软件质量报道 QualityReport

MeterSphere

白皮书 v2.0

持续测试白皮书

The White Paper of Continuous Testing

“软件质量报道”公众号及 MeterSphere 开源社区联合出品

2023 年 9 月

持续测试

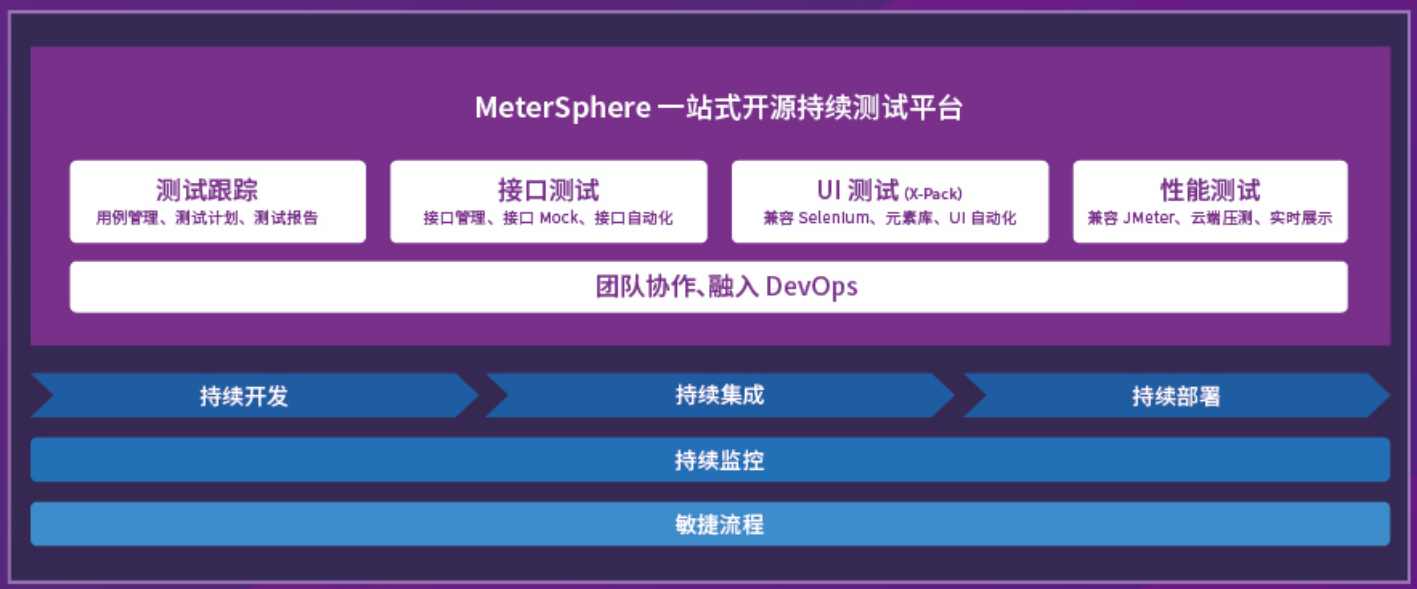
让测试不再成为持续交付的瓶颈

MeterSphere

MeterSphere

一站式开源持续测试平台

www.metersphere.io



目录 CONTENTS

1. 引言

1.1 编写本白皮书的目的

软件测试作为软件质量保证的关键手段，自从软件工程诞生之日起，就一直伴随着软件研发模式的迭代升级而持续自我更新。软件工程诞生五十多年来，历经瀑布模型、V 模型、敏捷模型等一系列的变革，慢慢形成业界认可的软件工程理论，并由此产生了测试驱动开发、持续集成、持续交付等一系列优秀实践。其中持续交付是企业最为关注的实践之一，也是敏捷和 DevOps 的核心诉求，它能够帮助企业更好地满足日益变化的业务需求。持续交付的实践过程倒逼软件测试，最终形成了与之适配的持续测试。

当前，随着企业数字化转型以及“软件定义一切”的深入发展，越来越多的企业面临着自身业务数字化转型的挑战，而这个转型过程高度依赖企业自身软件研发能力的构建和持续提升。因此，越来越多的非传统软件公司需要进入软件研发领域，并着手构建符合当下企业数字化转型要求的软件研发能力，包括与之适配的软件测试能力。

然而，关于持续测试（Continuous Testing）的优秀实践和方法还主要停留在大、中型互联网企业的内部，并未形成整个行业的共识。为此，我们希望通过本白皮书系统、深入地介绍“持续测试”理念的内涵以及如何在企业和团队内实施，并给出了相关的流程、方法和工具，旨在推动持续交付在更广泛的范围内真正落地，持续提高企业的数字业务竞争力。

1.2 如何组织、发布和维护本白皮书

本白皮书由“软件质量报道”微信公众号和 MeterSphere 开源社区共同编写而成。编写团队基于贴近企业实践的视角和态度进行策划和组织，结合自身对软件测试发展历程以及持续测试的理解，积极听取行业内其他专家的观点，并认真借鉴优秀公司所积累的经验，在此基础上完成白皮书的编写任务。

全文从持续测试产生的背景与意义、如何落地持续测试、持续测试成熟度能力模型（Continuous Testing Maturity Model，简称为 CTMM）以及持续测试企业案例等几个方面展开讨论，并介绍了持续测试产生的原因、持续测试落地的关键要素，以及如何借助 CTMM 来完成对团队测试能力的评判等内容。

本白皮书采用线上渠道（网站、微信公众号等）为主的分发模式，自持续测试白皮书 V1.0 版本发布以来，编写团队一直通过线上渠道广泛收集大家的意见，并基于这些建议进行更新，

从而形成《持续测试白皮书》v2.0 版本 。v2.0 版本的白皮书仍然通过线上渠道再次对外开放下载。欢迎业界各位同仁积极提交修正和改进意见，持续完善本白皮书，使之更加准确、全面和深入。

1.3 本白皮书给企业带来的价值

本白皮书为企业数字业务的资深测试人员、资深开发人员、高级质量管理人员、项目管理人员以及高层管理人员所准备。希望通过本白皮书能够让相关读者理解持续测试理念的内涵，以及在企业数字业务中积极采纳持续测试的价值和必要性。当然，持续测试包含的范围非常广泛，从面向开发人员的测试实践到传统测试团队的系统测试、验收测试，乃至业务系统发布后的线上测试等。一本薄薄的白皮书肯定无法详尽方方面面，期待本白皮书能作为一个引子，帮助大家推开持续测试这扇大门，并给大家在徜徉这扇大门背后的世界时提供一个相对合理的指引。

2. 持续测试产生的背景与意义

过去几年间，企业的数字化转型成为 IT 领域中不同角色都在共同关注的热点。数字业务的爆发，必然会导致企业对软件研发的强烈需求。但是，回顾软件发展的历程，毫不夸张地说，软件研发对于很多企业来说是一场高风险的投入，面临着非常多的不确定因素和较高的失败率。所以，过去的几十年，整个软件行业都在寻找降低软件生产风险和提升软件生产效率的有效方法，这种现实的需求驱动了包括如“持续开发”、“持续集成”、“持续测试”、“持续部署”，以及“持续监控”等一系列具体实践。其中“持续测试”作为企业业务数字化转型成功的基石之一，受到了广泛关注。

2.1 持续测试产生的背景

相对“CI/CD”、“DevOps”等理念的发展，“持续测试”在时间轴上属于后来者，然而也正是由于前者早期在企业内部得到了深入的实践和运用，才推动了“持续测试”的发展。

2.1.1 敏捷与 DevOps：软件研发模式的变革

敏捷与 DevOps 模式借鉴新产品研发模式、传统制造业精益生产的理念，打破了传统“瀑布模型”软件生产模式中各个阶段及角色的边界，将软件研发中各个不同角色紧密融合到一个短周期的软件研发过程中，从而实现“快速迭代、持续发布”的目标。它既强调全方位的测试能力，又在测试、开发和运维过程中良好地融合了自动化测试能力，同时它更关注利用自动化测试能力在持续交付流水线全过程中及时、准确地给团队提供当前版本的质量和使用体验反馈，从而切实保障软件持续交付过程中的质量，具体如图 1 所示。形象地说明了敏捷与 DevOps 模式不仅需要让软件交付流水线运行得更快，还需要能够以最低的代价覆盖交付所面临的业务风险，保障生产所开发的软件产品足够安全可靠。

图1 持续测试在敏捷与 DevOps 实践中的定位

敏捷与 DevOps 模式相对于传统的瀑布模型来说，其有如下几个方面的关键变化：

▶ 敏捷与 DevOps 模式更加积极地去面对软件研发过程中频繁变化的业务需求，认为需要具备支撑软件业务需求不断变化的能力。为此，其在软件研发和运维过程中引入短周期持续迭代模式，通过拆分业务需求、反复迭代的方式来响应业务的需求变化；

▶ 敏捷与 DevOps 模式特别强调“业务价值交付”这一关键要点，认为软件研发过程中所有的设计都应该服务于“将软件业务价值更快、更好地交付给最终业务用户”这一宗旨。而且，这个业务价值的交付必须经过最终业务用户认可才行。所以，软件研发过程要能够更加高效地交付业务价值，并支持在最终用户那里快速验证交付的业务价值；

▶ 敏捷与 DevOps 模式始终追求激发人在软件研发流程中的主动性，践行“人和互动比流程和规章更加重要”的原则。所以，其研发流程设计中突出为人员提供环境和支持，相信团队可以依靠自身的能力来完成软件业务价值的最终交付。

通过以上对比可以发现，这种模式成功的关键在于，团队能够以“业务价值的持续交付”为核心追求。为适应这种模式，软件研发进入了迭代周期通常为 1 至 4 周的持续迭代模式。软件研发团队的各个角色需要通力合作，在如此短的迭代周期内完成从需求分析、任务规划、代码实现、质量测试乃至上线发布的全过程。显然，这也意味着软件生产流水线上的各个环节都必须进行彻底的变革以满足如此高频的迭代节奏。而这种持续不断的业务价值交付过程也被定义为敏捷与 DevOps 最为核心的实践模型，即持续交付。

2.1.2 持续交付催生持续测试

如前所述，敏捷和 DevOps 的发展目标是加快软件的交付速度和质量，最终目的是持续交付。持续交付是企业对其软件研发能力的核心诉求，其关键抓手就是实现软件业务价值持续、高效地向业务用户交付这一闭环。

整个持续交付的流程包括计划、编码、构建、测试、发布、部署、运维和监控环节，并实现整个过程的持续迭代闭环。对于持续交付中的任何一个环节，为了满足敏捷及 DevOps 的要求，其都要遵循以下几个方面的要求：

▶ **执行过程需要“足够得快”，以满足整体迭代的效率要求。**这要求每个阶段都需要引入尽可能多的自动化工具和能力；

▶ **执行效果需要“准确且有效”，以保障整体迭代的质量要求。**这要求每个阶段都准确且有效达到最核心的质量诉求，而不是追求“大而全”的完整质量管理逻辑；

▶ **执行衔接需要“平滑而有序”，以保障整个迭代闭环能够高速运转。**这要求每个阶段在设计过程中都要考虑前后衔接，并在团队组织和工具链上实现打通。

按照以上要求，软件行业先后引入了 Scrum 敏捷研发模型、持续构建、持续部署、持续监控等一系列优秀实践。而在当下，相比于其他环节，测试领域成为企业落地敏捷及 DevOps 实施的最大瓶颈。具体来说，这个瓶颈体现在两个问题上：一是速度问题，一是价值问题。

速度问题主要体现在执行效率低，反馈时间长。

首先，传统软件测试的执行方式以手动测试为主。如今，软件自动化技术已经非常普及，软件生产流程中的代码检查、构建和部署等多个环节都已经实现大规模自动化，但软件测试的自动化比例仍然相对较低。

参考业内市场调研，即使相对容易自动化的接口测试自动化占比也未达六成，而其他如集成测试、UI 测试、回归测试等测试类型的自动化占比更低。没有足够自动化测试的支持，测试的迭代效率必然会成为瓶颈。尤其是当下企业对持续迭代的周期要求越来越短，这个瓶颈会变得更加突出。众所周知，当下很多时候测试被当成影响整个研发效率中的首要问题。

当然，执行效率低会带来另外一个问题，就是反馈时间太长。在测试没有执行完之前，开发团队几乎得不到测试关于当前产品质量和风险的任何反馈。而测试在产品没有最终完成编码前，也无法介入其中开始工作。一旦到测试介入测试并发现产品缺陷的时候，很多时候都已经为时已晚（要么就是没有足够时间来修复，要么就是缺陷修复的代价太高）。

无论是执行效率慢，还是反馈时间晚，最终都体现为测试工作未能很好地完成，导致产品发布延期，或者无奈之下只能带着巨大业务风险发布产品。

价值问题主要体现在缺少足够的“认可”，无论是从业务部门还是从研发部门。

在实际工作中，测试的结果往往得不到足够的“认可”，从而让整个团队对测试工作的价值产生怀疑。这种不能取得信任的原因主要有以下两个方面：

▶ **测试结果的稳定性不高，频繁出现明显的“误报”现象。**这种误报有可能是因为测试执行过程中的错误导致，也有可能和测试数据或者测试环境相关。频繁出现的“误报”导致大家对测试结果产生怀疑，进而不信任测试质量；

▶ **测试过程未能充分反馈软件中的业务风险。**现代软件涉及的功能和场景越来越复杂，但现实条件又决定了测试无法覆盖软件中的所有功能点和可能存在的应用场景。从商业角度来看，不同功能点及其应用场景的商业价值是不一样的，甚至相差巨大。传统软件测试用例设计的指导理念过于强调测试用例的逻辑完备性，而忽略了不同测试用例覆盖的业务风险差异巨大。于是，实际工作中经常出现重视逻辑和代码覆盖率，而忽略业务风险覆盖率的问题。测试工作花费了大量时间设计和维护了众多的无太大业务风险价值的测试用例，而忽略了对关键路径和重大业务风险点的覆盖。这种问题最终呈现出来的结果就是，业务团队无法从测试报告明确判断当前版本的业务风险到底有多大。

随着敏捷和 DevOps 研发模式的推广，以上这些问题变得越来越明显。测试越来越成为企业软件研发流程敏捷化的障碍。为了解决以上问题，各种测试理念和测试工具不断涌现。其中，为适配“持续交付”这一敏捷和 DevOps 核心目标的“持续测试”理念最为关注，并且得到了越来越多企业的采纳。

2.1.3 企业数字化转型加速持续测试的发展

随着过去 20 年互联网和移动互联网的普及，互联网和移动互联网已经深入到人类生活的每一个角落，众多企业不得不进行数字化转型来

保持与其客户、合作伙伴的交互和协作。为此，企业数字业务变得越来越重要。由于企业自身业务的复杂性和多变性，企业数字业务也必然面临着复杂性和多变性的挑战。这种挑战让越来越多的企业意识到必须要建立自己的数字业务运营和发展能力，而不能仅仅依赖于外部的某些超级数字化平台和特定供应商。

企业需要将其日常业务运营的方方面面迁移到数字世界，并保持在线运营。显然，这种迁移高度依赖企业业务软件的支撑。软件在企业内的角色定位由此从后台走到前台，不再仅仅是企业生产运营中的一个工具，而是融入企业业务运营的各个角落。如果企业需要建立数字业务运营和发展能力，企业自身就必须具备软件研发和运营能力。为此，企业不得不自己组建软件研发团队，发展自己的软件研发和交付能力。

但是，如果回顾过去几十年软件行业的发展，软件研发对于很多专业软件企业来说都是一种高风险的投入，面临着非常多的不确定因素和较高的失败率。何况现在有很多非软件企业需要在零基础上建设如此复杂的企业业务软件研发和交付能力。幸运的是，通过过去 20 多年软件及互联网行业的发展，软件行业初步寻找到了降低软件生产风险和提升软件生产效率的有效方法。这种方法体现在软件研发模式和软件交付模式两个方面：

▶ 软件研发模式实现了从传统瀑布模型向敏捷模型转换，以适应越来越快速的软件研发迭代要求；

▶ 软件的交付模式从线下模式（盒装软件）转为线上模式（SaaS，软件即服务）。由于整个社会网络带宽、容量和通信能力的大幅度提升，让软件在线访问和使用成为可能。

2.2 持续测试的定义和特征

目前，大家普遍认为持续测试是指软件持续交付流水线中的一种可随时开展且具有连续性的测试自动化测试流程。在 DevOps 中，持续测试是软件测试的一个子类别，其中软件在软件开发生命周期 (Software Development Life Cycle, SDLC) 的每个阶段都需要经过测试和验证，执行持续测试的主要目的是通过尽早和持续地测试，在持续交付过程的每一步评估软件的质量和性能。从持续测试的定义可以看出，持续测试是基于自动化测试能力，但更是一种融入持续交付实践的测试活动运行方式。与持续交付其他阶段的实践类似，持续测试实践最关键的特点在于“持续”二字。其特征包括以下四点：

▶ **全流程平滑有序：**将传统瀑布模型下的测试活动分别向软件研发运维流水线的左侧和右侧进行彻底移动，从而让测试活动在覆盖软件交付流水线的全过程上没有停顿、没有阻塞；

▶ **准确且有效：**被测系统往往很复杂，不可能做全回归测试，而是要推行精准测试，提升测试效率；

▶ **足够快：**以快速反馈为主要导向，整个测试过程要快，一方面依赖高度自动化测试（自动化测试占比应该超过 85%），另一方面依赖业务端到端的探索式测试；

▶ **高度集成：**以融入持续交付流水线为载体，测试活动将伴随软件研发流水线的每一次流动、每一个版本而频繁发生，实现测试与持续构建、持续集成、持续部署、持续运维等全面集成。

2.3 持续测试的内涵

企业在大规模推广持续测试时，一定要深入理解持续测试的内涵。持续测试本质上是企业测试人员和测试实践的一种结合，具体体现在测试文化的培养、能力的全面发展、质量的全流程跟踪三个方面。

2.3.1 持续测试是敏捷文化的体现

过去企业采用的瀑布式交付模型类似一个一个烟囱，团队之间在合作的同时，彼此之间也存在着竞争。在敏捷文化的理念中，强调的是无缝隙的组织协作，打破割裂。DevOps 流水线工具的建设虽然重要，但是敏捷文化才是 DevOps 的根本，两者互为补充，缺一不可。所以很多企业在建设敏捷文化的过程中，都将团队协作写进了价值观和企业文化里，其中也包含了测试文化。通过测试文化的建设，提升企业内部交付和测试流程的速度，也避免陷入部门墙割裂的问题之中。

和 DevOps 需要敏捷文化支撑一样，持续测试的落地也需要测试文化的支撑。企业只有把持续测试理念作为测试文化的一种，才能让它发挥应有的作用。通过持续测试文化的建设，影响参与到测试中的每个人员，逐步建立起对测试的正确态度，以下几个方面都是测试文化在团队落地的具体体现：

▶ 代码扫描出代码有问题，研发团队就应该立即整改，而不是依赖测试团队通过测试来发现代码造成的系统维护或者业务问题；

▶ 开发工程师不仅要开发功能，而且要对所开发的功能的质量负责，如编写测试代码或者从源头上保证软件的质量，不断提升测试的质量和效率；

▶ 测试工程师需要认真对待测试，如果测试人员为了测试而测试，就无法测出很多边界情况，又或者过分地追求高“覆盖率”而“自作聪明”。这样做的最终结果显而易见，那就是导致软件质量不可信，从而违背了测试的真正目的——交付高质量的软件；

▶ 测试团队在保证软件质量上需要采用高标准高要求，在软件加入新特性、新功能乃至新架构时仍能保证其质量。遇到软件的故障问题需要严肃对待，有可能一次的忽略就会导致生产环境出现故障。在测试方面需要发散性思考，通过各种测试方法和实践尽量多地覆盖各种软件使用场景。

当然上述的几个方面只是测试文化中常见的几个层面，在测试文化建设的过程中企业不能仅仅依赖测试人员来建立良好的持续测试文化。持续测试文化作为敏捷文化的一种，对不同的团队都有着不同的要求，比如产品经理也需要了解测试流程，并参与其中，开发人员在追求开发效率的同时也需要关注代码质量和单元测试质量等，最终的目的是将测试文化融入整个企业交付体系。

2.3.2 持续测试是测试全方位的有机融合与发展

根据测试方式的不同，测试可分为手工测试、自动化测试、探索式测试、随机测试等。企业可以结合现有的测试团队和测试工具、平台选

择不同的测试方式，并不是说一定要进行自动化测试或者探索式测试才能满足测试需求。从持续测试的定义中也可以看出手工测试也是持续测试的一种，但是需要满足可持续能力。企业管理者也深知持续测试带来的管理和经济效益，近几年来在逐步推动团队开展持续测试体系的建设。MeterSphere 开源社区按照不同企业成功的经验，总结出持续测试需要具备以下几个方面的均衡发展能力：

▶ **团队人员协作能力培养：**持续测试能够携不同团队群体共同参与到测试当中。从质量内建来看，测试的执行应该贯穿测试人员、测试经理、项目经理乃至开发人员、运维人员。所以持续测试应该具备不同人员的分权分域管理和跨部门测试协作能力。

▶ **多种测试类型一站式支撑：**持续测试应该具备多种测试类型。从能力角度看，持续测试应该包含不同的测试能力，包含传统的功能测试、广泛使用的接口测试、多种场景的性能测试、业务收益最高的 UI 测试、基石保证的安全测试等；其次，从场景角度看，持续测试应该包含面向不同场景的测试，比如混沌测试、精准测试等测试类型；

▶ **与周边系统无缝集成：**在测试数据共享互通上能够和企业现有的项目管理系统、DevOps 流水线等无缝集成。从持续测试的本质来看，持续测试的建设目标不仅仅是提高测试效率，其最终目标是提高软件的交付质量。而在持续测试落地过程中汇集了测试过程中产生的大量测试数据，这些测试数据同时也是企业宝贵的数据资产。一方面需要保证这些测试数据资产能够安全存储，另外一方面还需要重视测试数据的安全分享，所以持续测试应该能够和企业现有的项目管理平台的数据实现联动和能力集成；

