专栏：云原生技术

开发外包场景下 DevOps 实施方法及实践

王保中，姚文胜，梁奂，乔宏明，陈泳
（中国电信股份有限公司研究院，广东 广州 510630）

摘　要：目前，DevOps 实施研究主要针对同一公司内部不同团队间协同的场景。首次明确提出了开发外包场景下的 DevOps 实施问题，总结外包场景下实施 DevOps 的意义，分析外包场景下开发和运营团队分属甲方、乙方两个不同企业时实施 DevOps 的难点，提出迭代实施 DevOps 的方法，说明如何选取合适的优化提升切入点，并给出实际运用此方法的实践验证情况说明。

关键词：开发运维一体化；迭代实施；持续集成；持续部署；系统工程；云原生
中图分类号：TP399
文献标识码：A
doi: 10.11959/j.issn.1000−0801.2020321

DevOps implementation method and practice in software development outsourcing scenarios

WANG Baozhong, YAO Wensheng, LIANG Huan, QIAO Hongming, CHEN Yong
Research Institute of China Telecom Co., Ltd., Guangzhou 510630, China

Abstract: The current DevOps implementation research mainly focuses on the scenario of collaboration among different teams within the same company. The DevOps implementation issues in the development outsourcing scenario for the first time were clearly put forward, the significance of implementing DevOps under the outsourcing scenario was summarized, the difficulties of implementation DevOps in the outsourcing scenario were analyzed, where the development and operation teams belong to two different enterprises of Party A and Party B, and the method of iterative implementation of DevOps was proposed, how to select an appropriate entry point for optimization and improvement was explained, and an explanation of the practical application of this method was given.

Key words: DevOps, iterative implementation, continuous integration, continuous deployment, system engineering, cloud native

1　引言

云原生不仅带来容器、服务网络、微服务、不可变基础设施和声明式 API 等技术上的变化，还带来多团队协同开发（Dev）和运维（Ops）管理上的变革。目前，大部分 DevOps 实施方法讨

收稿日期：2020−10−20；修回日期：2020−12−05
基金项目：国家重点研发计划基金资助项目（No.2018YFB1802400）
Foundation Item: The National Key Research and Development Program of China (No.2018YFB1802400)

论的场景主要是针对同一企业内的开发和运营团队之间的协同，但是在现实生产中更为常见的场景是开发外包，即开发和运营团队分属甲方、乙方两个不同的企业，因此甲方、乙方环境下 DevOps 实施方法的研究具有更广泛的实用价值和重要意义。本文从 DevOps 的核心理念出发，针对开发和运营团队分属甲方、乙方企业场景下，提出一种开发外包场景下迭代实施 DevOps 的方法。

在 2009 年的 Velocity 大会上，John Allspaw 和 Paul Hammond 做了题为“每天部署 10 次：Dev 和 Ops 在 Flickr 的协作”的演讲[1]，讲述了他们如何建立 Dev 和 Ops 共享的目标，运用持续集成相关技术，极大地提高了部署这一瓶颈环节的工作效率。这次演讲引起听众广泛的共鸣，意义深远，被当作 DevOps 理念正式确立的标志。自此，DevOps 在全世界蓬勃发展，经过 10 年的不断完善，目前已经进入了 Gartner 技术成熟度曲线图中的生产成熟期（plateau of productivity），表明在企业内部引入和推广的案例开始激增。

另一方面，随着数字化运营能力对于一个企业越来重要，IT 开发外包的情况也越来越普遍，实施 DevOps，能提高甲方、乙方的合作效率，提高甲方对开发过程的标准化管理能力，从而将乙方的开发能力聚合成为甲方的开发能力。因此，外包场景下 DevOps 实施能力日益成为企业的核心竞争力之一。

1.1 DevOps 的定义

截至目前，DevOps 并没有权威的定义。有人甚至以“盲人摸象”作比喻，主张每个人都可以有自己的 DevOps[1]，第一个人说它最像一棵树，第二个人说它像一块毯子，第三个人说它像一条蛇。

一个比较全面的定义是：DevOps 是对敏捷软件开发与精益生产思想的演进，应用于 IT 端到端的价值链中，使业务基于现代信息技术，并通过文化、组织与技术变革来获得更大的成功[1]。这个定义的一个特点是比较全面，尤其他指出实施 DevOps 的目标是使业务获得更大的成功。另一个特点是没有提出一个直接明了的指标作为目标，因此一方面可以认为这个定义实操性不强，另一方面可以认为这反而为在实施 DevOps 过程中制定一系列针对性强的 KPI 目标值留下了空间。

