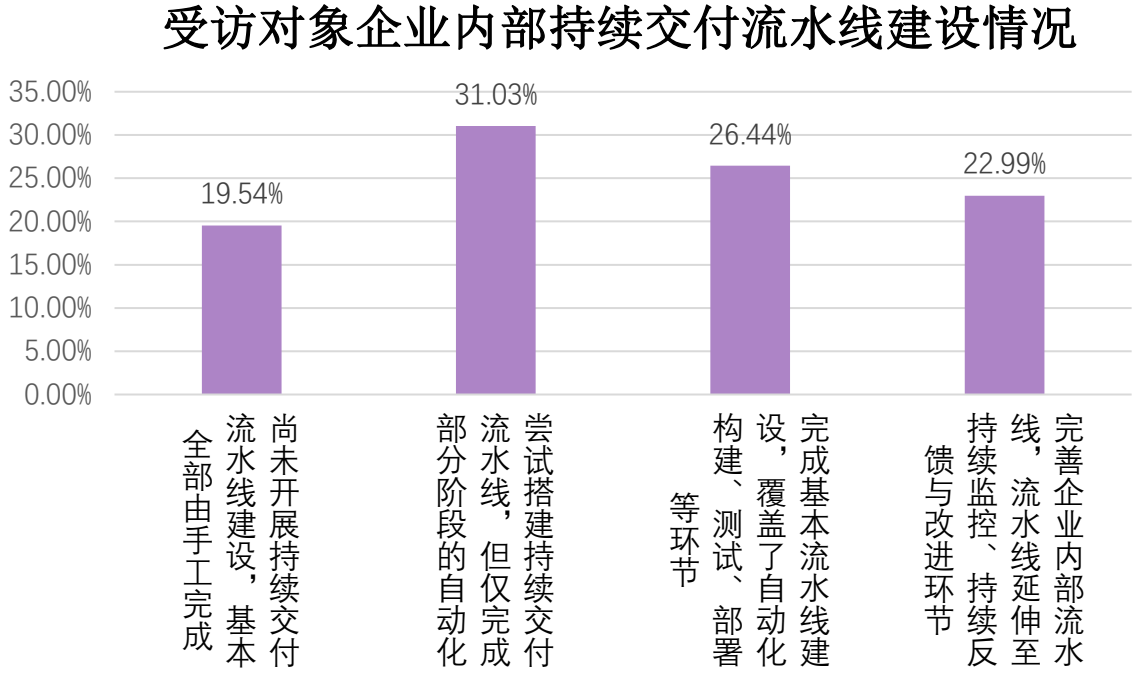


有43.7%的企业员工数超过1000，而员工人数小于500的企业占比达到了47.1%，说明受访对象覆盖了各个规模企业的员工。

企业专职安全人员规模不超过10人的比重高达52.9%，安全的重要性没有得到充分认识，这或许跟受访对象企业中中小型企业数量较多有关。

流程模式与持续交付建设





敏捷开发模式和**DevOps开发模式**占比最多，分别是**56.32%**和**28.74%**，且**敏捷开发模式**占比相比2023年上升了近20%，说明**敏捷开发模式**仍然是当前环境下企业的优先选择，这反映了当前软件开发领域对**快速迭代**和**高效协作**的强烈倾向。

已经开始着手建设持续交付流水线的企业占比相对2023年的**90.3%**略显下降，达到了**80.46%**，有一定数量的组织对建设持续交付流水线仅完成了部分自动化阶段，但也有不少组织已经完成了基本流水线的建设或者更进一步地完善了企业内部的流水线。**这反映出当前企业对于自动化流程的重视程度和热情依然很高，希望通过自动化流程提高软件交付的效率和质量**。

02

研发效能度量

- **新研发效能标准**
- **部署频率与交付周期**
- **变更失败比例与平均修复时长**
- **有效会议时间与客户满意度**
- **代码返工率与平均故障间隔时长**

新研发效能标准

参数	5分	4分	3分	2分	1分
部署频率	1天2次或多次	1天1次~3天1次	3天1次~1周1次	1周1次~1月1次	低于1月1次
交付周期	1周以内	1周~1月	1月~6月	6月~1年	1年以上
平均修复时长	少于15分钟	15分钟～3小时	3小时～1天	1天～1周	1周以上
变更失败比例	0~5%	5%~10%	10%~15%	15%~30%	30%以上
有效会议时间	30分钟以上	20分钟~30分钟	10分钟~20分钟	5分钟~10分钟	5分钟以内
代码返工率	低于3月1次	1月1次~3月1次	1周1次~1月1次	1天1次~1周1次	1天2次或多次
平均故障间隔时长	大于1月	1周~1月	1天~1周	1小时～1天	少于1小时

本年度采用了一种综合评估方法，该方法在原有的**部署频率**、**变更失败率**、**平均修复时长**以及**交付周期**四个核心度量指标的基础上，引入了三个新的评估维度：**有效会议时长**、**代码返工率**和**平均故障间隔时长**。这种评估体系的扩展能够从团队协作效率、代码质量、产品市场表现及产品可靠性等多个角度综合评估研发效能成效。针对每项指标，设计了一个**五分制**的评分模型，允许对各项指标进行量化评估。通过对这七个参数得分的**等权重累加**，最终得出了一个综合的研发效能评分。基于此总评分，特将性能表现位于前**20%**的团队定义为**高效能团队**，而余下的**80%**则被归类为**一般效能团队**。这种方法提供了一个结构化和量化的框架，以评估并区分不同团队在DevOps实践中的效能水平。

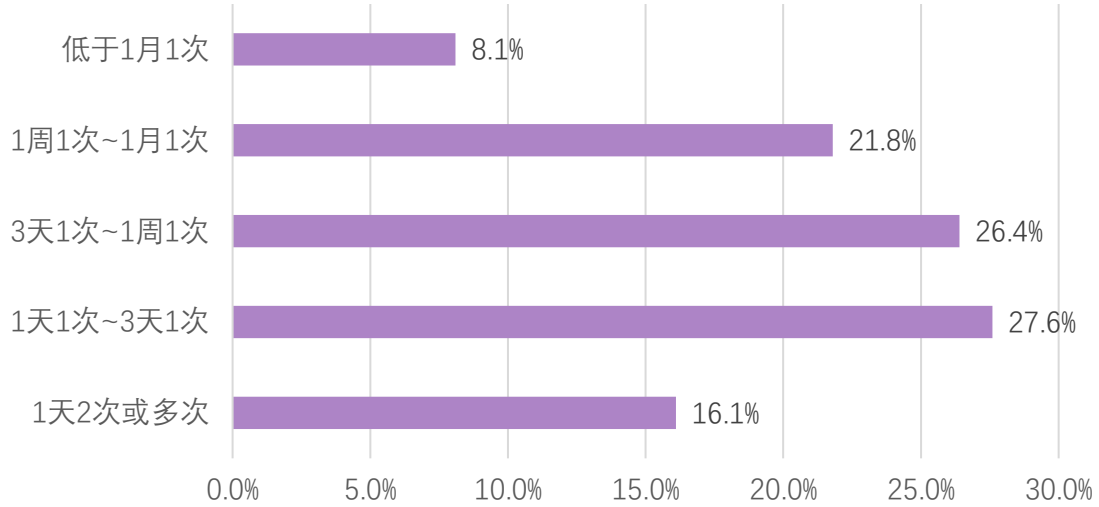
新研发效能标准

参数	高效能团队均值表现	高效能团队中位数表现	高效能团队众数表现	一般效能团队均值表现	一般效能团队中位数表现	一般效能团队众数表现
部署频率	1.2天1次	2天1次	2天1次	1.4天1次	5天1次	5天1次
交付周期	1.4月	0.6月	0.6月	3.2月	2月	0.6月、2月
平均修复时长	1.7小时	38分钟	38分钟	6.4小时	2小时	38分钟
变更失败比例	4.7%	2.5%	2.5%	12.5%	12.5%	12.5%
有效会议时间	32.4分钟	15分钟	15分钟、45分钟	19.9分钟	15分钟	25分钟
代码返工率	1.6天1次	1天3次	1天3次	1天1次	2.5周1次	3周1次
平均故障间隔时长	1.4月	3.5月	3.5月	5.5天	13.5小时	13.5小时

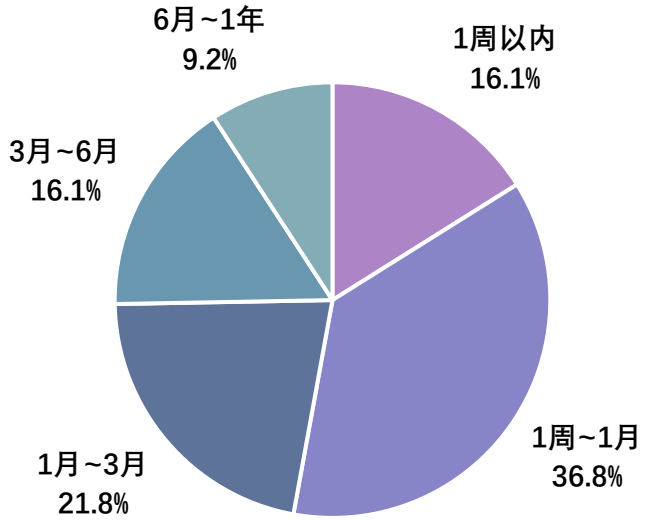
经过对受访对象团队的组织效能划分后，分别统计高效能团队和一般效能团队的各项参数的**均值**、**中位数**和**众数**，发现无论是否是高效能团队，各项参数的中位数表现差别不大，但从均值表现和众数表现(尤其是均值表现)来看，高效能团队**明显更好**，且与2023年相比，整体表现有所上升，高效团队与低效团队的差距减小，表明了业界的不断成熟。

部署频率与交付周期

受访对象的部署频率分布



受访对象的交付周期分布

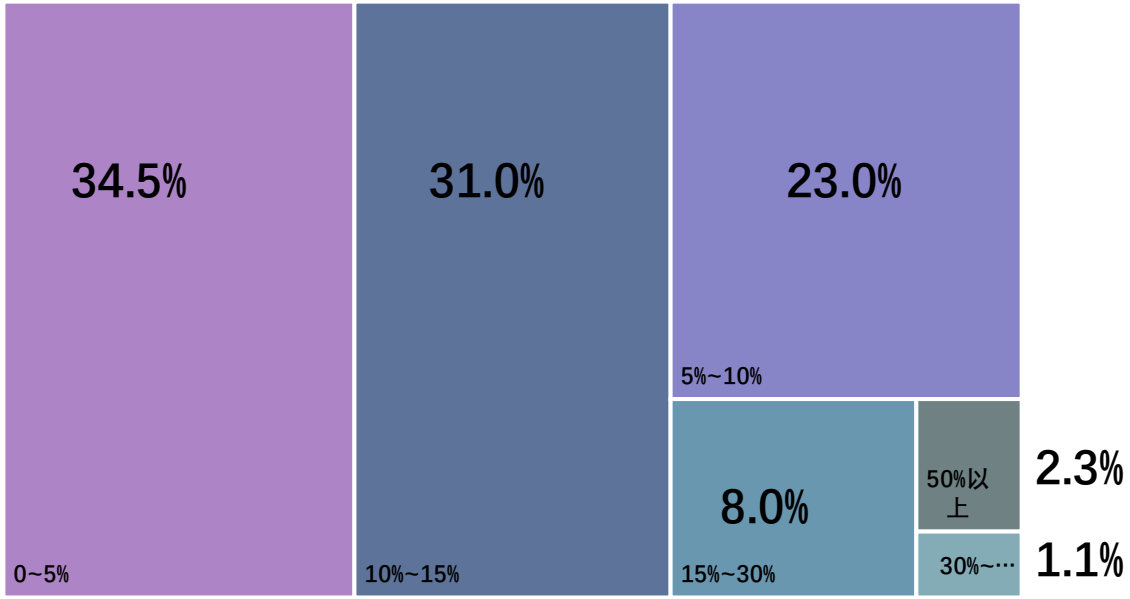


高频率部署也意味着快速和持续的部署，这有利于组织对市场需求及时反应，使组织能够通过自动化的构建、测试和部署循环来快速交付高质量的软件。本次调查的受访对象中70.1%能够完成单周部署，91.9%的受访对象能完成单月部署，只有极少数的受访对象部署频率低于1月1次。

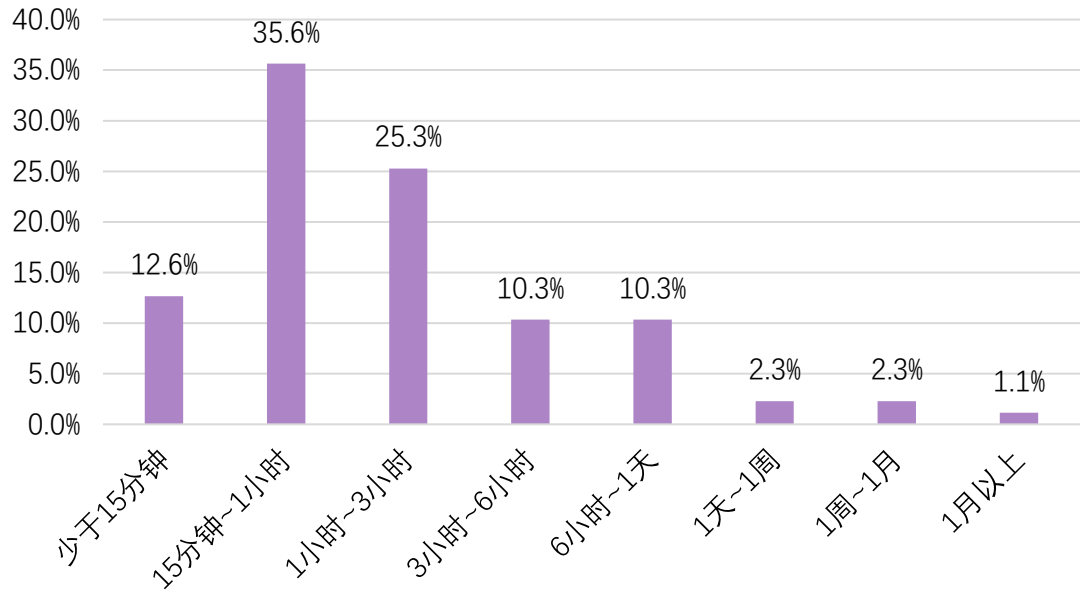
较短的交付周期表明团队能够快速响应市场变化和客户需求，这是在竞争激烈的市场环境中获得优势的关键。本次调查的受访对象中，有52.9%的受访对象表示他们能在一月内完成交付，有74.7%的受访对象能够在三个月内完成产品的最终交付，这些数据可能表明DevOps和敏捷实践在行业中已被广泛接受和应用，尤其是在那些需要快速迭代和持续交付的环境中。