▶ **测试资源动态化调度：**持续测试应该具备测试资源的动态化调度能力。测试资源往往是宝贵的，持续测试应该能够通过资源池的方式支持测试机的动态扩容和缩减。所有的测试任务，无论接口测试、性能测试还是 UI 自动化测试都可以采用资源池的方式进行调度与运行。一方面，可以同时满足几人到几百人团队的高频测试需求，另外一方面，也可以提高测试资源的利用率；

▶ **测试能力服务化：**持续测试应该具备测试的服务能力。不同的测试类型在开展上有可能千差万别，持续测试需要具备将不同的测试能力抽象为统一测试服务的能力。这样一来，在场景上可以帮助测试团队从测试支撑逐步转变为测试赋能，比如常见的一种持续测试场景：测试人员可以将一些冒烟测试场景授权给开发人员，每次迭代开发人员可以进行自测，这就是一种测试服务的体现。

持续测试本身可能是一个复杂的、多层次的过程。企业需要通过改进和反思现有的测试流程和体系，才能更好地推动持续测试在企业内部的开展。当然在这个过程中，可以借助同类型企业的落地经验或者借助不同的平台加速推动持续测试。

2.3.3 持续测试是全流程测试的新形态

随着软件测试技术的发展，很多企业测试团队的工作由原来测试发现缺陷逐渐发展成为测试提升软件质量。在持续测试中强调测试活动应该贯穿于整个软件生命周期中。从产品发布计划开始，直到交付、运维，测试融于其中，并与开发紧密结合，无论是需求阶段、开发阶段还是测试阶段，都需要确定在当前阶段的测试内容，并且对测试内容进行有效跟踪，以确保每个阶段的测试质量。从行为上来说，一切测试活动都可以算成是持续测试，既包括测试左移，也包括测试右移；既包括测试环境准备工作，也包含具体的测试活动；既包含测试前的架构需求调研，也包括部署后的运维监控和性能告警等。从前后顺序而言，主要包含以下几个环节：

▶ **测试规划阶段，尽早制定测试计划。**测试计划在整体的测试过程中有着承上启下的作用，即衔接测试需求、测试目标与测试最终报告。测试计划的制定与执行直接影响着整体测试的效果与质量，一个完善的测试计划不仅包含了测试任务分配、测试人员协调管理等统筹方面的内容，同时还包含了不同测试项的具体执行内容；

▶ **确保测试环境和测试数据的质量。**高质量的测试环境和测试数据能够最大程度地保证测试用例的执行质量，同时稳定的测试环境和测试数据本身也是一种质量的体现。这对测试人员提出了更高的要求，需要测试人员能够充分理解系统架构、产品部署方式、数据库表结构、运行日志分析等内容；

▶ **测试过程有序且易跟踪。**测试的最终结果不应该只通过测试报告进行体现，而是需要结合具体的测试过程。实现测试执行过程的跟踪可以有效地进行测试过程量化，比如一轮测试中，过程性需要跟踪每天多少测试用例被执行、每个测试人员平均测试多少测试用例、每个用例的时间轴节点等；

▶ **推动测试左移，实现测试与开发并行工作。**测试执行应该前置到软件开发生命周期的早期，多种工程实践可以帮助团队实现左移。比如重视测试评审，接口的定义支持从开发工具直接推送，优先投入接口定义的开发和自动化测试，引入代码扫描判断是否满足编码规范和工程标准，围绕需求（故事）持续进行测试用例的编写和修改，并且与开发保持进展协同，为开发提供必要的测试支持，使得测试与开发的工作实现同步进行；

▶ **建立完善的测试反馈机制。**建立可持续的测试反馈机制，可以帮助软件产品更好地迭代。建立完善的反馈机制也能够更早地发现问题，比如测试自动化报表生成、测试报告推送机制、缺陷自动提交能力、测试结果消息通知等；

▶ **将回归测试变为一种常态。**从软件质量保证角度来看，回归测试是必需的一种测试。作为软件生命周期的一个组成部分，回归测试在测试过程中占有很大的工作量比重，软件开发的各个阶段都会进行多次回归测试。常态型的回归测试必须采取一些较为有效的方法。比如将能自动测试的内容采用自动化测试（最大程度提高自动化率），可以通过接口自动化、UI 自动化乃至性能自动化等多种测试方式结合，最终保证回归测试的全面性；

▶ **应用部署之后，关注测试右移。**持续测试强调产品上线后持续关注和监控软件的运行状态，及时发现问题并跟进解决，将影响范围降到最低。比如生产系统的全链路压测、业务数据脱敏和多维度分析、灰度 / 金丝雀发布、业务场景 API 接口拨测等都是测试右移的实践。

2.4 持续测试的意义和价值

企业实施推广持续测试的意义和价值，一方面在于提高企业软件交付效能，另一方面也能够保证企业业务数字化转型的拓展。

2.4.1 提高企业软件交付效能

持续测试本质上还是一种测试，所以同样需要通过有效跟踪每个软件交付阶段的质量，在最大程度上降低测试人员、资源等的浪费、缩短测试周期、降低质量风险等，以便更好地促进持续交付。这种从开始到结束阶段全程质量保证的测试，确保了最终软件高质量的发布，在持续测试中体现以下在几个维度：

▶ **持续测试有效地提高了测试效率，降低了时间成本。**在传统的软件开发过程中，开发团队必须等待测试人员的测试结果反馈，这个过

程中还包含了测试环境准备等工作，但是开发人员的开发工作是不会等待测试人员反馈的。所以在测试人员的反馈之后，开发人员有可能因为新的代码问题不得不返工，付出更高的修复成本。通过持续测试，开发人员可以更快地获得对新提交代码的反馈，从而节省修复时间和金钱；

▶ **持续测试间接地改进了用户体验，提升了产品质量。**持续测试中强调测试人员站在用户的角度，模拟各种测试场景的用例设计。所以，易用性测试、A/B 测试等这类验证测试效果差异性的测试，更能提高用户使用体验的满意度，这也是持续测试理念的核心所在；

▶ **持续测试潜在地降低了故障成本。**在一些逻辑业务复杂的系统中，一个小型的错误就会产生连锁反应，从而导致不必要的停机时间，产生负面影响。对于小型企业组织来说，这可能会造成致命的打击。持续测试可识别在软件中可能不可见的错误，并有助于避免业务中断，间接地降低了软件故障率。

2.4.2 保障企业业务数字化转型的拓展

当前，企业数字业务拓展的关键是实现用户、业务和数据的在线化，并基于以上三者实现业务运营的持续优化和创新。为此，敏捷和 DevOps 成为企业数字业务中的主流软件研发模式，而持续测试则在其中承担着企业数字业务拓展中的质量保障责任。具体来说，这种特殊质量保障的价值体现在以下几个方面：

▶ **持续测试是保障企业业务数字化转型可以在业务风险可控情况下推进的关键。**显然，数字业务的业务风险控制是企业数字化转型中的底线，没有这个保障，企业不敢也无力进行彻底的数字化转型变革。快速迭代的数字业务更加重了企业对于这个问题的担心。持续测试实践强调风险控制内建在软件交付流水线的全过程，并强调通过合理的测试用例设计方式覆盖尽可能多的业务风险。同时，持续测试会将测试活动融入到交付流程中，通过自动化手段及时高效地发现软件业务风险。这样一来，快速迭代的数字业务全过程都有不断展开、有效设计的测试活动来控制其业务风险；

▶ **持续测试是保障企业数字业务创新的基石之一。**业务创新意味着需要尝试全新的东西。不管是全新的领域还是全新的模式，都意味着带来更多未知的风险。业务风险的控制方式多种多样，但对于数字业务来说，其业务软件的风险保障是最关键的一环。持续测试的理念是让测试的活动尽早发生，及早发现质量和业务风险，纠正产品设计和规划中的问题，帮助业务创新度过最容易夭折的孵化阶段。同时，当业务创新进入高速成长阶段，持续测试整个机制能够有效降低快速成长项目的质量风险，避免高速成长的项目因为质量风险而遭遇突然“死亡”；

▶ **持续测试是帮助研发团队进行测试转型升级的重要抓手。**企业要想大力拓展数字业务，其关键在于整个数字业务团队自身的转型升级。在大部分企业内，其研发团队中的质量保障由于历史遗留业务或者团队认知的原因，仍然在按传统方式运营。如果需要改变这种现状，企业需要一种明确的方法论和指导实践帮助团队形成新认知、达成新共识、执行新变革。

具体来说，这种抓手体现在三个方面：其一，持续测试强调全流水线都需要展开测试活动，这对于提升研发和运维团队的测试意识有很大帮助；其二，持续测试强调自动化测试能力的提升，这对传统测试团队实现自动化测试转型有巨大帮助；其三：持续测试强调基于交付流水线的协同，这让不同环节测试活动的衔接变得非常关键，从而能够有效促进研发团队内部不同团队在测试活动上的协同和价值认同。

3. 如何落地持续测试

企业落地持续测试的方法可能千差万别，但是方式上有普遍相似的地方。本章将从不同的维度详细介绍企业应该如何落地持续测试，首先企业应该确定整体指导框架，选择合适的实践框架作为抓手，最终依托平台或者工具逐步开展持续测试。

3.1 落地持续测试的指导框架

持续测试作为软件测试领域当前最为推崇的理念之一，在越来越多的企业内部被推广和落地。尽管每家企业落地实践都有其自身特点，但是总结起来仍然可以归纳为如图 2 的整体实践框架。

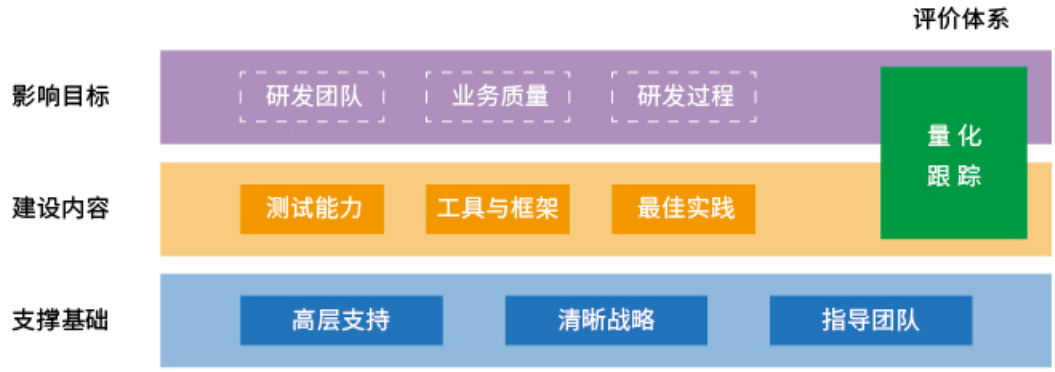


图2 持续测试落地整体框架

1. 首先，整个持续测试实践的支持基础在于取得公司高层支持，并能制定清晰的转型战略、建立指导团队；

2. 其次，在完成支撑基础工作后的建设内容包括：研发团队测试能力建设、工具与框架建设以及优秀实践落地；

3. 最后，通过这些工作形成对于研发团队、业务质量和软件研发过程的改变和影响，将持续测试实践融入到公司研发人员、软件研发流程和每个业务系统的质量保障之中。同时，整个过程需要通过合理的量化跟踪体系来帮助我们量化持续测试建设的进展和已经产生的相关影响进度。接下来我们将逐一讨论这个框架中各个要素的内涵。

3.1.1 取得高层的支持

与任何一项重要的企业实践一样，持续测试落地的第一个关键要素也是需要取得企业高层的支持。这一点之所以如此重要，是因为如下几个原因。

1. 首先，团队要意识到持续测试实践的落地并不是一个简单的事情。它的起因可以是公司内一个小团队的尝试，但是要想在整个公司层面上进行推广和落地，高层支持是帮助内部团队解决实践落地过程中各种问题的有力保障，尤其是在面对跨部门的沟通和协调时，高层支持是这些工作得以推动的关键所在；

2. 其次，高层支持对于持续落地的意义还在于，高层更容易将具体的软件工程实践价值和企业业务发展价值建立正确的关联，能让公司从业务拓展的角度来意识到持续测试实践落地的意义，更容易形成公司内部广泛的共识，可以大幅度地减小在实际落地过程中的摩擦和阻力。

所以，作为该项实践在企业内落地的推动者，其需要做的第一件事情就是——寻求企业内高层的支持。不过，在寻求高层支持时候，推动者一定要避免把持续测试描述成为一个“银弹”，让企业高层和非技术人员对此产生误解和不切实际的预期。推动者需要明确告知高层这一实践在推动过程中的困难、挑战和风险，需要得到的具体支持，以及这个实践会给企业带来的具体收益（例如成本收益、质量收益、效率收益等），并且最好能够提供相对可量化的指标。一旦有可以持续跟踪的量化指标，推动者则更容易得到高层的有力支持。

3.1.2 制定清晰的战略

在得到高层支持后，制定清晰的战略就变成了另外一个非常重要的因素。一般来说，这个战略包括业务层面、技术层面和团队组织架构层面。

▶ 从业务层面看，战略一定要紧紧围绕企业整体的业务目标展开，优先选择能够带来明确业务价值的项目和团队展开，尤其要避免落入纯粹技术层面上的“先易后难”逻辑中。这样容易导致实践推动过程中业务价值无感的尴尬境况；

▶ 从技术层面上看，转型战略一定要抓住持续测试中的核心技术手段（即自动化测试）的能力建设。无论是测试工具还是业务可测试性上，都需要向自动化测试手段倾斜。只有在实践过程中不断加强测试环节的自动化占比，才有可能不断释放持续测试实践的价值红利，得到公司高层和业务团队的持续支持；

▶ 从团队组织架构层面看，战略需要考虑两个问题：一是研发团队内测试活动的组织安排和调整；二是持续测试实践如何影响软件研发流程中的跨团队协作。

作为持续测试实践的推动者，制定好清晰的战略是其推动具体实践落实的关键指引，也有利于整个公司在具体实践过程中保持一致的理解。

3.1.3 成立指导团队

成立某种形式的指导团队是一个普遍的实践，也被认为是企业持续测试落地的关键要素。这个团队可能是在企业组织架构上真实存在的团队，但更可能是一种“委员会”性质的虚拟团队。总结来说，这个团队主要的职责包括如下几个方面：

▶ 帮助企业制定清晰的战略落地路径和执行计划；

▶ 提供必要的组织协调、资源支持和技术支持能力；

▶持续跟进实践落地的进度，及时反馈出现的问题并推动解决。

之所以需要建立这样的团队，是因为持续测试实践并不仅仅是一个技术方案或者工具平台的采纳，而是一种新的理念和运作模式的改变。这个过程较难很好地嵌入到已经存在的日常软件团队工作中，或者说简单嵌入会因为当前运作模式的惯性导致落地实践变形。

通常来说，这个指导团队是由以下几种类型的人员所构成的：

▶来自公司或者软件研发团队的高层代表。其主要职责是为团队持续取得高层的支持，并确保整个实践落地过程中不背离企业最核心的业务价值；

▶来自持续测试领域的专家。其主要职责是在实践落地过程中提供专业的技术培训和沟通协调支持，确保持续测试中的关键理念在落地过程中不走样。这类专家可以是来自企业内部，也可以是来自外部的领域内专家；

▶来自早期实践团队的领导。早期实践团队一般是持续测试实践落地的拥护者，而且也是企业实践落地的直接操作者。这类人员可以及时反馈实践落地过程中遇到的企业特有问题，并能结合企业实际情况给出建议。而且，在实践推广环节，这类成员会是最好的“现身说法”案例，可以大大提升整个实践的推广速度。

3.1.4 提升团队测试能力

企业内任何实践的落地，都依赖于企业内部人员观念的转变和能力的提升，持续测试也不例外。在这一点上，研发团队人员能力的提升显得尤为重要。由于之前长期的错误观点引导，导致国内研发团队在测试能力上的投资明显落后于开发和运维能力，大量测试外包的采纳又让广泛的测试能力提升变得不太现实。但是，在持续测试实践过程中，内建质量变成最核心的质量管理手段，这要求对于整个软件研发团队整体全面地提升测试能力。为了达成这一目标，企业研发团队需要从培训和实践两个方面入手。

能力培训

如同任何一项组织转型，能力培训都是不可或缺的环节。由于持续测试强调内建质量的能力，这要求整个持续交付流水线上的不同角色都参与其中，所以持续测试的培训工作不仅仅指测试团队的岗位，还包括开发团队、配置管理团队等岗位。当然，培训工作需要针对不同岗位有不同的侧重点。一般来说，常见岗位对于持续测试能力培训的侧重点如表 1 所示。

表1 不同岗位持续测试能力培训的侧重点

岗位名称	培训重点
开发人员	重点关注在开发环节有效提升软件内建质量的实践，包括单元测试、组件集成测试，以及相应的工具、自动化运行 / 回归测试方法等；
测试人员	重点关注测试用例设计与管理、接口与 UI 测试自动化、性能测试设计与自动化，以及测试数据管理、服务虚拟化相关工具平台使用等；
运维人员	重点关注线上压力测试、流量数据脱敏和存储、线上问题反馈，以及相关工具和自动化运行 / 回归方法等；
配置管理人员	重点关注持续交付流水线如何支持持续测试实践，包括如何集成单元测试、静态代码检测、如何在日常构建和部署中触发自动化测试等。

实践考核

任何一项能力的提升仅仅靠培训和理论学习是远远不够的，更需要在日常工作中反复实践和积累经验。这就需要研发团队各个不同角色都在测试能力上投入时间和精力，也意味着在考核体系上需要额外认可这部分的投入回报。为此，需要鼓励将持续测试实践融入到研发团队的日常工作考核过程中。这种考核需要包括软件内建质量提升的最终效果考核，也需要包括其中关键提升过程的考核。

▶ 正常情况下，有效持续测试实践的落地必然会带来比较明显的线上缺陷率的下降。因此，这个指标可以作为考核整体研发团队持续测试实践落地一个效果指标；

▶ 针对软件研发过程的不同阶段制定相应的过程性考核指标，包括如单元测试覆盖度、测试用例业务风险覆盖度、测试自动化占比等考核指标；

▶ 对于跨团队的持续测试整体运行效率设立过程考核指标，包括回归测试频率、执行时长、执行成功率等考核指标。

以上这些考核指标多以项目团队为维度进行考核。通过团队实践考核来肯定团队在持续测试方面的投入。当然，这些考核指标也会反向推动团队更积极地投入资源和精力到持续测试实践的落地过程之中。

3.1.5 选择合适的支撑工具和框架

在解决团队和人员能力的同时，选择一个合适的支撑平台和框架同样也是落地持续测试转型实践的关键因素。当前，软件测试工具及框架非常多，无论是商业的还是开源的，都可以供大家选择。由于持续测试强调测试质量管理是一个贯穿整个软件生产过程的活动，企业很多时候都需要选择多个工具和框架来完成这个目标。如果按照工具使用的目标活动进行划分，可以把持续测试的工具按照表 2 进行划分。

表2 不同测试工具和框架的分类

目标活动	工具与框架定位	典型工具与框架
编码活动	帮助团队完成包括单元测试、集成测试和代码检查等。	JUnit、PyUnit、CppUnit、SonarQube
测试活动	帮助团队完成功能测试、接口测试、界面测试、兼容性测试、验收测试，以及性能测试等。	RobotFramework、Postman、Selenium、Appium、MeterSphere
运营活动	帮助团队完成线上系统的功能、性能、容量问题的测试和验证等。	JMeter、阿里云 PTS、MeterSphere
测试管理	帮助团队对不同阶段的测试进行统一管理和协调。	Jira、TestLink、MeterSphere

尽管持续测试涉及的工具和框架非常多，但是企业进行工具选型还是要回到持续测试理念的关键诉求上来，重点需要考量如下几个方面：

▶ **工具和框架是否可以提供良好的自动化测试能力?** 自动化测试是持续测试的基础，离开这个能力谈持续测试的工具都面临着无法落地的窘境。而且，由于持续测试强调不同阶段都需要内嵌测试质量保障，所以不同活动的工具和框架需要按照测试活动的类型融入不同的自动化测试能力；

▶ **工具和框架是否可以很好地和团队持续交付工具链进行融合?** 持续测试是持续交付体系中的质量保障理念，其需要融合到持续交付工具链里面工作才能够达到最终的目标；

▶ **工具和框架需要提供合适的使用门槛。** 过高的使用门槛，例如需要专业编程技能、复杂的操作界面等，经常成为工具从企业内精英团队向其他团队推广的事实障碍。

考虑到开发团队日常工作的主要环境是集成开发环境（IDE），面向开发团队的持续集成工具和框架普遍都需要融入到 IDE 中。但是，市面上其他类型的持续集成工具相对分散，缺乏类似于 IDE 这样的集成整合平台。为此，这些工具及框架当下一个明显的发展方向就是“平台化”。一般来说，这种类型的持续测试平台需要具备以下几个方面的特征：

▶ **整合当前最普遍使用的测试能力工具。** 相对于各种工具提供的单一能力，持续测试期待能够整合多种测试需要的能力，并且能实现不同能力之间的合理衔接和转换。通常，这种平台需要整合的工具和能力包括用例管理能力、接口测试能力、性能测试能力、UI 测试能力等；

▶ **实现测试活动的全生命周期线上化管理。** 除了整合各种测试能力工具外，同样需要提供完整的测试流程跟踪能力，并且实现测试设计、测试计划、测试执行和测试报告等不同阶段工作的关联和整合，从而实现测试活动的全面线上化管理，增强人员之间的沟通协调效率；

▶ **支持不同团队的协同。** 如前所述，持续测试强调测试活动需要在不同环节持续发生，这就需要平台能够支持开发团队、测试团队、测试管理团队等不同角色的协同；

▶ **支持和外部系统的集成。** 具体表现为对于企业内包括需求管理、缺陷跟踪、构建工具、部署工具等持续交付工具链的内置对接，以及二次开发的支持能力。

3.1.6 重视行业优秀实践

尽管不同企业的测试活动运营情况以及人员能力各有不同，但仍然有很多大家共同认可的持续测试落地优秀实践。成功落地这些行业优秀实践会使得企业持续测试推进过程变得事半功倍。目前，行业内持续测试优秀实践主要集中在如下几个方面：