本文采用一个实操性强的定义：DevOps 是一套实践方法，在保证高质量的前提下缩短系统变更从提交到部署至生产环境的时间[2]。这也是维基百科目前所采用的关于 DevOps 的定义。这个定义有以下特点。

（1）界定了企业 IT 部门的整体目标。包括从开发到部署上线的 IT 端到端全流程，突破了对开发部门、运维部门分别制定不同 KPI 的做法，为开发部门和运维部门全面合作奠定了基础。

（2）简明清晰。在保证质量的前提下，直接以 IT 端到端价值链时间长度为 KPI 目标值。

（3）具体界定了 IT 端到端价值链的起点和终点。起点是系统变更提交，狭义而言，即开发人员提交（commit）新开发的代码的时间，这标志着基本开发过程结束，部署工作开始。但从广义上讲，起点应该从业务构想的提出开始。

制造业精益一个基本概念叫价值流，类似地，DevOps 也有一个概念“技术价值流”[3]，定义为业务构想转化为向客户交付价值的、由技术驱动的服务所需要的流程。一个技术价值服务，即一项工作任务在工作流程中从左向右流动，左边的起点源头是业务构想的提出，右边的终点是一项为客户带来价值的服务，具体而言，就是一个生产环境中正常运营的软件系统或者技术服务。从开发到部署至生产环境，包括多乙方外包供应商的生产流水线[4]，如图 1 所示。

1.2 DevOps 的核心理念及实施三步法

DevOps 从提出到现在能如此深入人心，不仅是云计算等技术的推动，也不仅是建设一个系统

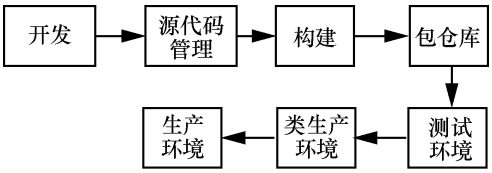


图 1　开发到部署至生产环境流水线

实现了持续集成、持续部署流水线，更重要在于其蕴含的核心理念：强调流水线整体效益的最大化，而不是流水线上单个组织或部门效益的最大化[4]。更有甚者，单个组织或部门效益的最大化，有时会成为整体效益最大化的障碍，比如，如果过份强调系统运行的稳定性，运维部门往往会利用各种上线审批流程来减少新功能部署上线的次数。而这是目前很多企业正在发生的情况。

为实施此核心理念，需要做到以下 3 点，对应于经典 DevOps 实施的三步法[3]。

第一，建立一个待优化的整体目标，并以此目标为导向，运用系统思维全面分析，建立端到端的流水线流程。

第二，局部目标与整体目标相协调。并不是说有了整体目标就可以忽视局部目标。局部目标能存在正是因为它有合理性，这就需要在流水线上各节点上的局部目标与整体目标相统一。有两个具体的做法[3]。

（1）将局部需求前置。不能等制品流到了某个节点才用这个节点的局部目标要求来检测它。因为这时一旦发现不符合局部要求，就要返工，造成资源的浪费，降低效率。一个典型的例子就是代码安全检查，等到开发完成才来进行代码安全检查就有些晚了。正确的做法是通过与整体目标协商，将局部检测要求转化成需求提前到流水线起始的开发设计阶段，这样的效率才是最高的。

（2）即时反馈。一旦在某个节点上检测发现问题，就应该立即改正，不能再流向下一节点，道理是显然的，因为流向后面节点所做工作都需要返工。但实际工作中，一个违反这个原则的案例是，修改一段代码后一直要等到流过集成测试、用户接受测试等环节后才知道这次修改是否正确。所以，建立自动测试流水线的原因不仅在于提高测试效率，更在于立即检测，尽可能避免让有问题的代码修改流入下一节点。

第三，持续优化。一项任务会流过流水线上的各个节点，整体表现由所有节点的表现决定。因此，整体表现优化涉及多个局部节点表现的优化，不能用瀑布式思维，试图将所有要优化的工作一次性全部实施完成，而应该采用迭代实施方法，持续学习与实验，它打造出一种高度信任的文化和一种科学的工作方式，并将对组织的改进和创新作为日常工作的一部分。每次迭代过程都要分析整个流水线中各节点的效率，找出其中的关键约束点[5]进行优化。然后进入下一次迭代，再次分析各节点的效率，找出本次迭代需要优化的关键约束点，这就是迭代实施的方法。