变更失败比例与平均修复时长

受访团队的变更失败比例分布



受访团队的平均修复时长分布

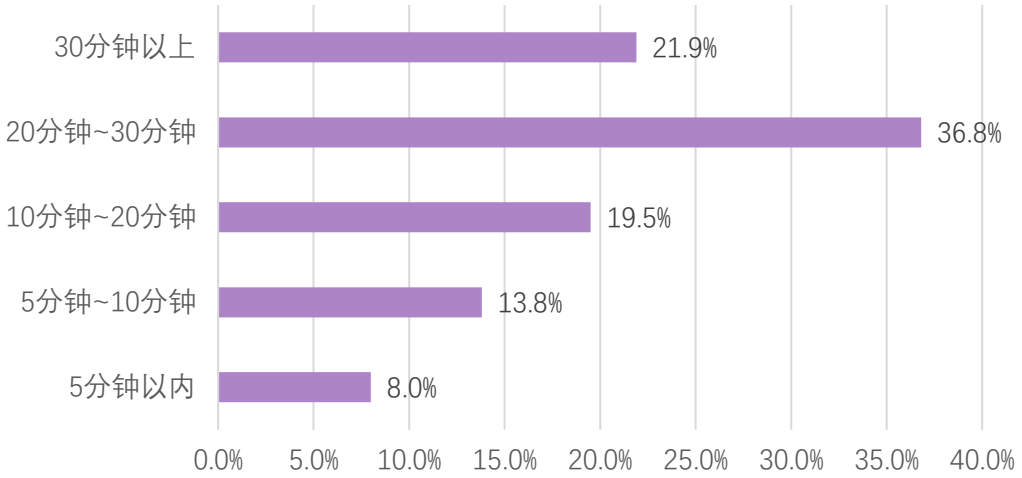


今年大部分的受访对象可以达到15%以下的变更失败率，排除掉不关注或不统计此数据的团队，这个比例达到了88.5%，整体表现较高，相较于2023年提高了3.4%，这表明了国内的DevOps团队在故障处理方面逐渐成熟的趋势。

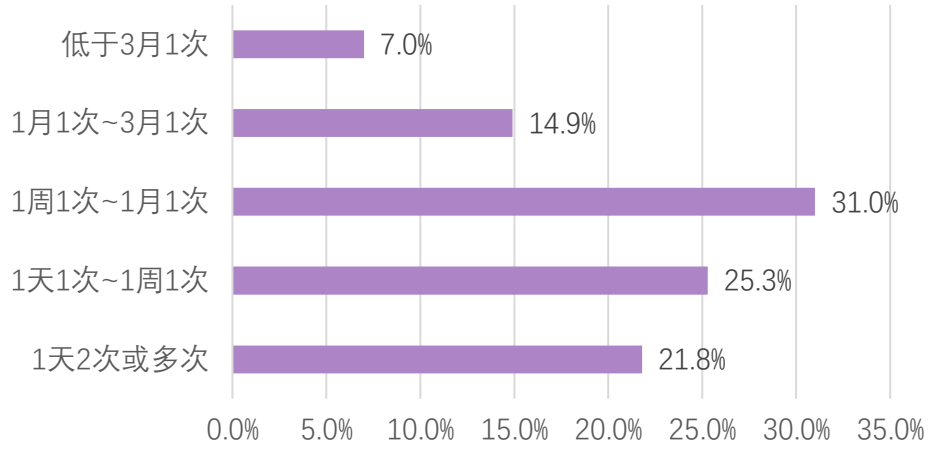
今年73.6%的受访对象能在3小时内解决突发故障，其中有12.6%的团队能够将平均修复时长控制在15分钟内，48.3%的团队能够将平均修复时长控制在1小时内，相较于2023年平均时长有明显的减少。这表明行业内对于迅速修复故障更加重视。

有效会议时间与代码返工率

受访对象的有效会议时间分布



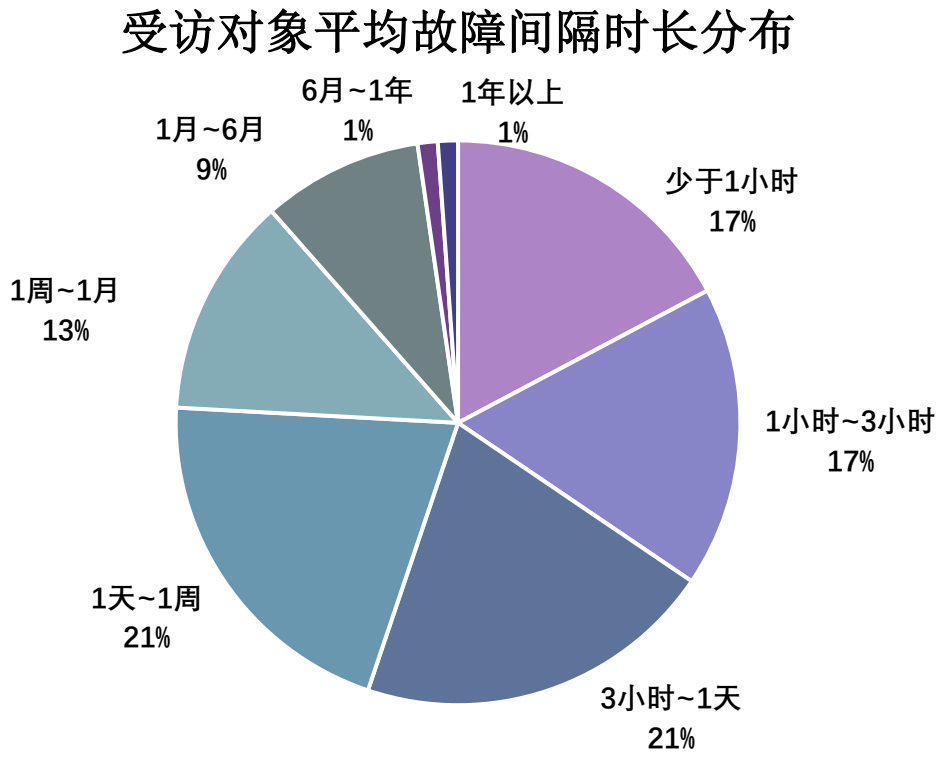
受访团队的代码返工率分布



有效会议时间是指**单位小时内针对项目进展和问题缺陷富有成效的讨论所占用的时间**。今年受访的团队仅仅有**21.9%**的团队认为有效会议时间超过30分钟，大部分团队(**78.1%**)的有效会议时间都低于30分钟，这可能表明大部分团队的会议缺乏明确的目标或存在沟通不畅的问题，依然存在提升效率的空间，需要引入优秀的实践指导。

代码返工率是指**软件开发过程中出现的代码缺陷或错误需要进行修复的频率**，**低返工率**通常意味着**高质量**的代码提交，这是软件质量和可靠性的重要指标。今年受访的团队中有**52.9%**的团队代码返工率低于1周1次，但其中处于1周1次~1月1次的占比**31.0%**，除此以外还有**25.3%**的团队代码返工率处于1天1次~1周1次，这些表明了参与调查的多数团队在代码质量管理方面的成熟度还不够。

平均故障间隔时长



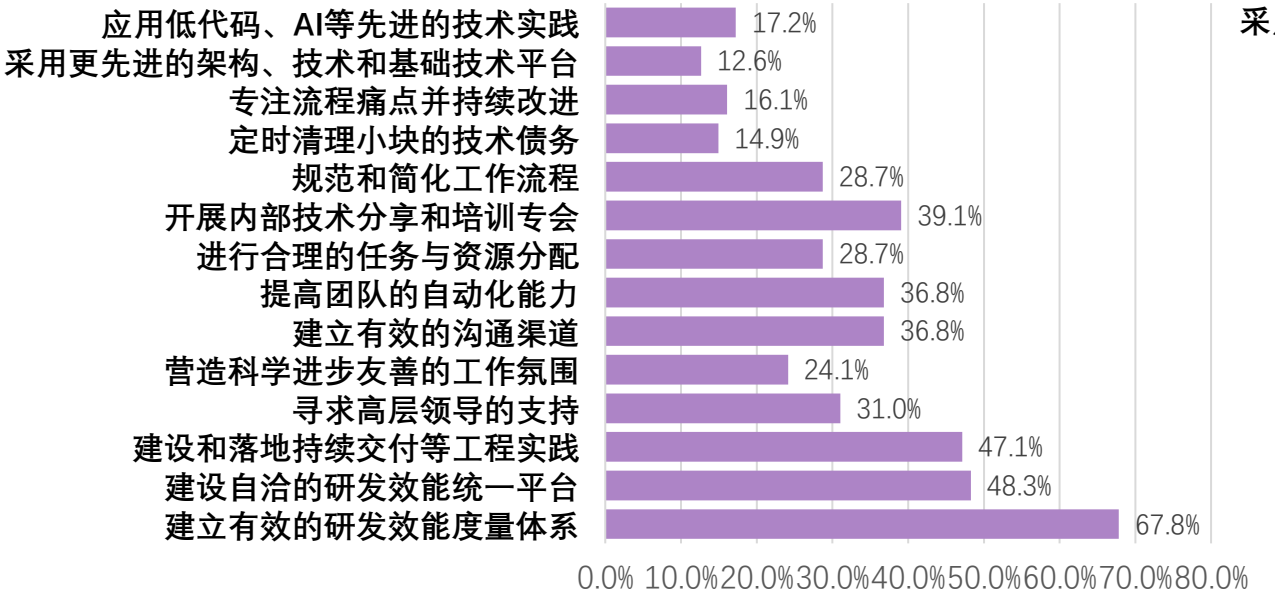
平均故障间隔时长MTBF是指**系统正常运行过程中两次相邻故障之间的平均时长**，MTBF是衡量系统**稳定性**和**可靠性**的关键指标，较长的故障间隔时长意味着系统更加稳定。今年的受访对象中仅有**24%**的团队平均故障间隔时长大于1周，且其中大于1月的仅占比**11%**，除此以外有**55%**的团队平均故障间隔时长小于1天，相较于2023年的**38%**已经有了显著的进步，这些表明了参与调查的近半数团队在风险管理和预防系统故障方面还面临着较大挑战，但整体正在向好的一面发展。