▶ **如何提升持续测试中的自动化占比?** 显然，这个问题是大家最为关注的优秀实践。团队可以通过测试类型迁移、自动化测试工具采纳等手段提升持续测试的自动化占比。按照行业经验，自动化测试用例比较理想的占比需要超过 85%；

▶ **如何提升持续测试中的测试稳定性?** 这里的测试稳定主要指自动化测试的稳定性。无论自动化测试占比有多高，如果自动化测试不能长期、稳定地运行，其价值都是值得怀疑的。在这方面，行业也积累了很多相关实践，例如测试环境管理、测试数据管理、服务虚拟化技术等优秀实践；

▶ **如何提升持续测试中的风险反馈质量?** 作为质量保障实践，其根本目的在于高效反馈当前软件的质量风险。而软件质量风险的反馈能力很大程度上依赖于测试用例自身的设计。类似，行业为此也提出了基于业务风险覆盖度的测试用例设计实践等；

▶ **如何保持测试的持续性?** 持续测试最终是为持续交付服务的，其相关实践也需要融合到持续交付流水线中，从而成为持续交付过程中的质量保障体系。而且，持续测试强调测试活动是伴随整个业务系统全生命周期的。即使系统已经交付上线，测试活动也不应该停止。这个融入过程也诞生了相关的优秀实践操作，值得团队学习和参考。

关于持续测试优势实践的企业案例介绍，具体内容可以参照本白皮书第五章节。

3.1.7 进行持续的量化跟踪

在推动持续测试实践的过程中，持续量化跟踪是另外一个非常重要的关键因素。只有可以量化的跟踪数据才能够给团队持续明确的反馈，也是持续争取到高层支持的最有效手段。这些量化指标可以是反馈效率、质量、成本或者业务风险覆盖等维度。以下是实践过程中常见的量化指标：

▶ **测试用例业务风险覆盖度：**根据业务系统进行业务风险评估后，设计的测试用例覆盖整个系统的业务风险比例。这是一个不同于传统测试用例逻辑覆盖度的指标，它用于反馈测试用户对于软件业务风险的管理，是一个比测试用例逻辑覆盖度更加重要的指标；

▶ **自动化测试覆盖率：**即自动化测试用例占所有测试用例的比例。这个指标反映公司和团队向自动化测试转型的进度。一般来说，这个指标达到 85% 是一个比较理想的状态；

▶ **测试用例回归通过率：**即每次测试计划中回归用例的通过率。这一指标主要反映整个测试用例运行的稳定性。持续稳定的回归测试是帮助团队提升对持续测试实践信心的关键；

▶ **测试成本节约率：**通过自动化测试执行的用例产生的成本节约。这一指标直接反映了公司在自动化测试投入的产出，是持续获取高层投入的重要指标；

▶ **测试平均执行时间：**测试用例的平均执行时间。这个指标一方面反映自动化测试的效率提升，另外一方面还能指引团队将自动化测试实践和企业内 CI/CD 流水线进行合理结合，落实提交即测试和每日构建即测试等优秀实践；

▶ **缺陷发现率：**即每个测试用例平均能发现的缺陷数。这个指标反映出测试用例设计的有效性，以及持续测试执行的效率。更有效的测试用例设计和更高效的持续测试运行都可以有效提升缺陷发现率。

当然，企业和团队在实践过程中还可以发现更多、更适合自己的量化指标。持续测试落地过程中不断积累这些指标是其重要工作之一。但是，在制定这些量化指标时候，团队需要注意两个方面的问题：

1. 量化指标一定是要有利于指导整个研发团队向正确的方向转型。错误的量化指标只会比没有量化指标还要糟糕，因为它会把团队带向更加错误的境况；

2. 量化指标一定是要能够通过技术手段持续进行跟踪。永远靠人工进行统计的量化指标基本上都难以持续。即使可以持续，也存在量化跟踪代价大、准确度低和反馈慢的问题。

另外，在本白皮书的第四章节会提出一个基于行业普遍实践的持续测试成熟度衡量模型。这个模型框架对于团队进行量化跟踪也有比较好的指导作用，大家可以作为参考。

3.2 持续测试落地的实践框架

任何成功的项目都少不了优秀的实践指导。持续测试的优秀实践框架自底向上进行阐述，主要关注解决测试稳定性、测试自动化、测试有效性和测试持续性这几个方面（如表 3、图 3 所示）。本章节将从这几方面深入展开探讨优秀实践的内涵，以及落地过程中的具体操作。

表3 持续测试落地过程中自底向上的实践框架

优秀实践类型	优秀实践主题
测试稳定性	实现测试数据集中管理；提供服务虚拟化能力；管理好测试环境与测试基础设施。
测试自动化	向接口测试自动化迁移；持续扩展自动化测试类型（比如全链路性能测试、安全测试等）；持续自动化降低自动化测试维护成本。
测试有效性	基于业务风险覆盖度的用例有效性分析；建设精准测试能力。
测试持续性	最佳方式融入持续交付流水线；推动 DevSecOps 实践；探索式测试；彻底地回归测试自动化。

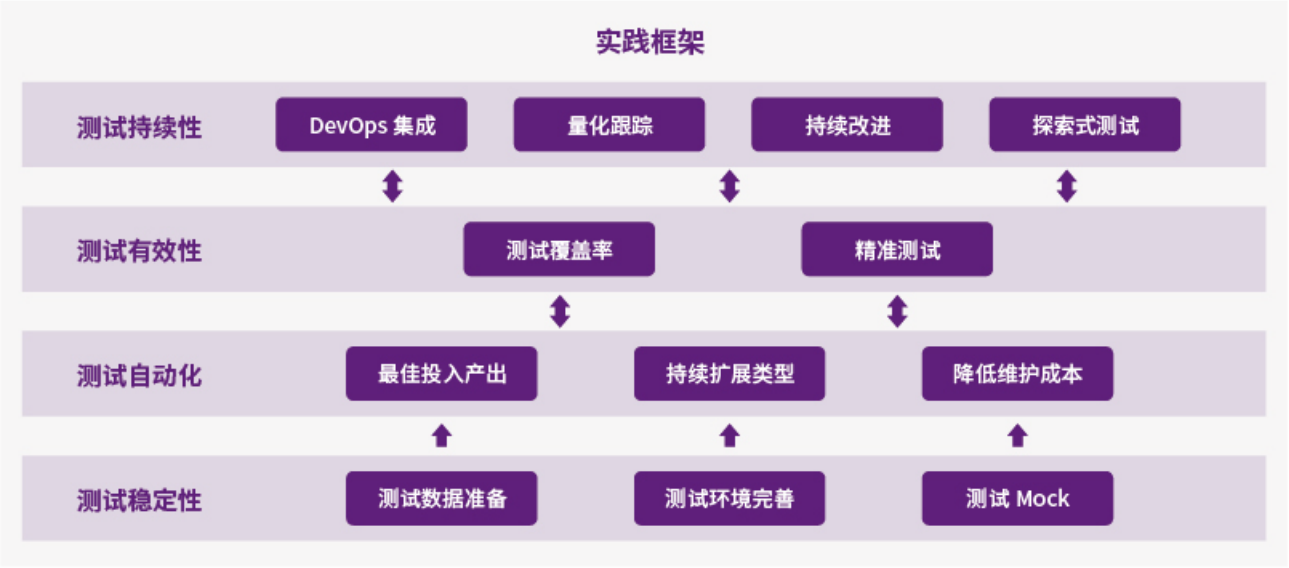


图3 持续测试落地的实践框架

3.2.1 保障测试的稳定性

3.2.1.1 确保测试环境的高可用性

测试环境准备是消耗测试团队很多时间的重要环节之一，也经常成为企业持续测试落地过程中的关键障碍。因为部署方式和使用目的的不同，测试环境分成如表 4 所示的不同类型：

表4 不同类型测试环境的分类

环境类型	部署方式	使用目的
本地测试环境	研发人员本机	研发人员用于单元测试或者组件测试
集成测试环境	测试环境独立部署	研发团队进行系统功能集成测试
验收测试环境	测试环境独立部署	产品团队进行系统功能验收和验证测试
兼容性测试环境	测试环境独立部署或外部租用部署	产品运行于多种运行环境的适配测试
性能测试环境	测试环境独立部署或外部租用部署	产品性能压测环境，通常包括大量压测集群
模拟仿真环境	类生产环境独立部署	研发团队进行真实流量验证和预发布测试

当然，企业还可能因为某些特殊测试需求而建设未在表 4 上包括的测试环境。由此可见，测试环境种类繁多，其维护成本也必然不低。尽管如前所述，通过集中的测试数据管理和服务虚拟化建设，可以有效解决测试环境对于外部数据和服务的依赖。但是，被测系统自身测试环境的部署和维护仍然充满挑战。这些挑战具体表现在以下几个方面：

▶ **不具备构建不同测试环境的条件。**这个挑战的原因可能来自成本的考虑，也有可能来自公司内基础设施建设和管理的原因。相关测试环境的缺少会明显影响测试活动的展开和持续运行；

▶ **不具备快速构建测试环境的能力。**即使企业有条件构建不同的测试环境，但是缺少快速构建的能力。不同测试环境从申请到搭建投产时间仍然会是一个巨大的瓶颈。在持续测试实践过程中，用户业务系统迭代速度快，其对环境要求的变化也会频繁，所以快速构建全新的测试环境能力是保障整个持续测试活动进行的重要因素；

▶ **缺乏环境一致性管理的能力和手段。**研发团队在测试活动时遇到的一个常见问题是测试环境的不一致性。包括系统自身版本一致性问题、依赖包一致性问题、基础设施环境一致性问题等。这些问题很多时候导致测试运行不稳定，对这类问题的排查也经常耗费研发团队大量的时间。

测试行业一直致力于解决以上这些问题。在过去不断的实践过程中，虚拟化与容器技术、云计算技术成为解决以上问题的关键所在。

▶ **云计算技术，尤其是公有云的出现让大量企业可以用非常低的成本高效获得测试环境。**无论是最基本的计算、存储和网络能力，还是各种高级的 PaaS 类服务能力，乃至各种兼容性测试需要用到的测试设备都能提供；

▶ **虚拟化技术和容器技术给多环境一致性管理带来了新的突破。**虚拟化技术通过机器镜像解决整个运行机器的环境管理问题，而容器技术则采用更为轻量的运行时封装模式进行构建，从而屏蔽了业务系统依赖的底层基础设施的差异。研发团队只需要在整个测试周期中管理好虚拟机及容器的镜像版本，就基本能够保障多个测试环境的一致性；

▶ **自动化技术大幅度提升环境搭建的效率。**无论是云平台提倡的基础设施即代码（IaC），还是容器领域的容器编排技术（以 Kubernetes 为代表）都让环境的自动化部署难题得到了极大地缓解。

持续测试团队在建设整个持续测试能力时，以上这些技术应该需要被重点考量并进入到整个实践规划过程中。当然，这些能力的建设和采纳可能需要涉及跨团队的协作，但这仍然是非常值得投入的事情，毕竟测试环境是一切持续测试的基础保障。

3.2.1.2 实现测试数据集中管理

测试数据的准备和管理是自动化测试落地的另外一个难题。在自动化测试过程中，测试初始环境的数据准备、测试过程数据的输入，以及测试结果的验证都离不开测试数据。为此，测试数据经常分成以下几类：

▶ **状态性数据：**描述测试用例开始前的系统状态数据。包括如数据库初始数据、系统依赖的磁盘数据、内存数据等。这类数据除了被系统自身使用外，还有可能用于系统依赖的外部服务虚拟化环境中进行状态初始化；

▶ **输入性数据：**描述测试用例执行的时候需要输入的数据。这类数据有可能直接嵌入到测试用例设计内部，也有可能需要依赖外部输入（比如文件等）；

▶ **验证性数据：**描述测试用例执行验证过程中的比对数据。类似于输入性数据，这类数据可能直接嵌入到测试用例设计内部，也有可能来自外部加载。

不管是以上哪种测试数据，随着测试用例规模的增加，其规模也在不断膨胀，维护成本也在快速上升。所以，集中化测试数据管理会成为很多持续测试团队必须要面对的问题。一般来说，这个集中管理平台需要具备以下几个方面的能力：

▶ **测试数据的输入和合成能力。**测试数据集中管理平台内的数据要么来自外部现成数据的输入，要么来自内部的自动合成。从外部输入角度看，需要支持实际使用到的各种数据和文件格式，并能够支持必要的数据格式检查，以保障输入数据的质量。而内部自动化合成能力也同样重要，因为实践过程中设计良好的合成数据能够帮测试团队覆盖大部分的测试场景。以最常见的“手机号”字段数据随机生成器为例，其应该具备批量生成包括不同运营商正确号码的能力，同时也应该能够伪造出一些常见的不合法格式的手机号。这样的数据合成能力会大幅度提升测试效率；

▶ **测试数据的版本管理能力。**测试数据集中管理平台需要具备支持对一个数据对象进行版本管理的能力，以适应不同版本测试用例和测试脚本的需要。对照测试数据的使用场景，其版本管理能力只需要能够对单一数据对象不同版本的标识和应用能力即可，不需要太多其他版本管理系统的高级特性；

▶ **测试数据使用场景集成能力。**无论是测试用例、测试脚本，还是服务虚拟化都有可能使用到测试数据。测试数据集中管理平台需要能和相关使用场景形成无缝的集成，让整个测试数据的使用过程变得简单直接。例如，一个接口测试场景需要一组输入数据，那么它应该可以在接口测试用例里面指定这组数据在集中测试平台中的数据对象地址和版本，然后在测试执行过程中能够通过标准协议自动获取相应的数据并用于测试。

对于自动化测试团队，当自动化测试用例到达一定规模后，并且测试数据已经成为日常自动化测试运行不稳定的主要原因之一时，就应该重视测试数据集中管理平台的构建。团队可以基于公司内部的数据管理平台自我研发，也可以采纳外部专业测试平台提供的类似工具。

3.2.1.3 提供全面的服务虚拟化能力

服务虚拟化（即日常说的 Mock 服务）是另外一个被经常提到的持续测试优秀实践。之所以服务虚拟化对于持续测试如此重要，是因为

它是解决自动化测试中外部依赖不稳定或者不可得的关键手段。用户的被测系统通常并不孤立存在，它依赖于各种外部系统才能正常运行。但是，在测试环节，这些外部系统依赖可能会因为各种原因不具备支持持续测试正常运行的能力。常见原因如下列表：

▶ 外部依赖系统还未开发完成，不具备对外提供服务的能力。这常见于多系统并行开发的场景；

▶ 外部依赖系统无法正常稳定访问。比如，外部依赖系统在测试环境未部署，或者在第三方不可控地方部署，无法在时间或者空间上满足自动化测试稳定运行的要求；

▶ 外部依赖系统在容量或者性能上满足不了当前持续测试的需求，导致部分测试用例无法正常完成；

▶ 外部依赖系统无法或者不适合模拟部分特别的场景或者数据，而部分测试用例又依赖于这些场景和数据；

▶ 外部依赖系统用于持续测试的成本太高，公司无法在持续测试环节承受这个成本。

需要强调的一点是，持续测试期待自动化测试能够持续、长期和高频率地运行，所以其对外部依赖系统的要求也是如此。这一点也更加重了持续测试对于服务虚拟化的依赖。团队在建设服务虚拟化能力时，需要注意以下几个方面的要点：

▶ 服务虚拟化需要能够快速、准确地模拟外部依赖系统。服务虚拟化是为了满足测试的需要，其实现过程是典型的“测试驱动开发”模式。在明确完测试需求后，整个实现过程不需要关注太多真实的业务逻辑，而是关注接收正确的输入，并快速产生预期输出即可；

▶ 服务虚拟化要尤其关注自身的稳定性。构建服务虚拟化的一个初衷就是解决外部系统依赖的稳定性和可获得性。如果服务虚拟化能力自身不能稳定运行，就无法支撑持续测试高效、高频的持续运行。

市面上广泛存在面向不同阶段需求的服务虚拟化框架，持续测试团队可以按照服务虚拟化使用的场景进行合理选择，以加速响应能力的建设。另外，服务虚拟化自身也需要合理的集中管理，并与测试数据、测试脚本等进行合适的联动。

3.2.2 提升自动化测试的效率

3.2.2.1 优先考虑向接口测试自动化迁移

如前所述，持续测试团队自动化测试占比超过 85% 是一个追求的目标。但是，如何达到这个目标就面临着测试用例类型的选择问题。当大家讨论自动化测试，通常想到的是端到端的 UI 自动化测试。使用类似于 Selenium、Robot Framework 或者 Appium 为主的框架编制脚本并运行。当然，这类自动化测试是非常有价值和必要的。但如大家所知，它也面临一些现实的问题：

▶ **自动化执行稳定性不够。**这类以 UI 为主的自动化测试脚本稳定性高度耦合用户系统的 UI 操作界面，不幸的是，UI 用户界面端是业务系统中变化最为频繁的部分之一。每次业务系统大的迭代都很大可能导致基于 UI 界面元素的自动化脚本失效，甚至有些时候需要完全重写自动化测试脚本；

▶ **自动化执行效率较低。**UI 自动化测试需要启动完整的应用客户端，模拟用户操作，获得界面反馈。这个过程运行链路较长，一个测试用例以分钟为单位运行是常见情况。当测试用例数量较多时候，一次小规模的自动化回归测试也需要几个小时，甚至是以天计算的时长。这样的效率会带来一个明显的副作用，就是它很难友好地集成到持续交付流水线中的每次构建中，从而实现高频率的持续运行，以便给研发团队提供及时的测试反馈；

▶ **测试用例粒度过大。**UI 测试一般以端到端的测试场景为主，而验证手段也以界面表现为主。一旦测试用例执行失败，其并不能给出相对精确的测试失败原因，给研发团队修复问题提供明确指导。

如上所述，利用 UI 自动化为主的测试用例类型很难达成推动测试自动化占比大幅提升的目标。相比于 UI 自动化测试，接口自动化测试更加适合成为测试自动化的主要测试用例类型，因为它非常适合持续测试理念下的自动化需求（如表 5 所示）。

表5 UI 测试与接口测试的对比

类型特征	UI 测试	接口测试
执行效率	端到端执行，每个用例典型执行时间在数分钟级别。	单接口或者基于接口编排的场景执行，每个用例典型执行时间在 1 分钟以下。
实现难度	依赖前端客户端录制技术，需要较为复杂的脚本编写能力。	基于接口规范进行设计，设计以界面操作为主，实现难度较低。
执行稳定性	高度依赖普遍频繁变化的前端 UI 界面，用例失效风险大。	依赖接口稳定性。大部分应用系统都会保持接口向前兼容，实践过程中稳定性有保障。
发现问题有效性	失败用例只能给出大体排查方向，需要团队深入寻找失败原因。	基于接口进行良好设计的用例能做到精准定位问题场景和原因。

基于以上分析可以看出，接口测试是整个测试自动化中最应该追求的测试类型。如果需要推动整个测试自动化占比，应该优先推动接口自动化的实践。一般来说，接口自动化测试用例应该占持续测试团队自动化用例超过 85%。

当前，对于大部分测试团队，为了推动测试自动化实践的落地，一个关键的工作就是实现测试自动化的工作重心从 UI 自动化向接口自动化迁移。这个迁移过程的典型过程如下：

▶ **首先，面向存量系统接口进行测试。**测试团队与开发团队需要在系统接口定义上形成更多的沟通，乃至形成统一的接口管理平台。测试团队需要基于当前接口定义进行接口测试用例设计、实现和执行，并通过接口测试自动化形成和开发团队的高效反馈机制。比如，实现接口测试每日回归，并及时和开发团队反馈回归结果。在这一阶段，接口测试用例的典型形式是单接口测试（围绕一个单接口形成正反结合的用例组合）；

▶ **其次，拆分现有自动化测试场景中的接口测试场景。**很多测试团队已经有成千上万的端到端 UI 自动测试脚本，并在日常过程中每天运行。测试团队需要去分析这些自动化测试场景，判断是否可以从其中剥离出相应的接口自动化测试场景。通常来说，优先选择通过接口测试去验证后端业务逻辑，将 UI 测试保留用于界面展现验证。通过这个拆分工作可以发掘出大量进行接口编排测试的场景；

▶ **最后，配合业务系统接口改造进行接口测试。**这是一个需要开发和测试团队共同努力的工作。不过好在随着前后端分离和微服务框架

技术的普及，越来越多的开发团队也有很强意愿进行相关的改造（这个改造工作对开发团队的价值也同样明显）。在这个改造过程中，测试团队需要尽早参与，争取按双方约定的接口规范尽早实现必要的接口测试自动化，从而通过接口自动化测试反向驱动开发业务改造过程中的质量保障，实现业务改造过程中的“测试驱动开发”。

总结来说，持续测试中的测试自动化实现高度依赖于接口测试自动化的实现。研发团队需要将其测试自动化的主战场切换到接口测试上，并通过以上介绍的相关实践方式推动接口测试自动化落地。

3.2.2.2 考虑安全测试自动化的引入

企业在日常运营中需要加强客户对企业产品的黏性，而业务系统是否能够安全、稳定地运行是非常重要的核心因素之一。在这种情况下，企业通过安全测试来识别和解决业务系统中存在的安全漏洞。

根据百度词条的解释，安全测试是指：在 IT 软件产品的生命周期中，特别是产品开发基本完成到发布阶段，对产品进行检验以验证产品符合安全需求定义和产品质量标准的过程。简单来说，安全测试就是通过测试，来确保软件产品信息安全的质量，以确保泄露数据的概率更低。

安全测试的目的是发现现有安全机制中的缺陷，发现软件应用程序的漏洞或弱点，主要目标是识别和评估系统的脆弱性，并确定数据和资源是否受到保护，免受潜在黑客的攻击。安全测试是在 IT 软件产品的生命周期中，特别是产品开发基本完成到发布阶段，对产品进行检验以验证产品符合安全需求定义和产品质量标准的过程。可以说，安全测试贯穿于软件的整个生命周期。而在所有安全测试中，代码质量（SAST）、Web 应用程序扫描 (DAST)、容器扫描 / 漏洞依赖分析、软件构成分析、自动漏洞扫描较为适合进行自动化安全测试。