2　外包场景下实施 DevOps 的难点分析

相比于开发团队与运维团队属于同一公司，开发团队与运维团队分属甲方乙方不同公司的环境下，实施 DevOps 更有难度。

首先，甲方、乙方的目标并不总是一致。目前大部分的软件开发外包合同都以业务需求书作为标的物，并且还有 1~3 年的免费维护期。对比 DevOps 的整体目标，在保证高质量的前提下缩短系统变更从提交到部署至生产环境的时间，就会发现这两者之间的差别：乙方以提交满足甲方业务需求书的软件产品包作为目标，而 DevOps 目标还包括将乙方提交的软件产品安装包部署到生产环境中并投入生产运营作为目标；要求乙方免费维护，意味着运营维护是没有价值，只有开发才是有价值的，这必然导致乙方在后期运营维护中减少投入，DevOps 倡导的开发团队和运营团队顺畅沟通的前提基础都不存在了。

因此，在甲方、乙方环境下实施 DevOps 首先要建立一个甲方、乙方共同的整体目标，最有

效的方法是通过合同的标的物，使甲方、乙方能在目标上保持一致：合同的标的物应该改变只含有业务功能要求的做法，增加将乙方提交的软件产品安装部署到甲方生产环境中；合同的标的物中还应该增加后期运营服务，合同费用应该包括运营服务费。运营服务费的计算不能只以服务时间为唯一的核算标准，否则对于乙方来说，服务时间越长而不是运营效率越高收益越大，这样又有背于提高整体效率的 DevOps 核心理念。

其次，不同企业之间有不同文化、不同的工作流程和机制。要整合在一起，有一个工作流程的适应问题，更有一个知识产权、管理权的协调问题。甚至物理距离也会是一个壁垒，社会科学研究已证实，超过 30 m 的物理距离会造成人们之间协作的壁垒[2]。要使甲方、乙方团队衔接在一起高效工作，不可能一蹴而就，必须采用迭代实施方法，建立实施 DevOps 的长效机制。

3 建立甲方、乙方场景下实施 DevOps 的长效机制

根据前文所的述 DevOps 实施三步工作法，甲方、乙方场景下通过 3 个步骤持续推进 DevOps 实施：建立端到端流程、建立反馈机制和迭代优化。

3.1 建立端到端流程，形成甲方乙方协同工作机制

3.1.1 全面分析，迭代实施

实施 DevOps，需要全面梳理从业务构想到在生产环境中部署完成端到端流程。人们往往只聚焦于需求实现的生产流程，而忽视了管理流程。管理流程分为项目工程管控类和财务相关管控类，如图 2 所示。

这些管理流程表面上与生产流程不直接相关，但是其中的审批环节会计入流水线的总耗时。如果忽视这些管理环节，会出现 IT 部门在开发部署时间方面压缩了很多，而工程管理部门、财务部门却改进很有限的情况，因此必须全面分析梳理端到端流程。最终达到的目标应该是所有的管理类节点所需的数据和信息都从生产流程中自动获取或计算，在流水线的各个环节中提交的交付物也应该避免重新填写，否则不仅重复劳动，更会出现数据在多处有复制版本的现象，需要花费额外时间进行核对，降低生产效率。

但问题的另一面是，如果一次性将所有的流程都搬到线上，工程量将十分浩繁。图 2 只是端到端流程的 1 级视图，如果要细化到 2 级、3 级流程视图，就会像集成电路图一样繁杂。所以，在实际的 DevOps 实施过程中应该坚持迭代实施的