03 研发效能实践

- 最佳实践
- 持续集成
- BizDevOps
- 架构治理
- 第三方库依赖冲突
- AI生成代码处理措施

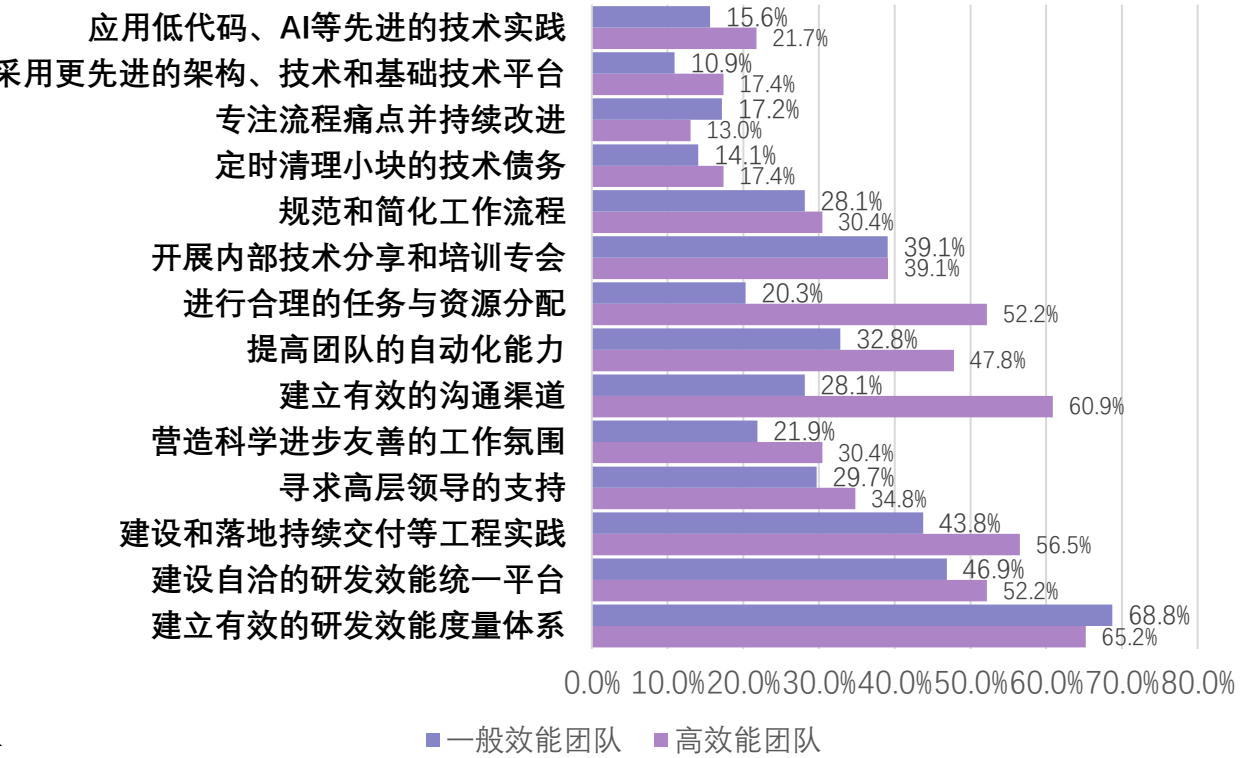
研发效能最佳实践

受访对象实施了哪些提升研发效能的最佳实践



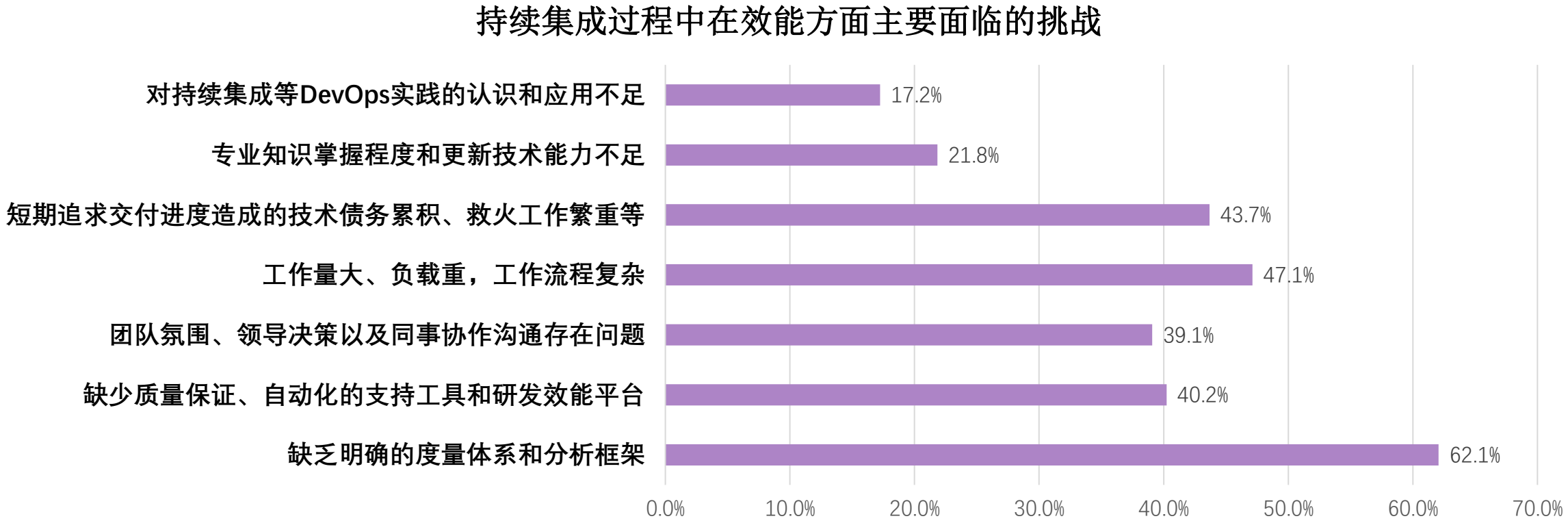
在提升研发效能的最佳实践中，建立有效的研发效能度量体系（67.8%）是最常实施的做法，显示许多组织认为**度量体系**对提升研发效能至关重要。建设自洽的研发效能统一平台（48.3%）也较为普遍，表明**统一平台**对提升效率有重要作用。相较于2023年，最佳实践的分布情况并未发生显著的变化。

高效与低效团队落实实践情况对比



高效能团队在许多关键最佳实践的落实上明显表现更好，整体全面领先一般效能团队，体现了最佳实践对于效能提升的直观作用。

持续集成-挑战

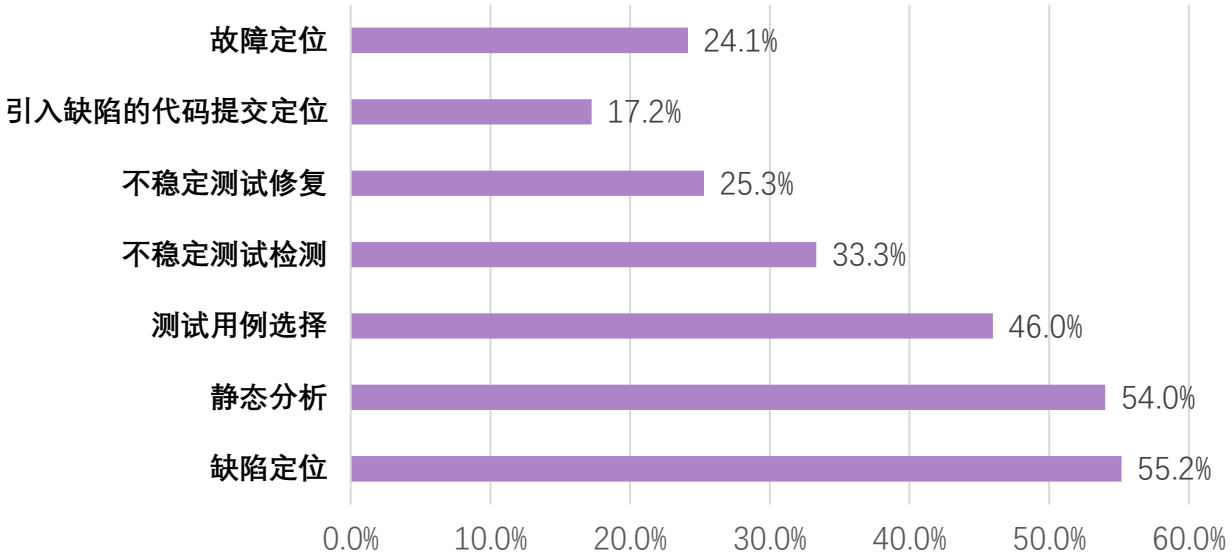


缺乏明确的度量体系和分析框架（62.1%）是最显著的挑战，表明需要引入明确的DevOps体系。人员管理（47.1%）和项目管理（43.7%）上的不足也是主要关注点，指出了管理对于研发效能的重要意义。

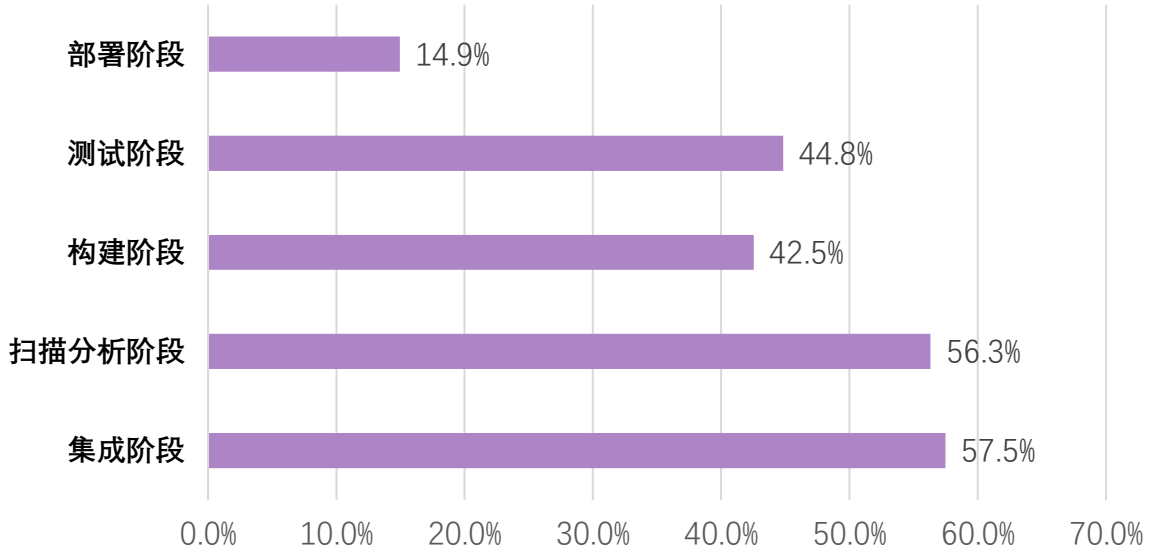
1. The State of Platform Engineering Report. (2023). https://www.puppet.com/resources/state-of-platform-engineering

持续集成-自动化/智能化技术

持续集成过程中使用了哪些自动化或智能化技术(多选)



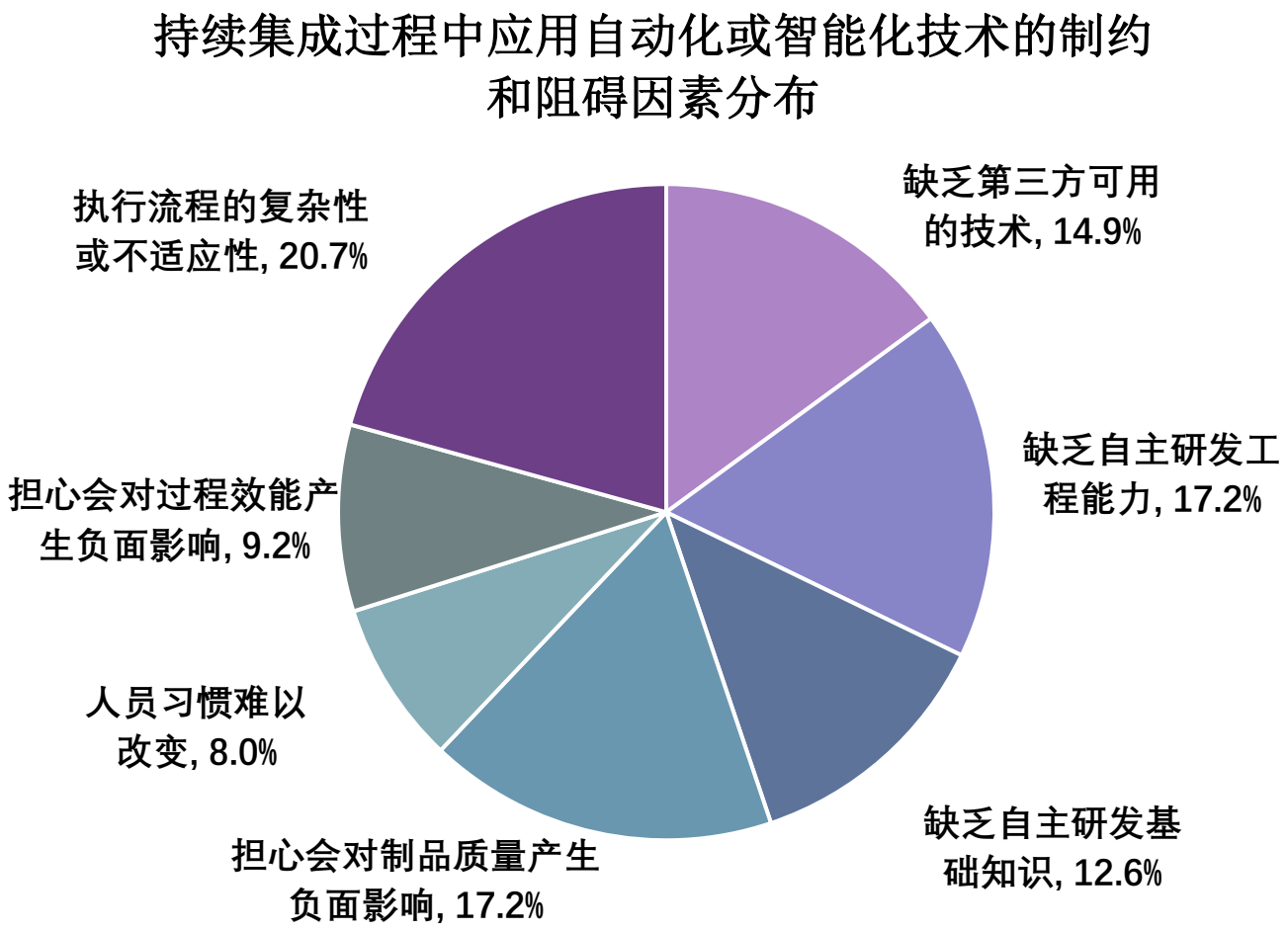
持续集成的哪些环节更需要这些自动化或智能化技术支持（多选）



缺陷定位（55.2%）和静态分析（54%）是持续集成过程中最常使用的自动化/智能化技术，半数以上的受访对象都采用了这些技术，同时调查结果表明集成阶段（57.5%）和扫描分析阶段（56.3%）是最需要自动化/智能化技术的环节，指明了**持续集成中自动化/智能化技术的发展方向**。

1. Rafi, Saima, et al. "Readiness model for DevOps implementation in software organizations." Journal of Software: Evolution and Process 33.4 (2021): e2323.

持续集成-制约和阻碍

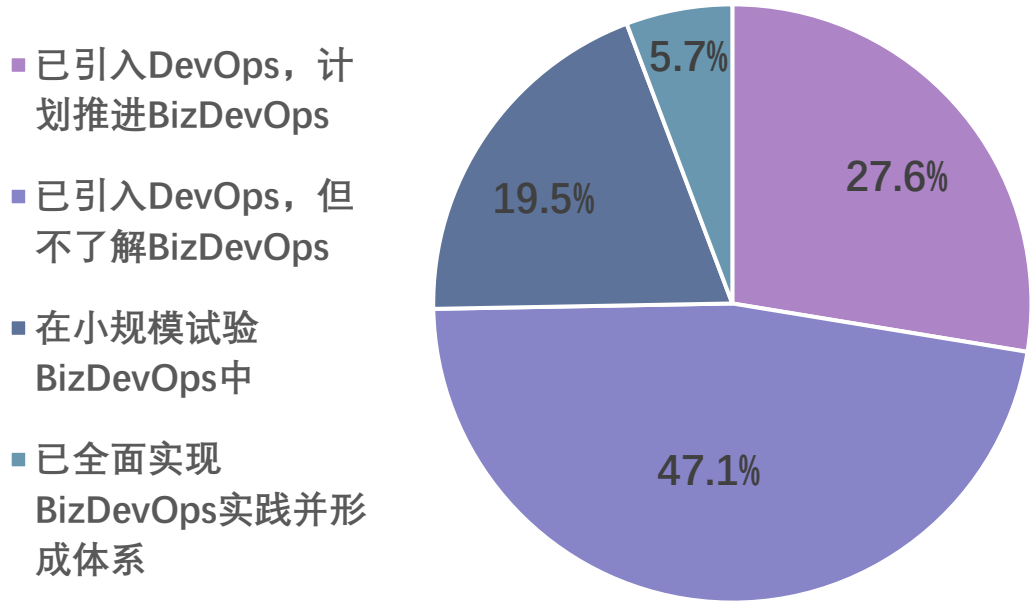


执行流程的复杂性或不适应性（20.7%）、缺乏自主研发工程能力（17.2%）和担心会对制品质量产生负面影响（17.2%）是制约和阻碍持续集成过程中应用自动化/智能化技术最主要的因素，分别代表了受访对象对于**持续集成自身、软件工程师和自动化/智能化技术三个不同方面的担忧**。结果表明业界目前对于自动化/智能化技术在持续集成中的应用仍未成熟。

1. Rafi, Saima, et al. "Readiness model for DevOps implementation in software organizations." Journal of Software: Evolution and Process 33.4 (2021): e2323.