3.2.2.3 合理评估哪些项目适合自动化

在项目中引入自动化测试前，需要从业务本身、项目团队两个维度对项目进行合理的评估，判断是否真的需要进行自动化测试。自动化测试虽然能够助力产品质量的提升，但是如果采用得不合理，反而降低了测试的能效。虽然行业中没有严格的标准评定什么样的项目适合开展自动化，但是结合测试特点，我们应该从项目版本改动周期、自动化测试建设时间、自动化建设成本等维度进行综合评估，具体地可以从以下几个方面进行权衡。

▶项目变动相对较少。测试脚本的稳定性决定了自动化测试的维护成本。如果项目需求变动过于频繁，而且每次变动都涉及测试脚本的调整，即使在一些平台或者工具的帮助下，脚本维护成本还是大于自动化所节省的测试成本，那么自动化测试就需要考虑是否有必要开展了。当然，如果能够有方法或者方案将变动的部分自动反馈到对应的自动化脚本的某些场景或者片段，就能够极大地减少测试人员对测试脚本的维护时间，这种情况下，则依旧可以开展自动化测试。如果项目中的某些模块相对稳定，而某些模块需求变动性很大，可对相对稳定的模块进行自动化测试，而变动较大的模块仍用手工测试或者其他的测试方式；

▶ 项目周期足够长。自动化测试从需求范围的确定，到自动化测试场景的设计，以及测试脚本的编写与调试，均需要测试人员投入相应的时间来完成。如果项目的周期比较短，短到都来不及编写自动化测试相关的内容，那么此项目的自动化测试也不宜开展；

▶ **测试成本可预估。**一个项目的自动化场景和要求是随着项目不停地迭代而越来越高标准的。当项目系统足够复杂的时候，自动化测试就不是一个人、一台笔记本能够完成的，需要团队人员和测试资源的长期投入。只有随着项目迭代的长期投入，才能更好地发挥自动化测试的价值。所以还需要充分评估未来项目上人力和物力的投入成本；

▶ **大量测试可重复、可回归。**如果项目中存在大量的测试需要重复执行（如定时任务），或者每周或者每天都需要一定的回归用例进行验证，并且这些测试和回归还占用了大量的人力时间时，那么就可以考虑将此部分的重复测试或者回归测试转换成相应的自动化测试。当然，是否将所有的测试都进行自动化，还是部分测试进行自动化，需要结合项目变动频率和项目周期进行整体评估。

3.2.2.4 持续降低自动化测试维护成本

自动化测试另外一大挑战就是维护成本问题。这个维护成本包括两个典型原因。**其一是自动化测试用例自身实现不稳定。**被测对象并没有发生变化，但测试用例因为自身实现不稳定而带来维护成本。解决这个问题一般需要去寻找更加稳定的方式设计和运行当前测试用例。比如，测试用例脚本中的断言判断规则是否可以更灵活，UI 测试中的元素 ID 寻找方式是否可以更智能等。当然，如果测试用例运行高度依赖外部测试数据或者服务，测试数据管理和服务虚拟化也会对降低这类测试用例维护成本有很大帮助。

其二是用例更新困难。随着业务系统的不断更新，自动化测试用例也要保持更新以维持有效性。如前所述，解决这个问题最主要的途径是实现测试用例类型的迁移，让自动化测试用例更多关联相对稳定的接口定义，而不是频繁变化的 UI。另外，模块化复用也是解决这个问题非常关键的手段。测试用例管理自身也可以分层。实践中可以把简单独立的原子场景进行编排，形成一个测试用例模块。更高级的测试场景利用这些测试用例模块进行组装。

例如，一个典型的电商系统通常包括如图 4 所示的完整流程：

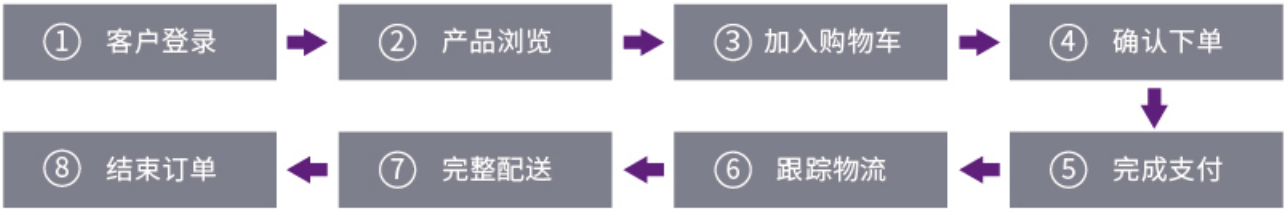


图4 典型电商系统的示例流程

假设需要对以上流程进行接口测试设计，比较合理的设计方案是对其中每个环节内的接口测试用例进行模块化管理，然后基于这些模块灵活组装不同新的复杂场景。以上每个环节内部也可以包括多个接口测试用例。例如“加入购物车”环节不仅仅要测试加入购物车的接口是否工作，还要验证购物车内查询接口运行是否符合预期。通过这种模块化管理和复用的方式，用户每个环节的更新变化大多时候都只关联到对应模块内的用例更新，而不需要更新上层依赖这个模块的复杂用例场景。

Codeless 自动化测试也被认为是降低自动化测试维护成本的重要手段之一，业界也在这些方面做不断的尝试。这类手段最主要的努力方向主要有以下几个方面：

▶通过引入更加友好的自动化测试交互界面，让大量的测试用例和脚本的维护可以用直观的界面交互方式完成，从而降低维护这些用例和脚本的成本。同时，这种模式也能有效降低测试人员的技术门槛；

▶通过引入用例执行过程中的高级“自愈”功能，让用例执行过程对于其背后依赖的环境和被测系统有更好的适配性。比如，Web UI 测试能够自动化尝试多种模式匹配界面上的目标元素；

▶通过引入更加友好的“录制—回放”机制支持用例和脚本的自动生成过程。例如，“录制—回放”工具越来越能和其他工具结合，实现基于数据驱动的用例生成机制，对降低用例维护成本的效果非常明显。

改进自动化测试用例维护成本的另外一个重要手段在于测试用例的设计思路。基于业务风险的测试用例设计强调用尽可能少的测试用例覆盖尽可能高的业务风险，从而避免维护大量无实质业务风险价值的无效用例，整体上减少测试用例的维护成本。关于如何有效进行基于业务风险的测试用例设计，请参考本白皮书后面的章节。

3.2.3 积极提高测试的有效性

3.2.3.1 基于业务风险覆盖度的用例设计

当讨论用例有效性问题时，首先我们需要明确一个现实：对于现代复杂软件系统进行全覆盖测试是一个几乎不可能（或者说在经济上不允许）的事情，这就意味着必须有所选择。在做这个选择时，大家需要回到测试活动的初衷，即保障产品质量风险，从而控制产品业务风险。因此，对系统测试场景选择遵循的第一原则就是场景的业务风险。而考核测试对产品质量风险的保障力度也就转换成为测试对于系统业务风险的覆盖度。为了度量这个覆盖度，我们一般需要进行以下两步操作：

第一步：对系统需求进行基于业务风险的分析

首先，团队需要理解什么是业务风险。对于任何一个软件系统需求，其背后都会对应业务场景，以及这个场景出现缺陷导致的业务影响。而业务风险可用如下公式进行定义：

业务风险 = 业务发生概率 x 业务影响程度

依照以上公式，可以看出高频发生的场景和带来严重业务影响的业务需求经常有较高的业务风险。团队在做系统需求的业务风险主要就是识别这两种情况，并最终结合上面的公式进行业务风险的判定和排序。

要分析业务发生的概率或者业务影响程度，都需要有一个参考基准。通常建议团队选择一个典型且简单的系统需求来做基准判定。例如，对于一个资产管理系统，我们可以把“查看资产详情”定义为基准场景。为量化分析业务风险，可以将其“业务发生概率”和“业务影响程度”都定义为 100。以此为标准依次分析出其他业务场景的业务风险。具体数值可能如表 6 所示：

表6 资产管理系统业务风险分析表（示例）

需求简述	发生概率	影响程度	业务风险值
查看资产详情（基准）	100	100	10000
增加全新资产	10	500	5000
删除存量资产	5	50	250
更新存量资产	50	100	5000
查询相关资产	200	200	40000
打印资产清单	10	10	100
生成资产报表	20	10	200

当然，团队还可以对以上每个场景内的业务风险再做进一步的细化分析。比如团队可以对查看资产详情内的各种子场景再展开分析，并确保子场景的业务风险累加数值和主场景风险值一致即可。在完成基于业务风险的需求分析后，研发团队基本清楚了业务系统真实的业务风险分布情况。这个业务系统风险的评估过程需要经常回顾，尤其是在不断积累生产运营经验后，需要团队来重新确认以前的判断是否合理。

第二步：按照业务风险覆盖度进行用例设计

完成业务系统需求风险评估后，团队需要以业务风险覆盖度为最终追求目标进行测试用例设计。整个用例设计过程的一个基本指导原则就是：用尽可能少的用例覆盖尽可能多的业务风险。具体来说，这个过程可以细分成如下两个过程：

▶ **首先，按场景的业务风险度进行场景选择。**这基本上都是从最高业务风险场景开始选择并进行测试用例设计的。至于需要达到多高的业务风险覆盖度，这会因不同系统和不同团队而异，但无论如何都需要有一个可以明确量化的指标来作为参考；

▶ **其次，对于每个选择场景进行详细用例设计。**对于每个场景，都可能有多维度输入，并且每个维度输入还会有多个可能选项。为此，软件测试领域有一系列的用例设计方法，例如组合覆盖、正交法、Pair-wise 等。其中，线性膨胀法（Linear Expansion）被认为是一个在用例数量和业务风险覆盖度之间达到很好平衡的方法。下面我们用一个具体例子说明线性膨胀法的工作方式。

如前面的资产管理系统，需要为“添加全新资产”进行测试用例设计。已知添加每个新资产需要提供“资产编号”、“资产类型”、“所属部门”三个必选参数。其中，按等价划分法把资产编号分成三类输入，即“非法资产编号”、“合法有效新编号”、“存量冲突编号”；资产类型有二类：即固定资产、流动资产；所属部门有三类：即业务部门、研发部门和后勤部门。

在线性膨胀法里，首先需要在多种可能组合中寻找到一个最主要组合，也是业务风险最大的一个组合。例如，团队一致认为“合法有效新编号”、“固定资产”、“业务部门”这样的组合为最主要组合。线性膨胀法就会先按这个主题组合设计第一条用例。然后，在每条新加用例保持和主组合只有一个维度的数值不同。以此类推，直到所有维度选型覆盖完成。这一场景的用例列表如表 7 所示。

表7 基于线性膨胀法设计的测试用例（示例）

用例编号	资产编号	资产类型	所属部门	备注
01	合法有效新编号	固定资产	业务部门	主组合
02	非法资产编号	固定资产	业务部门	资产编号发生变化
03	存量冲突编号	固定资产	业务部门	
04	合法有效新编号	流动资产	业务部门	资产类型发生变化
05	合法有效新编号	固定资产	研发部门	所属部门发生变化
06	合法有效新编号	固定资产	后勤部门	

通过表 7 可以看出，线性膨胀方法的完整测试用例数量为 6 个。相比较来说，组合覆盖需要设计 18 个测试用例（18=3x2x3），而正交法虽然只需要设计 3 个测试用例，但是其对业务风险的覆盖度又明显低于线性膨胀法。

线性膨胀法还会有另外一个优势。就是当用例运行失败后，可以非常方便地定位问题。实际工作中主要组合是被反复测试的场景，一般容易保障其成功率。一旦其他组合出现问题，由于和主要组合只有一个维度的输入值不同，非常容易定位到哪个维度带来的错误。甚至，通过友好设计的用例命名就可以帮助我们知道问题的精确方向。

经过基于业务风险覆盖率的测试用例设计，团队测试用例的有效性会得到大幅度提升，可以有效避免大量无明显业务价值的测试用例浪费团队日常实现和维护时间。而且，由于其能有效帮助定位问题，测试用例运行报告会对开发团队带来明显的问题定位价值，更容易体现测试活动的业务价值。

3.2.3.2 积极评估核心业务的全链路压测

全链路压测是指全业务覆盖、全链路覆盖的系统压测方式。开展全链路压测需要在生产环境实施压测，采用线上的真实数据、模拟线上的真实流量请求、模拟同等的用户规模、模拟同等的业务场景，来对完整的业务系统从网络、应用、中间件、数据、服务器、外部依赖等进行全覆盖的压测，并持续调优的过程，以找到系统的瓶颈点，对系统容量进行真实客观的预估。企业在开展全链路测试之前应该做好相应的自身评估：

▶ 结合业务实际需要，并不是所有的系统都需要进行全链路压测。大家熟悉的各种国内电商大促活动，其背后涉及的系统和模块极其复杂，传统的压力测试已经无法满足对这些大型系统的性能评估，所以才需要全链路压测，以此来评估各个环节的系统承压能力的瓶颈和容量。所以企业应该先重点从业务需求点出发，评估全链路压测的投入产出比；

▶ 坚持以自动化的方式从局部到全链路逐步推进。对于绝大部分测试团队来说，全链路压测的落地要求都显得过高，而且其中还会涉及大量的跨团队协作。所以，压力测试实践落地更适合在局部先展开。这个“局部”可以是局部的组件和接口，也可以是局部的场景。通过从局部进行实践，找到关键的性能瓶颈并指导开发团队进行改进，或者指导运维团队进行合理的容量规划。与此同时，压力测试的自动化执行是一个更为重要的目标。只有能够自动化执行压力测试，才能够实现在系统迭代过程中持续稳定地运行，从而及时发现系统的性能滑坡问题；

▶ **做好压测业务模型和数据模型梳理。**全链路压测在线上系统进行开展，所以首先应该将核心业务和非核心业务进行拆分，确认流量高峰针对的是哪些业务场景和模块，针对性地进行扩容准备，而不是为了解决海量流量冲击而将所有的系统服务集群扩容几十倍，这样会造成不必要的成本投入。其次全链路压测的所有数据都会在线上系统实际产生，需要做好流量染色、数据隔离、数据脱敏等方案设计；

▶ **适当引入外部专家服务。**全链路测试是一个非常依赖于执行人员经验和素质的测试类型，而这些经验和素质需要大量的实践积累，同时全链路压测也涉及企业的多个团队的合作。适当引入外部全链路专家服务是解决这一问题的有效手段。外部专家可以在三个方面帮助到企业：一是在测试用例设计和流量模型规划上，由于外部专家长期执行全链路测试，更有经验发现系统的性能瓶颈，并针对性地设计测试用例和流量模型；二是对于测试结果的解读和解决方案的指导，外部专家可以更深入地读懂测试结果并提出针对性的解决方案，包括对于性能瓶颈的定位和系统容量的评估；三是作为外部引入专家，可以独立于企业不同的团队，作为不同团队的桥梁，推动和促进不同团队的合作。

3.2.3.3 建立精准测试实践能力

测试有效性的另外一个重要实践就是精准测试。顾名思义，精准测试就是让测试活动能够有针对性地精确发生。它强调整个测试活动应该在明确、精细的度量指标指导下进行，并且需要对整个测试活动进行持续有效监控，及时采集到度量指标的反馈来促进测试过程的不断完善。从以上定义可以看出，精准测试需要包括如下构成要件：

▶ **度量标准：**一组精确监控测试活动质量的度量标准和数据。由于软件产品本质是由软件代码构建而来，所以软件系统精准测试最常用的度量指标自然就和代码关联起来。所以，大家经常采用的软件精准测试度量标准有代码覆盖率、分支径覆盖率、函数方法覆盖率等；

▶ **测试活动：**对目标系统的测试活动。在精准测试下的测试活动与其他测试活动大体相同，但是精准测试下的测试活动中需要持续收集和监控度量标准下的度量数据；

▶ **度量分析：**根据测试活动中产生的度量数据进行集中分析，发现测试活动中的盲点和潜在问题；

▶ **过程改进：**根据度量分析的结果，指导测试活动的改进。例如补充部分测试用例，修改测试用例的运行顺序等。

按照以上构成要件，推行精准测试的团队需要建立以下能力：

▶ **建立度量数据的采集能力。**精准测试依赖度量数据，所以团队需要能够在测试活动进行过程中全程监控并采集到需要的度量数据。由于软件精准测试的度量标准基本是和代码相关的，这个采集能力通常需要给被测系统嵌入“插桩”组件（通常是在构建过程中集成到被测系统中），收集测试活动时候的代码具体运行情况；

▶ **建立度量指标的分析评价能力。**分析采集到的度量数据，结合对于系统源代码的静态分析，从而可以将原始度量数据转换成为度量指标维度的数据，并按照系统的属性和级别制定相应的评价模型；

▶ **建立代码与用例的双向关联机制。**精准测试的最终目标仍然是改进测试过程，所以需要建立测试用例和系统代码之间的双向关联机制。让基于代码的度量指标能够关联到相关测试用例，从而达到改进测试过程的目标。

在具备以上精准测试能力后，团队进入精准测试落地运行的环节。和其他持续测试优秀实践类似，整个持续测试的运作过程仍然需要和持续交付流水线的深度融合。融合后的整体运作模式如图 5 所示。

图5 精准测试落地环

1. 将精准测试的度量数据采集过程融入到持续构建过程中。可以通过构建过程加入“插桩”组件或者在代码内框架加入“染色”日志等方法，让构建出来的系统可以在测试过程中持续对外输出需要的度量信息；

2. 将构建过程产生的制品包通过持续部署过程部署到测试环境，以便接下来测试活动的展开；

3. 基于用例库进行持续测试活动。需要注意的是，这里的持续测试活动可以是各种类型的测试活动，包括单元测试、集成测试、功能测试乃至性能测试等。所以，精准测试实践可以应用到持续测试中的各种测试类型上；

4. 基于测试活动产生的度量数据，结合对代码库的静态分析，将度量数据输入到度量库中。团队再基于代码和用例之间的关联关系将度量数据用于指导改进测试活动。

在实践过程中，精准测试的效果非常明显，尤其是在智能筛选回归用例集、快速定位缺陷具体位置等方面有良好的表现。而且，由于精准测试对于测试改进有精确指导方向，能够帮助团队发现传统方式不容易覆盖的业务逻辑和代码片段，从而将整个测试用例的代码覆盖率提升到传统用例设计很难达到的新高度。

3.2.4 思考和探索测试的持续性

3.2.4.1 最佳方式融入持续交付流水线

持续测试一个重要的实践是要将测试活动融入到持续交付流水线，从而实现测试活动在交付过程中的内建。考虑到持续交付最核心的载体就是交付流水线，实践过程中普遍会将测试活动集成到持续交付流水线中。但是在这个集成过程中，需要确保以下两个方面的影响可控：

▶ 对于持续交付执行效率的影响。持续交付执行的频率非常高（如每日构建或者提交时构建），且开发团队高度依赖这些活动的及时反馈来开展日常工作。所以，这些活动的执行时间不能太长。例如，提交时构建需要在 30 分钟级别完成，而每日构建也需要在 3~5 个小时内完成。加入测试工作势必影响它的执行时间，团队需要控制被集成的测试活动对整体执行时间的影响；

▶ 对于持续交付执行稳定性的影响。持续交付过程稳定运行是其提供反馈的基础保障。被集成的测试需要确保其自身的稳定运行，从而不会影响整个持续交付过程的稳定性。

一方面，需要确保被集成到持续交付流水线中的测试活动能够稳定高效运行，另外一个方面还需要追求测试对质量保障的有效性。这就要求能够对测试用例进行有效的影响分析，确定不同阶段应该执行的测试范围。整个测试范围选择的基本策略可以从当前软件变化 / 生产缺陷和业务风险覆盖度这两个维度考虑，推荐如表 8 进行交付流水线的融入策略选择：

表8 测试范围选择的基本策略

	高业务风险覆盖用例	中业务风险覆盖用例	低业务风险覆盖用例
有明显变化或生产缺陷场景	高频融入流水线 （如每日构建）	高频融入流水线 （如每日构建）	中频融入流水线 （如版本回归）
无明显变化或生产缺陷场景	中频融入流水线 （如版本回归）	中频融入流水线 （如版本回归）	低频融入流水线 （择机执行）

如前所述，以上策略在应用时还需要考虑前面提到的测试用例运行效率和稳定性。对于越需要高频融入流水线的用例，越需要花费时间保障其中执行效率和稳定性。需要注意的是，测试范围的选择是一个持续的过程，团队需要在每个研发周期内进行相应的回顾，并判断当前测试范围的选择是否仍然适合。所以，这也给自动化测试工具提出了要求，需要能够非常方便地调整融入到流水线上的测试范围。

3.2.4.2 推动 DevSecOps 的落地实践

顾名思义，DevSecOps 是 DevOps 概念的延续，DevSecOps 强调组织中的每个部门都同样负责在软件开发周期中每个阶段集成的安全性。DevSecOps 在兼容 DevOps 优势的同时，它可以最大限度地减少任何产品的漏洞，并使其完全准备好供最终用户使用。因为 DevSecOps 每个流程和相关工作流程都通过严格的安全检查实现自动化，因此可以更准确地满足安全要求。从 Gartner 发布的《2022 年软件工程技术成熟度曲线》（Hype Cycle for Software Engieering, 2022）中也可以看出 DevSecOps 位于第五阶段：生产成熟期。DevSecOps 已经成为主流采用的可以落地实践的方法和手段，落地时需要重点关注以下几点：