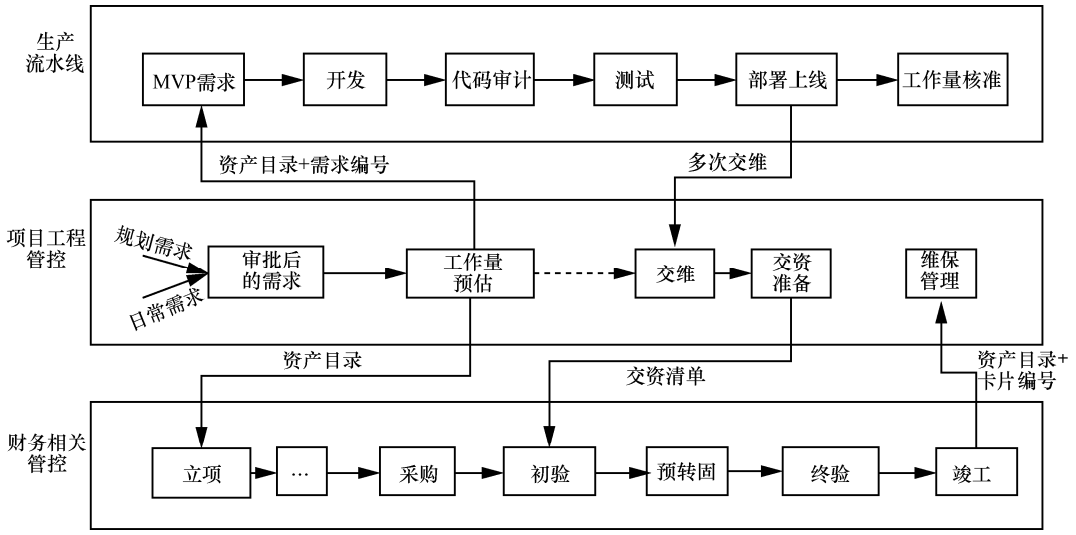


图 2 从业务构想到在生产环境中部署完成端到端流程图

方法。

迭代实施与分步实施有相似之处，但最大的区别有两点：

（1）迭代实施以总体目标为导向，每一次迭代都会以缩短从系统变更到部署上线的总时长为目标来规划迭代实施内容，而分步实施一般会按照难易程度来规划实施内容。因此，迭代实施的每一次迭代应该都能带来总体目标的改善，而分步实施是否会带来总体目标的改善有着不确定性；

（2）迭代实施从第一次迭代开始就应该建立全流程看板[3]，例如：可以按照图 3 的视图建立全流程看板。即使有些流程环节暂时没有线上实现或者与其他系统的接口没有打通，也应该手工填报流入、流出该节点的时间，从而计算该节点的处理历时。这样在全流程看板中，就能够看出哪个环节是瓶颈，从而制订下一次迭代需要优化的内容。

3.1.2　清晰定义甲方、乙方交接流程

从开发到在生产环境中部署上线的端到端流程，必然包含开发到运维的交接过程。一项工作在团队之间交接时，需要大量的沟通请求、委派、通知、协调的工作，而且经常需要排优先级、调度、消除冲突、测试和验证[3]。在甲方、乙方场景下，交接的成本更高。为了加快实现开发到运维的工作快速地从左向右流动，一方面要尽可能地减少交接次数，另一方面对于必要的交接，必须清晰地进行定义，交付物的内容、交接双方的责任、交接合作的方式都要事先给予明确的说明。

图 3 中，交维环节是甲方、乙方进行运维责任的交接。它既是生产类流程中开发阶段结束、部署阶段开始的重要标志，也是工程管控流程中的运维责任由乙方向甲方转换的控制节点，还是财务管控流程中固定资产交接的一个重要触发点，所以交维环节是端到端流程的一个关键点，根据迭代实施方法，首先需要对交维环节进行定义明确。

在交维环节乙方至少应该提交的交付物如下。

- 业务需求文档：细化到页面设计（不含美工）。
- 技术需求文档：明确技术路线、模块分解等。
- 详细设计文档：说明模块间接口、表设计等。
- 安装说明：包括所需软硬件资源及版本。
- 操作说明：以“Use Story”为线索进行说明。
- 安装包：对版本、主要功能进行说明。
- 安装脚本：要求能一键安装。
- 测试用例文档：以“Use Story”为线索进行说明。
- 测试结果文档：包括自动测试结果和人工测试结果文档。
- 故障处理/运维文档：包括监控点布置说明。

在项目开始阶段就清楚定义在交维环节应该提交的交付物，不仅能提高开发运维团队之间的交接效率，更重要地，有利于乙方在开发阶段就将运维需求尽早纳入开发需求中进行设计，实现 Ops 与 Dev 更紧密地结合[3] 。

3.2　通过自动化建立快速反馈机制

在清晰得出甲方、乙方交接各环节的基础上，应逐步实现交接的自动化。凡有交接必有规范，凡有规范尽可能自动化。一般地，人们对自动化部署、自动化测试、自动集成、自动化发布比较重视，而对交维过程的自动化比较忽视。

在交维环节，乙方会提交许多相关文档给甲方。但这些静态文档与系统代码是否相符、质量如何，如果不实现自动化，甲方无法给出快速的反馈意见。交维环节可以通过以下步骤逐步实现自动化，加快对甲方对乙方的反馈。