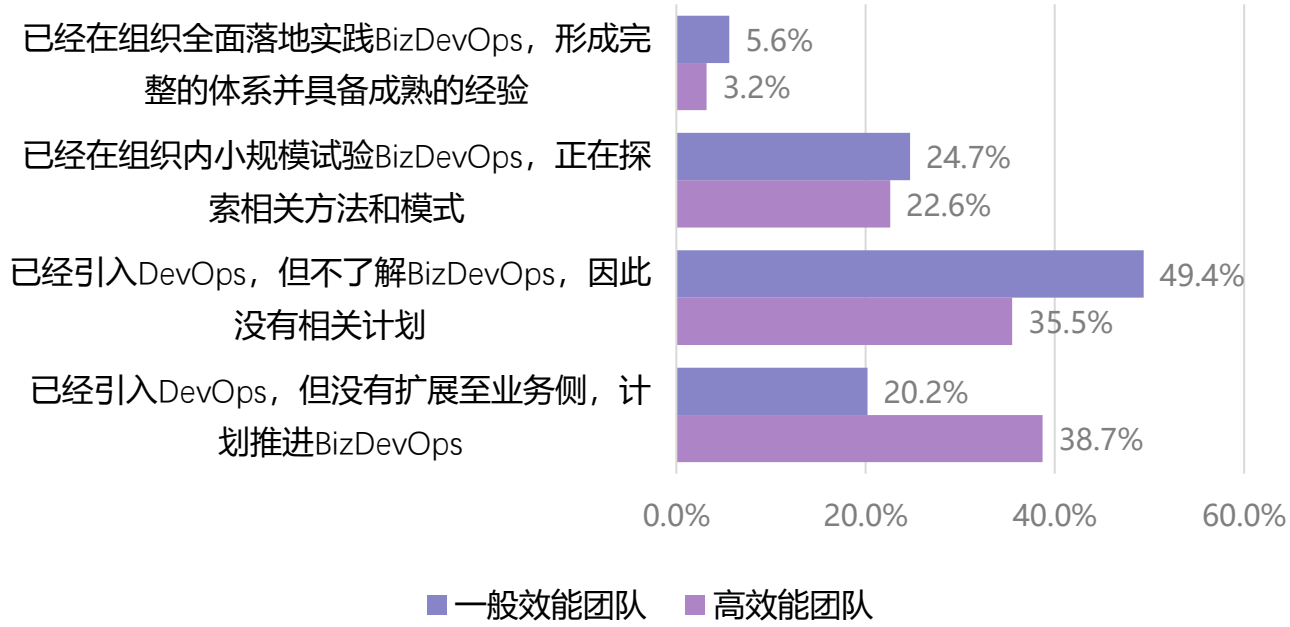
BizDevOps

受访对象将DevOps的实践向业务侧扩展的现状与计划



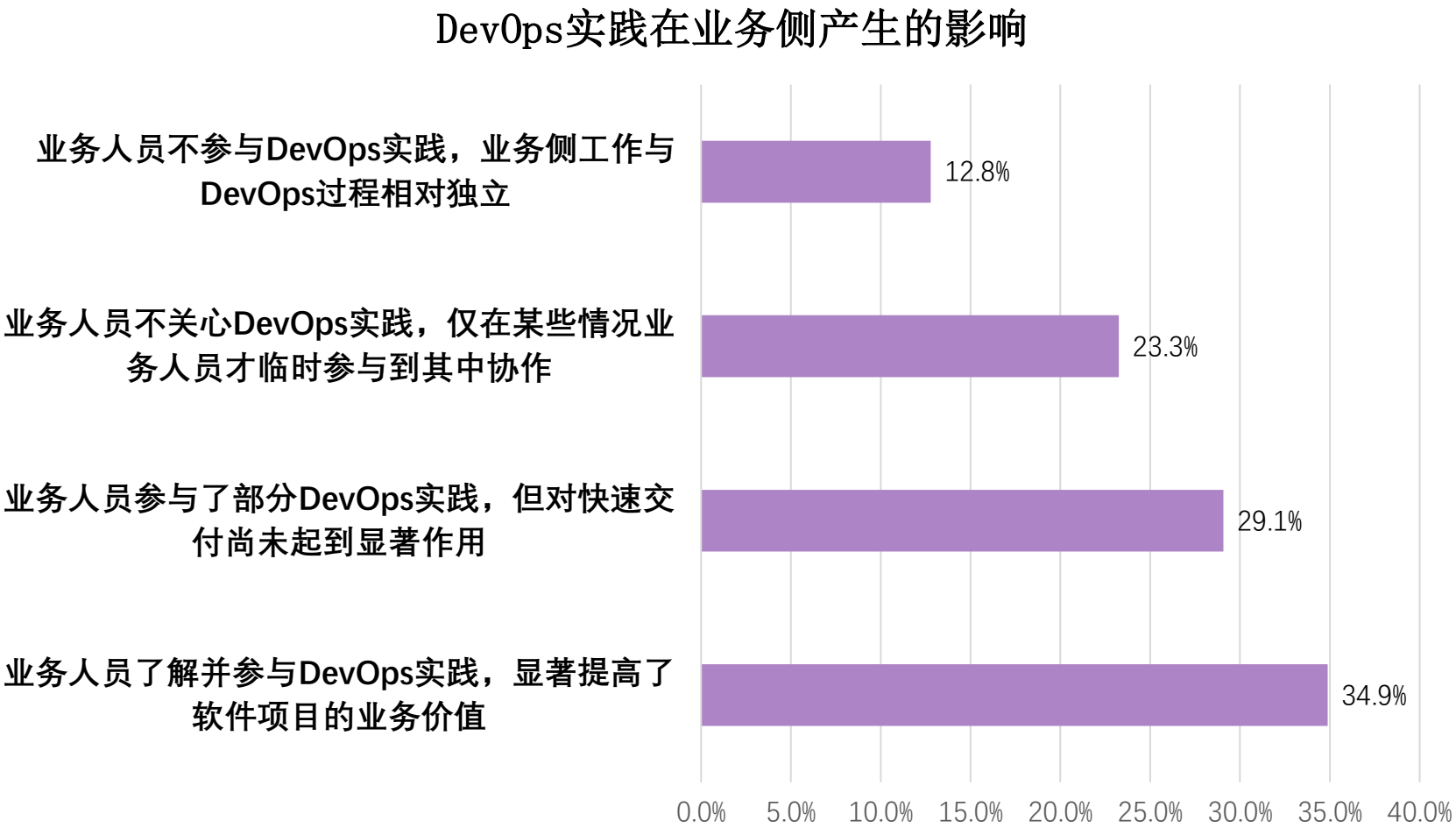
大多数组织已引入DevOps，但对BizDevOps缺乏认识，因此尚未制定相关计划（47.1%）。只有少数组织正在积极探索或已经成功实施BizDevOps。

高效能团队与一般效能的DevOps业务侧拓展情况对比



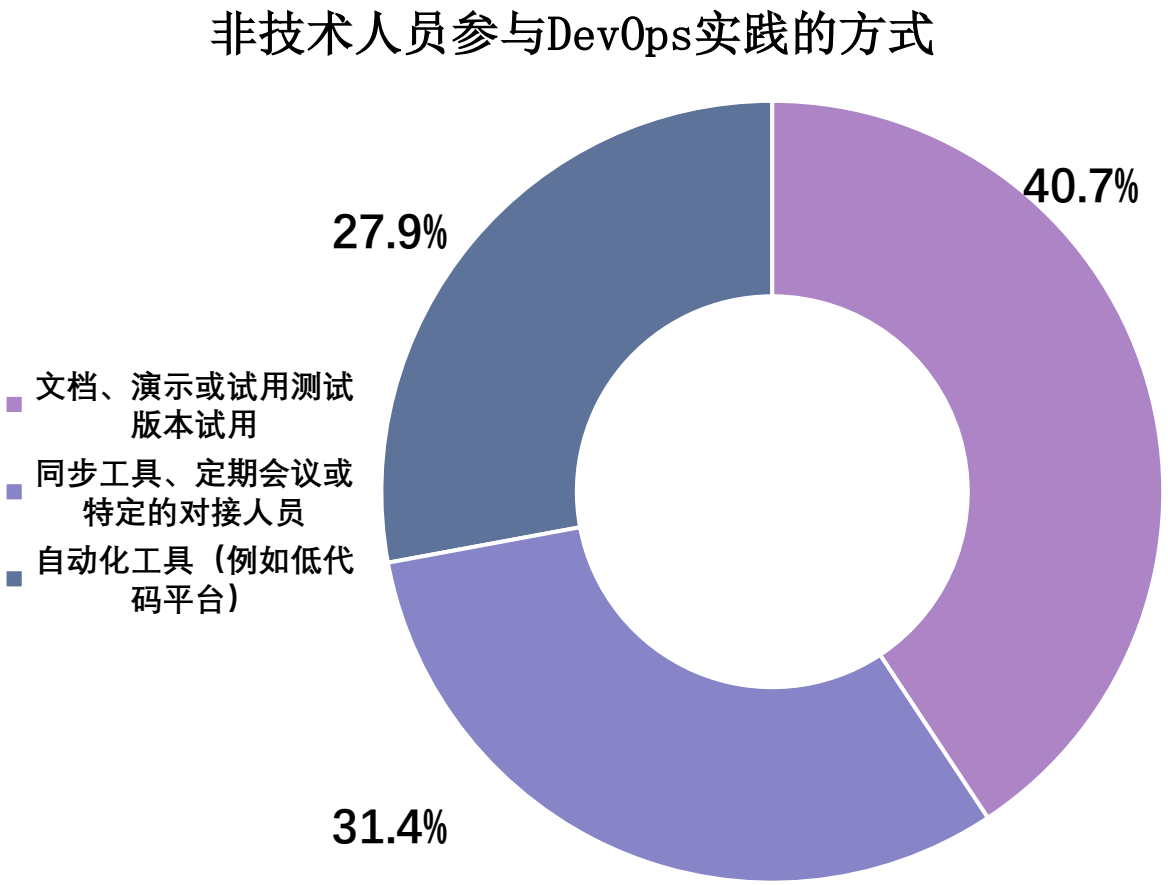
高效能团队和一般效能团队的BizDevOps实施情况普遍较低，但高效能团队更倾向于计划推进BizDevOps的实践（38.7%），而一般效能团队中更多（49.4%）表示已经引入DevOps但不了解BizDevOps，因此没有相关计划。

BizDevOps-影响



一部分业务人员了解并积极参与DevOps，这对提高软件项目的业务价值和驱动业务发展起到了积极作用。然而大多数（29.1%）业务人员虽参与DevOps工作，但对快速交付业务和市场需求的产品目标影响不大。这表明尽管有业务团队的参与，但DevOps实践与业务需求之间的融合还需进一步优化。但还有相当比例的业务人员对DevOps的工作了解不足，甚至完全不参与。

BizDevOps-方式

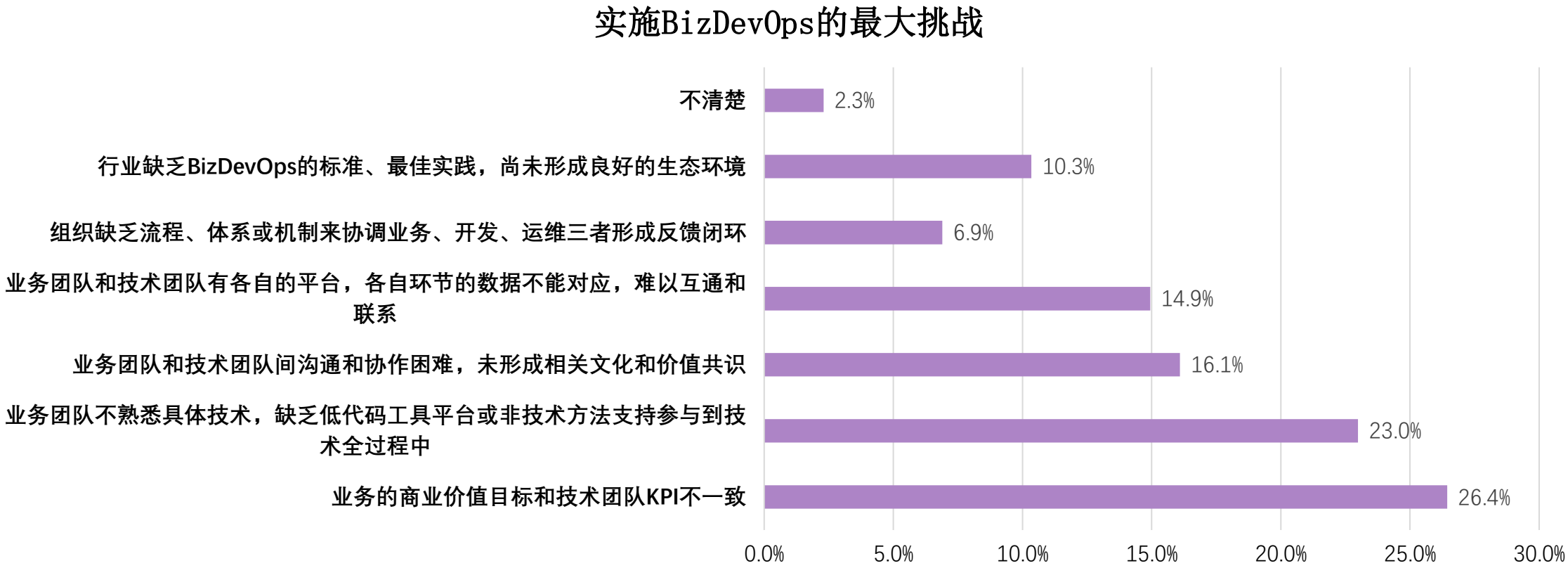


文档、演示或测试版本试用（40.7%）是最普遍的方式，非技术人员通过**阅读文档、观看演示或试用测试版本**来理解和反馈DevOps实践的成果。这种方式有助于他们更好地理解技术团队的工作进展，同时提供实际的业务视角和反馈。

同步工具、定期会议或特定的对接人员（31.4%）侧重于通过**定期的沟通和专门的对接人员**来确保非技术人员与DevOps团队的持续交流，有助于实时解决问题和快速响应业务需求变更。