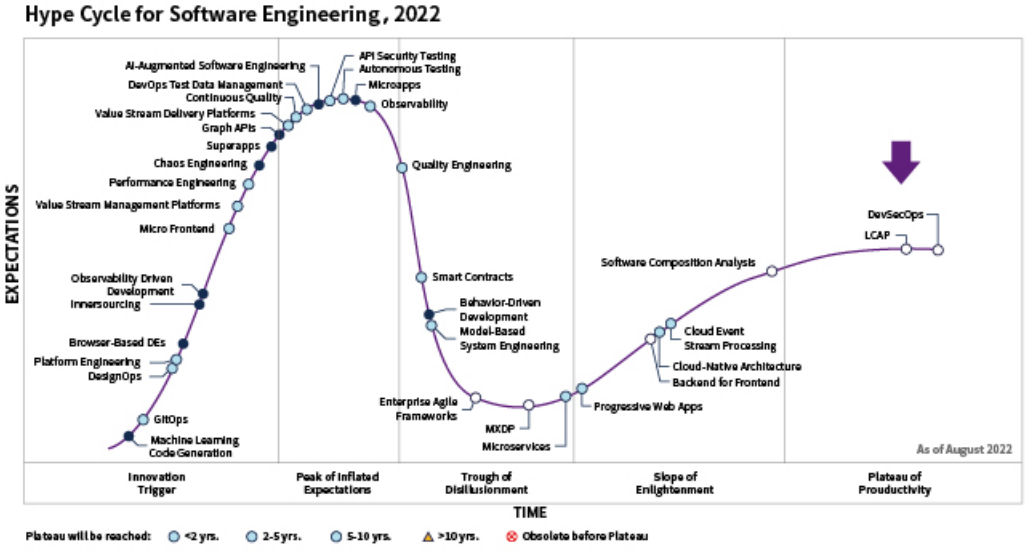


图6 Hype Cycle for Software Engieering, 2022

1. 安全左移：将安全程序（代码审查、分析、测试等）移动到软件开发生命周期（SDLC）早期阶段，从而防止缺陷产生和尽早找出漏洞。通过在早期阶段修复问题，防止其演变为需花费巨资加以修复的灾难性漏洞，安全左移可达到节省时间和金钱的目的；

2. 安全工具：集成安全工具是 DevSecOps 的基础之一。企业可以将安全工具集成到 DevOps 管道中，来确保安全问题能在整个开发生命周期中持续得到处理。包含但不局限于静态分析安全测试（SAST）工具、动态应用安全测试（DAST）工具、主机安全扫描、容器安全扫描等；

3. 注重团队安全意识培养：每一个实施 DevOps 框架的组织都应该转向 DevSecOps 思维模式，并且提升所有参与产品需求开发、技术开发、测试、运维等人员的安全意识和安全能力。

3.2.4.3 积极尝试探索式测试

以上描述的各种持续测试技术架构对于判定规范明确的测试非常有效。但是，在实际测试实践中，测试团队仍然需要进行一些非明确规范的测试工作。业内把这种无明确判定规范，但体现设计、执行和学习同时进行的测试工作统称为探索式测试。由于其无明确判定规范，探索式测试一般无法有效自动化，需要大量的手工工作。尽管持续测试非常强调自动化测试，并以此为基础融入到企业软件持续交付流水线中，但这并不意味持续测试就否定了探索式测试的必要性。相反，持续测试仍然认可探索式测试是整个敏捷测试中的重要一环。其主要原因在于：

▶ **探索式测试可以帮助测试团队发现很多其他测试方法发现不了的问题。**软件系统的质量不仅仅只有“是否按预期运行”这一个评价标准，系统是否“好用”很多时候是另一个关键的评价标准。探索式测试可以帮助团队发现系统的设计逻辑是否合理，有无让用户困惑，某个功能和其他功能是否协调等诸多这样的问题；

▶ **探索式测试是一个增强整个软件研发团队对于软件产品质量理解的有效手段。**在实践过程中，测试团队经常召集包括开发、运维、设计、产品等多个不同角色参与产品的探索式测试（例如“bug bash”活动）。这种测试是一个非常好的契机，让软件生产过程中各个不同角色的人重新理解产品质量，并进行跨角色的沟通，有效地发现潜在缺陷和现有测试的局限性；

▶ **探索式测试还是对新功能进行常规自动化测试前的有效测试手段。**不难想象，自动化测试对于回归测试非常有效。但当新功能开发完成后，进行以主流程为目标的局部探索式测试可以非常快速地发现常见缺陷，并快速修复。在实践过程中，这种测试很多时候是由测试团队提供用例和工具，开发团队在完成开发后快速执行的。这种场景下，探索式测试表现出来的执行效率和灵活性通常会非常好。

正因为以上多种原因，探索式测试仍然应该是持续测试中非常重要的组成部分。但是，由于以手工执行方式为主、测试质量高度依赖参与人员的特征，团队在执行探索式测试需要注意以下几个方面：

▶ **严格明确探索式测试的范围。**避免将很多可以明确自动化的功能扩散到探索式测试范畴，从而降低了整个测试的效率和质量。在探索式测试过程发现了部分有明确功能规范的缺陷，应该更新到自动化测试用例进行持续的自动化回归验证；

▶ **探索式测试设计和安排要确保“探索式”特征。**在探索式测试过程（尤其是如“bug bash”这样的活动）中，测试团队给出的测试指引应比较宽泛性，要避免给出那种明确测试路径和预期结果的指引；

▶ **要寻找更广泛的人群参与探索式测试。**除了上面提到的软件研发团队中的多个角色，还可以寻找公司内其他团队，乃至部分种子客户参与进行测试。越广泛的测试群体意味着更多的测试视角和更广泛的测试覆盖度；

▶ **对探索式测试进行认真进行总结和回顾。**测试团队需要牵头完成对每次探索式测试的总结，形成明确的测试报告，并召集参与人员对测试执行情况回顾分析，形成可执行的改进意见。

3.3 持续测试落地的技术平台

任何实践的落地都少不了工具或者平台的支撑，持续测试也不例外。但是建设持续测试的工具或者平台并非易事。持续测试工具和平台的选择不仅需要评估支持的测试类型和功能，还需要评估学习成本、技术发展、与 CI/CD 生态系统的集成能力等。

目前持续测试的支撑工具或者软件分为商业软件和开源软件两种。企业如果选择商业的持续测试软件需要考察供应商的能力，如果是开源软件需要重点关注开源社区和项目的活跃度。市场上满足持续测试落地能力要求的平台大体分为两大类：工具能力集成平台和一站式服务平台。

▶ **工具能力集成平台：**此类平台往往是企业自研为主，集合企业自身测试和工具使用进行研发，重点是整合现有团队使用的测试工具和碎片化管理的测试用例。在使用上，测试人员重点还是需要精通不同测试工具和框架的使用方式；

▶ **一站式服务平台：**一站式服务平台和工具能力集成平台最大的区别为，一站式对底层工具进行了封装和屏蔽，使用人员无需过多关注底层工具如何使用，只需要重点熟悉平台服务化界面和使用，因为平台提供了一致性的操作和体验。在一致的体验之上再提供一致的测试服务，比如统一的 API、标准的 DevOps 流水线集成方案，以及统一的数据管理仓库等。

3.3.1 工具能力集成平台

工具能力集成平台的前提条件为测试工具化。测试工具化不同于测试工具，测试工具化需要企业在开展具体测试时候，具体的测试内容可以由测试工具自动开展，比如很多企业基于 Jenkins + JMeter 集成方案落地，实现了发版后的接口自动化测试。而工具能力集成平台的平台特点在于不会改变现有测试人员的工作习惯，将不同的测试工具化的能力进行集成，最终形成如下图的通用架构能力。工具能力集成平台最大的优点是解决了测试工具碎片化问题，对测试脚本和用例进行了统一的管理；缺点也很明显，一般只有测试管理界面，没有提供测试脚本编排、编写能力，对使用的测试人员自身能力要求较高，并且在平台的维护和使用上，需要团队投入更多的精力。

图7 工具能力集成平台通用架构

如上图工具能力集成平台通用架构所示，为了企业能够更好地落地和推广持续测试，工具能力集成平台需要考虑几个层面的能力：

▶管理层：具备用例管理能力，包含不同类型测试用例脚本管理能力，比如 JMeter 脚本、Python 脚本等；测试文件管理，比如 CSV 文件，图片管理等；

▶调度层：能够将用户执行的测试任务，基于数据层组装成不同的可执行测试脚本，并且按照测试类型分配到不同的测试工具上，更进一步可以基于运行层的资源状态，自动调度最优资源进行测试；

▶工具层：包含不同类型的测试工具，提供不同类型的测试能力，常见的为 JMeter、Selenium、Python 环境等，工具层是测试用例运行的载体；

▶运行层：不同的测试工具部署的环境，包含虚拟机、容器或者物理机等。

3.3.2 一站式服务平台

一站式服务平台和工具能力集成平台最大的区别在于，一站式服务平台在使用上屏蔽了底层工具的差异，提供了一致性的用例编写和设计界面。并且在测试能力上提供测试服务化能力和更强的持续集成能力，一切测试皆服务。下图为一站式服务平台通用架构图：

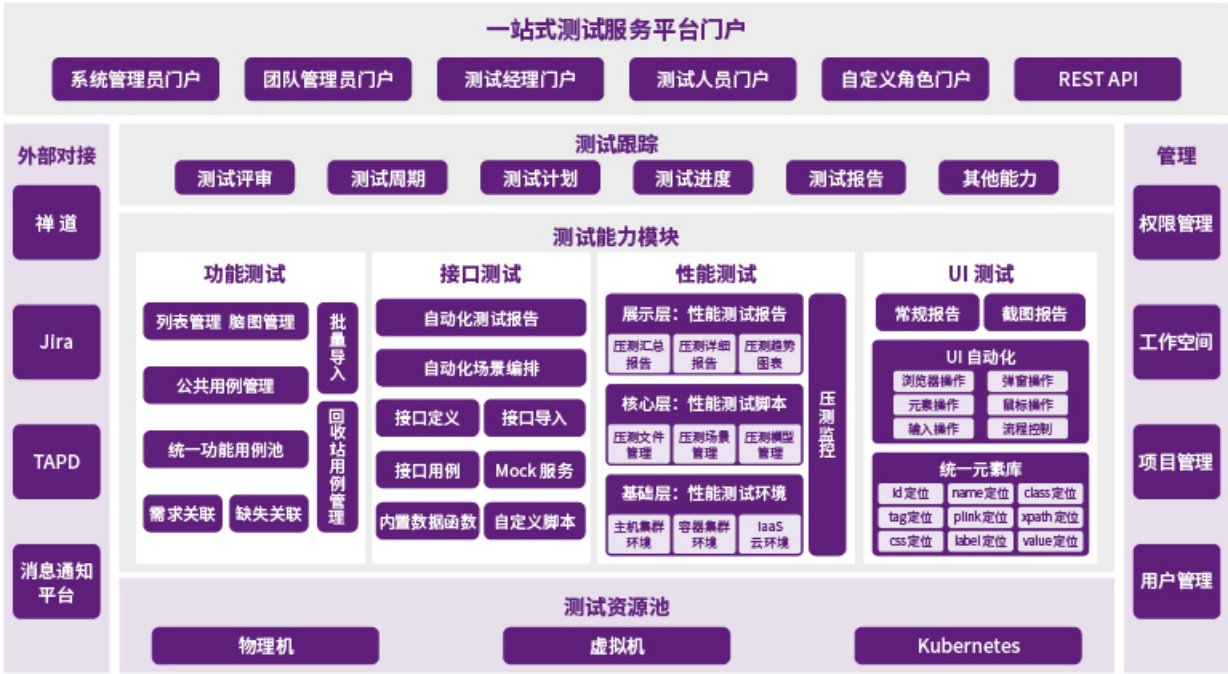


图8 一站式服务平台通用架构

从持续能力来看，非常关键的一点在于一站式服务平台架构如何满足企业和用户的需求设计。具体来说主要包括以下几个方面：

▶统一的用户 / 租户体系：一站式服务平台虽然能够提供多种的测试能力，但是需要将这些能力提供并展示给最终的用户，所以建立一个统一的逻辑用户 / 租户体系，并和企业现有的组织管理架构进行映射是不可或缺的关键能力之一；

▶完整的权限管理体系：测试平台的使用者不仅仅是测试人员，同时也有开发人员、运维人员等。测试平台需要具备权限管理体系，实现人员、角色和功能权限的解耦，从而可以让企业非常方便地建立一套符合自己企业内部实际情况的测试权限管理体系，让不同的角色人员在测试平台中依据其职责边界开展工作；

▶完备的 API 访问接口：大部分企业建设的 DevOps 平台或者 CI/CD 工具链，都已经提供相关的 API 访问接口。一站式服务平台在这方面的要求也必不可少，它必然要和其上层或者同级服务进行数据交互；

▶测试工具的兼容能力：一站式测试在具备不同的测试能力之外，还需要重视企业存量测试资产数据。企业测试经过了很长时间的累积，在不同的工具上累积了大量的测试用户或者测试数据，一站式服务平台需要提供用户一键导入现有测试工具的数据至平台中的能力，比如导入 PostMan 接口用例、JMeter 接口测试、Swagger 接口等脚本、Selenium IDE 自动化脚本等，最终实现统一的管理；

▶灵活的协议插件体系：虽然软件测试包含了常规的 HTTP 协议测试，但是不同企业、不同行业、不同的场景，对于测试的协议有着不同的区别。一站式测试平台需要能够将测试协议的支持与平台进行解耦，通过协议插件的方式，灵活地支持用户通过自定义开发协议插件方式在平台中开展其所需协议的自动化测试和脚本编写能力；

▶大规模横向扩展能力：不同的企业测试团队人员规模从几人至几十、几百人都有。一站式服务平台支持测试资源的动态扩容，所有的测试任务，无论接口测试、性能测试还是 UI 自动化测试都可以采用资源池的方式进行调度与运行，可以同时满足几人到几百人团队的高频测试需求。

当前市面上定位于持续测试的一站式服务平台的项目和产品在不断涌现，其中最具代表性的产品有来自国外的 Tricentis Tosca 和来自国内的一站式开源持续测试平台 MeterSphere。

表9 两款具有代表性的持续测试平台对比

	Tricentis Tosca	MeterSphere
自动化能力	支持基于模型的 API 和 UI 测试 支持服务虚拟化能力 支持自动化性能测试能力 支持测试数据管理能力	支持类似 Postman 体验的接口测试 支持接口管理功能 支持兼容 JMeter 的性能测试能力 支持 Web 端 UI 测试 支持测试数据管理功能
测试管理能力	支持测试用例管理、测试计划规划、测试报告生成 支持基于风险的测试需求分析与测试用例设计	支持测试用例管理、测试计划规划、测试环境管理、测试资源管理、测试报告生成
团队协同	支持多租户体系 支持基于角色的权限管理体系	支持多租户体系 支持基于角色的权限管理体系
外部系统集成能力	支持主流持续集成工具对接 支持主流缺陷管理工具对接	支持主流持续集成工具对接 支持主流缺陷管理工具对接
产品分发方式	SaaS 分发为主	开源 + 企业版软件订阅 + SaaS
产品支持方式	线上支持	社区支持、线上支持、线下专业服务

从表 9 可以看出，Tricentis Tosca 平台和 MeterSphere 平台在功能完整性上相差不多。而在产品分发和支持方面，由于 MeterSphere 来自国内团队，且采用开源方式运作，有着明显的本土优势和社区化优势。

3.4 积极拥抱新技术

和其他任何领域类似，测试领域也在不断涌现出新的技术。持续测试团队需要对此保持必要的敏感度和积极拥抱的态度。目前，在测试领域有以下几个方面的新技术值得测试团队高度关注：

▶ **利用线上历史流量指导测试工作。**在测试领域，如何真实模拟用户的线上行为永远是一个关键的问题，而线上历史流量则是最接近这个模拟需求的。这个领域涉及非常多的技术环节，包括流量的抓取、存储和回放等，有非常多可以研究和探讨的地方。最近各个测试领域的技术峰会可以看出这个领域保持着极高的热度，这也是大型互联网公司测试团队重点尝试的方向。建议测试团队对此保持高度关注，有可能的情况下可以在局部项目进行尝试，并积累技术能力和操作经验；

▶ **应用人工智能（AI）技术到测试领域。**人工智能是一个存在已久的话题，尤其是进入 2006 年后，随着深度学习（Deep Learning）技术的突破，机器已经在某些特有领域具备了类似（甚至超过人类）的技能。例如，最近 GPT-4 等人工智能（AI）语言大模型的出现，GPT-4 是一种基于 RLHF（人类反馈强化学习）、多模态的语言大模型，开始具有 AIGC 和通用人工智能（AGI）能力。GPT-4 能够执行一系列复杂的任务，如代码生成、错误检测、软件设计、自动生成测试脚本等。基于此方面，持续测试可以利用 GPT-4 等技术，构建虚拟测试机器人协作测试团队工作。另外，精准测试也是从人工智能受益比较大的领域之一，结合人工智能、代码静态分析和存量用例库进行综合分析，可以用于自动推荐新的可能测试用例。总体来说，这个领域值得关注，但是离实践落地还有一定距离，建议测试团队保持关注，及时了解行业进展。

当然，测试领域还有很多其他新技术的发展，这里不再一一列举。作为测试领域的从业人员，保持开放态度，并能积极思考新技术和实际工作之间的关联性是一个好的思维实践。

4. 持续测试成熟度能力模型

持续测试作为一项复杂且长期的企业实践，合适的度量模型是执行过程中的重要一环，一个合适的度量模型能够对持续测试的落地起到积极推动的作用。所以根据以上章节的阐述和行业内的实践经验，本章构建了以下持续测试成熟度能力模型。期待这个模型框架可以帮助企业更好地定位自身当前状态，并能建立相对清晰的努力方向和路径。

4.1 成熟度模型级别说明

从持续落地的深入程度进行不同模型基本的定义，分为 Level 1-4 四个级别，分别对应原始使用尝试阶段、初步尝试阶段、高级实践阶段和专家深入阶段。具体不同模型成熟度级别详细地定义如下：

▶ **Level 1：团队处于持续测试实践的原始阶段。**用例管理分散且用例设计质量高度依赖测试团队个人的能力，测试执行以手动测试为主，无任何高级的持续测试实践被常规化采纳；

▶ **Level 2：团队处于持续测试的初步尝试阶段。**开始积极引入各种自动化测试能力，融入到持续测试实践体系。但是整个努力仍然在局部和初级阶段，还未形成稳定的常规实践；

▶ **Level 3：团队处于持续测试的高级实践阶段。**已经在各个方面形成较为稳定的持续测试实践落地，并且在局部已经达到较高的实践能力。团队日常工作高度依赖这些持续测试实践运营；

▶ **Level 4：团队处于持续测试的专家深入阶段。**团队已经熟练掌握持续测试各个方面的关键实践，团队日常运营效率高，对软件测试的其他环节还能形成了非常有力的支持。与此同时，团队积极探索超出现有持续测试实践的其他方面工作。

4.2 成熟度模型的构成

本着紧贴用户实践的原则，本成熟度模型选择从具体的持续测试实践维度来定义。具体的维度包括测试用例设计与管理、测试用例执行、测试环境管理、交付流水线集成和高级测试实践采纳。整个成熟度模型定义如表 10 所示：

表10 持续测试成熟度模型的级别定义

测试维度		Level 1	Level 2	Level 3	Level 4
测试用例设计与管理	测试用例管理模式	无	低	中	高
	测试用例设计	无	低	中	高
测试用例执行	接口测试与集成测试自动化	无	中	高	高
	UI 自动化测试	无	低	中	高
	基于脚本的手工测试	高	中	低	低
测试环境管理	测试数据管理	无	低	中	高
	服务虚拟化	无	低	中	高
	测试环境能力	低	低	中	高
交付流水线集成	CI/CD 流水线集成	无	低	中	高
高级测试实践采纳	探索式测试	无	低	中	高
	非功能性测试	低	低	中	高

上表中各个维度的“无、低、中、高”评判标准的描述如表 11 所示。

表11 持续测试成熟度模型分维度评价标准

评定维度\评定级别	无	低
测试用例管理模式	用例管理以线下个人管理为主，通过简单的文件方式进行共享和协作。	建立了用例线下用例协作方式，有较为规范的线下用例评审机制。
测试用例设计	测试用例设计无团队层面统一指导思路，以个人设计为主。	初步具备测试用例设计的逻辑覆盖度思维，用例设计会进行日常评审。
接口测试与集成测试自动化	无常规性接口与集成测试自动化实践。	初步引入脚本或者工具进行接口与集成自动化测试，但以单接口为主，且接口与集成测试自动化占比较低（低于 50%）。
UI 测试自动化	无 UI 自动化测试，以手工测试为主。	引入脚本或者相关工具进行局部的 UI 测试自动化，并未变成测试团队常规测试内容。
基于脚本的手工测试	无基于脚本的手动测试。	在局部有相应基于脚本的手动测试。
测试数据管理	测试数据以个人管理为主，生成机制随机，质量无法保障。	通过文件或者数据库形成初步的测试数据集中管理，但是无合理的版本管理机制，无数据生成和质量保障机制。
服务虚拟化	未建设任何服务虚拟化。	通过脚本或者工具构建了基本的服务虚拟化，并在测试自动化中使用。
测试环境能力	测试环境不完备，部署实践长，无有效的一致性管理能力。	初步具备分工明确的不同测试环境，部署过程仍以手工或者脚本为主，环境一致性管理问题仍然比较严重。
CI/CD 流水线集成	未和 CI/CD 流水线形成任何形式的集成。	和 CI/CD 流水线形成初步集成，可以在流水线中触发自动化测试。
探索式测试	无任何特意组织和规划的探索式测试。	在软件研发局部阶段有嵌入必要的探索式测试，例如新功能开发后的快速功能测试。
非功能性测试	无任何有意组织和设计的压力测试、安全测试和兼容性测试等非功能性测试工作。	通过脚本或相关工具对于系统局部进行包括压力测试、安全测试和兼容性测试工作，但未形成规律的非功能性测试。