（1）自动化部署：甲方根据安装说明中的软硬件资源要求准备好测试用软硬件资源，利用安装脚本一键安装好测试版本。

（2）自动化测试：在安装好测试版本的基础上，执行自动化测试脚本，自动生成自动化测试报告。而对于人工测试结果，甲方进行人工测试时能做到实时反馈结果给乙方。

（3）详细设计文档验证，根据文档中说明的接口，进行接口的自动测试，将测试结果与文档中设计的结果进行对比。

（4）故障处理/运维文档验证：根据文档中法人监控点布置，对监控点进行可视化监控验证。对某些监控点进行异常值设置，查看监控视图是否产生告警信息，查看运维操作是否能够有效恢复故障。

3.3 成立运营社区，形成甲方、乙方持续改进共赢机制

如上文所述，实施 DevOps 不是上一套 DevOps 平台就可以一劳永逸的，而应该运用系统论[6]整体思维，采用迭代实施方法，不断优化改进。成立运营社区是一个好办法，将甲方、乙方都纳入运营社区，有利于把局部发现转化为全局优化、将日常工作的改进制度化。运营好社区，需要注意以下几点。

3.3.1 建立持续改进机制

（1）成立 DevOps 推进组织

《组织管理的技术——系统工程》[6]提出：“这就要求以一种组织、一个集体来代替先前的单个指挥者，对这种大规模社会劳动进行协调”，在外包场景下实施 DevOps，涉及多个单位的合作协同，应该成立 DevOps 推进组织，主要职责包括统一定义术语，设定实施 DevOps 每次迭代优化内容，进行 DevOps 实施培训、社区议题设置等。

（2）统一定义术语

有一个重要而常被忽视的问题是统一的术语定义，在外包场景下沟通不顺畅往往会追溯到甲方乙方对术语定义的不统一。

1）系统、应用、模块

何谓系统、应用和模块？例如，电信企业营业厅中受理系统一般都分为前端和后端，后端分为多个模块，其中某些模块的功能又会被手机端的自助受理应用所复用。在微服务架构下，系统、应用、模块等界定没有清晰的标准。一旦 DevOps 平台建成启用，多个项目组向其中提交代码，进行版本控制，进行看板的监控，如果甲方、乙方对系统、应用、模块没有统一的定义，会很快引起歧义。

按照简单实用的原则，“应用”对应着一个独立的安装包，是 DevOps 平台进行版本控制的最小单元。废止“模块”，就是说“模块”是乙方内部的概念，对于 DevOps 平台和甲方而言是不可见的。“系统”由“应用”组合而成，具有明确而独立业务功能的 IT 单元，是一个体现业务目标的概念。

2）项目、需求、迭代

首先需要指出，“项目”这一术语有两种不同的语境，分为“开发项目”和“业务项目”。开发项目，是与应用系统紧密相关，系统应用与开发项目是一对多的关系。业务项目，强调业务目标。例如，“XX 企业云化改造项目”。

项目、需求、迭代之间的概念模型如图 3 所示。

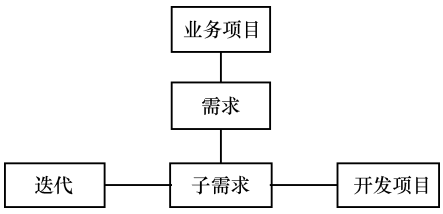


图 3 项目、需求、迭代版本概念模型

其中的关键点是：业务项目是工程管控类和财务相关管控类的概念，开发项目才是 DevOps 平台中的概念，必须将两者术语的内涵和对应关系定义清晰，甲方乙方沟通起来才能顺畅；将需求按照系统的维度分解成子需求后，每一个子需求与开发项目是一一对应关系。简单地，开发项目=系统+子需求。因此，建议对开发项目的命名

遵循以下标准：系统名称（或系统迭代版本）+立项年度+子需求概要。比如：A 系统 2019 年智能化改造项目、A 系统 2020 年模式自动发现及加载项目。这种命名方法有助于解决一般 DevOps 实施中的一个难题：对单个开发项目的管理比较完备，而对于基于系统维度的跨年度多期项目的分析管理功能比较弱。