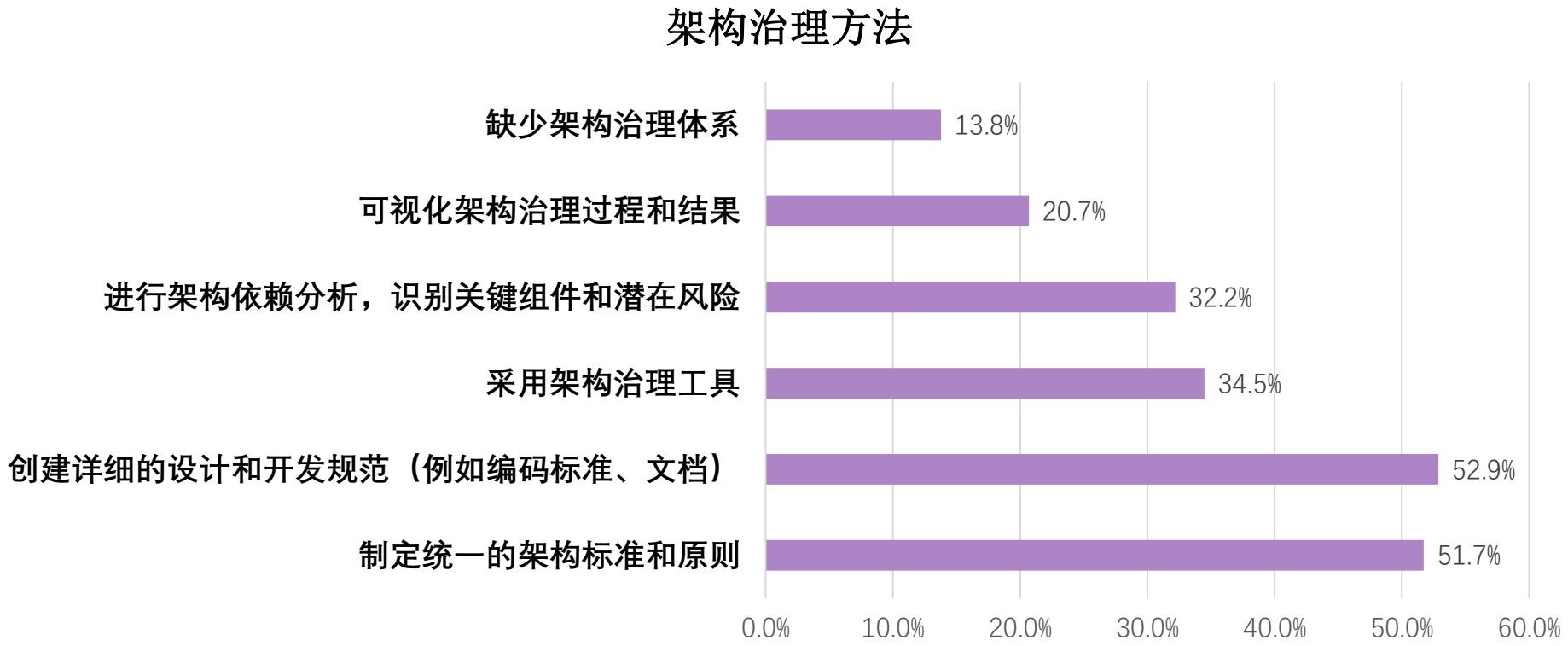
自动化工具对业务数据持续分析和反馈，业务人员和技术人员在共享环境中协作（27.9%）是利用**自动化工具**，如低代码平台，实现业务人员和技术人员的紧密协作。通过这些工具，非技术人员可以更直观地参与需求管理和数据分析，促进业务与技术的融合。

BizDevOps-挑战



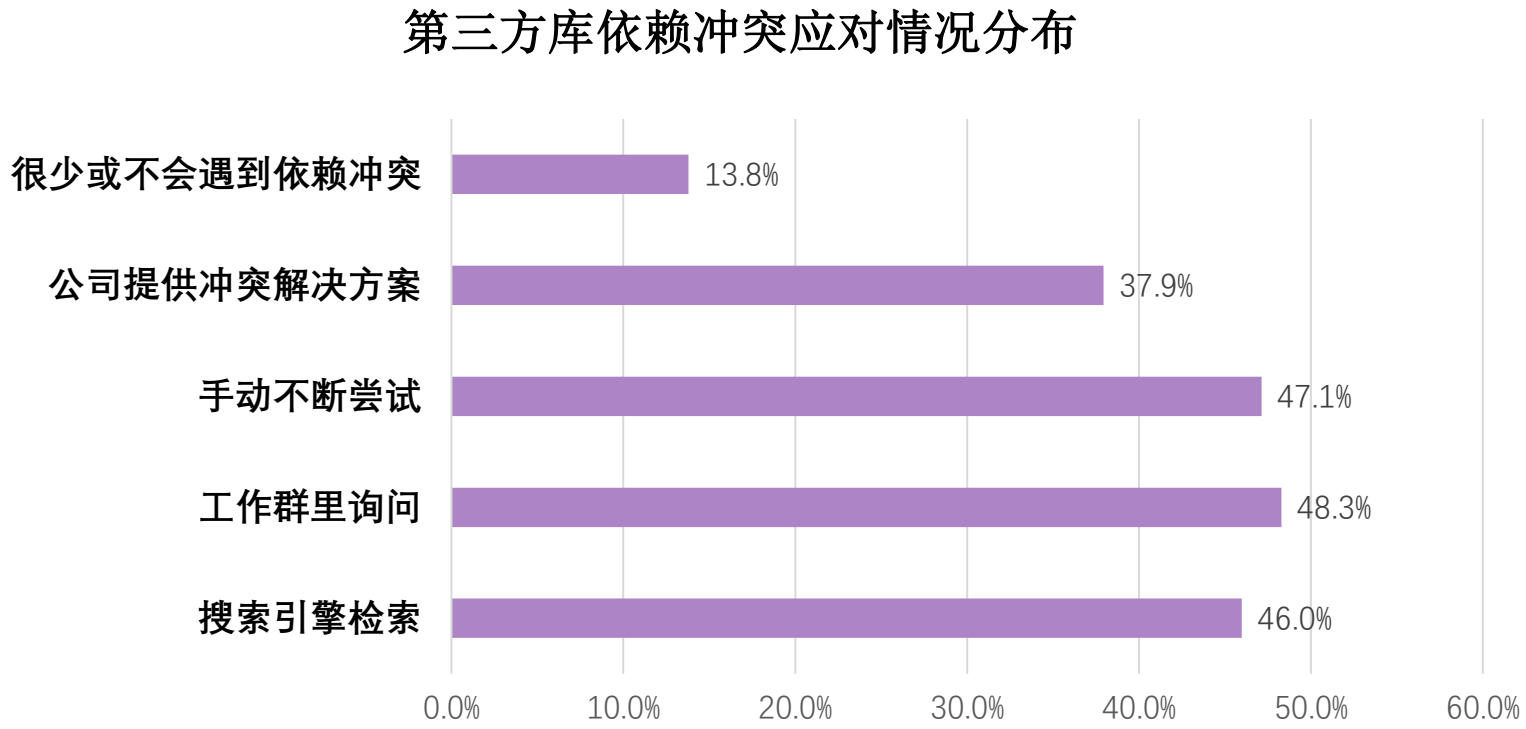
组织实施BizDevOps面临的挑战主要是对技术不熟悉（23.0%）和目标不一致（26.4%）:**业务团队不熟悉技术细节**，缺乏适当的工具或非技术手段参与技术过程，导致需求与产品功能不匹配，未能形成共同的文化和价值观导致的**无效开发和滞后反馈**，以及商业价值目标和技术团队KPI不对齐。寻求**统一业务和技术目标**的方法是实现有效的BizDevOps实践的关键。其次是沟通和协作障碍（16.1%），业务和技术团队之间缺乏有效的沟通和协作。相较于2023年，影响实施BizDevOps的挑战并未出现明显变化。

架构治理



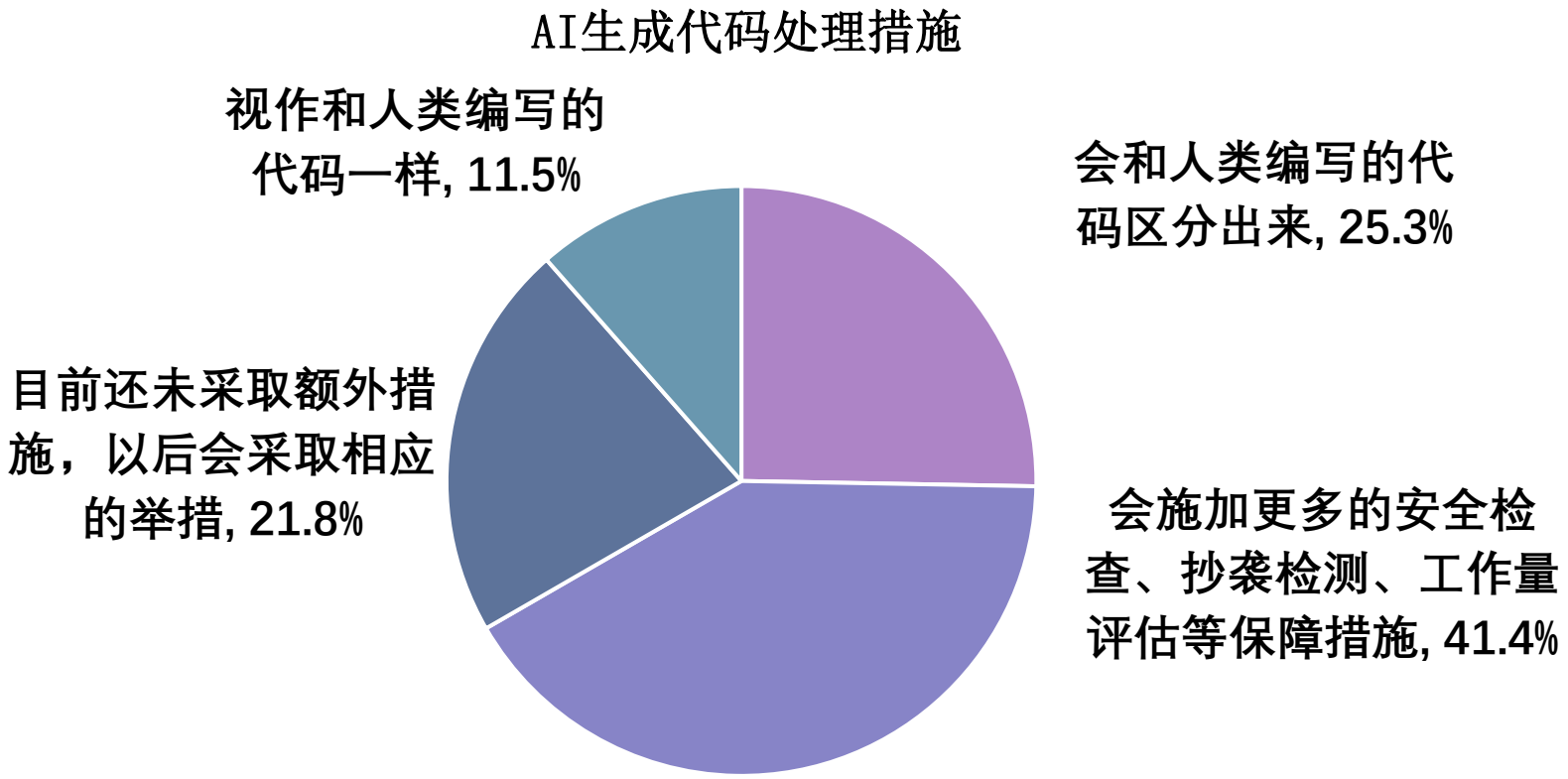
创建详细的设计和开发规范（52.9%）与制定统一的架构标准和原则（51.7%）是软件研发实践中最常被使用的架构治理方法，体现出了**良好项目管理的重要意义**，同时也表明业界对于项目管理的重视程度。

第三方库依赖冲突



大部分受访对象（86.2%）在实践中都会遇到第三方库依赖冲突的情况，对于第三方库依赖冲突的应对方式以软件工程师自发解决为主，仅37.9%的受访对象会有公司提供冲突解决方案，表明业界整体未能建立完善的知识库。

AI生成代码处理措施



对于AI生成的代码，受访对象普遍表现出了对于AI生成代码的不信任，仅11.5%的受访对象会完全信任AI生成的代码，将其视作和人类编写的代码一样，63.2%的受访对象表明需要对AI生成的代码进行额外的检查与评估，结果体现了目前AI生成代码技术依然存在不足，无法得到业界的信任。

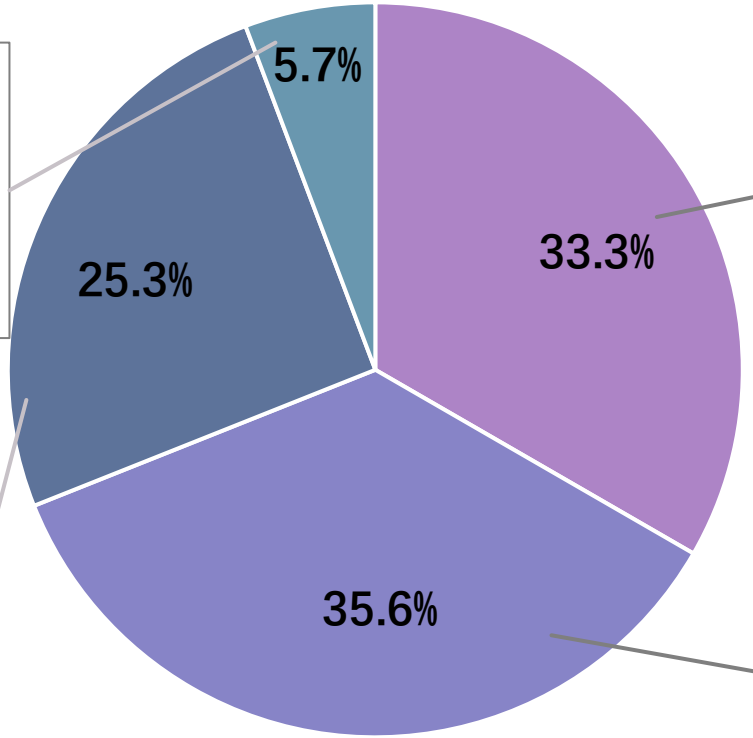
04

DevOps安全实践

- 安全实践成熟度
- 安全实践频率
- 安全实践效率
- 安全测试实践
- 开源组件库安全
- 安全实践挑战
- 安全管理实践

安全实践成熟度

受访对象的DevOps安全实践成熟度分布



完全融入，显著成果（2023年为5.8%）

仅有不到6%的受访对象表示他们已经在技术、流程、工具和文化等各个层面实现了安全与DevOps的深度融合。

安全与DevOps独立（2023年为36.4%）

占比超过三成的受访对象完全没有将安全纳入DevOps流程或仅仅在开发完成后才进行安全审查，说明至少在流程层面缺乏两者的融合，安全措施往往难以及时跟进产品开发和迭代，成为后置操作。