中	高
用例管理以线上集中平台协作为主，用例评审可以线上展开，但用例管理与用例执行未形成关联和互动。	用例管理以线上集中平台协作为主，支持线上用例评审，用例管理与用例执行可联动。
有初步业务风险覆盖率意识，能对项目业务风险进行初步评判并以此指导用例设计。	熟练业务风险覆盖率评估思路，能够持续以此为指导优化测试用例设计。
逐步形成相对常规的接口与集成测试自动化实践，支持单接口和场景化接口与集成测试自动化。	接口测试自动化占比超过 50%，且保持持续的优化。接口自身的设计、实现和变更与测试自动化形成良好的互动。
UI 测试成为版本发布日常测试工作中的一部分，且以自动化执行为主。	UI 测试常规化运行，且以自动化为主，执行稳定性有较好的保障机制。
测试团队日常工作大幅度依赖基于脚本的手动测试。	测试团队几乎全部依靠基于脚本的手动测试，无常规自动化测试实践。
有集中的测试数据管理平台，平台数据进行合理的版本管理，有较为丰富的数据生成和生产数据脱敏机制，数据质量稳定。	测试数据集中管理，数据版本和质量可靠，测试数据与测试用例和自动化测试场景形成良好关联和互动。
有集中的服务虚拟化管理平台，能够支持常见的服务虚拟化需求，能在测试自动化中使用。	有集中服务虚拟化管理平台，支持丰富的服务虚拟化需求，服务虚拟化可以与测试数据、测试用例及自动化测试形成良好的关联和互动。
具备相对完整的不同测试环境，部署过程基本实现自动化，测试环境一致性能力基本形成。	有完备的测试环境管理机制且能高效一致地部署，不同测试环境的部署可以融入到不同测试需求中按需实时建设。
自动化测试已经嵌入到研发团队的每日构建或者提交构建，测试运行时间合理，且运行结果文档能得到研发团队认可。	无论是测试左移还是测试右移实践都已经嵌入到 CI/CD 流水线。在保障整个流水线平滑运行的同时，能够有很好地测试自动化运行质量，研发及运维团队高度依赖测试团队的质量反馈。
在软件研发各个阶段都有意设计相应的探索式测试环境，并有明确的测试计划和流程。	探索式测试成为软件研发各个阶段中的必要阶段，并且形成广泛的参与度和高效的测试反馈机制。
有完整的规划、设计和执行，测试自动化执行程度高。针对非功能性测试的结果能形成有效反馈和改进计划。	非功能性测试成为系统性能、容量、安全和兼容性反馈的主要手段。且为此建立了广泛的工具和平台，测试自动化、常规化运行，能够支持生产流量乃至生产环境的测试。

4.3 如何提升成熟度

对于不同企业和团队来说，都可以参照以上成熟模型来做自我评估。但是不同企业和团队在提升成熟度拓展的路径可能会因为实际情况不同而有所不同。这里推荐一种典型地提升成熟度的路径（以企业和团队当前处于 Level 1 为例）供大家参考。具体如图 9 所示：

图9 持续测试成熟度模型的提升路径

图 9 整体描述了一种持续测试程度模型的提升路径，每个阶段的工作内容以及评判标准。关于每一步骤的详细描述如下：

1. 发展起步：重点提升测试执行的自动化程度。尤其是要将自动化测试的主要努力转向接口自动化领域。这一提升过程可以选择部分项目或者部分团队开始尝试。但是，这个尝试不同于之前的大部分自动化测试尝试，它一定要求自动化测试用例占比到一个较高的比例，让选择的项目或团队能形成真正依赖自动化测试的运营习惯。在达成这个运营方式过程中会让企业对自动化测试有更加深入的理解，各种原来未曾遇到的问题和挑战会逐步暴露出来，并最终形成合适的解决方案。与此同时，团队可以初步尝试将部分测试活动合适地嵌入到持续交付流水线中，并在局部尝试测试环境管理的改善，以及采纳高级测试实践；

2. 关键提升：重点在于落实测试用例设计与管理，以及改进自动化测试稳定性上。一方面，企业和团队需要回顾分析大量已经存在的测试用例，判断是不是有大量无效的测试用例在浪费维护和运行的时间。这种有效性分析可以从已经实施自动化测试的项目和团队开始，逐步扩散到其他项目。通过这个阶段的工作来形成测试团队对于有效测试用例的认知，并逐步构建出测试用例有效性的评审机制。测试团队还需要基于有效性分析的结果对存量测试用例进行改造，而这一过程也是企业实现测试用例自动化改造的契机。另一方面，企业和团队需要构建测试用例的线上管理和协作能力，让测试团队的日常工作主要在线上完成。与此同时，保障自动化测试稳定运行也是这一方面的工作关键点。尤其是在测试数据管理、服务虚拟化和测试环境部署保障上。这一阶段努力成功的重要标志是，自动化测试结果“误报率”大幅度下降，集成和回归测试通过率大幅度提升，且能够给产品提供有效的质量反馈；

3. 成熟落地：全面提升整个持续测试运行效率，建立测试及时反馈机制。将自动化测试集成到项目日常的 CI/CD 流水线中，并根据不同运行时间点选择合适的测试用例集去运行。这一阶段的工作让研发团队能够及时获得测试反馈，改进产品质量。同时，持续高频率执行的自动化测试结果可以让整个团队清晰地判断软件产品的业务风险走势，增强业务部门对于软件发布的信心。另外，建立稳定的高级测试实践运行体系也是这一阶段的重要职责。引入如探索式测试、压力测试、安全测试、兼容性测试等高级测试手段，将测试覆盖的风险从功能层面向更深入的层面推进。

CASE STUDY

案例研究

农银金科 DevOps 与持续测试应用实践

持续测试助力 DevOps 建设

农银金融科技有限责任公司（以下简称为农银金科），是农银集团全资子公司，成立于 2020 年。作为农银集团金融科技输出的重要窗口和平台，农银金科面向集团内外部客户提供信息化建设服务、场景金融生态建设、科技输出和创新技术研究等业务。

一、农银金科 DevOps 建设

为了打破研发、测试和运维人员之间的沟通壁垒，让公司的产品交互能够更加的顺畅和高效，农银金科内部以 DevOps 的标准来践行软件交付相关的工作，以实现开发运营一体化。其中，自动化测试是 DevOps 体系中的重要一环，这也成了农银金科团队引入和应用 MeterSphere 一站式开源持续测试平台的原生动力。

■ **DevOps 工作流与工具链** 依托 DevOps 平台，农银金科落地了端到端的 DevOps 工作流，建立了覆盖研发、测试、部署等全流程的工具链体系，全面实现工程管理的线上化、自动化。

■ **打造产品版本管理中心** 除了 DevOps 平台中大家比较熟悉的需求管理和流水线功能以外，值得一提的就是农银金科团队还依托 DevOps 平台打造了一个产品版本管理中心，建立了产品版本管理的机制，并将需求、测试计划、流水线、代码基线、制品、缺陷、测试报告、变更信息进行集中管理，建立起全链路质量追踪的通道，有效追踪研发过程度量数据的同时，融入了自动创建分支、自动收集质量数据、自动部署等多种自动化能力，在企业里逐步树立起了产品版本管理的意识。

二、MeterSphere 开源持续测试平台在农银金科的应用

1. 测试平台选型

农银金科在成立之初，为了提升研发及测试的效率，让研发过程更加规范，同时也考虑到需要去建设 DevOps 一体化的流程，就产生了引入一款自动化测试平台的构想，希望通过这个平台去承载包括测试跟踪、接口测试和性能测试等相关的工作。

在产品选型的过程中，农银金科主要考虑了以下几个方面的能力：

■ **DevOps 的支持能力：** 能够很好地融入到 CI/CD 的流程中；

■ **一站式的能力：** 测试工作所需使用的测试管理、性能测试、接口测试等测试工具相对较为分散，不利于统一的资产管理，资产的后期管理与维护成本高，因此希望有一个平台能够聚合多个工具，提供一站式的测试能力；

■ **具备可拓展性：** 能够提供 API 接口或者有良好的库表结构，便于组织进行测试能力延伸，帮助测试人员去进行数据分析等；

■ **易用性强：** 平台学习成本较低，测试人员不需要经过大量的培训，就可以快速投入使用。

基于以上这些考虑，农银金科进行大量的工具调研及选型工作，包括一些开源工具和商业工具，比如 JMeter 等，最终选择了 MeterSphere 开源持续测试平台，并且成功地引入和全面应用到了农银金科研发管理体系中。

2. 一站式测试管理与执行平台

引入 MeterSphere 平台以后，农银金科将 MeterSphere 平台定位为公司的一站式的测试管理和执行平台。

■ **平台承载农银金科测试过程中所有测试资产的管理工作，以项目维度去进行统一的管理。**

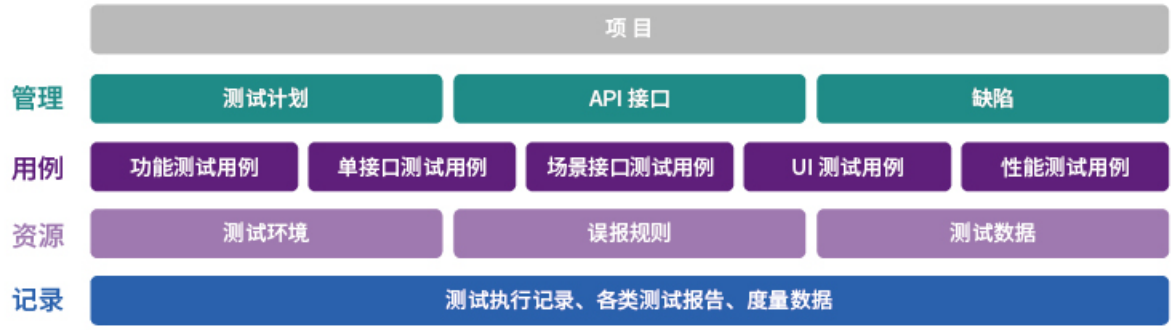


图10 项目维度统一管理

测试人员可以根据发版计划创建版本测试计划，并关联相关的测试用例，再基于测试结果一键创建缺陷，实现需求 / 用例 / 缺陷之间的双向可追溯。

■ **全面落地测试左移，团队的工作协同度提升。**

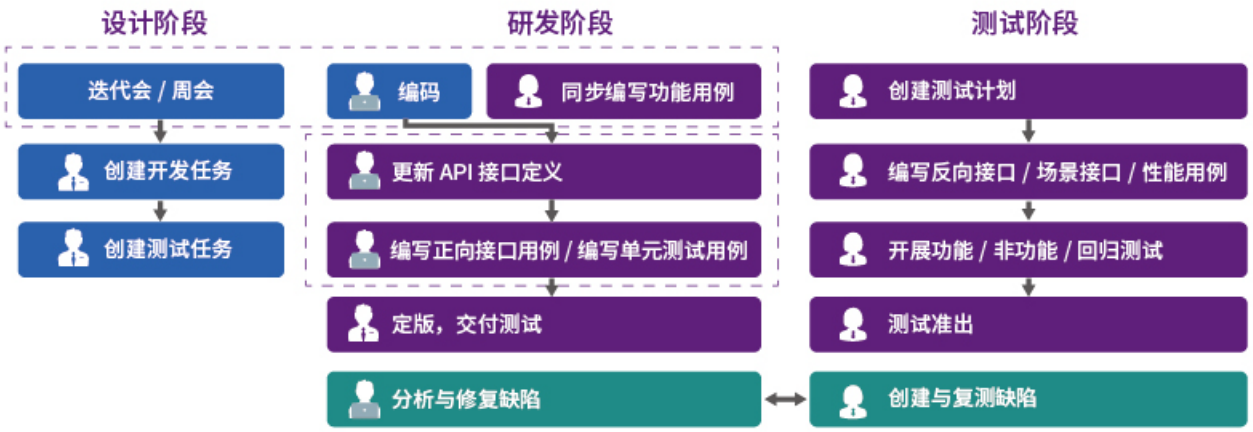


图11 测试左移落地过程

项目开始实施时，研发、测试人员通过迭代会尽早达成需求一致，测试人员提前开展功能测试用例的编写工作，开发人员负责更新 API 定义并充分参与到接口测试工作中，接口的更新实时触达测试人员。通过这样的形式，测试人员能够尽早地启动测试，同时开发人员也参与到了部分测试工作中。

■ **MeterSphere 深度结合 DevOps 平台流水线与产品版本管理，实现了自动化调起和自动化收集结果等功能。**

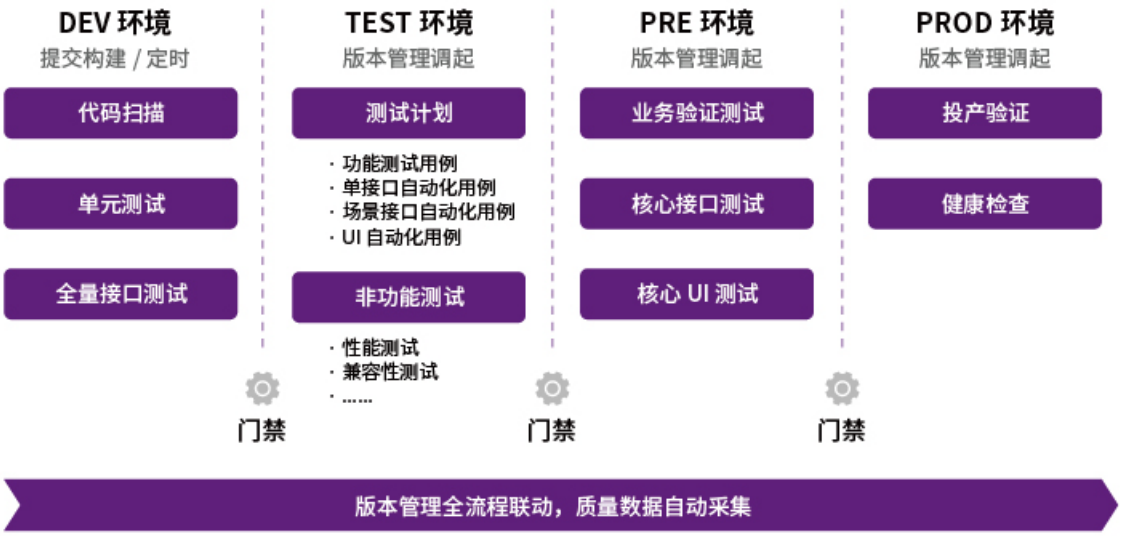


图12 自动化调起与自动化收集结果

可以看到在上图中对应的四套环境里面，各个项目会按需执行不同的测试内容。比如在 DEV 环境用户可以通过流水线，自动触发全量的接口自动化测试；在 TEST 环境，会由测试经理去触发整个测试计划的自动执行；进入到 PRE 环境和 PROD 环境，测试人员可以选取关键、核心的用例进行二次的验证。在每个环境之间，通过质量门禁来卡控测试执行的结果，以确保质量的可控性。全流程通过产品版本管理来实现联动，并实现质量数据的自动采集。

3. MeterSphere 平台各模块应用情况

■ **测试计划管理** 在测试计划管理模块，团队跟随应用版本计划来创建测试计划，并且在农银金科的 DevOps 平台进行关联绑定，集中管理版本测试资产，直观掌握测试进度。

■ **功能用例管理** 测试人员按需采用表格 / 脑图 / 在线编辑等方式，快速完成功能用例的编写，项目团队可以在线评审用例。各个项目功能用例按模块分类管理，对应关联到 DevOps 平台上的需求，建立起了需求、用例、缺陷这三个测试关键要素的双向追溯关系。

■ **接口用例管理** 在接口用例管理部分，是由开发人员去维护接口列表，测试人员进行接口用例的更新，实现了高效信息共享与工作协同。目前，农银金科内部可以保证所有项目的接口在 MeterSphere 平台得到 100% 的覆盖。

■ **UI 测试用例** UI 测试部分，农银金科在 MeterSphere 平台的元素库中将各个项目按照功能模块进行了拆分，集中管理便于维护。因为平台已对脚本类操作做好了封装，测试人员可以轻松完成 UI 测试用例的编写。调试过程清晰可见，测试结果支持生成截图与报告。MeterSphere 的 UI 测试功能具有低学习成本、低设计成本、低维护成本和高执行效率的特点，可以有效补充到团队在接口测试方面没有触达的测试点。

■ **性能测试** 在性能测试方面，MeterSphere 平台可以基于接口用例直接一键生成性能用例。测试人员无需重复编写脚本，只需要做一些参数的配置，比如说并发数、压测时间等，就可以去调度压测集群执行相关任务了。同时 MeterSphere 平台还支持实时查阅测试报告。

■ **DevOps 流水线** 农银金科内部是通过开发 MeterSphere 插件的形式来实现任务调度的。最终的实现效果是流水线可以按需灵活调起测试计划、单接口、接口场景、UI 场景等各种类型的自动化测试，实时输出测试结果，与持续集成联动实现部署即测试。

三、MeterSphere 平台落地效果

■ 企业级的测试平台，助力落地持续测试

农银金科运用 MeterSphere 平台，在内部建立起了测试管理体系，形成了与 DevOps 平台全面集成、覆盖全测试类型的统一测试管理平台。平台全面承载公司测试计划、接口定义、测试用例等关键资产的集中管理与自动化执行工作，实现了测试工作的集中化、规范化、自动化，全面提升了公司的测试效率。通过产品版本管理与流水线自动化测试相结合，有效落地了持续测试。推动实现测试工作的左移，测试资产的更新维护更加及时，开发人员和测试人员的协同更加高效，研发过程质量自检更加充分。

■ 通过 DevOps 认证评级

2022 年 7 月，农银金科顺利通过了信通院 DevOps 能力成熟度评估认证，取得了 DevOps 能力成熟度持续交付三级、系统与工具（流水线部分）优秀级双项证书，成为业内首个同时获得工具和持续交付两个标准评估的银行系金融科技公司。

■ 产品的质量数据可追溯

MeterSphere 平台协助农银金科团队以产品版本管理为主线，建立起需求、测试计划、测试用例、缺陷与需求、代码版本、构建、发布等要素之间的关联关系，实现了数据全流程双向可追溯，结合测试数据看板，帮助团队逐步形成全流程质量内建的意识。

在质量数据可视化方面，团队还使用了 FIT2CLOUD 飞致云旗下的开源数据可视化分析工具 DataEase，搭建了农银金科的测试质量大屏，帮助团队直观地了解当前测试平台的应用情况，也给上层领导提供了更多的决策依据。

截至目前，MeterSphere 平台在农银金科已经承载超过 70 个项目的用例的管理，在平台创建了超过 11 万个功能用例、15000 多个接口用例，及 25000 多个接口和场景用例，在 UI 测试模块的用例数量也接近 1000 个。

图13 农银金科测试质量大屏

四、未来展望

在深度使用 MeterSphere 开源持续测试平台的同时，农银金科还希望这个平台能够在未来不断成长，尤其是在持续测试“一站式”的能力方面，希望能够拓展新的功能疆界。

■ **进一步提升与 DevOps 平台的集成能力** 希望 MeterSphere 平台能持续完善与 DevOps 平台对接的相关能力，提供更加顺畅的对接体验。例如，让缺陷管理更加方便、信息对接更加实时、用户体验更加丝滑等，探索更多的应用可能；

■ **扩展 UI 测试能力** 在 UI 测试方面，希望能够持续丰富操作场景，提供 MeterSphere 专属的界面录制工具，UI 测试支持资源池运行，UI 执行场景支持实时视频观看等，持续提高 UI 测试的执行效率和测试稳定性；

■ **扩展数据统计、数据报表等能力** 基于 MeterSphere 平台打造更加多维、服务不同层级用户的测试数据度量视图，帮助企业高管、项目经理、测试人员等角色从不同角度进行测试进度和测试质量的跟踪分析；

■ **持续延伸测试平台能力域** 希望 MeterSphere 的研发团队探索移动端兼容性测试、安全测试等融入平台的可能性，打造一个真正的集成全测试类型的统一平台，让用户能够聚焦在一个平台完成所有测试工作。另外，还希望尝试应用精准测试、AI 技术融入等技术方向，让测试更加精准、自动和智能。

CASE STUDY

案例研究

致远互联基于 MeterSphere 构建敏捷测试平台与 DevOps 体系

DevOps 集成、一站式测试

北京致远互联软件股份有限公司（以下简称为“致远互联”）成立于 2002 年，总部设立在北京，是一家集产品的设计、研发、销售及服务为一体的高新技术企业，为客户提供专业的协同管理软件产品、平台、解决方案及云服务，是中国协同管理软件领域的开创者和持续引领者。致远互联主要为政企客户提供产品及解决方案，并不断向大协同市场领域拓展。公司坚持平台化产品的发展路线，基于自主研发的 V5 协同管理平台，开发了面向中小企业组织的 A6 产品，面向中大型企业和集团型企业组织的 A8 产品，以及面向政府组织及事业单位的 G6 产品。

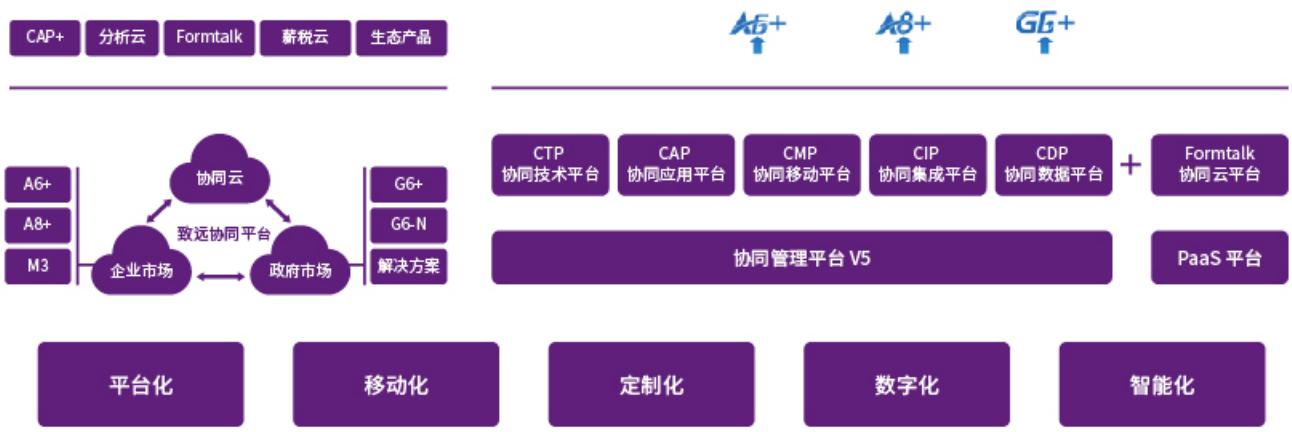


图14 致远互联产品家族

一、致远互联的业务背景

致远互联旗下的软件产品采用微服务架构设计，产品业务复杂度较高，整体由 20 个微服务构成。主打产品协同运营平台（即 Collaborative Operation Platform，简称为 COP）是面向数字化组织的企业级服务新品类，能够覆盖企业办公常见业务终端，统一接入企业业务流程建设体系。