3）建立统一的系统名称清单

对于新建系统，应该建立报备流程，才能避免前清后乱。常见的一个误区是，将新建系统报备流程与系统的维护职责、固定资产的移交关联起来，固然这样流程很完备，但是实际执行过程中由于协调难度过大，反而使报备流程卡在其中。例如，甲方业务部门为抢抓商机需要紧急开发一个系统，甚至都来不及与甲方 IT 维护部门协调，就直接找到一个乙方开发商进行了系统的开发。经过市场的验证，这个系统使用情况良好，值得继续保留。此种情况下，甲方 IT 维护部门认为这个应急系统不符合总体的系统架构规范，不愿意接手维护责任。这个系统就成为一个技术债务，游离在甲方应用系统名称清单之外。鉴于此，建议应用系统报备流程是一个简短流程，与应用系统的维护职责是否移交无关，以保证能形成一个完整的甲方系统名称清单。

建立统一的系统名称清单使用规定。比如，规定必须列入系统名称清单后才能启动运维责任移交流程，才能获得后续优化改进的投资等。这种做法的好处是，通过各种使用规定，强化系统名称清单的权威性，从而增强需求提出部门、开发部门主动与维护单位沟通协调的积极性。

3.3.2　建立甲方、乙方共赢机制

利用 DevOps 平台，甲方能够实现与乙方开发过程的衔接，同时又能极大提高开发部署的自动化程度。但要把甲方对乙方开发过程标准化的管理能力转变成甲方的核心竞争力，必须建立甲方、乙方共赢机制，才能使乙方自愿地由自身自由掌控的开发过程转变为受甲方控制的开发过程。乙方能从 DevOps 实施中获得的益处是，按照甲方的 DevOps 规范进行开发过程的标准化管理，能够增强对甲方运营需求等非功能性需求的理解，提高乙方的开发效率，减少返工，从而达到甲方、乙方共赢的目的。为达到此目的，建立运营社区，形成甲方乙方持续改进沟通的机制和甲方、乙方共建共赢生态，这成为必然的选择。

4　选取合适的优化提升切入点

建立甲方乙方共赢机制，实现甲方乙方在 DevOps 平台端到端流程的无缝衔接，不是 DevOps 实施的结束，而是利用 DevOps 平台优化整体目标的开始。DevOps 的整体目标是提升从技术业务构想到部署运维的可靠性和频次，所以接下来要利用 DevOps 平台，实现对流程中约束点进行优化。

如何选择流程中的约束点？这个看似复杂的问题有一个简洁而有效的解决方法[4]：看在哪个节点前堆积的半制品量最多，就说明这个节点是关键约束点。在这个关键约束点的效率没有提升之前，优化其他节点的效率对整体目标的优化是没有贡献的，甚至会起负作用。

从图 1 可知，可以从以下两个节点场景切入进行优化。

场景 1　乙方向甲方提交二进制安装包到包仓库中，然后由甲方负责在测试环境中进行安装，进行用户接受测试、回归测试等一系列测试后，并部署到生产环境中，从而完成新版本的发布工作。

场景 2　乙方向甲方提交源代码，由甲方构建生成二进制安装包，然后甲方按照场景 1 完成新版本的发布。在此场景中，甲方能实现对乙方开发过程的管理，比如，实现对乙方开发时间的管理、对架构的管理等，还可以对乙方提交的源代码进行代码审计和安全检查，从而增加系统代

码的可靠性。但在实践中，乙方往往会担心对源代码的知识产权被甲方移作他用，这需要甲方、乙方在合同中对源代码知识产权的保护有详细的约定和较强的互信基础。

最终状况下，DevOps 平台应该能实现场景 2 下完整的 CI/CD（continuous integration/ continuous deployment）流水线。但在现实中部署节点前积压的工作肯定更多，所以场景 1 能实现从安装包到测试环境的秒级自动部署，是实施 DevOps 一个好的优化提升切入点。

4.1 从安装包一键部署到类生产环境

从技术上讲，利用 Jenkins+Docker 成熟的开源组件，可以实现从安装包到准生产环境的秒级一键部署，但更需要关注的是 DevOps 实施带来的管理方式上的改变，具体包括以下几个关键点。

（1）系统应用管理：即前文所述，建立一个全企业唯一的系统应用名称清单。

（2）软件资产管理：以企业级的系统名称清单为线索，建立全企业唯一的软件资产库。改变传统的软件资产分散在各个项目组进行管理的状况，实现全企业利用统一的规范进行版本管理。

（3）应用配置管理：将系统应用部署所需要的软硬件资源说明、系统应用配置信息（如：数据库地址、端口等）从安装包中独立出来管理，严禁乙方将配置信息硬编码到代码中。这样通过不同的配置信息就能实现同一安装包的多场景自动安装部署。