部分整合，成果初显（2023年为26%）

四分之一左右的受访对象表示已在DevOps流程中嵌入了一定程度的安全实践，例如在CI/CD管道中加入了自动化安全测试，整体效益还需进一步验证和优化。

尝试整合但有问题（2023年为31.8%）

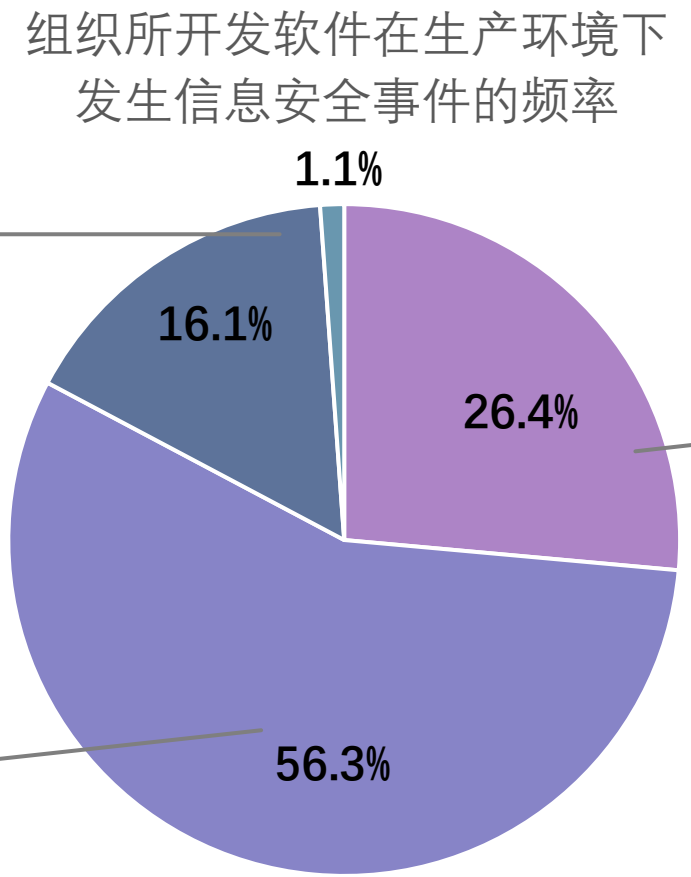
也有三分之一的受访对象虽然意识到安全需要前置融入DevOps，但在实际实施中遇到了技术、流程、工具或文化等方面的阻力。例如缺乏统一工具链，导致整合效果不佳。

结论：DevOps安全实践成熟度整体偏低。

安全事件频率

时常发生（16.1%）和频繁发生（1.1%）

虽然此类情况的比例相对较低，但对于这部分组织而言，频繁的安全事件意味着较大的运维风险，需要优先解决。



从未发生

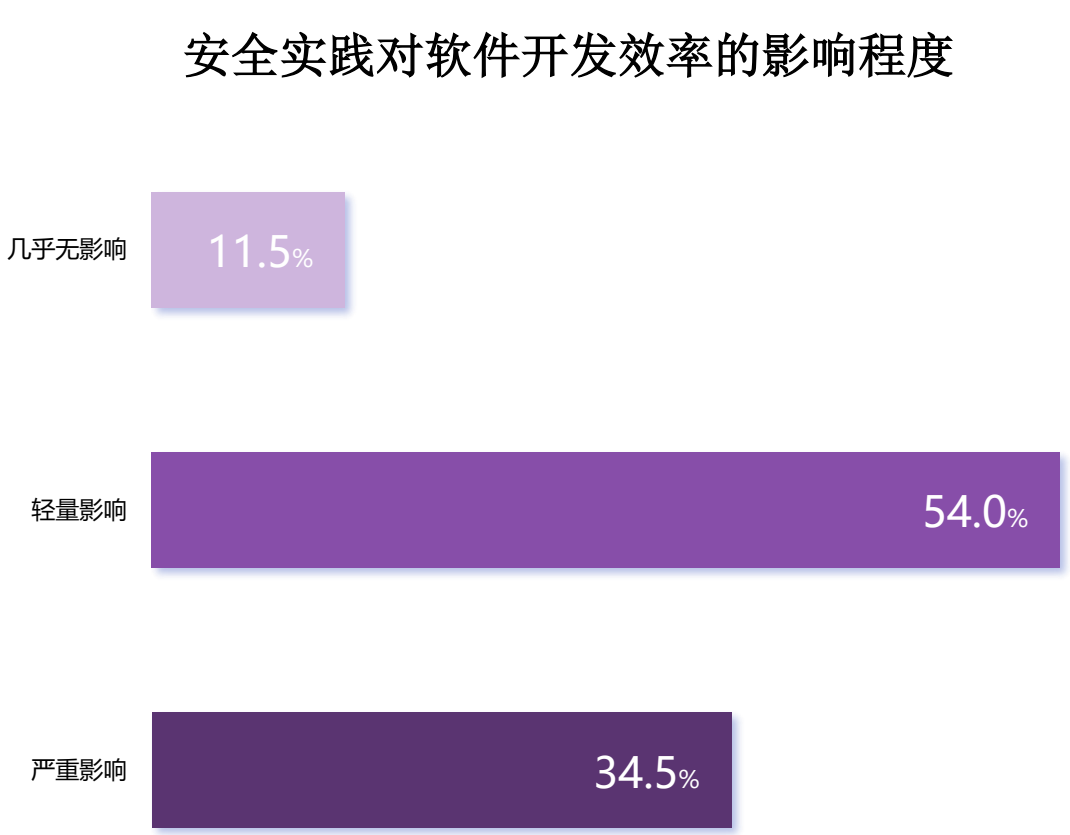
约四分之一的受访对象表示未曾发生过信息安全事件，这在一定程度上体现了他们在安全管理和风险控制方面的有效性。但也需要警惕可能存在“未知的安全盲区”，或者在监测与报告机制上尚待完善。

偶尔发生

超过半数的受访对象表示在过去一年中偶尔遇到信息安全事件。这说明多数组织尚无法完全避免安全风险，现有的安全措施与流程只能在一定程度上减少事故发生频率，但仍需持续改进。

结论：生产环境下软件信息安全事件在大多数组织中仍较为常见。

安全实践效率



- **轻量影响（2023年为52%）**

多数受访对象认为安全实践会在某些阶段拖慢开发节奏，可能体现在对流程的额外检查、审计资源的调配，以及在工具上需要适配和培训等方面。

- **严重影响（2023年为33%）**

这部分受访对象表示所在组织往往在技术、流程、工具或文化上尚未做好与安全深度融合的准备，导致安全审计与开发流程之间存在冲突（例如在时间节点或质量准入标准上不一致），进而严重影响效率。

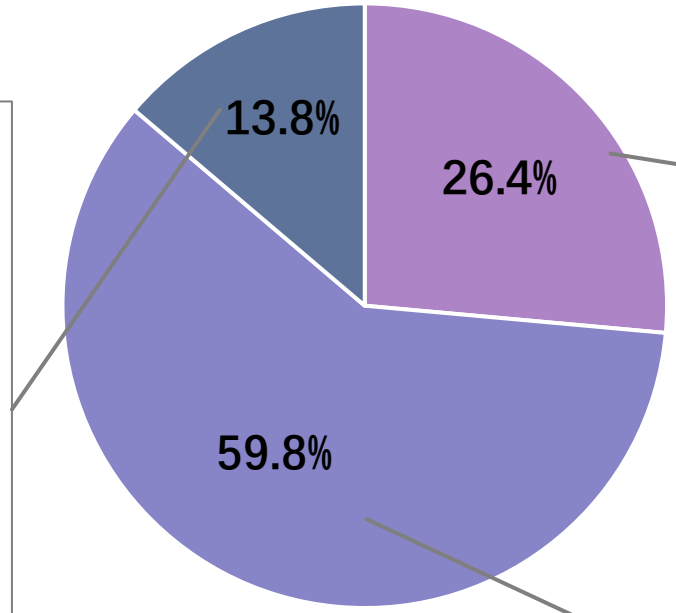
- **几乎没有影响（2023年为15%）**

这类受访对象表示所在组织可能已经引入成熟的DevOps安全实践，能够通过自动化工具与规范化流程来降低安全测试或审计带来的额外负担。

结论：安全实践对软件开发效率造成轻微影响。

安全测试实践

受访对象的软件研发流程与安全测试工作的结合程度



完全集成且自动化安全测试

仅有不到两成的受访对象表示实现了将安全测试工具和流程无缝集成到研发流程中，并且通过自动化手段减少人工介入。这类组织通常在DevOps安全实践上更为成熟，能够在开发早期及持续集成过程中快速发现并修复安全漏洞。

不进行安全测试工作或独立执行

超过四分之一的受访对象表示要么不进行任何安全测试，要么把安全测试独立在研发流程之外，通常在开发或上线完成后才进行安全检查。这种模式下，安全测试成为事后补救措施，难以及时识别和修复漏洞。

部分集成安全测试工作，但需要人工执行

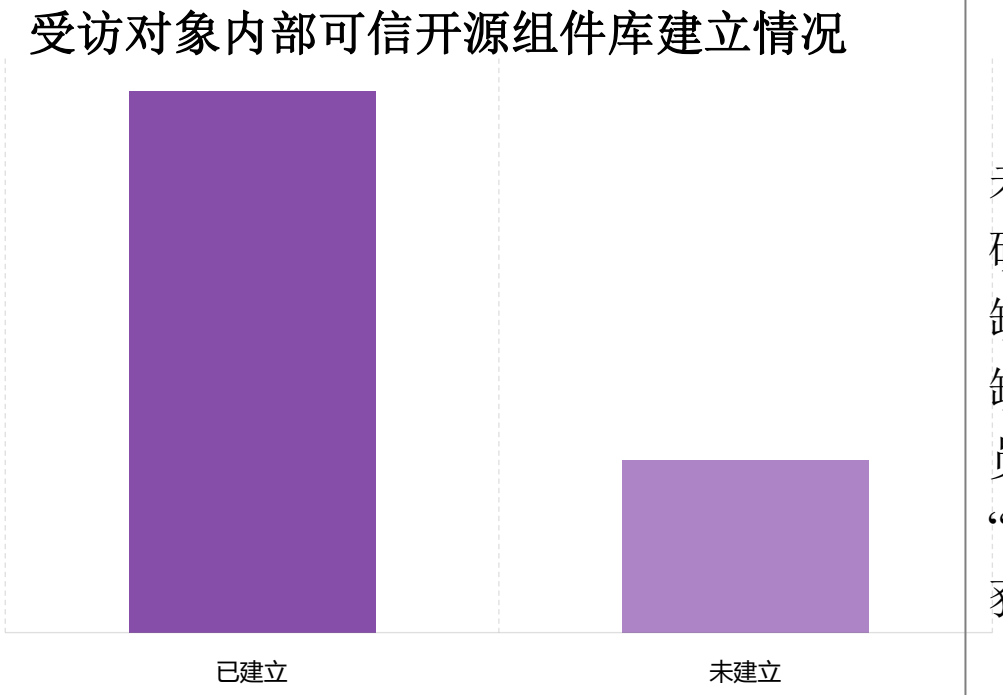
将近六成的受访对象表示虽然在一定程度上将安全测试工作融入研发流程，但主要还是依赖人工执行，难以满足DevOps快速迭代的需求。人工测试相比自动化测试不仅效率相对较低，也容易产生漏测问题。

结论：研发流程与安全测试结合度低，缺乏自动化技术。

开源组件库

已建立内部的可信开源组件库（75.9%）

已建立内部可信开源组件库的受访对象可能会遇到如何持续维护组件库的安全性和质量，如定期检测组件漏洞和潜在合规问题；需要投入一定的人力和基础设施（如制品库管理工具、CI/CD集成）来确保库的高可用性和可靠性等挑战。



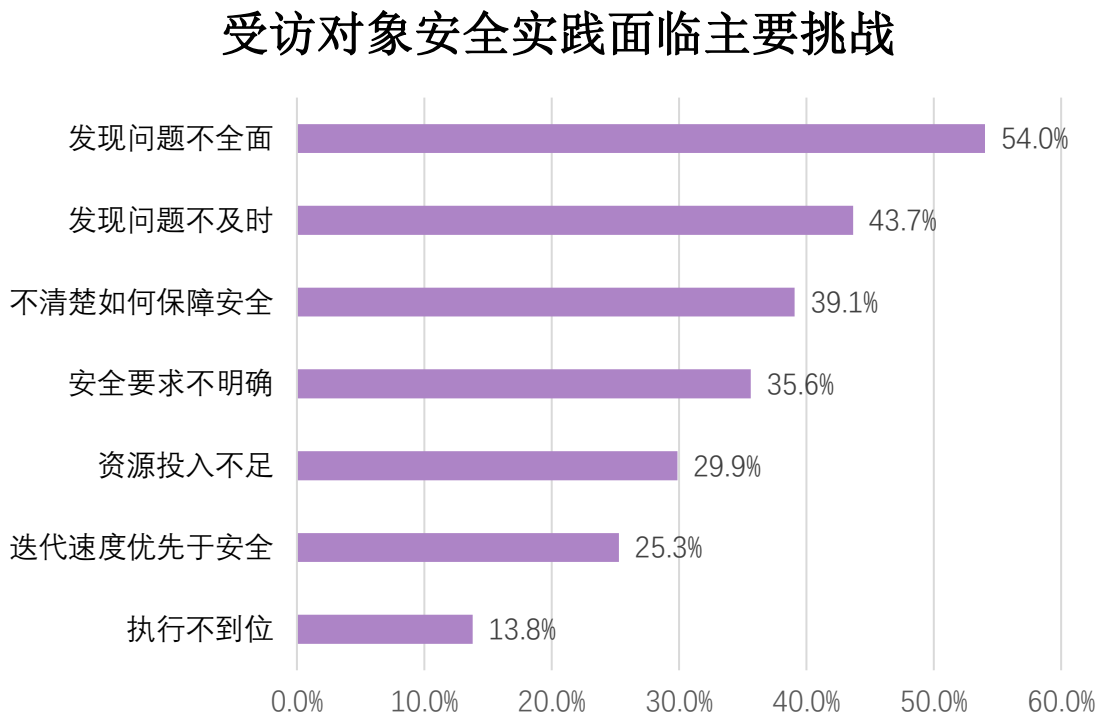
尚未建立内部的可信开源组件库（24.1%）

未建立的可能原因是组织规模、研发成熟度或安全意识尚不足，缺乏对开源组件潜在风险的重视；缺乏相应的管理工具、流程和人员来运营、维护一个内部库；以“就地取用”的方式直接从公共源获取依赖，未形成统一管理。