图15 企业数字化业务框架

由于业务复杂度高，且产品的业务生态体系庞大，致远互联的测试团队不断探索，希望能够搭建具有控制产品质量、测试流程、考核指标、持续集成等能力的一体化测试平台。

二、使用 MeterSphere 平台的契机

致远互联的测试团队规模为 50 人，分不同测试小组进行业务、接口、性能、UI、安全等测试工作。出于大规模团队测试协作的要求，且从自身产品理念——“协同办公”出发，在搭建测试平台过程中，测试团队经理找到了 MeterSphere 一站式开源持续测试平台。希望能够基于 MeterSphere 构建起协同研发体系工作的测试平台，并且打通产研 DevOps 流程，满足团队管理与降本增效的需求。致远互联选择 MeterSphere 开源持续测试平台的理由如下：

■ **具备团队协作能力：** MeterSphere 与致远互联自身的协同办公、数字化组织建设产品理念不谋而合。测试过程的数据、测试脚本、测试报告可按照业务组进行整合，方便对团队考核做数据支撑；

■ **使用体验佳：** MeterSphere 的界面简洁易用，具备接口管理、接口用例管理、自动化测试的编排能力，支持自动输出测试报告的能力。同时，测试同学上手度高，大幅缩短了自动化测试时长；

■ **DevOps 集成能力：** 团队对代码质量要求高，代码构建需要提前做 P0 场景冒烟测试及自动构建日常自动化测试任务。MeterSphere 平台具有构建完整的流水线任务的能力。通过对 MeterSphere 的调研、落地使用，并且对 DevOps 流水线集成系统做了适当改动之后，致远互联构建了自己的“敏捷测试平台与 DevOps 建设系统”。

三、敏捷测试平台与 DevOps 体系的建设方法

基于致远互联自研的 DevOps 平台与 MeterSphere 集成的设计思路如下：

1. 基于 MeterSphere 的 OpenAPI 接口，通过测试平台微服务统一触发冒烟测试和接口功能测试自动化任务。通过流水线构建 Dev 环境，触发对应微服务接口场景执行，异步返回执行结果到流水线服务。如果通过，则控制推送镜像至 Test 环境，并且通过配置文件控制执行并发数、执行环境，最终将集成执行过程数据收集，并生成测试报告和触发消息；

2. 测试平台端创建初始化、冒烟、功能测试、还原等目录，分别在对应目录下实现对应接口场景：流水线构建 Dev 环境，触发对应微服务接口场景执行（可配置执行目录），异步返回执行结果到流水线服务。如果通过则控制推送镜像至 Test 环境。CI 构建时，测试同学只需在对应目录下新增或移出场景，即可同步 CI 构建时的执行场景。

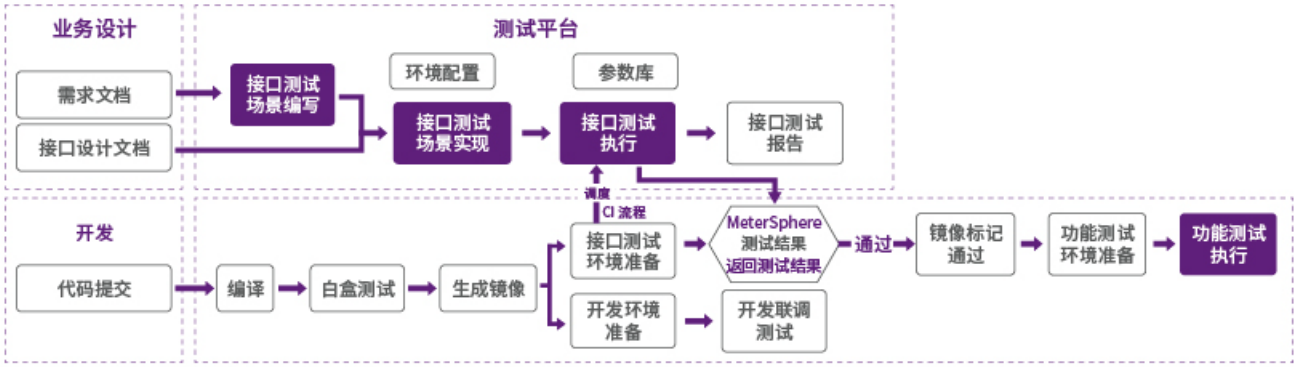


图16 自研 DevOps 平台与 MeterSphere 集成设计思路

统一调度 CI/CD 流水线集成过程如下：

① 开发代码提交到阿里云上做代码集成编译并生成对应的镜像后，通过 MeterSphere 平台拉起接口测试环境，触发任务调用 MeterSphere 冒烟接口自动化任务搭建，快速完成 P0 场景冒烟测试。

■ 若 P0 冒烟测试通过，则标记镜像成功并构建提测到测试环境，通知测试人员进行最新的功能测试；

■ 若 P0 冒烟测试未通过，则构建任务回滚，由开发人员重新修改自测再提测。保证测试环境功能稳定及代码提测质量。

② 在测试团队端划分多个测试小组，当产品给出需求文档后，测试人员将统一完成基于具体功能的接口测试场景。在 MeterSphere 平台按测试目录实现脚本编写，再由测试平台自动配置每晚跑相关功能的自动化测试任务。最后，测试报告会通过内部聊天工具“致信”机器人生成对应消息通知到指定人。

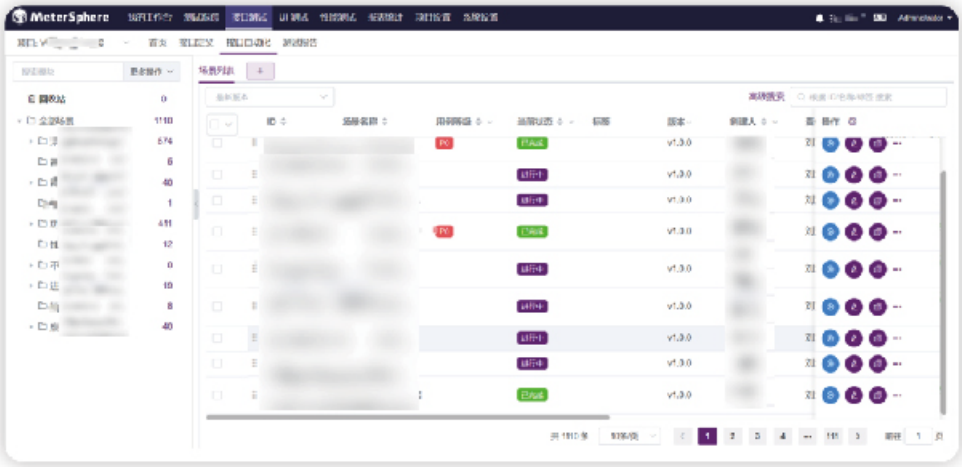


图17 接口测试场景

③ 所有执行过程数据由测试平台统计生成测试报告。每个微服务工程下自动化用例执行数量、通过率由统一收口展示，这样可以直观看到每个测试小组所负责模块的自动化覆盖情况，为后续产品发布及项目总结做数据支撑。同时，测试报告可以穿透到 MeterSphere 平台检查未通过的接口用例，进而再对测试脚本做进一步的调试修改。

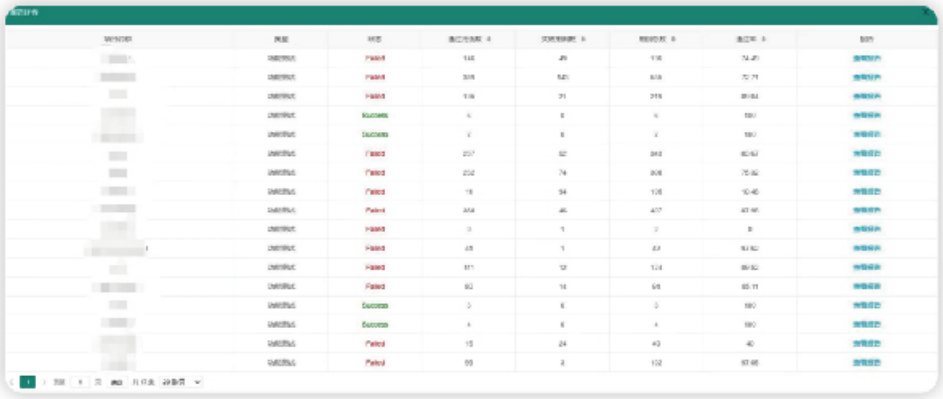


图18 统计生成测试报告

④ 利用平台优势，通过对研发代码构建回滚次数、测试脚本数量、测试执行过程通过率、接口覆盖率的统计，项目管理团队可以对研发质量和测试质量进行整体评估。对团队产出效能、投入产出的评比及后续岗位晋升都可以提供更有说服力和支撑力的数据支撑。

四、总结

致远互联基于 MeterSphere 构建的敏捷测试平台实现了以下成果：

■ 测试平台通过调度 OpenAPI 接口将场景执行方式由目录控制，使业务组使用更加灵活；

■ 过程数据收集的颗粒度更高，能够收集到更多关于测试管理过程的数据；

■ 任务调度更灵活。测试平台可以配置 Java 线程，通过调整接口自动化的执行线程，满足统一场景对于不同并发线程的需求，同时也便于集成多种流水线环境；

■ 由测试平台统一调度自动化任务，减少了人为干预造成的资源撞车，降低沟通成本，较以往的自动化构建效率明显提升；

■ 目前，致远互联已经在 MeterSphere 平台完成项目管理数 17 个。接口场景数量超过 2600 个，接入用户数量 50 人。

五、后续测试平台完善计划

■ **测试平台与精准测试的集成** 目前，致远互联的测试团队正在初步推动并搭建精准测试平台相关建设内容，后续和测试平台做数据打通与集成，实现测试构建后精准推送受影响的功能测试用例到相关测试人员处，并自动拉起构建接口自动化任务，快速完成测试任务；

■ **契约测试探索** 致远互联产品微服务架构体系庞大，为了确保微服务之间足够兼容及通信协议调度正常，后续将引入契约测试，实现问题的快速定位和测试前移。

CASE STUDY

案例研究

商米科技基于 MeterSphere 的全球化云服务接口测试实践

团队协作、一站式管理、测试提效

上海商米科技集团股份有限公司（以下简称为商米）是一家致力于为商用领域提供智能 IoT（Internet of Things，物联网）设备及相应配套的“端、云”一体化服务的物联网科技公司。从 2016 年推出第一款智能手持终端至今，商米已经推出移动、金融、台式、自助、网络 / 视频等丰富的商用 IoT 设备，覆盖多种业态场景。2020 年，商米正式对外发布了 BIoT（Business Internet of Things，商业物联网）战略，希望通过智能商用 IoT 设备、商用操作系统与 IoT 云管理平台所构成的商米产品及服务体系将数字化带入到每个商业场景中。与此同时，商米还向合作伙伴提供标准化应用开发服务与高可靠全球化设备远程管理与控制服务，赋能合作伙伴快速实现对设备及应用的商业化管理与运营。

一、背景介绍

自 2016 年开始生产智能设备至今，商米的设备已经从一代机演进到二代机，且正在进行三代机的开发。商米的产品影响范围遍及全球，拥有全球化的服务、全球化的技术支持和全球化的线下数据建设。在平台升级至 BIoT 战略之前，公司使用的早期 SUNMI 云架构如图 19 所示。早期 SUNMI 云的模式相对比较简单，即线下设备搭载定制于 Android 的 Sunmi OS（Sunmi Operating System，商米操作系统），再通过 Sunmi OS 搭载各种软件，商米则通过一套内部管理平台和一套合作伙伴的开放平台，形成了简单的联动生态。

图19 早期 SUNMI 云架构

后期升级至 BIoT 模式后，商米业务架构的复杂度加大。除了在基础的各种系列设备上搭载 Sunmi OS，以及搭载 OTA（Over The Air，空中下载）、远程协助、应用商店、商米收银台、流量服务、证件云解等各种设备软件以外，商米还为客户提供了两种开放平台以及两种解决方案。在商米的开放平台上，客户可以通过开放平台去看护、管控自己的设备，在自己的渠道上添加各种黑、白名单，并定向推送各种应用版本至终端的某一台设备上。商米可以为客户定制软硬件一体的解决方案，也可以直接提供一些通用的方案。

图20 商米业务架构现状

二、商米平台的基本现状

■ 目前商米拥有多个技术栈，线上 ECS（Elastic Compute Service，云服务器）的数量已经达到 500 多台，有近 20 个线上集群、近千个活跃服务、上百个域名（不包括通过 CDN 加速的域名），统一网关的每日调用量已达到 6 亿次；

■ 商米云平台对应多种线下环境；

■ 商米的设备需要保证网络安全并具有一定的记忆力，所以需要在网关添加各种加密、解密插件；

■ 商米的后端技术栈主要包括 Java、Golang 等；

■ 近年来，商米架构逐渐走向微服务化和云原生化，这使得平台架构更加复杂；

■ 业务接口量暴增至几千个，在这种情况下，如何保障接口与技术栈质量对商米技术团队来说就成了一项较大的挑战。

三、测试中的技术难点与技术选型

1. 技术难点

商米从 2020 年开始走向国外。为了满足各个国家对于数据安全的要求，商米已经在海外建立了多个数据中心。在设备跟随客户移动、客户可能移动至不同地域的情况下，为了满足每个数据中心将业务数据留存在当地的要求，商米打造了全球化的运维平台架构。商米有一套全球通用的大数据分析平台，对其所有数据中心进行统一的管控运营。但是由于商米在不同地域的数据中心使用不同的云，在中国数据中心使用的是阿里云，在海外的数据中心使用的则是谷歌云或亚马逊 AWS。要打通这些不同的云平台存在一定难度。

2. 技术选型

目前，商米云平台测试主要包括接口测试和 UI 测试。其中接口测试的接口按照属于研发域与测试域进行区分，又可以划分为外部接口和内部接口。

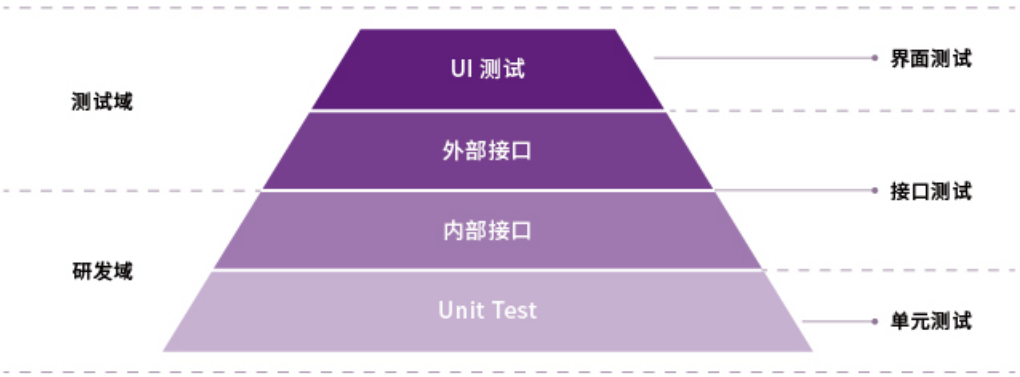


图21 商米云平台测试梯队

商米测试团队进行的主要是外部接口测试和 UI 测试。基于这一点，团队早期制定了接口测试的平台化方案，基于 JMeter 进行二次开发，通过 Jenkins 触发流水线，也自研了接口测试框架，但实际效果并不理想。这套平台化方案存在的问题包括：

① 早期业务的接口规范与所使用的技术栈不统一；

② 早期测试人员的个人能力有限，不同测试人员使用的测试工具难以通用；

③ 因为商米的业务往往有各种各样的加签规则，以及多种不同的接口，仅靠一款固定工具难以满足业务需求，需要团队持续投入，去定制开发并维护工具，成本过高；

④ 早期商米测试团队对前端的外部接口仅进行手工测试。后期团队进行基于代码方法的测试，也因为 UI 元素识别的难度和成本较高而暂时没有进行 UI 测试。

四、为什么选择 MeterSphere ?

对于测试工作来说，成本、准确性和速度这三个方面应该达成三角平衡。在进行测试平台选型时，商米测试团队对照自身诉求和痛点梳理、对比了很多测试平台工具，希望能够测试平台能够满足以下条件：

① 成本方面

■ 团队希望所选用的测试工具可以满足测试代码及其依赖环境无需在本地部署的条件；

■ 希望能够避免云上部署的资源浪费情况。

② 准确性方面

■ 团队希望选用的测试工具可以实现测试数据、用例的统一协作。因为测试团队中每个人都有一套自己的测试脚本，但不同的脚本在合并地址时往往会出现冲突，后续也只能由不同的脚本负责人各自进行维护，依然难以达到协调的效果，所以需要一个平台工具来帮助全体测试人员展开统一协作；

■ 团队经调研、实践发现，自研的测试框架往往不太稳定。而一般大公司开发的测试框架又相对局限，只能支持某一种协议，或者只支持某一种使用方式。如果要应用到具体的纵向场景中，会有很多需要定制开发的内容。商米的测试团队希望能够解决这个问题。

③ 速度方面

■ 平时测试人员自行整理，或者根据一些原有的开源框架输出的测试报告都不够清晰美观，且写报告的过程耗时较长；

■ 虽然商米的测试团队有自己的 DevOps 流水线，但早期流水线的结构较为简陋，只能支持测试任务的并行。而基于成本的限制，测试用的设备也需要控制在一定数量内，这对测试速度产生了负面影响。

2022 年 4 月，商米的测试团队接触到了 MeterSphere 开源持续测试平台，发现它能较好地满足以上测试工作中的诉求，试用后很快就选定其作为商米的测试工具。对商米测试团队来说，MeterSphere 的优势如下：

① MeterSphere 平台支持云化部署；

② MeterSphere 支持 Kubernetes 集群的动态调度，避免环境长期弃置造成资源浪费，一次性使用后即可直接销毁；

③ MeterSphere 支持测试团队方便地进行多人协作，且提供长期维护的稳定版本；

④ MeterSphere 平台可以通过一对多的方式同时控制多个节点，能够为测试团队提供接口测试与压力测试方案；

⑤ 测试团队之前在 JMeter 上的测试工作量较大，沉淀较多，但依然可以通过 Jar 包的方式将这些测试工作的内容全部快速导入到 MeterSphere 平台。

五、MeterSphere 平台的落地成果

商米研发阶段的流水线包括需求管理、交付协作、提测、开发验证、研发自测、SIT（System Integration Testing，系统集成测试）验证、预发验证、上线审核、生产发布等多个环节。其中，MeterSphere 测试平台主要应用于商米的研发自测和 SIT 验证环节。测试团队通过 Kubernetes 集群的方案部署了 MeterSphere 平台，并将其接入了商米的 DevOps 流水线。目前商米使用阿里云的云效平台作为 DevOps 平台，主要使用其中的代码管理和流水线功能部分。团队希望产品上线之后，依然能有监控线上环境稳定情况的途径。在流水线接入 MeterSphere 平台之后就基本可以实现自动化测试的良好运行，并得到及时的结果反馈。

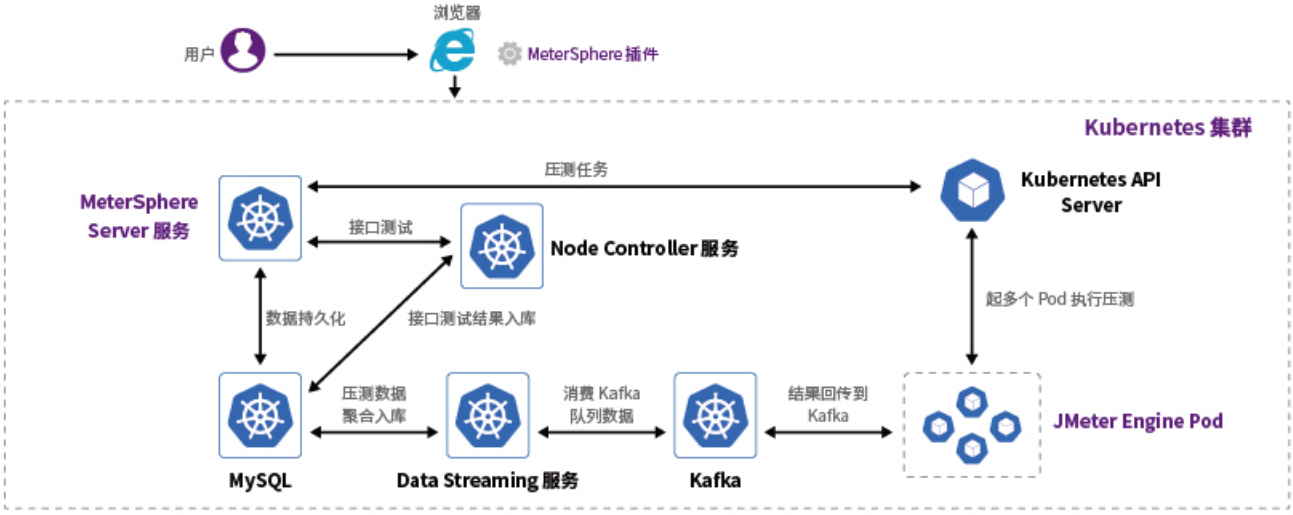


图22 流水线接入 MeterSphere 平台

以下是 MeterSphere 平台在商米测试工作中的一些实际应用场景：

■ **接口场景化** 在接口中编排测试步骤，通过这些步骤将流程链路串联起来。

■ **用例跨环境执行** 需要测试的接口可以分布在多个设备上。由于端上的组件和网络库的版本不一样，它们的验签方式也不一样。为了验证这些签名，让业务逻辑在每个版本上都能成功实现，使用的测试平台需要具有同时满足多种环境配置的能力。MeterSphere 能够很好地满足这一需求。此外，团队在接口测试方面，也尽可能避免了一套数据只能在一套环境上执行的情况，而是通过 MeterSphere 平台使得一套数据可以在多套环境上执行。这样就可以直接满足线下环境、预发环境等在内的所有阶段的签名验证。