（4）基础设施管理：目前的理念是基础设施即代码[7]。一般地，通过 Kubernetes 实现对硬件资源的统一管理调度，为自动部署准备好硬件环境。

（5）自动部署：将以上（1）~（4）的准备工作做好后，就只剩下纯技术操作，即通过 Jenkins 读取系统应用配置信息，将安装包从软件资产库中检出，并自动安装到基础设施上。

4.2 建立发布到生产环境的流水线

部署到生产环境中，包括两个动作：

动作 1 将一个安装包在生产系统中部署成功；

动作 2 通知用户进行使用。

其中，动作 1 与部署到测试环境的操作基本相同，动作 2 才是部署到生产环境中要着重要考虑的问题。

为了使用户平滑升级到一个系统应用的新版本，应该采用灰度发布的方式，将用户分批导流到一个系统的新版本上，这样对用户的影响才降低到最少，升级的风险才降低到最小。灰度发布需要甲方提前向乙方说明此要求，这样乙方开发阶段就应该考虑部署到在生产环境中的需求，局部需求前置，是实施 DevOps 一个关键要点。

5 实践验证

电信运营商在 IT 领域广泛采用了外包模式，诸多 IT 系统的研发和运维需要电信方运维团队和合作伙伴研发团队的协作。近年，IT 上云是中国电信云网融合战略的重大举措之一，目标是完成中国电信 IT 系统云原生架构改造，涉及 31 个省公司上万人 IT 开发和运维团队的协调、上千个 IT 系统的改造。以此为契机，中国电信运用此迭代实施 DevOps 方法，利用全集团统一的 DevOps 平台，将全部上云系统纳入管理，为 1 000 多个系统提供代码托管、安全扫描和 CICD 流水线服务，为持续迭代优化打下了基础。

5.1 成立 DevOps 推进组织

中国电信成立了 IT 上云推进组织，其中，中国电信股份有限公司研究院作为专门的 DevOps 推进技术支撑中心，负责制定 DevOps 系统开发和实施推进规划，交由中国电信系统上云社区讨论后实施。中国电信系统上云社区由集团公司部门组织建立，成员包括 31 省 IT 开发运维人员和各 IT 系统开发商。统一 DevOps 推进技术支撑中心有利于持续迭代推进 DevOps，避免“毕功于一役”式的项目实施方式。开发运维社区有利于运维团队和开发团队的沟通交流，实施推进规划的

需求来源于社区中不同团队的经验分享和难点痛点说明，经过讨论后再发布实施有助于凝聚不同团队间共识，与仅通过行政命令强制实施相比，增加了甲方、乙方实施的主动性，提升了实施效果。

5.2　选好切入点

选好切入点，使第一次迭代就能体现业务上的收益，对于 DevOps 转型的持续推进尤为重要。各省公司 IT 运维流程、各开发商的开发管理流程各不相同，管理水平参差不齐，希望优化提升点侧重点差别很大。开发运维社区开会讨论收集的切入点主要包括：

- 架构管理，希望通过 CI 过程增强甲方对乙方开发商架构管控能力；
- 度量管理，通过 CI 过程中对乙方开发过程的管理，核算甲方投资的合理性；
- 需求管理，实现业务需求与多次迭代技术方案的直接关联。

这些当然有其合理性，但根据上文所述的理由，最后形成共识，以自动化部署作为切入点。并且，在运维社区的讨论过程中，也认识到各省公司、开发商优化侧重点的不同，将纳入 DevOps 平台管理的系统分为 3 个等级——L1、L2、L3：L1 级只提交安装包和自动化安装脚本，由平台实现自动化部署；L2 级在 L1 的基础上，提交源代码和自动编译、Docker 镜像打包脚本，并且通过 DevOps 平台实现代码的版本管理；L3 级在 L2 的基础上，实现对 CI 过程管理。通过分级管理，既统一了切入点，又可以利用一些系统探索下一个迭代的优化举措。

5.3　持续迭代优化

完成了自动化部署这个切入点的推进工作，接下来就要开始下一个迭代优化周期。优化提升包含两个方面：（1）优化 CICD 流水线的关键环节，从而提升从开发到部署至生产环境的效率；（2）为达成此目标，优化 DevOps 平台相应的功能。工作流程是，首先在开发运维社区中展开讨论，然后由 DevOps 推进技术支撑中心制定统一的优化规划。目前中国电信已经将安全检测能力接入 DevOps 平台，从而纳入 L2 级管理的系统在开发阶段就进行安全检测，改变以前只能等乙方开发商交付产品件才进行安全检测的被动局面。这种改变在较短的时间内就实现了，证明迭代实施方法在外包场景下推进 DevOps 转型的有效性。