结论：大多数组织已建立内部的可信开源组件库。

安全实践挑战

问题：您所在组织在应用安全性上的主要挑战是？



- **发现问题不全面（54.0%）**

超过一半的受访对象表示无法全面发现安全问题。很大程度上与安全测试技术、工具性能不足等因素相关。

- **发现问题不及时（43.7%）**

将近一半的受访对象认为安全问题难以及时暴露。这可能意味着DevOps的持续监控、自动化扫描、实时告警等机制并没有很好地与安全实践融合。

- **不清楚如何保障安全（39.1%）**

约四成的受访对象对“安全该怎么做”并不清晰，说明多数团队仍缺乏对不同层面安全需求（如Web安全、云安全、供应链安全）系统性的安全培训或DevOps安全实践的经验积累。

- **安全要求不明确（35.6%）**

三成以上受访对象指出所在组织的安全目标或规范不够清晰，导致开发或运维人员对安全工作的优先级认知不足，执行时缺乏统一的标准或策略，也与管理层缺乏明确的安全投入和考核机制有关。

- **资源投入不足（29.9%）**

近三成的受访对象认为当前资源（人员、预算、工具等）不足以支撑全面且有效的安全防护。尤其在快速迭代的DevOps实践下，安全通常被认为是额外负担，难以获得足够资金与人力支持。

- **迭代速度优先于安全（25.3%）**

四分之一的受访对象指出，交付速度与商业需求常常压过安全优先级，导致对安全问题的关注度下降。“先上线、后补救”的模式虽然满足了业务诉求，但可能为后续的漏洞修复或安全事件埋下隐患。

- **执行不到位（13.8%）**

虽然所占比例最低，但这部分受访对象表示所在组织已经制定了相对完善的安全规范或流程，但在落地执行过程中，受制于团队意识、流程管控或绩效考核等因素，安全实践未能严格执行。

结论：发现问题不全面和不及时是安全实践过程中的主要挑战。

安全管理实践

受访对象中的安全/DevSecOps工程师表示其所在组织在安全管理实践方面相对完善：

- 使用的安全审计工具**高效，能发现安全问题**
- 使用的开源组件**保留了完整的软件材料清单（SBOM）**
- 对软件供应商**进行严格且持续的风险管理**
- 对第三方软件库**每月**进行定期安全检查和管理
- 如果发现第三方软件库存在安全隐患，采取的措施是

  （1）对不安全库进行评估，然后决定继续使用还是停用；

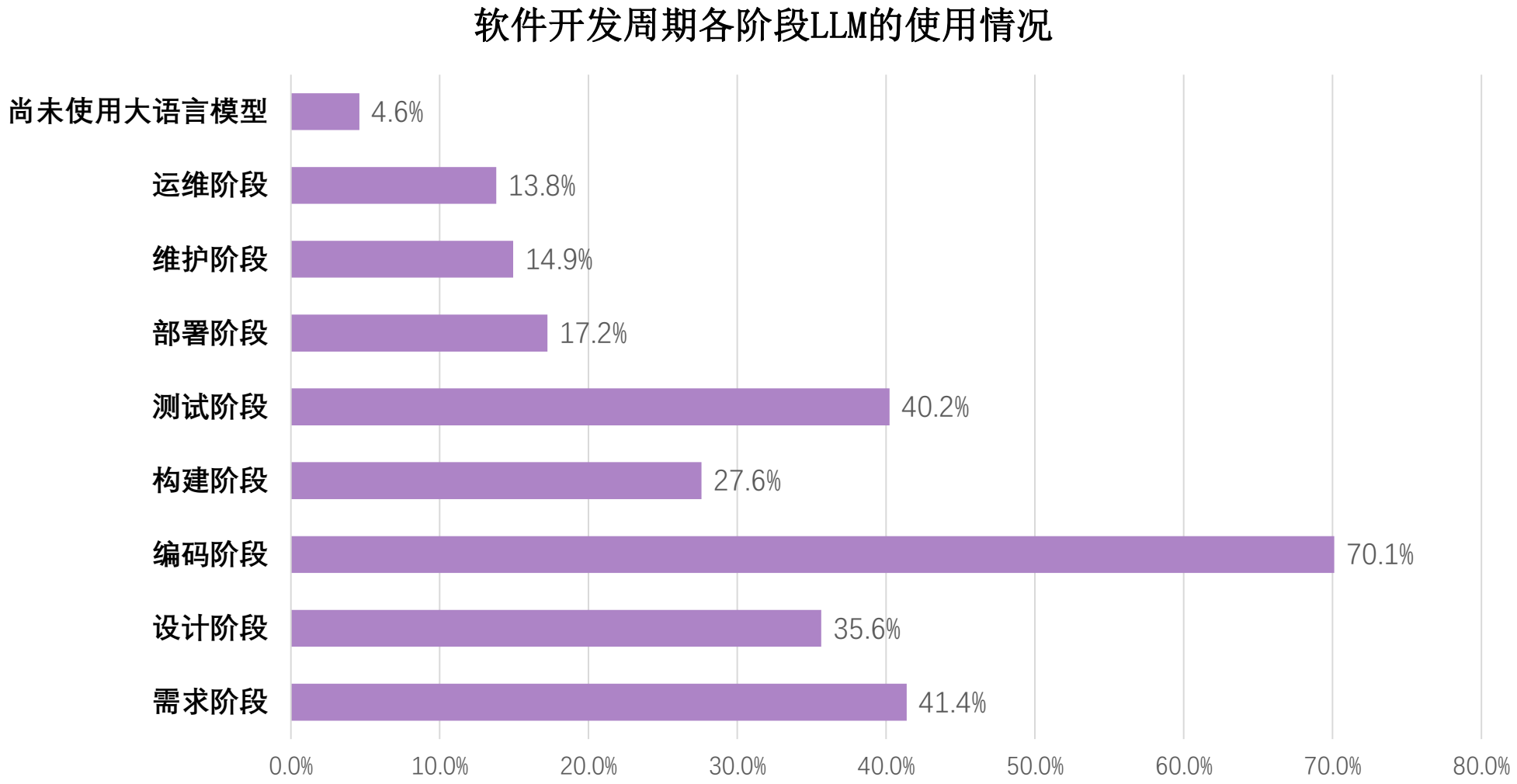
  （2）根据外部建议，将库升级到安全版本

05

大模型与研发效能

- 大模型使用现状
- 大模型对研发效能影响
- 大模型应用挑战

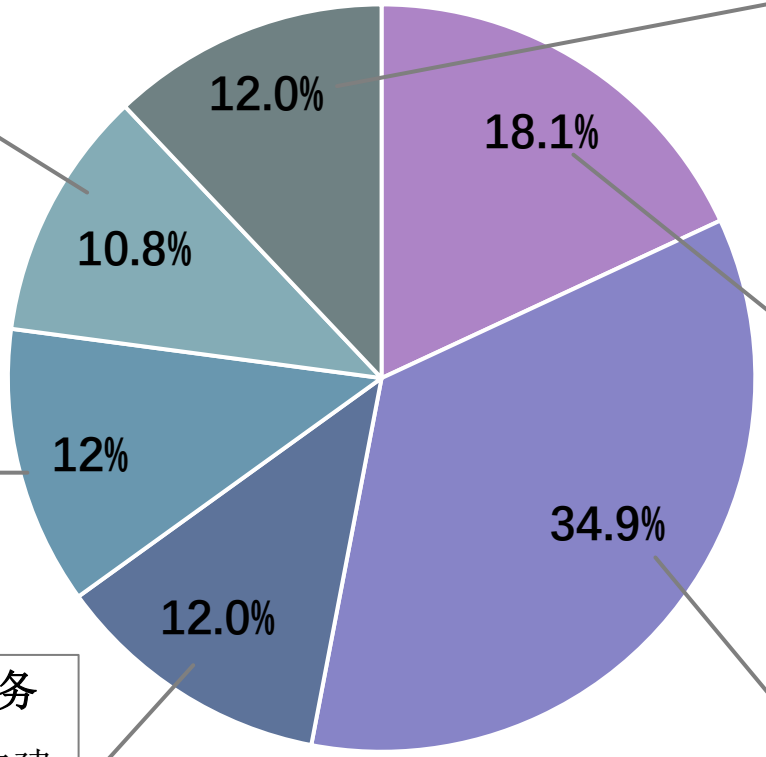
大模型使用现状



结论：大模型在软件开发生命周期的应用已经从单点（编码环节）逐渐延伸到需求、测试等多个阶段

大模型使用现状

受访对象对LLM投入规划情况分布



正在/已经基于现有的第三方LLM，开发相关工具及服务

极少数受访对象表示所在组织已经从“计划”走向“实践”，在现有LLM的基础上开发了自定义应用、插件或解决方案。

计划自主开发新的LLM及相关工具和服务

少数受访对象表示所在组织对于模型定制化需求和数据安全管控要求较高，希望基于自身业务特点，完全掌握模型研发及底层技术。

正在/已经自主开发新的LLM及相关工具和服务

极少数受访对象表示所在组织已经着手自行构建LLM模型和工具链，意味着很少一部分组织具备了成熟的技术、大规模数据的储备以及预算投入。

正在/已经引入第三方服务

少数受访对象表示所在组织已经落地了第三方LLM服务，并开始在实际业务流程或产品中进行应用与验证。

计划直接引入LLM的第三方服务

少数受访对象表示所在组织希望通过直接购买或订阅第三方LLM服务的方式，降低技术门槛与研发成本，尽快将大语言模型应用于业务场景。

计划基于现有的第三方LLM，开发相关工具及服务

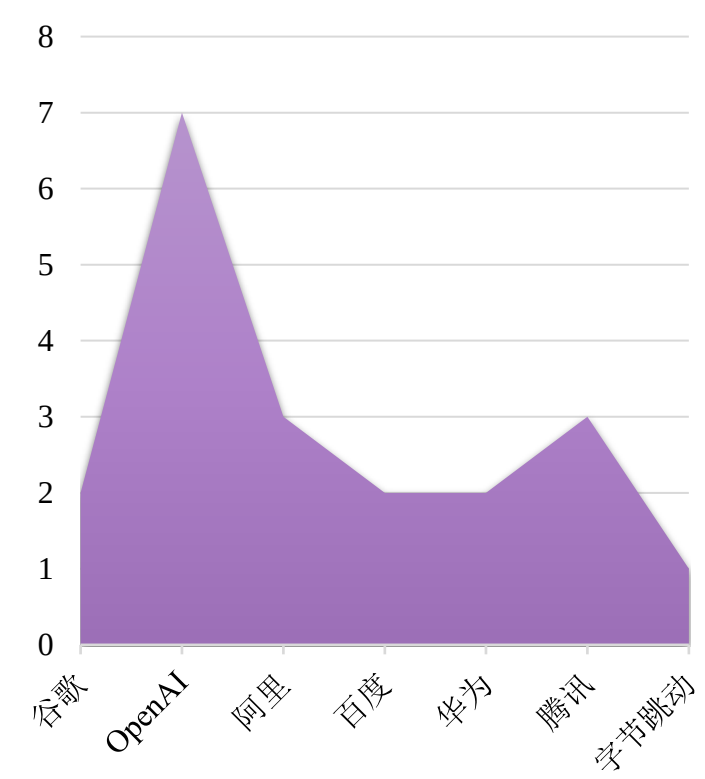
大多数受访对象表示所在组织倾向于在现有成熟LLM基础上进行二次开发，从而缩短研发周期，快速推出具备自身特色的应用或服务。

结论：对LLM的投入规划主要仍处于计划中状态并大多使用第三方模型或服务。

大模型使用现状

问题：您所在组织目前主要使用哪家供应商提供的API？

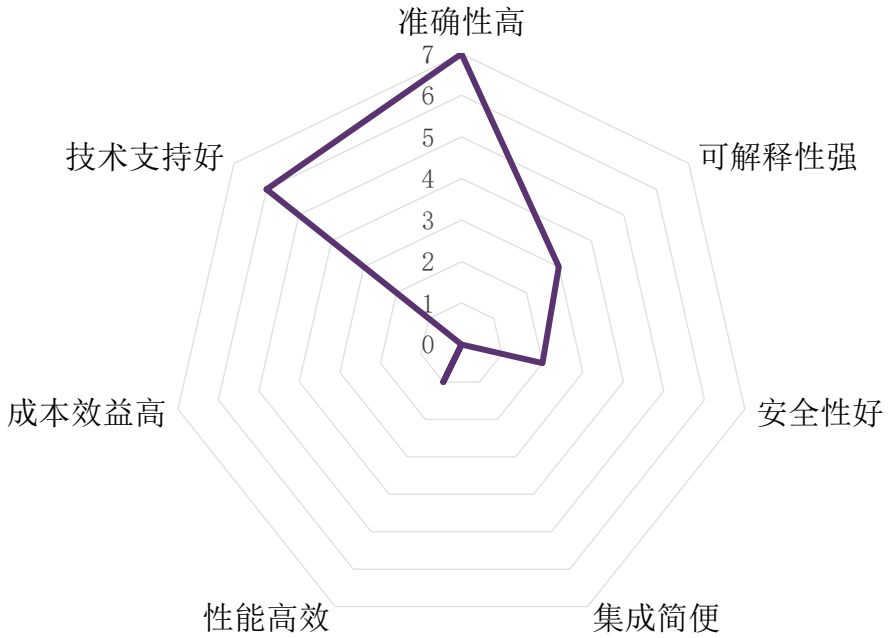
- **OpenAI（35.0%）**依旧是占比最高的供应商，体现了其在模型成熟度、性能、生态和社区支持方面的吸引力。
- **阿里 / 腾讯（15.0%）**作为国内的主流云服务与技术提供商，其研发的模型能够提供本地化及行业定制化的解决方案；与其云原生产品或其他企业服务进行打通，适合国内业务场景。
- **谷歌 / 百度 / 华为（各10.0%）**谷歌在 NLP 领域研究积累深厚，提供多种云端 AI 服务，但部分功能或服务对国内团队而言需要克服网络与政策障碍；百度作为国内较早投入大模型研发的厂商之一，也在不断完善文心一言及相关服务；华为研发的大模型更多聚焦于国内政企安全、国产化替代、硬件/软件协同优化等场景。
- **字节跳动（5.0%）**相比其他厂商，字节跳动在大模型API的对外服务上仍处于起步阶段，但最新的豆包大模型展现出巨大潜力。