图23 满足多套环境配置

■ **接口同步插件** 为了更好地支持整个研发测试协作，测试团队也在研发端配合制作了一款 IDE 插件。这款插件支持研发团队将通过注解的方式写出来的接口信息自动投入到 MeterSphere 平台，及时将接口变动情况同步给测试人员。测试人员可以直接根据接口变更信息去维护对应的用例，有效降低了沟通成本。

■ **海外拨测节点** 由于具有全球化的业务背景，商米需要有能力去巡检全球各个数据中心的服务，确认各集群是不是正常可用、CDN 有无异常，以及网络链路是否通畅。目前商米还不能满足所有端上的 APN（Access Point Name，接入点）设置，但在 MeterSphere 平台接入、DevOps 的末端上线之后，商米可以通过全球各节点去巡检各服务、业务、域名及网络链路是否正常，及时排查业务中出现的问题。

■ **结合 DataEase 制作仪表板大盘** 商米的测试团队还使用 FIT2CLOUD 飞致云旗下的另一款开源产品——DataEase 开源数据可视化分析工具制作了测试仪表板大屏。测试人员可以通过仪表板监测接口测试的用例数量、API 接口或场景自动化的用例数量、用例数量等级的分布、用例增长数量、用例数量增长趋势等指标，更好地辅助测试团队内部进行日常管理工作和问题识别工作。团队现在也将这个仪表板融入了生产质量的全生命周期管理，更加方便地进行缺陷和故障的排查。

图24 商米测试质量大屏

六、未来规划

引入 MeterSphere 开源持续测试平台半年时间以来，商米的测试团队获得了良好的使用体验，提高了工作效率，并通过 MeterSphere 平台解决了一系列测试工作中遇到的问题。在之后的工作中，团队希望能够继续开发 MeterSphere 平台的各种功能，也结合自身业务总结出了一些想通过 MeterSphere 平台实现，或希望 MeterSphere 平台未来能够支持的功能：

■ **代码覆盖率检测** 目前 MeterSphere 平台还无法进行代码覆盖率的检测。希望未来 MeterSphere 平台可以并入全量或增量的代码覆盖率检测功能，或者搭建覆盖率平台，并将其与 MeterSphere 平台的接口测试打通；

■ **全球服务压测** 由于自身采用的是全球化的服务模式，商米的测试工作也会面临容量规划的问题。商米在各地区的业务增速并不均衡，工作人员无法做出准确的预判，也不能及时感知容量的使用情况，往往在线上业务的内存资源告急或发生故障时才能收到反馈。团队希望在全球化的业务中，可以根据每个地域或节点的实时用户量，通过 MeterSphere 平台去建立每个地区的用户压力模型，并进行全球化的服务压测；

■ **基线性能测试** 以往商米的性能基线测试一般是在其他测试工作结束之后进行的。团队希望以后能通过 MeterSphere 平台将性能基线测试加入整个发布流水线之中，希望业务应用在上线前可以通过基线性能测试。因为早期测试代码效率较低，且目前商米业务量增长需要提升整体的开发测试效率，团队需要以基线性能测试为门禁，将其加入商米的测试环境，防止线上事故发生；

■ **UI 自动化回归** 商米的测试团队希望 2023 年能够引入 MeterSphere 的 UI 自动化功能。商米云服务有很多外部站点平台，比如合作伙伴平台、内部管理平台、门户网站、其他开发者中心等，对于这些 Web 网站，团队希望通过使用 MeterSphere 的 UI 自动化测试能力降低测试的工作量。

CASE STUDY

案例研究

国信证券基于开源工具的系统质量保障实践

质量管理、业务监控

国信证券股份有限公司（简称“国信证券”）是一家全国性大型综合类证券公司，在全国 117 个城市和地区共设有 58 家分公司、183 家营业部。国信证券拥有国信期货有限责任公司、国信弘盛私募基金管理有限公司、国信资本有限责任公司、国信证券（香港）金融控股有限公司四家全资子公司，50% 持股鹏华基金管理有限公司。公司及子公司经营范围涵盖证券经纪、融资融券、代销金融产品、做市交易、期货经纪、资产管理、投资咨询业务、创业管理服务业务等。

一、国信证券系统测试的背景及面临的挑战

1. 国信证券系统测试的背景

作为深圳本地的中国头部证券公司，国信证券对系统质量有着极高的要求，对测试质量保障的要求也随之提高。通常来说，金融行业的信息系统通常具有以下的一些特点：

① 双态 IT 系统架构，技术复杂度高 金融信息系统中一般会同时拥有稳态和敏态这样的“双态”业务系统。其中，稳态业务系统主要是指承载用户开户、业务办理、证券买卖、委托撤单、交易报盘、统一清算等证券公司最核心业务的系统。稳态业务的建设一般较早，国信证券的稳态业务系统设计采用了较为传统的总线模式，其中有的系统已经在 Windows 操作系统上运行了大约 20 年的时间；

敏态业务系统则是为了满足基金业务多样化和创新的需求，承载智能选股决策分析等类别的新生业务系统。敏态业务诞生较晚，其系统技术架构相较于稳态业务更新，大部分采用的是微服务架构，少部分采用容器化部署方式，以便更好地满足业务快速迭代的诉求。

证券业务的敏态系统更偏于面向互联网用户侧，用户侧业务从敏态系统进入后，通过稳态的核心业务长链路到达交易所等地。基于“稳态”与“敏态”共存的系统状况，测试产品的选型需要可以同时较好地满足“双态”模式下业务系统的测试要求。

② 多模式交付形态，系统数量多 除主要的业务系统外，国信证券还拥有运营平台、资讯平台等多类支撑系统。对于这些支撑系统，如果在市场调研之后与公司业务需求较为匹配，会通过外购的方式来引入这些成熟的系统产品；如果有一定的定制化需求，则会与厂商合作共建；除此之外，公司有时也会采用内部完全自研的方式进行系统建设。这些支撑系统的来源不一，交付模式不同，导致整个系统的质量管理更为复杂。

③ 行业监管严格，容错率极低 金融行业的信息系统直接关系到用户的资金安全。如果未能及时排查并解决系统中出现的故障，就很容易导致用户在使用的过程中出现资产损失。所以监管方对于公司内 IT 团队的要求也会更加严格，一旦系统出现问题，必须在规定时间内快速处理，否则团队将面临严重的问责与处罚。

2. 金融信息系统面临的挑战

国信证券的金融系统主要分为重要系统和一般系统。这两类系统的侧重点不同，需要采用不同的方式去维护，这也就意味着它们在系统的建设和运维过程中将面临着不同的挑战。

① 重要系统 重要系统是包括开户业务、交易业务以及一些清算业务等在内的系统。重要系统大多与客户资金等重要信息相关，是公司重

点保障的对象。负责的 IT 团队需要具有对 Bug 排查的敏锐度和速度，因而会投入比较多的资源，加强测试、验收、运维，以保障其运行质量。

重要系统质量保障面临的挑战主要有：

■ **业务系统上线的验证依赖外部同事** 目前，国信证券内部共有各类型的组件 350 个，应用系统每年会有约 3000 次变更。并且，公司每个星期都会配合上交所、深交所进行一些通关测试，每两周会有定期的系统重启，每个月会有灾备演练，每半年会有年度灾备演练。每次通关测试或系统重启等活动后，都需要再次对业务系统进行一次全方位的验证。

因监管要求，验证工作更依赖营业部的工作人员来协助进行。但营业部的工作主要是服务客户，营业部的工作人员对于 IT 系统的了解往往不够深入，且跨部门的沟通协作也容易导致信息缺损，在系统验证的过程中就可能会出现一些疏漏。

■ **业务故障报告依赖客户** 国信证券的业务故障报告遵从“客户发现问题并反馈→客户经理反馈→一线应急人员上报→二线运维人员反馈→开发人员处理”的流程。业务故障报告的整个流程链路过长，反馈耗费的时长同步增加，问题现场回溯和排障的困难也会影响修复时间。这种延误和低效率可能触及证监会的监管红线，也容易导致影响的范围进一步扩大，把小问题变成大问题。

总结来说，重要系统的风险管控主要体现在验收测试以及验收之后持续运营的过程当中。

② **一般系统** 国信证券对一般系统的投入资源相比重要系统而言要少一些。对于一般系统，通常仅有一个项目经理来负责管理，包揽技术管控、需求设计、系统测试、系统运维等多种工作。

对于一般系统来说，面临的主要挑战就是无法有效地管控厂商提供系统的质量。国信证券一般是通过直接外购的方式获得一般系统，但厂商的产品质量往往也是参差不齐。在系统交付模式中的需求设计、开发编码、系统测试等部分主要都是供应厂商自发进行，甲方难以介入并提前进行质量管控。只有在系统交付时，才会由外包同事现场进行一轮纯手工的验收测试，系统交付的质量风险较高，缺陷修复后的回归验证周期也比较长。而从质量保障的角度看，一般系统的风险管控在系统上线前就要开始进行。

图25 一般系统交付流程

二、质量管理与业务监控的建设思路

针对不同种类系统面临的问题，国信证券整理了一些解决问题的思路，具体如下：

1. 落实“三个统一”的标准化厂商质量管理

“三个统一”包括统一测试管理、统一标准交付物以及统一工具，三者之间互相关联、互相促进，可以达成“交付系统的质量与效率同步提升”

的目标。其中，在统一测试管理的场景下，可以更好地推进测试工具统一；统一的测试工具及相应的统一测试标准可以促进标准交付物的形成；标准交付物的成型和数量增长也将反哺统一测试管理。通过这样的管理思路，国信证券可以逐步提升统一标准化厂商质量的过程管理工作。

图26 “三个统一”标准化厂商质量管理

2. 业务层主动监控，提前感知系统的可用性故障

针对公司内重要系统的痛点，为达成线上验收、客户报账等重要业务事项的质量保障，国信证券的 IT 团队选择尝试“测试右移”的思路，将测试环节右移到生产环境一侧。在生产环境中自动化模拟用户的业务调用操作，可以进行实时的业务拨测，提高发布验证的效率，提前感知系统的可用性，并保障系统的稳定运行。

在公司业务变更时，团队通过在生产环境中运行接口测试或自动化测试，也可以快速发现生产过程中或系统变更时出现的故障，为系统排障争取更多的时间，减少对用户体验的负面影响时长。

图27 生产环境自动化业务巡检

3. 开源工具助力传统企业低成本快速构建 IT 基础设施

基于前面两条建设思路，国信证券的 IT 团队决定采用“工具先行”的模式，借助低成本的开源测试工具，快速构建 IT 基础设施并保障其运行质量。之所以决定选用开源的测试工具，而不是选择购买闭源产品、定制或自研测试软件，国信证券的 IT 团队有以下几点考量：

① 开源工具的成本较低 除去产品费用等显性成本较低外，开源工具的学习成本、使用中节约的时间成本等隐性成本，相较于其他测试工具也更具优势。因为开源工具的流通更广，员工有过该软件学习和使用经历的可能性就更高，也更容易在丰富的社区教程帮助下学会软件的操作技能，无形中节约了测试团队的学习成本；

② 周期短，见效快 已有的测试工具会比定制或自研的测试工具更加成熟，可以达到“开箱即用”的效果，将之接入业务系统的难度更低，建设周期更短；

③ 技术更先进 开源测试工具往往采用主流技术栈，技术选型具有先进性，可以在一定程度上降低软件学习与使用的门槛；

④ 产品生命力强 开源软件的用户更加广泛，且产品会随着大量用户的不断反馈而持续迭代，更有持续演进的动力，生命力比闭源软件、定制软件或自研软件更强大。

三、MeterSphere 在国信证券的落地实践

1. 开源工具选型

在经过详细的调查选型后，国信证券的 IT 团队选择了 MeterSphere 一站式开源持续测试平台。MeterSphere 最吸引团队的优势包括：

① 开源社区活跃度高 MeterSphere 的用户群广泛，开源社区活跃度非常高。目前 MeterSphere 在 GitHub 已获得 10000 多个 Star、2000 多次 Fork，已经在很多企业中被落地使用；

② 功能全面 作为一款一站式的开源持续测试平台，MeterSphere 中包含了国信证券的测试工作所需要的测试跟踪、接口测试、性能测试等几乎全部功能；

③ 兼容 JMeter 在引入 MeterSphere 之前，国信证券的性能测试工作主要是基于 JMeter 展开的。公司的交易系统等稳态系统中有一些非通用的数字协议，也是通过基于 JMeter 研发的扩展插件去进行兼容和支持的。由于 MeterSphere 也可以兼容 JMeter，引入 MeterSphere 平台后可以方便、快速地在二者之间完成测试工作的对接；

④ 扩展性强 MeterSphere 可以兼容多种协议，拥有丰富的插件体系。这就意味着 MeterSphere 作为一款软件具有充足的扩展性，能够更好地适应国信证券内部的诸多系统；

⑤ 测试环境适配 MeterSphere 支持容器化部署，便于信创环境等多种测试环境的快速迁移和与测试工具的适配；

⑥ 采用主流技术栈 MeterSphere 主要采用 Java 语言编写，使用 Spring Boot（后端）、Vue.js（前端）等常见的技术栈，与国信证券的业务系统和 IT 团队成员技能契合度高。

2. 业务监控的落地实践

① 业务监控全方位覆盖 国信证券的 IT 团队通过 MeterSphere 的接口自动化功能，以“金太阳”移动 APP 为试点，实现了对包括交易、理财、开户、首页、行情、资讯等在内的核心业务系统内业务监控的 100% 覆盖，以及对自建数据中心与行情云上机房的 100% 覆盖。覆盖接口共 591 个，接口覆盖率达到 94.5%。业务监控保持 7*15 小时（09：00~23：59，节假日不休）的不间断运行。

② 监控告警发现、推送、响应、处理的闭环实现 在 MeterSphere 的协助下，国信证券实现了监控告警发现、推送、响应、处理的闭环管控。在业务监控保持 7*15 小时（09：00~23：59）不间断运行的情况下，当业务系统中出现错误，业务监控就会发出对应告警，并将告警推送至统一告警事件平台以及负责人微信，快速触达负责问题处理的 IT 工作人员。

同时，平台可以针对这些告警做出分类分级的处理，协助工作人员快速进行问题的优先级排序并制定处理决策。对于告警的故障问题，MeterSphere 也可以给出机房定位、接口定位、错误内容和错误时间等清晰的信息提示。IT 团队还对业务监控的参数进行了染色，便于知悉错误的提交者是内部工作人员还是外部用户。

在公司业务系统变更时，可能会短时间内触发大量告警。IT 团队设计了告警的可配置自动恢复策略。在系统变更结束、服务恢复正常之后，不用浪费人力去一个个地手动排查处理，这些失效的告警可以自动恢复正常。

③ 实现对私有协议及各种非标准协议的支持 国信证券的业务系统中还涉及多种私有协议和非标准协议，IT 团队在 MeterSphere 平台通过可扩展的插件机制解决了这问题。在整个过程中，按照“依赖厂商 - 合作共研 - 自主开发”三步走的模式，目前已经扩展了 4 个插件，包括飞致云的客户成功团队帮助开发的两种插件、国信证券 IT 团队与飞致云共同研发的私有 TPC 网关插件，以及国信证券自主研发的华锐极速交易插件。后续国信证券将根据业务系统的要求，根据测试需求自主扩展 MeterSphere 平台的插件体系。

④ 在应用变更、系统重启及日常业务巡检方面发挥价值 通过 MeterSphere 平台建设的业务监控体系在国信证券业务系统变更、重启及日常的业务巡检中发挥了很高的价值。业务监控体系上线一年间内，IT 团队共发现了 61 个线上问题，其中 72% 的问题是在日常巡检中被发现，28% 的问题是在应用变更和日常重启中被发现。作为金融类企业，国信证券的线上故障率在同类企业中一直处于较低水平。

3. 基于 MeterSphere 进行业务系统质量管理

① 明确系统质量过程管理规范

通过使用 MeterSphere 测试平台，国信证券打破了系统供应商的黑盒子，明确了对合作厂商交付系统质量的全过程管理规范，即将测试管理的过程“左移”到厂商交付之前。

在系统需求设计之后，国信证券会对此需求设计进行初步评审，然后系统供应商再开始进行开发编码。开发编码结束后，团队会与厂商先

沟通进行一次用例评审，让厂商依据要求进行系统测试，并要求厂商对产出物进行标准化管理。最终，国信证券在进行业务系统的内部测试验收时，会对此过程执行过程审计，通过审计后方可开放系统的线上部署。

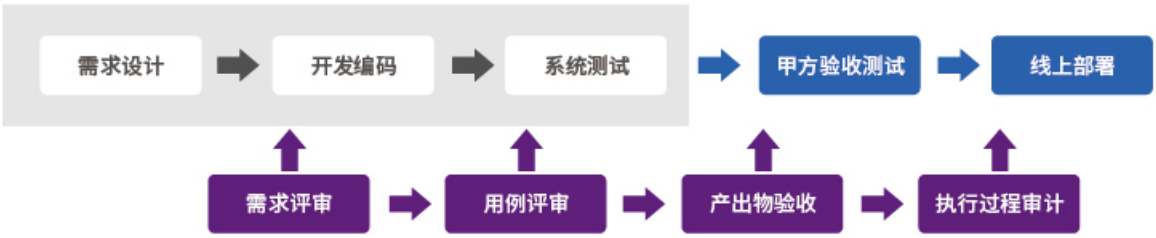


图28 系统质量全过程管理

② 统一平台，标准化交付物，快速验收存档

国信证券要求系统供应厂商与国信证券使用数据兼容的统一测试平台——即 MeterSphere 一站式开源测试平台。在交付业务功能的基础上，厂商需要向国信证券同步交付编写的功能测试用例、接口自动化用例与性能测试用例等，通过 MeterSphere 平台即可将这些测试相关的交付内容直接导入至国信证券的内部环境。这样系统采购之后的维护期就可以在系统供应厂商无需介入的情况下，快速上手执行系统升级或服务迁移等操作，并直接运行已有的用例开展测试工作，实现系统交付的无缝衔接以及厂商和交付系统的标准化管理。

③ 面向不同项目及角色，提供多样服务能力

MeterSphere 平台为国信证券 IT 团队中不同的项目与角色提供多样化的服务能力。

■ 对于开发人员来说，MeterSphere 支持多种接口协议的调试、编辑、回放，同时提供前后端分离开发模式的接口 Mock 服务，还能实现接口测试一键转性能测试，有效提升了软件开发的效率；

■ 对于测试经理来说，MeterSphere 可以系统化地管理测试过程中的资产数据，便于进行用例的储存和复用，也可以直接在 MeterSphere 平台对用例进行在线评审。在 MeterSphere 的“测试管理”模块中，可以对测试任务进行线上分配，并对工作人员的测试进度进行实时追踪。对于在 MeterSphere 平台进行的测试工作，也可以更规范地进行测试过程的审计，即时更新审计结果并为执行动作留痕。审计中发现的缺陷则可以进行分类处理，保障交付系统的质量；

■ 对于应用运维人员来说，MeterSphere 平台可以进行业务变更后的验证，提高验证效率，加速业务更新上线。在业务系统的日常巡检中，MeterSphere 平台也发挥了非常大的作用，当发现系统问题时 MeterSphere 可以及时进行统一告警，并将告警消息推送至负责人微信等，减小问题的影响时长和影响范围，帮助提升整个系统的稳定性。

6. 参考资料

[1] Wolfgang Platz, Cynthia Dunlop,《 Enterprise Continuous Testing：Transforming Testing for Agile and DevOps》，2019

[2] 朱少民，《敏捷测试：以持续测试促进持续交付》，人民邮电出版社，2021

[3] 朱少民，《全程软件测试（第三版）》，人民邮电出版社，2019

[4] 乔梁，《持续交付 2.0 业务引领的 DevOps 精要》，人民邮电出版社，2019

[5] 陈冬严等，《精通自动化测试框架设计》，人民邮电出版社，2016

[6] Gartner,《Take a ‘Shift Left’ Approach to Testing to Accelerate and Improve Application Development》, 2019

[7]Gartner,《China Summary Translation: 'Magic Quadrant for Software Test Automation'》, 2019

[8]Gartner,《Hype Cycle for Software Engieering, 2022》

关于“软件质量报道”公众号：

“软件质量报道”公众号致力于创建健康、安全、绿色的软件生态，分享软件质量管理、软件测试的思想、方法、技术与优秀实践，追踪软件质量领域的热点，及时报道软件质量管理的成功案例或质量事故等。

软件质量报道公众号

关于 MeterSphere 开源项目：

MeterSphere 是 FIT2CLOUD 飞致云旗下品牌。作为一款一站式的开源持续测试平台（github.com/metersphere），MeterSphere 涵盖测试跟踪、接口测试、UI 测试和性能测试等功能，全面兼容 JMeter、Selenium 等主流开源标准，有效助力开发和测试团队充分利用云的弹性进行高度可扩展的自动化测试，加速高质量的软件交付。自 2020 年 6 月发布至今，MeterSphere 开源项目得到了开源社区的广泛认可和积极反馈，并已经在众多企业内落地使用。截至 2023 年 8 月，MeterSphere 开源持续测试平台项目在软件托管平台 GitHub 上的 Star 数量已经超过了 10,000 个，累计安装部署次数超过 175,000 次。

MeterSphere 公众号

编者注：《持续测试白皮书》由“软件质量报道”公众号及 MeterSphere 开源项目组联合撰写，版权归属于“软件质量报道”及 MeterSphere 开源项目组。