6　结束语

云原生架构使多个微服务能组织成为功能强大的系统，也使多团队之间的开发和运维高效协作成为可能。外包场景下实施 DevOps，通过设定开发运维共享的整体目标，采用迭代实施的方法，成立 DevOps 推进组织，建设甲方、乙方共生的开发运营生态社区，逐步把多个乙方组织的开发运维能力聚合成甲方的核心能力，实现甲方、乙方共赢的整体目标。

参考文献：

[1] 奥列格•斯克伦尼科. DevOps 精要：业务视角[M]. 北京：清华大学出版社, 2020.
SKRYNNIK O. DevOps–a business perspective[M]. Beijing: Tsinghua University Press, 2020.

[2] 伦恩·拜斯, 英戈·韦伯, 朱黎明. DevOps: 软件架构师行动指南[M]. 胥峰, 任发科, 等译. 北京：机械工业出版社, 2017.
BASS L, WEBER I, ZHU L M. DevOps: a software architect's perspective[M]. Translated by XU F, REN F K, et al. Beijing: China Machine Press, 2017.

[3] KIM G, HUMBLE J, DEBOIS P, 等. DevOps 实践指南[M]. 刘征, 王磊, 马博文, 等译. 北京: 人民邮电出版社, 2018.
KIM G, HUMBLE J, DEBOIS P, et al. The DevOps handbook how to create world-class agility, reliability, and security in technology organizations[M]. Translated by LIU Z, WANG L, MA B W, et al. Beijing: Posts&Telecom Press, 2018.

[4] SHARMA S. DevOps 实施手册在多级 IT 企业中使用 DevOps[M]. 万金, 译. 北京：清华大学出版社, 2018.
SHARMA S. The DevOps adoption playbook aguide to adopting DevOps in a multi-speed IT enterprise[M]. Translated by WAN J. Beijing: Tsinghua University Press, 2018.

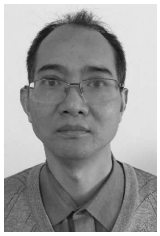
[5] 基恩·金, 凯文·贝尔, 乔治·斯帕福德. 凤凰项目—一个 IT 运维的传奇故事[M]. 成小留, 刘征, 等译. 北京: 人民邮电出版社, 2019.
KIM G, BEHR K, SPAFFORD G. The phoenix project a novel

about IT, DevOps, and helping your business win[M]. Translated by CHENG X L, LIU Z, et al. Beijing: Posts & Telecom Press, 2019.

[6] 钱学森, 许国志, 王寿云. 论系统工程（新世纪版）[M]. 上海: 上海交通大学出版社, 2007.
QIAN X S, XU G Z, WANG S Y. Theory of systems engineering (new version)[M]. Shanghai: Shanghai Jiao Tong University Press, 2007.

[7] MORRIS K. 基础设施即代码：云服务器管理[M]. 金明, 钱伟, 等译. 北京: 人民邮电出版社, 2018.
MORRIS K. Infrastructure as code managing servers in the cloud[M]. Translated by JIN M, QIAN W, et al. Beijing: Posts & Telecom Press, 2018.

[作者简介]



王保中（1967− ），男，博士，中国电信股份有限公司研究院云网运营技术支撑中心高级工程师，长期从事 IT 规划和大型 IT 系统开发项目管理工作，主要研究方向为云原生系统迭代式开发和运维管理。

姚文胜（1969− ），男，中国电信股份有限公司研究院云网运营技术支撑中心高级工程师，主要从事云网融合平台规划研发建设、云原生 DevOps 研发实施推广、IT 运营创新和产业合作等工作。

梁奂（1970− ），男，中国电信股份有限公司研究院工程师，长期从事电信 IT 系统规划与 IT 服务管理方面的工作。

乔宏明（1979− ），男，中国电信股份有限公司研究院工程师，主要研究方向为与 IT 规划和与 IT 相关的技术。

陈泳（1975− ），男，中国电信股份有限公司研究院云网运营支撑中心工程师，主要从事企业上云规划、DevOps 推广实施、PaaS 平台检测与运营等方面的工作。