结论：大模型或相关AI服务供应商市场呈多元化格局，OpenAI 占据领先，国内厂商百花齐放。

大模型使用现状

受访对象选择LLM的API供应商考量因素



- **准确性高（36.8%）**

在LLM及相关API服务的选择中，“准确性”是最主要的考量因素。精确的输出和可靠的推理能力对于产品质量、用户体验以及后续衍生功能至关重要。

- **技术支持好（31.6%）**

有约三成的受访对象表示更看重供应商的技术支持能力，尤其是在模型使用过程中可能遇到的配置、定制或故障排查等问题。如果厂商能够提供及时、专业的技术服务，能够大大降低企业在部署和维护上的难度和风险。

- **可解释性强（15.8%）**

少数受访对象重视模型的可解释性。尤其在需要合规审计、风控、医疗等对解释性或可追溯性要求较高的领域，解释模型输出结果的原因十分重要。

- **安全性好（10.5%）**

部分受访对象关注API背后的数据安全与隐私防护机制，包括访问控制、数据加密、日志追踪等，以确保模型的使用符合内部安全策略和行业合规要求。

- **性能高效（5.3%）**

少数受访对象将“高性能”视为主要考量指标，如延迟、吞吐量和资源消耗。在实时性需求较高的业务场景中，模型的运行速度和可扩展性尤为关键。

- **集成简便 / 成本效益高（均为0%）**

在本次调查样本中，尚无受访对象将“集成简便”或“成本效益”作为首要理由。这可能意味着当前各组织更倾向于在准确性、安全合规、技术支持等方面投入，而对接与价格方面的考量相对次要，或已被列入基本门槛。

结论：多数组织优先关注模型的输出质量和完善的服务保障。

大模型对研发效能影响

- **10%~20%（31%）**

大多数受访对象认为LLM能够带来效能提升，但实际效果并非十分显著，通常能为研发流程中某些环节带来一定帮助。

- **20%~30%（28%）**

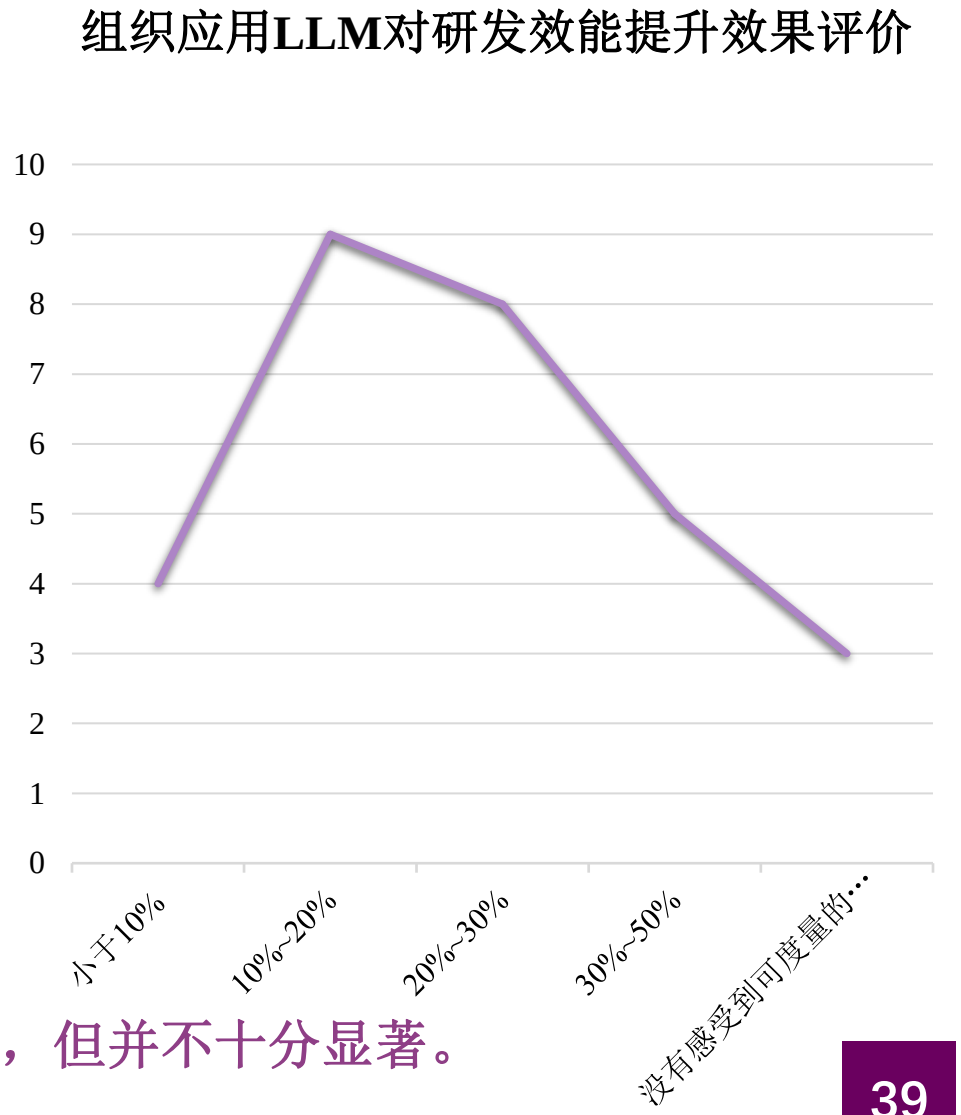
近三成受访对象认为LLM在加速研发方面发挥了较为明显的助力。

- **30%~50%（17%）**

近两成受访对象给予了更高的提升幅度评价，LLM的助力对研发效能的提升十分突出。

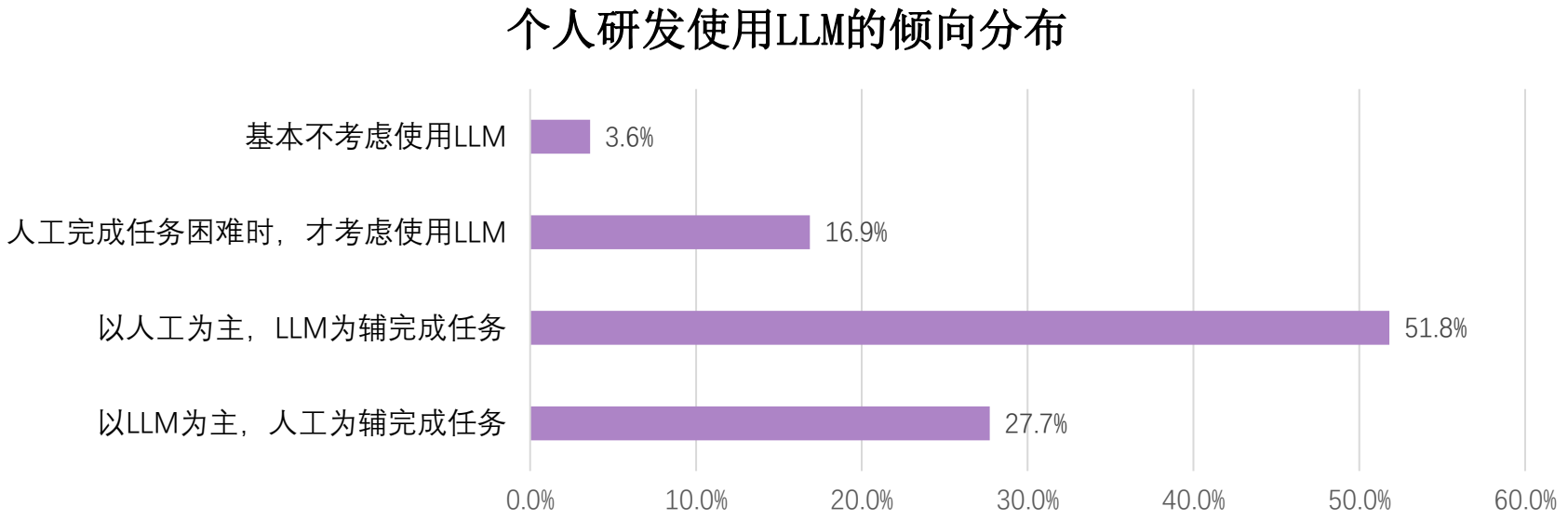
- **小于10%（14%）和没有感受到可度量的提升（10%）**

极少数受访对象认为LLM对研发效能的影响十分有限，甚至没有作用。



结论：应用LLM对研发效能有提升效果，但并不十分显著。

大模型对研发效能影响



- **以人工为主，LLM为辅（51.8%）**

超过半数的受访对象将LLM视作一种辅助工具，用于减轻重复性或简单的工作量，而非主导个人研发工作流程。

- **以LLM为主，人工为辅（27.7%）**

近三成的受访对象已经尝试将LLM放在更核心的位置，作为研发任务的主要执行者，例如直接使用模型生成和优化代码或文档，人力主要参与审核、微调与把关。

- **仅在人工完成任务困难时才考虑使用LLM（16.9%）**

约两成的受访对象表示只有在遇到特别棘手或耗时的任务时，才会启用LLM“救场”。LLM相当于一种“应急工具”，并未被纳入日常研发工作，可能由于模型本身存在使用门槛较高、性能不足的问题，或者缺乏与模型适配的研发场景与习惯。

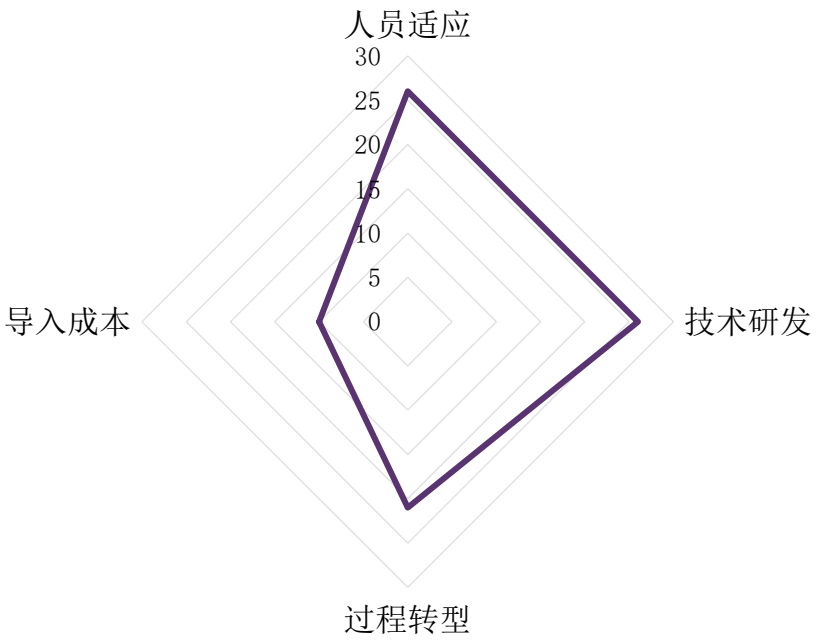
- **基本不考虑使用LLM（3.6%）**

极少数受访对象表示不使用LLM，可能存在对数据安全、隐私、或模型本身的准确度、效率与成本等顾虑。

南京大学 NANJING UNIVERSITY

结论：大模型及相关服务在研发时主要充当研发人员的辅助工具角色。

大模型应用挑战



- **人员适应（31.3%）**

随着LLM技术的引入，如何让团队成员熟悉新工具、接受并掌握新的工作方式成为重要难点。这包括对模型使用方法、风险管理以及与研发过程融合等方面的学习与适应。

- **技术研发（31.3%）**

大多数受访对象表示组织在LLM的技术实现、业务场景对接和深度定制上同样面临挑战。例如，需要在模型选择、自主研发或二次开发、基座模型训练、性能优化等多维度进行投入。

- **过程转型（25.3%）**

少数受访对象认为所在组织现有的流程与文化可能难以快速适配LLM，如研发过程的调整、审核机制需要更新、部门协作方式需重新梳理等。

- **导入成本（12.1%）**

极少数受访对象认为所在组织的挑战存在于导入成本，可能包括硬件投入、API费用、研发运维开支以及相关人员培训成本等。

结论：大模型应用挑战主要存在于人员和技术方面的双重需求。

实验室简介

软件研发运维一体化(DevOps)是软件工程发展中的第三次重大变革，而随着DevOps 的快速普及，软件研发效能已成为“泛DevOps 时代”软件企业实现“降本增效”和“持续创新”的核心竞争力。南京大学**软件研发效能实验室**(DevOps+ Research Laboratory)在国内率先开展以**软件研发效能**及**DevOps**为中心的全方位教学、科研和产业合作的学术团队，研究方向主要包括**软件及系统工程过程**、**软件体系结构**、**软件质量与安全**、**智能运维技术**，**经验软件工程**、**区块链与智能合约技术**、**面向大数据与人工智能的软件工程**等。

实验室承担了多项国家级与江苏省重大科研项目，包括主持国家重点研发计划、江苏省重点研发计划、国家自然科学基金等项目；协同挪威科技大学、(美)密歇根大学、清华大学开展《信息技术国际合作伙伴》(IPIT)计划；与华为、腾讯、 中兴、百度、阿里巴巴、美团、星环科技等知名企业建立多方面长期**科研合作**。实验室曾主办APSEC、EASE、ICSSP等软件工程学科重要国际学术会议以及中国软件大会的系列论坛，自2016年起发起**DevOps中国年度调查研究**，(南京)全球软件开发者大会(**NJSD**)和互联网架构峰会(**IAS**)等

实验室现有全职教师8人（**张贺、邵栋、荣国平、匡宏宇、李杉杉、刘博涵、周鑫、杨岚心**），客座教授5人，企业导师5人，在读博士生18人，硕士研究生100 余人。截至2024年6月，师生已在ICSE、FSE/ESEC、ASE、TSE、TDSC、ESE、JSS、IST、SPE等国际顶级软件工程会议和期刊发表论文**两百**余篇，其中**12**篇获国际最佳论文奖/最具影响论文奖。

版权声明

DEVOPS+
软件研发效能实验室